MSc program
“Biomedical Engineering”

MSc Thesis
of Angeliki Bicaku
A.M.: 1096175

Title:
Neuromorphic Photonics based on Vertical Cavity Surface Emitting Lasers

Supervisor:
Asst. Professor Dimitris Alexandropoulos

Thesis Number: 1096175
Patras, 15/07/2024
The thesis reports original material developed by Angeliki Bicaku and does not violate Intellectual Property Rights in any way. Reuse of existing material is properly referenced, while potential use of figures, graphs, etc is being reproduced after receiving the respective license from the IPR holder.
Acknowledgements

First and foremost, my deepest gratitude goes to my beloved family. Their continuous support and encouragement have been my bedrock, without which this journey would have been insurmountable. To my friends, thank you for being my anchors, offering solace and joy through the highs and lows.

I am profoundly grateful to my supervisor, Associate Professor Dimitrios Alexandropoulos, whose guidance has been invaluable. His faith in my abilities and insightful feedback, along with his professionalism, have shaped the course of this research.

Special thanks go to his exceptional team of PhD students, especially Mr. Panagiotis Georgiou and Mr. Christos Tselios. Their continuous assistance, collaboration, and expertise have been crucial to the success of this work. Their dedication and willingness to help have made a significant impact on my research journey.

I extend my appreciation to all my teachers in the Interdepartmental Master’s Program in Biomedical Engineering. Special thanks to the program director, Professor Konstantinos Moustakas, for his leadership and support.

I am also grateful to the Department of International Affairs at the University of Patras, especially Professor Michael D. Logothetis, and Mrs. Polyxeni Christia, International Relations Administrative staff, for supporting me before and during the Erasmus+ placement at RWTH Aachen University. Additionally, I would like to thank the University of Patras and the Department of International Affairs for the grant that enabled me to realize the Erasmus+ placement in Aachen, Germany.

During my traineeship, I had the opportunity to work under the guidance of Prof. Dr.-Ing. Max Christian Lemme and Professor Alwin Daus at the Chair for Electronic Devices. Their valuable knowledge on neuromorphic devices enriched my experience significantly.

Inspired by Sir Isaac Newton’s words, "If I have seen further, it is by standing on the shoulders of giants," I have been fortunate to be supported by individuals with immense knowledge and patience throughout this journey. This work is as much yours as it is mine.

Thank you all for being an integral part of this achievement.
Abstract

Traditional von Neumann computers, based on complementary metal-oxide-semiconductor (CMOS) technology, inherently lack the ability to process complex data or learn in a manner akin to the human brain. This limitation has driven significant research efforts toward developing neuromorphic (brain-like) systems that emulate the brain’s information processing mechanisms. While electronic approaches to artificial neuronal models have been explored for decades, they face intrinsic constraints such as reduced bandwidths, slower operation speeds, and crosstalk issues.

In contrast, photonic approaches for neuromorphic systems exploit the advantages of optical electronics, such as ease of analog processing and the ability to achieve full parallelism by transmitting multiple signals on a single waveguide at the speed of light. Vertical-cavity surface-emitting lasers (VCSELs) are particularly promising due to their mature technology, ultra-small footprint, low cost, high-speed operation, and potential for large-scale integration. This study explores the dynamic behaviors of optically injected Quantum Dot (QD) spin-VCSELs and their potential applications in neuromorphic photonics.

Dimensionality reduction techniques such as UMAP, t-SNE, and PCA, followed by clustering methods including K-means, DBSCAN, and Agglomerative clustering, were employed to analyze the time series data of the laser’s ground state (GS) and excited state (ES) emissions. UMAP provided the most distinct and well-separated clusters, particularly for the ES-minus dataset, resulting in accurate categorization of dynamic states into spiking and chaotic behaviors, with further differentiation of sub-categories within spiking behaviors such as low-intensity spiking, chaotic spiking, bursting spiking, and normal spiking behavior. These findings underscore the potential of VCSELs in creating advanced, efficient, and high-speed computing systems, paving the way for future innovations in adaptive control, learning, and sensory processing.
Extended Abstract

Η εξέλιξη των νευρομορφικών συστημάτων στοχεύει στην αναπαραγωγή της εξαιρετικής αποδοτικότητας και του ικανοτήτων μάθησης του ανθρώπινου εγκεφάλου, υπερβαίνοντας τους περιορισμούς των παρελθόντων αρχιτεκτονικών von Neumann. Οι φωτονικές προσεγγίσεις, ειδικά τα VCSELs (λέιζερ επιφανειακής επιστροφής κάθετης κοιλότητας), παρουσιάζουν μια υποσχόμενη οδό για την επίτευξη της ταχύτητας και της διασυνδεσιμότητας που απαιτούνται για την προηγμένη νευρομορφική υπολογιστική.

Η έρευνα αυτή επεκτείνεται στις δυναμικές συμπεριφορές των οπτικών οπτικών Quantum Dot (QD) spin-VCSELs, διερεύνοντας την καταλληλότητά τους για εφαρμογές νευρομορφικής φωτονικής. Τα QD spin-VCSELs προσφέρουν λειτουργία ψηλής ταχύτητας και χαμηλής κατανάλωσης ενέργειας, καθώς και τη δυνατότητα παραγωγής διεγερτικών σημάτων αιώνων σε υπονανοδευτέρα διάστημα.

Τα QD spin-VCSELs προσφέρουν λειτουργία υψηλής ταχύτητας και χαμηλής κατανάλωσης ενέργειας, επιτρέποντας υπερταχύτητα γνωστική υπολογιστική με υψηλή αποδοτικότητα.

Η έρευνα αυτή εξελίχτηκε μέσω αριθμητικών προσομοιώσεων χρησιμοποιώντας συγκεκριμένες εξισώσεις ρυθμού για QD spin-VCSELs, οι οποίες αναλύτηκαν με τη μέθοδο του Μέγιστου Ευθέτη Lyapunov (LLE) για τη χαρτογράφηση των περιοχών διέγερσης και σταθερότητας.

Η ανάλυση της Ισχύος Φασματικής Πυράνοτητας (PSD) χρησιμοποιήθηκε για την εξαγωγή χαρακτηριστικών όπως η προεξοχή ή η πλάτος των ορυφών. Επιπλέον, η ασυμμετρία και η υφιστάμενη ολοκλήρωση τη βάση των χαρακτηριστικών που στη συνέχεια χρησιμοποιούνται για τη μείωση διαστάσεων.

Η απόδοση αυτών των μεθόδων συσταδοποίησης αξιολογήθηκε χρησιμοποιώντας μετρικές όπως η μέση βαϑμολογία σιλουέτας (mean silhouette score), ο προσαρμοσμένος δείκτης Rand (adjusted Rand index), η κανονισμοποιημένη αμοιβαία πληροφορία (normalized mutual information), η αριθμητική αμοιβαία πληροφορία, ο δείκτης Fowlkes-Mallows (FMI) και την καθαρότητα της συστάδας (purity). Τα αποτελέσματα έδειξαν ότι η UMAP, σε συνδυασμό με τις K-means και Agglomerative Clustering, παρείχε απολύτηρη αναγνώριση των συστάδων.

Τα ευρήματα αυτής της μελέτης υπογραμμίζουν τις δυνατότητες ενσωμάτωσης της φωτονικής με τη νευρομορφική επεξεργασία για την ανάπτυξη προηγμένων υπολογιστικών συστημάτων. Οι γνώσεις που αποκτήθηκαν από θεωρία και προσαρμογή των ενεργειών στις προηγμένες εφαρμογές, αξιοποιούντας τα εγγενή πλεονεκτήματα των φωτονικών συσκευών για την επίτευξη εξελιγμένων υπολογιστικών δυνατοτήτων.
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>Two-Dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>Three-Dimensional</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>ARI</td>
<td>Adjusted Rand Index</td>
</tr>
<tr>
<td>ASE</td>
<td>Amplified Spontaneous Emission</td>
</tr>
<tr>
<td>CB</td>
<td>Conduction Band</td>
</tr>
<tr>
<td>CCA</td>
<td>Curvilinear Components Analysis</td>
</tr>
<tr>
<td>CBE</td>
<td>Chemical Beam Epitaxy</td>
</tr>
<tr>
<td>CW</td>
<td>Continuous Wave</td>
</tr>
<tr>
<td>DDEs</td>
<td>Delay Differential Equations</td>
</tr>
<tr>
<td>DBI</td>
<td>Davies–Bouldin Index</td>
</tr>
<tr>
<td>DBR</td>
<td>Distributed Bragg Reflector</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>Density-Based Spatial Clustering of Applications with Noise</td>
</tr>
<tr>
<td>DFB</td>
<td>Distributed Feedback Laser</td>
</tr>
<tr>
<td>DR</td>
<td>Dimensionality Reduction</td>
</tr>
<tr>
<td>ES</td>
<td>Excited State</td>
</tr>
<tr>
<td>FCA</td>
<td>Free Carrier Absorption</td>
</tr>
<tr>
<td>FCD</td>
<td>Free Carrier Dispersion</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FMI</td>
<td>Fowlkes-Mallows Index</td>
</tr>
<tr>
<td>FWHM</td>
<td>Full Width at Half Maximum</td>
</tr>
<tr>
<td>GS</td>
<td>Ground State</td>
</tr>
<tr>
<td>GELs</td>
<td>Grating-Enhanced Lasers</td>
</tr>
<tr>
<td>InP</td>
<td>Indium Phosphide</td>
</tr>
<tr>
<td>ITO</td>
<td>Indium Tin Oxide</td>
</tr>
<tr>
<td>KL</td>
<td>Kullback-Leibler</td>
</tr>
<tr>
<td>LCP</td>
<td>Left Circular Polarization</td>
</tr>
<tr>
<td>LIF</td>
<td>Leaky Integrate-and-Fire</td>
</tr>
<tr>
<td>LLE</td>
<td>Largest Lyapunov Exponent</td>
</tr>
<tr>
<td>LTP</td>
<td>Long-Term Potentiation</td>
</tr>
<tr>
<td>LTD</td>
<td>Long-Term Depression</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>MBE</td>
<td>Molecular Beam Epitaxy</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MNIST</td>
<td>Modified National Institute of Standards and Technology</td>
</tr>
<tr>
<td>MOCVD</td>
<td>Metal-Organic Chemical Vapor Deposition</td>
</tr>
<tr>
<td>MVU</td>
<td>Maximum Variance Unfolding</td>
</tr>
<tr>
<td>Nf</td>
<td>Noise Figure</td>
</tr>
<tr>
<td>NMI</td>
<td>Normalized Mutual Information</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>PCM</td>
<td>Phase Change Memory</td>
</tr>
<tr>
<td>PCMs</td>
<td>Phase Change Materials</td>
</tr>
<tr>
<td>PS</td>
<td>Polarization Switching</td>
</tr>
<tr>
<td>PSD</td>
<td>Power Spectral Density</td>
</tr>
<tr>
<td>QD</td>
<td>Quantum-Dot</td>
</tr>
<tr>
<td>QW</td>
<td>Quantum-Well</td>
</tr>
<tr>
<td>RCP</td>
<td>Right Circular Polarization</td>
</tr>
<tr>
<td>RF</td>
<td>Radio-Frequency</td>
</tr>
<tr>
<td>SA</td>
<td>Saturable Absorber</td>
</tr>
<tr>
<td>SFM</td>
<td>Spin Flip Model</td>
</tr>
<tr>
<td>SNN</td>
<td>Spiking Neural Networks</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>SOA</td>
<td>Semiconductor Optical Amplifier</td>
</tr>
<tr>
<td>SLs</td>
<td>Semiconductor Lasers</td>
</tr>
<tr>
<td>STDP</td>
<td>Spike-Timing-Dependent Plasticity</td>
</tr>
<tr>
<td>t-SNE</td>
<td>t-Distributed Stochastic Neighbor Embedding</td>
</tr>
<tr>
<td>TO</td>
<td>Thermo-Optic</td>
</tr>
<tr>
<td>UMAP</td>
<td>Uniform Manifold Approximation and Projection</td>
</tr>
<tr>
<td>VB</td>
<td>Valence Band</td>
</tr>
<tr>
<td>VCSEL</td>
<td>Vertical-Cavity Surface-Emitting Laser</td>
</tr>
<tr>
<td>VCSOAs</td>
<td>Vertical-Cavity Semiconductor Optical Amplifiers</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very Large Scale Integration</td>
</tr>
<tr>
<td>WCSS</td>
<td>Within-Cluster Sum of Squares</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Deep learning is the leading force in high-performance computing, quickly nearing the limits of available computing power. Orange dots indicate the total compute power, normalized to petaflop/s-day, used to train various neural network architectures. Blue dots illustrate the trend of Moore’s law. (A petaflop/s-day equals $10^{15}$ operations per second for one day, totaling $8.64 \times 10^{19}$ operations.) [1].

2.2 Speed and efficiency metrics for various neuromorphic hardware platforms are compared. In the upper right, the Hybrid III-V/Si (SNN PIC) and Sub-λ (subwavelength photonics) platforms are shown. Other points represent the latest electronic neuromorphic hardware. Shaded areas approximate qualitative trade-offs for each technology [1].

2.3 Variations in spike processing time scales (pulse width and refractory period) between biological neurons (left), electronic spiking neurons (center), and photonic neurons (right) [2].

2.4 A selection of recently demonstrated photonic neural network architectures [3].

    a. A multiwavelength broadcast-and-weight photonic neural network composed of MRR weight banks (WEI) and balanced photodiodes (BPD) for linear weighting and summing, and MRR modulators (MOD) for nonlinear activation [4].
    b. A coherent all-optical feed-forward network composed of MZI meshes configured to implement a matrix using singular value decomposition: red meshes implement a unitary matrix and blue meshes implement a diagonal matrix [5].
    c. A WDM-based all-optical neural network using PCM. A collector made of MRRs multiplexes optical pulses from the previous layer (bottom) and a distributor broadcasts the input signal equally to the PCM synapses of each neuron (top) [6].
    d. A multilayer perceptron implemented with superconducting optoelectronic network platform with No neurons [7].
    e. Diffractively coupled photonic nodes forming a large-scale recurrent neural network. A spatial light modulator (SLM) encodes the networks’ state and a digital micromirror device (DMD) creates a spatially modulated image of the SLM’s state. The output is obtained via superimposing the detected modulated intensities [8].
    f. Left, diffractive deep neural network based on coherent waves. Right, the network comprises multiple transmissive (or reflective) layers; each point on a given layer acts as a neuron, with a complex-valued transmission (or reflection) coefficient [9].

2.5 Neuromorphic photonics utilizes recent advances in photonic integration, scalable computing models like spiking and deep neural networks, and a robust integrated photonic ecosystem using III-V materials and silicon.

2.6 The structure of an individual neuron as observed under a fluorescence microscope [10].

2.7 Schematic of biological neuron (Reproduced image [10]).

2.8 Changes in membrane potential during an action potential. The red curve represents the action potential after anodal stimulation, while the blue curve represents the normal action potential. [11].

2.9 The release of neurotransmitters is a fundamental aspect of communication at chemical synapses. Upon depolarization of the presynaptic membrane, voltage-gated calcium (Ca$^{2+}$) channels open, allowing Ca$^{2+}$ influx into the cell. The influx of calcium ions into the synaptic vesicles causes them to fuse with the presynaptic membrane, releasing neurotransmitter molecules into the synaptic cleft. The neurotransmitter subsequently diffuses across the synaptic cleft, where it binds to ligand-gated ion channels in the postsynaptic membrane. This results in either a localized depolarization or hyperpolarization of the postsynaptic neuron. [12].

2.10 Differences between biological neurons, artificial neurons, and spiking neurons [13].

2.11 Spiking neural networks represent information through events over time rather than binary bits. Since the occurrence time of a spike is analog and its amplitude is digital, these signals employ a mixed-signal or hybrid encoding approach [14].
2.12 (a) Biological neurons form complex networks through synapses, linking the axons of presynaptic neurons to the dendrites of post-synaptic neurons. This nonlinearity is crucial for their computational capabilities. (b) The optical perceptron mimics this process by applying a nonlinear transformation to a sum of optical fields from neurons in layer $l$, represented as $E_j^l$, scaled by synaptic weights $W_j$, which produces the output field of the neuron, $E_j^l$. (c) Vertical-Cavity Surface-Emitting Lasers (VCSELs) can be effectively integrated with external optical fields. Their inherent dynamics allow them to exhibit a nonlinear response to such optical injections. (d) When connected through photonic links, VCSELs offer a potential pathway for the creation of advanced photonic neural networks [15].

3.1 Diagram illustrating a VCSEL device. This vertical-cavity surface-emitting laser is compact and includes two doped distributed Bragg reflectors (DBRs), a central cavity with the active region, at least one oxide aperture, and ring-shaped n- and p-contacts. Inset: Scanning electron microscope image showing the central cross-section of a VCSEL with two oxide apertures. Reproduced with permission from the referenced source. [15].

3.2 Types of Polarization: (a) Linear Polarization, (b) Circular Polarization, and (c) Elliptical Polarization. [16].

3.3 Schematic illustrating the key components of the SFM model, which is based on the four-level Maxwell–Bloch equations. The carrier densities $n^+$ (for spin-down) and $n^-$ (for spin-up) correspond to the right ($E^+$) and left ($E^-$) circularly polarized field components, respectively. The characteristic decay rates for spin and birefringence are denoted by $\gamma_s$ and $\gamma_p$ [17].

3.4 Bucket model for lasers: (a) In a conventional laser, water added to the bucket represents carrier transitions and the overflow indicates the lasing threshold, with small leaks showing spontaneous emission. (b) In a spin laser, the bucket is split into two halves for different spin populations (hot and cold water). The partition allows some mixing due to spin relaxation. Unpolarized pumping (pink) is an equal mix of two polarized contributions (red and blue) [18].

3.5 Interaction between master and slave QD spin-VCSEL. The image was reproduced from [19].

3.6 Diagram illustrating the energy levels (WL, ES, GS, VB) and carrier transitions in the 8-level system utilized in the SFM for QD spin-VCSELs [20].

3.7 Illustration of the Largest Lyapunov Exponent method. The figure shows the divergence or convergence of trajectories over time, represented by intensity and carrier density plots [17].

4.1 Steps in the analysis pipeline followed in this work.

4.2 Extrema Analysis map in the $k_{\text{inc}} - \Delta f$ plane for ES.

4.3 The boundary of locked and unlocked regions of the map in Figure 4.2.

4.4 Illustration of positive and negative skewness.

4.5 Comparison of Distributions with Varying Variances and Kurtoses.

5.1 Representation of Power Spectral Density of spiking time series data.

5.2 Representation of Power Spectral Density of spiking time series data.

5.3 Representation of Power Spectral Density of chaotic time series data.

5.4 PCA of Spiking and Chaotic Data (ES plus time series).

5.5 t-SNE Projection of ES-plus Time Series Data using Features [skewness_value, kurtosis_value, median_prominence, num_pks, range_width].

5.6 t-SNE Projection of ES-plus Time Series Data using Features [skewness_value, kurtosis_value, std_prominence, num_pks, range_width].

5.7 t-SNE Projection of GS-minus Time Series Data using Features [skewness_value, kurtosis_value, std_prominence, num_pks, range_width].

5.8 t-SNE Projection of GS-minus Time Series Data using Features [skewness_value, kurtosis_value, median_prominence, num_pks, range_width].

5.9 Elbow Method for Optimal $k$.

5.10 UMAP Projection of Time Series Data with Best Parameters (ES Plus).

5.11 UMAP Projection with K-means Clustering for ES Plus Time Series (min_dist=0.1, n_neighbors=15). Mean silhouette score: 0.88, Davies-Bouldin Index: 0.45.

5.12 Silhouette Plot for UMAP Reduced Data with Median Prominence (ES Plus).

5.13 Elbow Method for Optimal $k$ (ES Plus, Median Prominence).

5.14 UMAP Projection with Best Parameters (ES Plus, Median Prominence).
5.15 UMAP Projection with K-means Clustering (ES Plus, Median Prominence) ....................... 58
5.16 UMAP Projection results for ES minus time traces of the dataset, (min_dist=0.1, n_neighbors=20) 59
5.17 Chaotic time series ($k_{inj} = 16.3205, \Delta f = 11.5385$). ........................................... 60
5.18 Bursting spiking time series data ($k_{inj} = 26.5306, \Delta f = 8.40154$). .......................... 60
5.19 Low intensity spiking behaviour observed mainly in ES Plus time trace ($k_{inj} = 18.3673, \Delta f = 5.38402$). .......................................................... 61
5.20 Dense spiking behaviour observed mainly on the ES plus time trace ($k_{inj} = 30.6122, \Delta f = 13.0769$). .......................................................... 61
5.21 Chaotic spiking behaviour present mainly in ES plus time trace ($k_{inj} = 14.2851, \Delta f = 6.92308$). 62
5.22 K-means Clustering ES plus time series (Mean silhouette score: 0.87 for 3 clusters). ......... 62
5.23 Clustering methods comparison: K-means, Hierarchical, and DBSCAN clustering on UMAP-reduced data (ES Minus, 2 Clusters) ...................................................... 62
5.24 Clustering methods comparison: K-means, Hierarchical, and DBSCAN clustering on UMAP-reduced data (ES Minus, 3 Clusters) ...................................................... 63
## List of Tables

2.1 Comparison of digital, analog, and hybrid systems in terms of robustness, expressiveness, power efficiency, and bandwidth limiter [21]. ...................................................... 15
2.2 Summary of various optoelectronic excitable devices. ...................................................... 17
2.3 Techniques for Artificial Optical Neurons Using VCSELs. ........................................... 19

4.1 Values of parameters for the QD spin-VCSEL [20] ...................................................... 34
4.2 Comparison of Dimensionality Reduction Techniques: PCA, t-SNE, and UMAP ............ 48

5.1 Metrics for widths and prominences of the peaks for spiking time series. ...................... 49
5.2 Metrics for widths and prominences of the peaks. ...................................................... 50
5.3 Metrics for widths and prominences of the peaks. ...................................................... 51
5.4 Clustering Performance Metrics and Parameters (UMAP 2 K-means) for skewness_value, kurtosis_value, std_prominence, num_pks, and range_width. ................................. 57
5.5 Clustering Performance Metrics and Parameters (UMAP 2 K-means, Median Prominence) 58
5.6 Clustering Performance Metrics Summary for ES Minus (2 Clusters) ......................... 61
5.7 Confusion Matrix for K-means Clustering (2 Clusters) ............................................. 63
5.8 Clustering Performance Metrics Summary for ES Minus (3 Clusters) ......................... 63
5.9 Confusion Matrix for K-means Clustering (3 Clusters) ............................................. 63
# Contents

Acknowledgements ii
Abstract iii
Extended Abstract iv
Acronyms v
List of Figures ix
List of Tables x

1 INTRODUCTION

2 FOUNDATIONS AND ADVANCES IN NEUROMORPHIC PHOTONICS 3

2.1 NEUROMORPHIC PROCESSING IN ELECTRONICS AND PHOTONICS ................................. 3
   2.1.1 Neuromorphic Photonic Technologies ........................................... 6
   2.1.2 Neuromorphic Photonics Approaches ............................................ 7
   2.1.3 Technological Challenges of Neuromorphic Photonics ......................... 8

2.2 BIOLOGICAL FOUNDATIONS ................................. 9
   2.2.1 Neuronal Cell Morphology Unveiled ........................................ 9
   2.2.2 The Dynamics of Membrane Potential .................................... 10
   2.2.3 The Pathways of Neural Communication: Synapses ....................... 10

2.3 ARTIFICIAL NEURONS ........................................ 11
   2.3.1 Spiking Neuron Model: Leaky Integrate-and-Fire (LIF) ................. 13
   2.3.2 Spiking Signals ........................................ 14

2.4 PHOTONIC NEURON MODEL ................................. 15

2.5 OPTOELECTRONIC DEVICES WITH EXCITABLE DYNAMICS ................................. 16
   2.5.1 Computing with spiking VCSEL Neurons .................................. 18
   2.5.2 Spike-based Processing Systems with VCSELs ......................... 20

3 TECHNICAL INSIGHTS OF VCSEL TECHNOLOGY 21

3.1 STRUCTURE OF VCSEL DEVICES ........................................ 21

3.2 FUNDAMENTALS OF LASER THEORY ........................................ 21
   3.2.1 Interaction Mechanisms ........................................ 22
   3.2.2 Polarization of light ........................................ 23

3.3 OVERVIEW OF THE SPIN FLIP MODEL (SFM) ........................................ 24

3.4 SPIN-VCSELs ........................................ 25
   3.4.1 Spin Dynamics in Quantum-Dot Spin VCSELs ........................... 25

3.5 STABILITY MAPS CALCULATION USING LARGEST LYAPUNOV EXPONENT METHOD (LLE) ................................. 30
   3.5.1 Practical Computation of LLE ........................................ 31

4 METHODOLOGY 32

4.1 DATASET GENERATION ........................................ 32
   4.1.1 Excitability Analysis of optically injected QD spin-VCSELs Using Largest Lyapunov Exponent (LLE) Method ................................. 32
   4.1.2 Extrema Analysis Maps and Dynamic Control ........................................ 33
   4.1.3 Numerical Simulations Setup of simulation parameters and solvers Tools and techniques (MATLAB) ........................................ 34
1. INTRODUCTION

The brain’s remarkable energy efficiency, capability in performing complex computational tasks, and adaptability set it apart from traditional von Neumann computer processors. These superior attributes are often linked to the intricate connectivity patterns within biological nervous systems; however, the cellular dynamics of individual neurons are equally crucial. Neurons process information through discrete events known as spikes, demonstrating a blend of analog and digital properties. This spike processing exploits the efficiency of analog signals while mitigating the noise accumulation typical in analog computation, representing an initial step towards achieving the impressive capabilities of bio-inspired processing.

Neurons also possess additional physical and signal processing features, such as hybrid analog-digital signals, representational interleaving, co-location of memory and processing, unsupervised statistical learning, and distributed information representations. These characteristics contribute to signal processing abilities that are both advantageous and often impossible with conventional methods. Engineers have long sought inspiration from neuroscience to distinguish key performance-enhancing properties from mere biological idiosyncrasies. By focusing on essential biological features and integrating non-biological elements, bio-inspired engineered systems could unlock new and practical computational possibilities.

A vast amount of microelectronic platforms have attempted to replicate neuron-like architectures while incorporating advanced technologies. However, most aim to match rather than surpass biological time scales. This raises the question: what kinds of signal processing might be achievable with a bio-inspired visual front-end operating ten million times faster than its biological counterpart? Microelectronic neural networks striving for high speed and extensive interconnectivity face a tradeoff between bandwidth and connection density, ultimately limiting their speed.

Traditional computer architecture, centered around a CPU and memory, is optimized for sequential, digital, and procedure-based programs. This centralized approach is not well-suited for the distributed, massively parallel, and adaptive computational models foundational to neural networks in artificial intelligence (AI). AI aims to replicate human-level accuracy in tasks challenging for conventional computers but easily performed by humans. Significant breakthroughs have been achieved with machine learning (ML) algorithms, particularly those based on neural networks. These algorithms process information in a distributed fashion and adapt to previous inputs rather than following explicit programming instructions, leading to applications in diverse fields such as language translation and personalized medicine [22, 23].

Neuromorphic engineering aims to integrate elements of machine learning (ML) and artificial intelligence (AI) algorithms into hardware that reflects their highly parallel nature [24]. This integration promises significant improvements in speed and energy efficiency for information processing. Neuromorphic hardware is not limited to ML but also finds applications in robotic control, mathematical programming, and testing neuroscientific hypotheses [25].

These massively parallel hardware systems necessitate extensive interconnections among components, such as neurons. Traditional metal wiring for each connection is impractical; thus, current neuromorphic electronics often use a shared digital communication bus with time-division multiplexing, balancing bandwidth and connectivity. However, the use of optical interconnects could resolve this trade-off, potentially accelerating both machine learning and neuromorphic computation significantly [3].

Examining the intrinsic physical characteristics of electronic and photonic platforms reveals that integrating both into a hybrid analog-digital system can harness the strengths of each. This integration allows for the design and implementation of more sophisticated and high-bandwidth processing algorithms than what is currently feasible with either technology independently. Photonic platforms, with their high speeds, expansive bandwidth, and minimal cross-talk, are particularly well-suited for ultrafast spike-based information processing with high interconnection densities. Additionally, the high wall-plug efficiency of photonic devices could potentially surpass electronic systems in terms of energy efficiency, allowing access to a low-power computational domain at the picosecond scale that is otherwise inaccessible [26].

The fusion of a processing paradigm with the physical phenomena that support it represents a significant advancement in efficiency and performance compared to methods that disregard physics entirely. Both biological...
thalamocortical structures and emerging mathematical models of network information integration demonstrate a clear distinction between computation and communication. While computation typically does not occur within the axons of neurons, these communication pathways and their versatile configurations are essential for the network’s overall informational complexity. Photonic spiking network processors, owing to their advantageous communication properties such as low dispersion, high bandwidth, and minimal cross-talk, can access a computationally rich and high-bandwidth domain unattainable by other technologies. This new domain, termed ultrafast cognitive computing, offers a novel processing paradigm with potential applications in adaptive control, learning, perception, motion control, sensory processing (including vision, auditory, and olfactory systems), autonomous robotics, and cognitive processing of the radio frequency spectrum [27].

The research conducted in this study investigates the complex dynamics of optically injected Quantum Dot (QD) spin-Vertical-Cavity Surface-Emitting Lasers (spin-VCSELs) and their potential for neuromorphic photonics applications. QD spin-VCSELs possess unique features that make them exceptional for neuromorphic photonics. They can generate and control excitatory spiking signals at sub-nanosecond rates, and their high efficiency and low energy consumption make them ideal for next-generation photonic neural network hardware. These lasers benefit from well-established fabrication processes, high wall-plug efficiencies, and the potential for integration into scalable 3D photonic circuits. Future developments, such as spin-VCSELs and high-beta VCSELs, are expected to enhance efficiencies and reduce energy costs, reinforcing their significance in neuromorphic computing.

Furthermore, the time series for ground state (GS) and excited state (ES) emissions were derived through the numerical solution of specific rate equations using the Largest Lyapunov Exponent (LLE) method. These data were then mapped onto a $k_{inj} - \Delta f$ plane to identify regions of excitability and stability, indicative of spiking and chaotic behaviors. The analysis included examining the Power Spectral Density (PSD) of the laser’s output to extract key features such as the number of peaks, peak prominence, and peak width, which revealed structured behaviors in spiking regimes and complex spectral landscapes in chaotic regimes. Additional features like skewness and kurtosis of the time series were also analyzed. For dimensionality reduction, techniques such as UMAP, t-SNE, and PCA were utilized, followed by clustering methods including K-means, DBSCAN, and Agglomerative clustering. These approaches facilitated the accurate classification of dynamic states and further differentiation within spiking behaviors. Performance metrics used in this analysis included the mean silhouette score, adjusted Rand index (ARI), normalized mutual information (NMI), accuracy, Fowlkes-Mallows index (FMI), and purity, offering a comprehensive assessment of clustering quality.

The subsequent chapters thoroughly explore these topics, covering the foundational principles, technical insights, and methodologies employed in this study, followed by a comprehensive analysis of the experimental results and a discussion on future work.

The chapters of this thesis are organized as follows:

- **Chapter 2**: Provides an in-depth review of the foundations and advances in neuromorphic photonics, including the biological foundations, artificial neurons, and photonic neuron models.
- **Chapter 3**: Explores the technical insights of VCSEL technology, discussing the structure, dynamics, and stability maps of spin-VCSELs.
- **Chapter 4**: Outlines the methodology employed in this study, detailing the dataset generation, feature extraction, and clustering techniques used.
- **Chapter 5**: Presents the experiments and results, including the analysis of the Power Spectral Density of the data and the dimensionality reduction and clustering evaluation.
- **Chapter 6**: Offers conclusions drawn from this research, highlighting the implications and future directions.
- **Chapter 7**: Discusses potential future work, exploring areas for further research and development in this field.
2. FOUNDATIONS AND ADVANCES IN NEUROMORPHIC PHOTONICS

2.1 NEUROMORPHIC PROCESSING IN ELECTRONICS AND PHOTONICS

Traditional computers rely on a centralized processing architecture, with a central processor and memory, which is optimized for running sequential, digital, procedure-based programs. However, this architecture is inefficient for computational models that are distributed, massively parallel, and adaptive, such as neural networks used in artificial intelligence (AI). AI aims to achieve human-level accuracy on tasks that are challenging for conventional computers but intuitive for humans. Significant progress in AI has been driven by machine learning (ML) algorithms based on neural networks, which process information in a distributed manner and adapt to past inputs rather than following explicit programming. ML impacts various aspects of life, including language translation and cancer diagnosis.

Neuromorphic engineering seeks to transition elements of ML and AI algorithms to hardware that mirrors their massively distributed nature, leading to potentially faster and more energy-efficient information processing. Neuromorphic hardware is applied not only to ML but also to robot control, mathematical programming, and neuroscientific hypothesis testing. This hardware relies heavily on massively parallel interconnections between neurons, making dedicated metal wiring for every connection impractical. Current neuromorphic electronics use shared digital communication buses with time-division multiplexing, trading bandwidth for interconnectivity. Optical interconnects, however, could eliminate this trade-off, accelerating ML and neuromorphic computing [1].

Figure 2.1: Deep learning is the leading force in high-performance computing, quickly nearing the limits of available computing power. Orange dots indicate the total compute power, normalized to petaflop/s-day, used to train various neural network architectures. Blue dots illustrate the trend of Moore’s law. (A petaflop/s-day equals $10^{15}$ operations per second for one day, totaling $8.64 \times 10^{19}$ operations.) [1].
Numerous technologies have successfully implemented large-scale spiking neural networks (SNNs) in electronics. Notable examples include Stanford University’s Neurogrid from the Brains in Silicon program [28], IBM’s TrueNorth [29] developed under DARPA’s SyNAPSE program, the HICANN chip from the University of Heidelberg’s FACETS/BrainScaleS project [30], and the neuromorphic chip from the University of Manchester’s SpiNNaker project [31], both part of the European Commission’s Human Brain Project.

Unlike von Neumann processors, which depend on a single point-to-point link between memory and the CPU, neuromorphic processors necessitate numerous interconnections, often involving hundreds of many-to-one fans-in per processor [32]. This demand for extensive multicasting creates a distributed communication load, leading to performance challenges due to capacitive loads and radiative physics in electronic links, alongside the typical bandwidth-distance-energy constraints of point-to-point connections. To achieve realistic scalability, systems often adopt a combination of crossbar time-division multiplexing (TDM) and packet switching [33].

In addition, research has shown that photonic technologies offer a promising alternative for these high-bandwidth applications. By taking advantage of the high-speed and low-latency capabilities of photons, it is possible to develop neuromorphic systems that can operate at much higher speeds and with greater energy efficiency than traditional electronic approaches. Photonic interconnections, in particular, can handle the massive data rates required for such applications, potentially overcoming the limitations imposed by electronic interconnects. A notable innovation in this field is a hybrid photonic manufacturing platform that integrates both active elements, like lasers and detectors, and passive elements, such as waveguides, resonators, and modulators, on a single chip. Utilizing this platform, neuromorphic photonic systems could potentially outperform neuromorphic electronic systems by 6–8 orders of magnitude in speed, particularly when considering the reduced bandwidth requirements of virtualized interconnects [26,34].

Photonics, established in telecommunications and data centers, has not yet been widely adopted in information processing and computing. Their development of complex photonic systems has been limited by the lack of a strong photonic integration industry. However, recent advancements have rapidly transformed the manufacturability of photonic chips, promising to achieve the same economies of scale that microelectronics has long enjoyed. The properties that make optoelectronic components excellent for communication do not align with the requirements of digital gates. Nevertheless, non-digital computing models, such as neural networks, are more suitable for photonic implementation. The goal of neuromorphic photonic processors is not to replace conventional computers but to enable applications currently beyond the reach of traditional computing, especially those needing low latency, high bandwidth, and low energy consumption.

Ultrafast neural networks have potential applications in fundamental physics breakthroughs like qubit readout classification, high-energy particle collision classification, fusion reactor plasma control and also nonlinear programming (e.g., solving optimization problems in robotics, autonomous vehicles, predictive control, and partial differential equations), machine learning acceleration (e.g., vector-matrix multiplications, deep learning inference, ultrafast or online learning), and intelligent signal processing (e.g., wideband RF signal processing, fiber-optic communication) [4].

Photonic circuits are well suited to implementing high-performance neural networks due to their advantages in interconnectivity and linear operations. Connections between artificial neurons are represented by a scalar synaptic weight, allowing interconnections to be modeled as matrix-vector operations, where each neuron’s input is the dot product of outputs from connected neurons, attenuated by a weight vector. Optical signals can be multiplied through tunable waveguide elements and added via wavelength-division multiplexing (WDM), carrier accumulation in semiconductors, electronic currents, or changes in material crystal structures induced by photons [35].

Neural networks require long-range connections for distributed information processing. Compared to metal wire connections, photonic waveguides experience lower attenuation and generate less heat over distance, with minimal frequency-dependent signal distortions. While electronic point-to-point links use transmission line and active buffering techniques, neural networks involve massively parallel signal fan-out and fan-in, making these techniques impractical for each physical connection. To avoid the trade-offs of electronic wiring, neuromorphic electronic architectures use digital time-multiplexing, allowing for larger neural networks at the cost of bandwidth. However, applications that prioritize bandwidth and latency can only be effectively addressed by direct, non-digital photonic broadcast interconnects [3].

Adopting a new architecture based on a photonic interconnection fabric with nonlinear optoelectronic devices offers distinct advantages in energy efficiency, bandwidth, and latency, overcoming many fundamental limitations of digital and analog electronics. This approach could be one of the rare feasible methods to achieve
ultrafast, complex on-chip processing without excessive power consumption. Historically, complex photonic systems have not been fully explored due to the lack of a robust photonic integration industry [26]. However, the rapid evolution of manufacturable photonic chips now promises to deliver economies of scale similar to those enjoyed by microelectronics. Notably, a new hybrid photonic manufacturing platform is emerging that integrates both active components (such as lasers and detectors) and passive components (such as waveguides, resonators, and modulators) on a single chip. Utilizing this platform, a neuromorphic photonic approach could potentially perform 6–8 orders of magnitude faster than neuromorphic electronics (Figure 2.2), particularly when considering the bandwidth reduction achieved through virtualized interconnects.

Figure 2.2: Speed and efficiency metrics for various neuromorphic hardware platforms are compared. In the upper right, the Hybrid III-V/Si (SNN PIC) and Sub-λ (subwavelength photonics) platforms are shown. Other points represent the latest electronic neuromorphic hardware. Shaded areas approximate qualitative trade-offs for each technology [1].

Optics has been acknowledged for a long time as an effective medium for matrix multiplication and interconnects. In the past, optical neural networks were initially developed by researchers like Psaltis [36]. Nowadays, the increasing demand for advanced computing and the progress in relevant technologies have brought Photonic Neural Networks (PNNs) back into focus. One of the key advancements is silicon photonics, which offers a significant improvement over earlier attempts. The first demonstration of photonic spike processing was achieved only in 2009 by Rosenbluth [37]. Spike processors, like gates in digital computers, act as baseline units of larger interconnected networks able to perform more complex computations. Rosenbluth’s photonic integrate-and-fire systems, based on optical spike processing devices, operated on picosecond width pulses and had an integration time constant in the order of 100 ps, about eight orders of magnitude faster than biological neurons. A bloom in research related to various aspects of photonic spike processing followed Rosenbluth’s first demonstration. Many of these proposals for spiking “photonic neurons” or “laser neurons” or “optical neurons” have been extensively reviewed. Silicon photonic platforms can integrate high-quality passive components with high-speed active optoelectronics, achieving a competitive integration density. A proposal for a scalable silicon photonic neural network was first introduced in 2014 and successfully demonstrated in 2017, alongside other silicon photonic neuromorphic systems [5]. On-chip silicon electronics enable calibration and control to mitigate component sensitivity issues, while advancements in on-chip optoelectronics facilitate cascadability and nonlinearity [38]. The potential of neuromorphic photonic systems to significantly enhance computing capabilities is becoming more attainable, with the possibility of achieving PetaMAC (multiply-accumulate operations) per second per mm² processing speeds and attojoule per MAC energy efficiencies [3]. Although photonics offers advantages in connectivity and linear operations over traditional electronics, challenges persist in the storage and retrieval of neuron weights within on-chip memory. Despite extensive research into optical memories and ‘in-memory’ computing, these solutions often fail to support high-frequency read/write operations [35]. To develop scalable neuromorphic photonic processors, there will be a need for seamless integration of electronics with hybrid electronic and optical memory systems, utilizing the strengths of both volatile and non-volatile memories in either digital or analog formats, depending on the specific application and computation requirements [3,39].
2.1.1 Neuromorphic Photonic Technologies

Marching ahead to enabling technologies that can surpass conventional electronics, neuromorphic photonics stand out as an alternative technology platform with immense potential to revolutionize AI hardware accelerators. This new scientific and technological field aims at exploiting photons to carry out matrix multiplications with almost zero power consumption, leveraging at the same time their virtually unlimited bandwidth, inherent parallelism and compatibility with semiconductor industry. In this context, a series of start-up companies such as Lightmatter, Lightelligence, Luminous and LightOn came to the fore in pursuing photonic neural network (PNNs) architectures to carry-out multiply-and-accumulate (MAC) computations with ultra-low power consumption, claiming orders of magnitude improvements in compute efficiency against digital and analog electronics counterparts.

Capitalizing on technology assets of neuromorphic photonics, a high proliferation of PNNs has been investigated by the photonics research community. This has focused on several platforms including bulk and diffractive optics as well as photonic-integrated-circuits (PICs) in an attempt to establish a technology roadmap for photonic accelerators that can lead to scalable and energy efficient deployments. While flavors of photonic accelerators and PNNs based on bulk and diffractive optics demonstrated credentials for increased scalability supporting millions of neurons with excellent accuracy benchmarks, challenges associated with inherently low reconfiguration speeds, power hungry implementations and complex co-integration with electronics created an opportunity for PIC based accelerators to dominate.

PIC-based accelerators relying on silicon photonics technology and hybrid integration approaches and exploiting emerging material platforms can leverage high-integration densities enabled by complementary-metal-oxide-semiconductor (CMOS) manufacturing and unleash Peta scale compute performance within a low energy envelope, promising energy efficiencies of just a few femto-Joules per operation (OP) [40]. Although such an alluring prospect has been highlighted many times in several studies and technology roadmaps published in the literature, PIC based accelerators still encounter a series of hurdles to overcome, calling for technology advances and renewed architectures in order to scale up and address requirements imposed by on-line training and fast reconfiguration of weights with low power.

PNNs comprise multiple neurons, that can handle input data signals based on the multilayer perceptron model so as to build several types of neural networks and respective layers such as feed-forward, recurrent and convolutional, to name few. In turn, a photonic neuron is composed of two major parts, namely its linear and its non-linear part. The linear part is responsible for carrying out linear operations such as multiplication and summation and the non-linear part is responsible for carrying out non-linear functions to endow photonic hardware implementations with intelligent data processing capabilities. Summation is accomplished by optical combiners and multiplexers, while multiplication is usually provided by variable optical attenuator (VOAs) modules through adjusting light absorption/amplification properties. Non-linear activation functions such as Rectified Linear Unit (ReLU), Parametric ReLU (PReLU), sigmoid and tanh can in principle be realized entirely in the optical domain via non-linear phenomena supported for instance by semiconductor-optical-amplifiers (SOAs) or through opto-electronic (O/E) conversions [41]. Other types of non-linearities are possible that do not rely on optoelectronics, such as saturable absorbers or nonlinear media such as lithium niobate.
2.1.2 Neuromorphic Photonics Approaches

Neuromorphic photonic approaches can be broadly divided into two main categories: coherent (single wavelength) and incoherent (multiwavelength) methods, as described in sources [6, 8] (Figure 2.4).

Coherent methods encompass systems like those based on reservoir computing [42] and Mach-Zehnder interferometers [5]. In reservoir computing, the weights of the hidden layers are fixed and immutable. Another approach utilizes silicon photonics to develop fully programmable neural networks [4], using a broadcast-and-weight protocol [16]. In this method, photonic neurons emit optical signals at different wavelengths, which are then multiplexed into a single waveguide, broadcast to all neurons, weighted, and detected by a photodetector. The connection weights between neuron pairs are individually adjusted by microring resonators (MRRs), and the use of wavelength division multiplexed (WDM) carriers prevents mutual interference when a single photodetector receives the signals. As a result, the neural computation remains entirely analog, avoiding the need for logic operations or sampling that would require serialization. This leads to significant improvements in energy efficiency, latency, crosstalk, and bandwidth compared to electronic neuromorphic circuits [1].

This approach has distinct advantages over previously mentioned methods, including the demonstration of capabilities such as fan-in, inhibition, time-resolved processing, and autaptic cascadability [4], while also introducing new challenges related to reconfigurability, integration, and scalability. Manipulating information carried by photons is more challenging than handling electronic signals, particularly for nonlinear operations and memory storage [3]. The photonic neurons discussed here address this issue by utilizing optoelectronic components (O/E/O), which can be integrated with standard electronics to provide reconfigurability. However, scaling up neuromorphic photonic circuits remains a challenge due to the lack of digital information, memory units, and a serial processor. This results in each element in a neural network requiring a physical unit, increasing size, area, and power consumption. Furthermore, it is important to consider the integration costs, as the benefits of using analog photonics (high parallelism and high bandwidth) must outweigh the costs of interfacing with
digital electronics, which requires both O/E and analog/digital conversion.

2.1.3 Technological Challenges of Neuromorphic Photonics

The resurgence of neuromorphic photonics has been fueled by advancements in three main areas: integrated photonics, machine learning algorithms, and analog photonic signal processing (Figure 2.5). Silicon photonics has played a crucial role in these technological strides.

Photonic processors integrate various components, including light sources, and passive and active devices. However, there currently isn’t a commercial fabrication platform that can simultaneously offer light generation, wavelength multiplexing, photodetection, and transistors on a single die. The leading devices in each category utilize different photonic materials such as SiN, Ge, InP, GaAs, and 2D materials, all of which require different fabrication processes like silicon-on-insulator, CMOS, and FinFETs. Silicon photonics stands out as an ideal platform to unify these components, thanks to its compatibility with foundry processes, compact device integration, and cost efficiency. This makes it possible to develop scalable photonic systems on a chip.

Figure 2.5: Neuromorphic photonics utilizes recent advances in photonic integration, scalable computing models like spiking and deep neural networks, and a robust integrated photonic ecosystem using III-V materials and silicon.

- **Lasers and Amplifiers**: Achieving on-chip optical gain and power necessitates the integration of active InP lasers and semiconductor optical amplifiers. This integration is currently achieved through methods such as III-V to silicon wafer bonding (heterogeneous integration) or precise assembly co-packaging (hybrid approach). Quantum dot lasers offer another promising route as they can be grown directly on silicon; however, their fabrication processes have yet to reach the reliability required for commercial standards [43].

- **Materials**: The development of energy-efficient and fast-switching optical and electro-optical materials is critical for applications like non-volatile photonic storage, weighting, high-speed optical switching, and routing, all with minimal power consumption. Existing platforms can implement neural nonlinearities using electro-optic transfer functions [21], but novel materials such as phase change materials (PCMs), graphene, and ITO-based modulators could significantly enhance performance. Plasmonic PCMs, capable of dual operation modes, can bridge optical and electrical signals [44]. There is an urgent need
for a general material design method to develop suitable photonic materials for various photonic components [45].

- **Electrical Control**: Integrating CMOS controller chips with silicon photonics is essential for electrical tuning control and stabilization. This can be achieved through various methods, including wire-bonding, flip-chip bonding, 2.5D integration (using interposers), 3D stacking (through-silicon vias), and monolithic integration. Each method presents its own set of performance and design trade-offs.

- **System Packaging**: For a photonic processor to function effectively, it must interface with a computer system, necessitating a design that is self-contained and resilient to temperature fluctuations, with robust electrical input/output connections [3]. Presently, manufacturers face challenges in integrating electrical/thermal components and chip-to-fiber interconnects within a single package.

- **Algorithms**: Significant advancements are required to map abstract neural algorithms onto photonic processors to make neuromorphic photonic platforms commercially viable. Currently, the literature focuses on individual devices and small control circuits. The ultimate aim is to enable neural network programming tools, such as TensorFlow, to directly configure a neuromorphic photonic processor [1].

## 2.2 BIOLOGICAL FOUNDATIONS

### 2.2.1 Neuronal Cell Morphology Unveiled

One of the fundamental structural units of the brain is the neuron. Typically, neurons are large cells and contain abundant cytoplasm. Their organization gives rise to the gray matter of the cerebral hemispheres, the brainstem, the tissue of sympathetic ganglia, and the spinal cord. The neuron consists primarily of the perikaryon or cell body, where electrical impulses are generated. It is characterized by a spherical, ovoid, or polygonal shape as demonstrated by Figure 2.6 and contains mitochondria, neurofibrils, microtubules, pigments, a large nucleus, rough granular endoplasmic reticulum, Nissl bodies, and the Golgi apparatus.

![Figure 2.6: The structure of an individual neuron as observed under a fluorescence microscope [10].](image)

Additionally, each neuron possesses an axon, a single cylindrical and long process with a constant diameter that starts from the axonal hillock and ends at the terminal arborization or synaptic knob. The axon is responsible for generating or rapidly transmitting nerve impulses to other cells, as it is surrounded by myelin, which facilitates the faster propagation of electrical impulses. Finally, neurons include numerous short processes called dendrites or dendritic spines as depicted in Figure 2.7 [46].
2.2.2 The Dynamics of Membrane Potential

In a neuron at rest, meaning it is not receiving any stimuli, there is a high concentration of sodium ions (Na\(^+\)) on the external surface of the cell membrane and a high concentration of potassium ions (K\(^+\)) and anions (PO\(_4\)\(^{3-}\), SO\(_4\)\(^{2-}\), etc.) inside. To maintain this uneven distribution of ions, the membrane has a Na\(^+\)/K\(^+\) pump, which for every three sodium ions it expels from the cell, brings in two potassium ions through active transport. There is no similar mechanism for the aforementioned anions, as they diffuse only slightly. This creates a potential difference, with a high concentration of positive ions outside and a high concentration of anions inside the neuron. This potential difference is approximately -70mV, with most of the open channels being potassium channels.

When a neuron receives a stimulus, the membrane potential changes, leading to the generation of an action potential. Voltage-gated sodium channels open, allowing Na\(^+\) to flow into the cell, causing depolarization as the membrane potential moves towards +30mV. Subsequently, these sodium channels close and voltage-gated potassium channels open, allowing K\(^+\) to exit the cell, initiating repolarization. This outflow temporarily causes hyperpolarization, where the membrane potential becomes more negative than the resting potential. Finally, the Na\(^+\)/K\(^+\) pump and other ion channels restore the membrane potential to its resting state of approximately -70mV. All these changes are demonstrated in Figure 2.8.

2.2.3 The Pathways of Neural Communication: Synapses

The connection points between neurons where functional communication occurs are called synapses. They play a crucial role as they are considered the primary means of storing information in the brain. In most synapses found in the cerebral cortex, the presynaptic terminal, which is part of the sending cell’s axon, and the postsynaptic terminal, which is part of a dendrite or the cell body of the receiving cell, are separated by a narrow gap known as the synaptic cleft.

Signal transmission between these two terminals occurs through the release of chemical substances from the presynaptic terminal when it is activated by an action potential as demonstrated in Figure 2.9. These chemical transmitters are enclosed in vesicles, traverse the synaptic cleft, and bind to open ion channels or metabotropic receptors on the postsynaptic terminal. Consequently, they receive or release positively or negatively charged ions from the cell, thereby adding or removing charge [46].

It is important to note that a neuron acting as a transmitter can be exclusively either excitatory or inhibitory, specifically releasing the same type of transmitter at all its terminals. The amount of charge transmitted to the postsynaptic cell per input spike essentially reflects the strength of the connection between two neurons and can permanently change, thus altering the network’s behavior. Long-term changes in synaptic strength
Changes in membrane potential during an action potential. The red curve represents the action potential after anodal stimulation, while the blue curve represents the normal action potential. [11].

are referred to in the literature as long-term potentiation/depression (LTP/LTD). LTP strengthens synaptic connections following high-frequency stimulation, enhancing synaptic transmission efficiency. In contrast, LTD weakens synaptic connections following low-frequency stimulation, reducing synaptic efficiency. Changes in synaptic strength caused by temporary spike patterns from presynaptic and postsynaptic neurons are described as spike-timing-dependent plasticity (STDP) [47,48].

Recent research has underscored the importance of both presynaptic and postsynaptic modifications in synaptic plasticity. While much attention has traditionally focused on postsynaptic changes, it is now recognized that presynaptic neurons also significantly contribute by modulating neurotransmitter release in response to action potentials. This dual regulation is vital for fine-tuning synaptic connections, which supports complex behaviors and cognitive functions [49].

Understanding the detailed structure and plasticity of synapses is crucial for unraveling the mechanisms of neural communication and developing therapeutic strategies for neurological disorders. Continuous research in this domain is essential for advancing our knowledge and treatment of conditions characterized by synaptic dysfunction.

2.3 ARTIFICIAL NEURONS

Neuroscientists are investigating artificial neural networks to emulate the brain’s "natural processing" capabilities. These networks, consisting of simple nonlinear nodes, can learn and adapt to execute specific tasks, a process known as "learning." Nowadays, neural networks underpin leading machine intelligence systems in areas such as speech recognition, natural language processing, and computer vision.

A neural network comprises three fundamental elements: a series of nonlinear nodes (neurons), an adjustable interconnection framework (network), and a method for representing information (coding scheme). A basic representation of a neuron includes three parts: synapses (connections), an adder (or linear combiner) that performs weighted summation, and a nonlinear activation function, which is typically monotonic and bounded.

The network structure is a weighted directed graph where the connections, called synapses, link neurons. Each neuron receives inputs as a weighted sum of the outputs from connected neurons. This input is processed, integrated, and transformed into a nonlinear response through an activation function.

In the evolution of computational neuroscience, three generations of neural networks have been studied. The first generation is based on the McCulloch-Pitts model [50], featuring a linear combiner followed by a binary activation function. These networks are capable of simulating any Boolean circuit, making them universal for digital computations.

The second generation introduced analog outputs with continuous activation functions, replacing binary thresholds. These networks can approximate any continuous function over a compact domain with high accuracy.
Figure 2.9: The release of neurotransmitters is a fundamental aspect of communication at chemical synapses. Upon depolarization of the presynaptic membrane, voltage-gated calcium (Ca\(^{2+}\)) channels open, allowing Ca\(^{2+}\) influx into the cell. The influx of calcium ions into the synaptic vesicles causes them to fuse with the presynaptic membrane, releasing neurotransmitter molecules into the synaptic cleft. The neurotransmitter subsequently diffuses across the synaptic cleft, where it binds to ligand-gated ion channels on the postsynaptic membrane. This results in either a localized depolarization or hyperpolarization of the postsynaptic neuron. [12].

incorporating the concept of time, these networks can develop recurrent connections, creating attractor states and enabling associative memory [51].

Traditional neural network models use "rate coding" to represent spike rates with an analog variable. However, fast analog computations in the visual cortex of macaque monkeys suggest that rate coding alone may not suffice. Studies from the 1990s showed that these monkeys could process and classify visual patterns within 30 ms, despite neuron firing rates being below 100 Hz.

This led to the hypothesis that neurons encode information through the precise timing of spikes, giving rise to the third generation of neural networks based on "spiking neurons." This generation introduces "temporal coding," where the timing of spikes conveys analog information, optimizing energy use. Research by Maass [52] demonstrated that this generation can generalize previous models, simulate real-valued neural networks, and offer greater noise resilience.

Earlier models did not fully utilize "time" as a computational resource. The third generation, however, utilizes spike timing to enhance neural network efficiency and robustness, representing a significant advancement in
neural computation and opening new possibilities for developing sophisticated, biologically-inspired neural network models.

2.3.1 Spiking Neuron Model: Leaky Integrate-and-Fire (LIF)

The leaky-integrate-and-fire (LIF) neuron model, which captures the spiking dynamics observed in animal nervous systems, is widely recognized as the leading model for studying complex computations in theoretical neuroscience [53]. Recently, LIF neurons have garnered significant interest from engineers and computer scientists for several reasons. Firstly, LIF neurons provide a powerful and efficient computational primitive from the perspectives of computational complexity and theoretical computability, capable of emulating both Turing machines and traditional sigmoidal neural networks [54]. Secondly, these neurons are robust and efficient processing elements, enabling the implementation of complex algorithms with minimal hardware resources [24]. Lastly, known pulse processing algorithms derived from neuroethology facilitate complex signal processing tasks. Pulse processing is already employed in analog VLSI to implement robust sensory processing algorithms [55]. These characteristics collectively underscore the potential of LIF neurons in advancing computational methodologies and hardware efficiency.

The basic biological configuration of a LIF neuron is shown in Figure 2.10 (SNN). It includes a dendritic tree for collecting and summing inputs from other neurons, a soma that functions as a low-pass filter and integrates signals over time, and an axon that transmits an action potential, or spike, when the integrated signal surpasses a threshold. Neurons connect through synapses, or extracellular gaps, where chemical signals are conveyed. The axon, dendrite, and synapse play crucial roles in the weighting and delaying of spike signals.

![Figure 2.10: Differences between biological neurons, artificial neurons, and spiking neurons [13].](image-url)
In the standard LIF model, neurons are modeled as an equivalent electrical circuit. The membrane potential \( V_m(t) \), the voltage difference across the membrane, serves as the primary internal (activation) state variable. Ions crossing the membrane experience a resistance \( R = R_m \) and capacitance \( C = C_m \) associated with the membrane. The soma acts as a first-order low-pass filter, or a leaky integrator, with the integration time constant \( \tau_m = R_m C_m \) determining the exponential decay rate of the impulse response function. The leakage current through \( R_m \) drives the membrane voltage \( V_m(t) \) to 0, but an active membrane pumping current counteracts this, maintaining a resting membrane voltage at \( V_m(t) = V_L \).

A neuron has:

1. \( N \) inputs, which represent induced currents through input synapses \( \sigma_j(t) \), continuous time series of either spikes or continuous analog values;
2. an internal activation state \( V_m(t) \);
3. a single output state \( O(t) \).

Each input is independently weighted by \( \omega_j \) and delayed by \( \tau_j \), resulting in a time series that is spatially summed. This aggregate input electrical current is

\[
I_{app}(t) = \sum_{j=1}^{N} \omega_j \sigma_j(t - \tau_j). \tag{2.1}
\]

The result is then temporally integrated using an exponentially decaying impulse response function, resulting in the activation state

\[
V_m(t) = V_L e^{\frac{t-t_0}{\tau_m}} + \frac{1}{C_m} \int_0^{t-t_0} I_{app}(t-s) e^{\frac{s}{\tau_m}} ds, \tag{2.2}
\]

where \( t_0 \) is the last time the neuron spiked. If \( V_m(t) \geq V_{\text{thresh}} \), the neuron outputs a spike \( O(t) = 1 \) and \( V_m(t) \) is reset to \( V_{\text{reset}} \). After spiking, there is a refractory period during which it is difficult to spike again; if \( O(t) = 1 \) then \( O(t-T) = 0 \) for \( T \leq T_{\text{refract}} \). Thus, the neuron’s output is a continuous time series of spikes.

The parameters that influence the device’s behavior are: the weights \( \omega_j \), delays \( \tau_j \), threshold \( V_{\text{thresh}} \), resting potential \( V_L \), refractory period \( T_{\text{refract}} \), and integration time constant \( \tau_m \). Three factors influence \( V_m(t) \): passive leakage of current, active pumping current, and external inputs causing time-varying membrane conductance changes. These are represented by the terms in the differential equation defining \( V_m(t) \):

\[
\frac{dV_m(t)}{dt} = \frac{V_L}{\tau_m} - \frac{V_m(t)}{\tau_m} + \frac{1}{C_m} I_{app}(t); \tag{2.3}
\]

if \( V_m(t) > V_{\text{thresh}} \) then produce a pulse and set \( V_m(t) \rightarrow V_{\text{reset}} \). \( \tag{2.4} \)

### 2.3.2 Spiking Signals

At the cellular level, the brain encodes information as time-based events or spikes, represented as “hybrid signals” with both analog and digital traits, as shown in Figure 2.11. Spike processing has been refined in both biological nervous systems and engineered neuromorphic analog VLSI systems, utilizing LIF neurons as basic processing units that encode data via the analog timing of spikes. This intricate blend of analog and digital processing enables scalable high-bandwidth photonic processing by addressing the analog noise accumulation problem and the digital processing bandwidth inefficiency.
The distinctive features of the spike processing model support a hybrid analog-digital approach, allowing photonics hardware to scale in processing complexity and efficiency. Table 2.1 provides a summary, comparing digital, analog, and hybrid systems in terms of robustness, expressiveness, power efficiency, and the determining factor of usable bandwidth. Spiking signals offer natural and accessible information transmission, forming the basis for some of the extraordinary capabilities observed in neuroscience studies.

**Table 2.1:** Comparison of digital, analog, and hybrid systems in terms of robustness, expressiveness, power efficiency, and bandwidth limiter [21].

<table>
<thead>
<tr>
<th></th>
<th>Digital</th>
<th>Analog</th>
<th>Spiking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robustness</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Expressiveness</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Power Efficiency</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bandwidth Limiter</td>
<td>Clock speed</td>
<td>Minimum SNR</td>
<td>Pulse width</td>
</tr>
</tbody>
</table>

### 2.4 PHOTONIC NEURON MODEL

A photonic neuron must execute three essential mathematical operations: vector multiplication (weighting), spatial summation (addition), and a nonlinear transformation (activation function). The initial stage inputs must be consistent with the output nature—in this instance, photons. With network scaling, additional hardware mechanisms are necessary to maintain signal integrity. The neuron must support a scalable number of inputs, termed "maximum fan-in" (Nf), which determines the network’s connectivity degree. Each neuron’s output power must be sufficient to drive at least Nf other neurons, ensuring "cascadability." This concept correlates with "thresholding," where the signal-to-noise ratio (SNR) at the output must be lower than at the input. Optical systems face challenges with cascadability, thresholding, and fan-in due to quantum efficiency (finite photon supply) and amplified spontaneous emission (ASE) noise, which degrades SNR.

The initial attempt to replicate spiking LIF behavior in ultrafast optical components involved a substantial fiber-based system that occupied an entire optical bench. This early photonic neuron, while bulky, power-intensive (2W), and inefficient (1%), managed to demonstrate crucial hybrid spike processing functions like integration and thresholding [37].

A remarkable parallel between photonics and neural computing is observed in the similarity between the equations describing the intracellular potential of a LIF neuron (Eq. 2.3) and the gain dynamics of a semiconductor optical amplifier (SOA) represented by the equation below.
\[
\frac{dN'(t)}{dt} = \frac{N'_{\text{rest}}}{\tau_e} - \frac{N'(t)}{\tau_e} + \frac{\Gamma \alpha(\lambda)}{E_p} N'(t) P(t) \tag{2.5}
\]

- Activation: \(\frac{dN'(t)}{dt}\)
- Active pumping: \(\frac{N'_{\text{rest}}}{\tau_e}\)
- Leakage: \(-\frac{N'(t)}{\tau_e}\)
- External input: \(\frac{\Gamma \alpha(\lambda)}{E_p} N'(t) P(t)\)

The variable \(N'(t)\) denotes the excited carrier concentration in an optically-coupled semiconductor junction, while \(\tau_e\) represents the carrier lifetime. \(P(t)\) is the incident optical power, and \(\frac{\Gamma \alpha(\lambda)}{E_p}\) is a wavelength-dependent light-matter coupling coefficient. The alignment of these equations highlights that the inherent physical behaviors of this standard electro-optical material can replicate the primary analog processes of neural dynamics, such as summation and integration. Despite their dynamic isomorphism, these equations operate on significantly different time scales; the time constant \(\tau_m = R_mC_m\) in biological neurons is about 10 ms, whereas \(\tau_e\) in SOAs typically ranges from 25 to 500 ps. This disparity suggests that it’s possible to emulate the neural integration function at time scales millions to billions of times faster, enabling the exploration of neuro-inspired processing techniques for high-speed applications.

In the active region of the SOA, electronic free carriers behave analogously to the neuron’s state. Input spikes cause rapid fluctuations in the carrier concentration, which then decays back to its resting level. Depending on their wavelength, optical inputs can either increase or decrease the SOA state. This effect is represented in equation (8.3) by the gain parameter \(\alpha(\lambda)\). Wavelengths within the SOA’s gain band will deplete the carrier concentration, while shorter wavelengths can enhance it, providing the opposite effect. Figure 8.4 demonstrates an experimental setup showing simultaneous optical excitation and inhibition in an SOA integrator.

### 2.5 OPTOELECTRONIC DEVICES WITH EXCITABLE DYNAMICS

Recently, a variety of optoelectronic devices have been developed that exhibit excitable dynamics analogous to physiological neurons. Excitable systems are generally defined by three criteria:

- They have a single equilibrium state where the system can remain stationary indefinitely.
- When triggered beyond a certain threshold, the system follows a typical path and emits a "spike".
- After emitting a spike, the system returns to its resting state during a "refractory period," during which it is less likely to emit another spike.

Among these devices, approaches based on Semiconductor Lasers (SLs) have been particularly prominent due to their ability to exhibit multiple neuron-like behaviors, such as excitability [56, 57] and complex nonlinear dynamics [58]. This has led to the development of a wide variety of artificial optical neurons using different types of SLs, including micro-disk [59], micro-pillar [60], and micro-ring lasers [61], quantum dot (QD) lasers [62], two-section lasers with saturable absorber regions [63], and vertical-cavity surface-emitting lasers (VCSELs) [56, 64–71]. Additionally, other types of optoelectronic devices have been developed to illustrate the diversity and application of these excitable systems, including two-section models in semiconductor lasers [56, 67, 72], photonic crystal nanocavities [73], polarization-sensitive vertical cavity lasers, lasers with optical feedback or optical injection, and linked photodetector-laser systems with receiverless connections or resonant tunneling [74, 75]. Table 2.2 summarizes some of these devices.
Table 2.2: Summary of various optoelectronic excitable devices.

<table>
<thead>
<tr>
<th>Device</th>
<th>Injection Scheme</th>
<th>Pump</th>
<th>Excitable Dynamics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polarization switching VCSELs</td>
<td>coherent optical</td>
<td>optical</td>
<td>optical interference</td>
</tr>
<tr>
<td>Microdisk laser</td>
<td>coherent optical</td>
<td>electrical</td>
<td>optical interference</td>
</tr>
<tr>
<td>Injection-locked semiconductor optical laser with delayed feedback</td>
<td>electrical</td>
<td>electrical</td>
<td>interference</td>
</tr>
<tr>
<td>Resonant tunneling diode photodetector and laser diode</td>
<td>electrical or incoherent optical</td>
<td>electrical</td>
<td>electrical tunneling</td>
</tr>
<tr>
<td>Semiconductor ring laser</td>
<td>coherent optical</td>
<td>electrical</td>
<td>optical interference</td>
</tr>
<tr>
<td>Semiconductor lasers with optical feedback</td>
<td>incoherent optical</td>
<td>electrical</td>
<td>stimulated emission</td>
</tr>
<tr>
<td>Two-section gain and SA</td>
<td>electrical</td>
<td>electrical</td>
<td>stimulated emission</td>
</tr>
<tr>
<td>2D Photonic crystal nanocavity</td>
<td>electrical</td>
<td>electrical</td>
<td>thermal</td>
</tr>
</tbody>
</table>

Initial investigations concentrated on discrete nonlinear fiber components, which were hindered by their intricacy, significant energy consumption, and substantial size. Despite some biological spiking characteristics being exhibited through excitability in semiconductor lasers near Z-symmetry and the polarization switching (PS) effect in VCSELs, it wasn’t until 2013 that a comprehensive excitable laser model was introduced to emulate the behavior of an LIF neuron.

- **VCSEL with a Saturable Absorber (SA)**
  The initial compact VCSEL-SA replicating the spiking behavior of a LIF neuron was introduced in 2013 [63]. When combined with VCSOAs, VCSEL-SAs have demonstrated potential in various spiking applications, such as spike encoding, spike memory, STDP learning algorithms, and pattern recognition [76, 77].

- **PS-based VCSEL**
  The polarization switching (PS) effect and nonlinear dynamics from polarized optical injection in VCSELs have been thoroughly explored to mimic spiking neuron functions. The VCSEL’s state, influenced by specific bias conditions, is governed by two orthogonal polarizations of the fundamental transverse mode. When subjected to external optical injection with orthogonal polarization, the lasing mode transitions from parallel to orthogonal polarization, simulating spiking neuron activation [68].

- **Micropillar Laser Neuron**
  Micropillar lasers display spiking characteristics akin to GELs, reacting to disturbances with subnanosecond spikes and exhibiting both absolute and relative refractory periods [78]. The spike latency offers temporal encoding, and coupled micropillar lasers have shown the ability to execute spike-based logic gates and temporal pattern recognition [79].

- **DFB Excitable Laser Neuron**
  In 2015, a two-section DFB excitable laser neuron on a hybrid III–V/silicon platform was suggested [80]. Recently, an integrated photonic spiking processor incorporating the analog O/E/O link was demonstrated, featuring nine two-section DFB laser neurons, pairs of high-speed balanced photodetectors (BPDs), and connecting metal wires [81].

- **PCM Neuron**
  Phase-change materials (PCM) can carry out the plastic weighting operation of synapses and the LIF functionality of spiking neurons. The GST-embedded MRR resonator-based spiking neuron prototype was first introduced in 2018 [82]. An all-optical spiking neurosynaptic system has successfully demonstrated pattern recognition tasks directly in the optical domain with both supervised and unsupervised learning.

- **Microring Neuron**
  Studies have examined the excitability of passive microcavities to replicate the spiking dynamics of biological neurons. Various nonlinear effects, such as free-carrier absorption (FCA), free-carrier dispersion (FCD), and thermo-optic (TO) effects, interact within a microcavity, resulting in excitability due to the disparity between rapid free-carrier dynamics and slower heating effects [79, 83].
Fano Laser Neuron and Quantum-Dot Laser Neuron

Other excitable lasers have been utilized to imitate spiking neurons. A Fano laser on a photonic crystal platform has been proposed to achieve an all-optical nonlinear activation function with a nanosecond refractory period [84]. The waveband switching effect in QD mode-locked lasers has been employed to obtain both excitatory and inhibitory spiking neurons [62].

2.5.1 Computing with spiking VCSEL Neurons

Utilizing photonic technologies holds immense promise for establishing parallel neural network (NN) interconnections, particularly when integrated with photonic neural network (PNN) computing [3,36]. As depicted in Figure 2.12 (b), a photonic perceptron gathers optical inputs from the network and applies a nonlinear transformation to them. Optical communication is inherently parallel, energy-efficient [26], and recent advancements in optical memristors suggest a path toward programmability [85]. However, developing competitive nonlinear photonic components for artificial neurons has been challenging.

Recent advancements have brought the energy consumption of innovative photonic devices closer to that of their electronic counterparts. Enhancing nonlinearity generally involves confining photons to a small space or increasing interaction time, often using optical resonators, such as lasers. Vertical-cavity surface-emitting lasers (VCSELs) stand out among semiconductor lasers due to their unique properties, making them ideal for next-generation PNN hardware. Figure 2.12 (c) shows a VCSEL post-fabrication and illustrates how a VCSEL-based neuron processes information. VCSELs benefit from high-yield, commercially mature fabrication processes, achieving wall-plug efficiencies above 30%. Their low lasing threshold currents and amplitude-phase coupling through Henry’s alpha factor enable a highly nonlinear response to optical inputs, making them suitable as all-optical and electro-optical artificial neurons. The production of optical excitable spiking signals in VCSELs has been observed, occurring at sub-nanosecond rates that are significantly faster than the biological neuron’s timescales [70,86]. VCSELs can be modulated at speeds exceeding 30 GHz [87], resulting in ultra-low energy consumption per nonlinear transformation, around 10 femtojoules (fJ) [88].

![Figure 2.12](image_url)

**Figure 2.12:** (a) Biological neurons form complex networks through synapses, linking the axons of pre-synaptic neurons to the dendrites of post-synaptic neurons. This nonlinearity is crucial for their computational capabilities. (b) The optical perceptron mimics this process by applying a nonlinear transformation to a sum of optical fields from neurons in layer \( l \), represented as \( E_{j}^{l-1} \), scaled by synaptic weights \( W_{lj}^{l} \), which produces the output field of the neuron, \( E_{i}^{l} \). (c) Vertical-Cavity Surface-Emitting Lasers (VCSELs) can be effectively integrated with external optical fields. Their inherent dynamics allow them to exhibit a nonlinear response to such optical injections. (d) When connected through photonic links, VCSELs offer a potential pathway for the creation of advanced photonic neural networks [15].

Given the parallel nature of a potentially passive and low-loss interconnect with over 100 connections per channel, PNNs utilizing VCSELs could achieve energy usage below 100 attojoules (aJ) per operation, compared to the 100 fJ to 1 picojoule (pJ) range in electronic circuits. This is a current snapshot, and future advancements such as spin-VCSELs [89], high-beta VCSELs using cavity quantum electrodynamics could further reduce energy costs to the fundamental physical limits of a few photons per operation [90]. VCSELs emit light vertically.
to their substrates, allowing efficient testing and integration with scalable 3D photonic circuits to establish PNN connections, as shown in Figure 2.12 (d).

Additionally, multiple photonic neurons can be realized by arranging VCSELs in arrays [88], or through spatial multiplexing of a multimode large area VCSEL to implement photonic neurons in spatial modes [91]. Moreover, two orthogonal polarization directions can extend the degrees of freedom [92]. VCSEL structures can also operate in an excitable regime, either using intra-cavity saturable absorbers [86] or through injection-locked induced excitability [70].

VCSEL-Neurons have been demonstrated at both short wavelengths (e.g., 850 nm) [56] and long wavelengths (1310 nm, 1550 nm) [64], [70]. Notably, short-wavelength VCSEL-Neurons have shown capabilities in activating, storing, and controlling excitable spikes under phase-modulated optical injection. Our research group has concentrated on neuromorphic systems with VCSEL-Neurons operating at longer wavelengths within the telecom windows of 1310 nm and 1550 nm, ensuring compatibility with current optical telecommunication networks and data center technologies. We have recently demonstrated the controllable excitation [70] and inhibition [93] of sub-nanosecond excitable spike patterns that replicate various biological neuronal responses.

Table 2.3 summarizes the techniques used to develop and investigate VCSEL-Neurons. Most of the research on SL-based photonic spiking neurons has focused on single devices. However, there is a consensus that interconnected architectures are essential for developing neuromorphic photonic systems for practical computing and AI applications. Consequently, our research has emphasized connectivity between different VCSEL-Neurons, demonstrating successful communication of excitatory [94] and inhibitory spiking signals between coupled VCSEL-Neurons at telecom wavelengths. Additionally, recent work has highlighted the potential of coupled VCSEL-arrays at short wavelengths for neuromorphic applications [95]. Numerical studies have investigated various interconnectivity architectures between coupled VCSEL-Neurons at telecom wavelengths using PS for operation [96]. Theoretical research has also described the potential of photonic neuronal models based on VCSELs with a saturable absorbing region and VCSELs combined with vertical cavity semiconductor optical amplifiers for different spiking processing tasks, including spiking memory, spike encoding, spike timing-dependent plasticity, and pattern recognition [45, 63, 76].

Table 2.3: Techniques for Artificial Optical Neurons Using VCSELs.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Wavelength</th>
<th>Experimental/Numerical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spike activation via polarization switching [69]</td>
<td>1550 nm</td>
<td>Experimental</td>
</tr>
<tr>
<td>Spike activation via electrical bias stimulation [93]</td>
<td>1310 nm</td>
<td>Experimental</td>
</tr>
<tr>
<td>Spike activation via phase-modulated optical injection [71]</td>
<td>980 nm</td>
<td>Experimental/Numerical</td>
</tr>
<tr>
<td>Spike activation via amplitude-modulated optical injection [97]</td>
<td>1310 nm</td>
<td>Experimental/Numerical</td>
</tr>
<tr>
<td>Spike inhibition via amplitude-modulated optical injection [93]</td>
<td>1310 nm</td>
<td>Experimental/Numerical</td>
</tr>
<tr>
<td>Spike activation/inhibition via saturable absorber region [45, 76]</td>
<td>850 nm</td>
<td>Experimental/Numerical</td>
</tr>
<tr>
<td>Networked/coupled spiking VCSELs [97, 98]</td>
<td>1310 nm</td>
<td>Experimental/Numerical</td>
</tr>
</tbody>
</table>

Theoretical investigations have also enhanced our understanding of neuronal dynamics in VCSELs through models like the SFM and Yamada models [99]. Extensive numerical studies have validated experimental findings and expanded the potential applications of VCSEL-based spiking neurons, ranging from image processing to information encoding and complex logic operations [96, 100–105].
2.5.2 Spike-based Processing Systems with VCSELs

The emergence of experimental demonstrators for spike processing systems using VCSEL neurons marks a significant advancement in ultrafast neuromorphic photonic computing. Recent studies have showcased various artificial intelligence (AI) applications such as image processing and temporal pattern recognition, benefiting from the rapid processing capabilities and energy efficiency of these systems [105–109].

In 2020, Robertson and colleagues demonstrated a method to classify 4-bit digital patterns using a single VCSEL-neuron. This system responded to specific patterns with rapid 100 ps spikes, while remaining inactive for non-target patterns [105]. Additionally, they showcased a coincidence detection task with the same VCSEL-neuron, identifying the simultaneous arrival of two different inputs within a narrow time frame of less than 420 ps.

Further applications include all-optical XOR classification on binary patterns, emulating the function of pyramidal neurons using a dual modulated optical injection system [106]. VCSEL-neurons have also been applied in image processing tasks such as edge-feature detection and binary convolution, which computes intensity gradients in images [107, 108]. These tasks were performed efficiently, with input data rates in the GHz range and low power consumption, utilizing commercially available VCSELs.

Robertson et al. expanded the utility of a single VCSEL-neuron to mimic a complete neuronal layer, effectively conducting image edge detection with minimal preprocessing. This involved encoding the Hadamard products of a target kernel with image pixels into bursts of input pulses, integrated by the VCSEL-neuron, facilitating optical spiking convolution and edge detection [110]. This experimental photonic VCSEL layer was also integrated with a software-implemented Spiking Neural Network (SNN), achieving over 97% precision in classifying complex images from the MNIST handwritten digit database.

The capabilities of VCSEL-neurons extend to spike information encoding, as demonstrated by Hejda et al. These neurons exhibited spike latency and an inter-spike refractory period dependent on the amplitude of the input stimulus, enabling precise spike-timing and spike rate encoding of digital signals at over 1 Gbps [109].

In further research, Hejda et al. leveraged the rate coding ability of VCSEL-neurons to rapidly encode color information in images, enhancing functionality in image processing tasks [111].

VCSEL-based optical neurons thus represent a promising platform for high-speed, low-energy neuromorphic photonic spike-processing systems. With functionalities like threshold-and-fire and integrate-and-fire, these neurons are well-suited for integration into expansive photonic SNN architectures, paving the way for advanced spike-based processing applications in AI. Consequently, VCSELs provide a robust, all-optical, and hardware-efficient solution, poised to revolutionize the future of ultrafast neuromorphic computing.
3. TECHNICAL INSIGHTS OF VCSEL TECHNOLOGY

3.1 STRUCTURE OF VCSEL DEVICES

Typically, VCSEL structures are fabricated using epitaxial growth techniques such as molecular beam epitaxy (MBE), metal-organic chemical vapor deposition (MOCVD), and chemical beam epitaxy (CBE). The Figure 3.1 illustrates the layered structure of a VCSEL device.

VCSELs are distinct in their structure, consistently designed with horizontally aligned high-reflectivity dielectric mirrors that enclose a gain medium within a very short cavity, typically half a wavelength in length. The structure is etched into pillars or mesas, with optical output occurring through the top mirror which usually has slightly lower reflectivity, ranging from about 99.0% to 99.8%.

The mirrors are generally distributed Bragg reflectors (DBRs), composed of layers of different semiconductor materials transparent at the lasing wavelength but with varying refractive indices. Optical gain is achieved using either quantum wells or quantum dot ensembles. VCSEL dynamics are often modeled using rate equation models, which are more complex for VCSELs than for standard edge emitters due to their ability to emit in orthogonal optical polarizations and the need to model carrier spin-populations, based on the spin-flip model [112].

A critical parameter for VCSELs is the carrier’s relaxation oscillation frequency, typically ranging from 0.1 to 10 GHz, which largely determines the bandwidth for nonlinear transformations in VCSELs. For optical information injection, crucial is the photon lifetime within the VCSEL cavity, generally between 1 and 5 ps. This parameter is vital for achieving optical injection locking with an external laser, which is feasible within a narrow frequency detuning range dependent on the ratio of injection to emission power and the photon lifetime, which scales the VCSEL’s frequency selectivity [113].

Figure 3.1: Diagram illustrating a VCSEL device. This vertical-cavity surface-emitting laser is compact and includes two doped distributed Bragg reflectors (DBRs), a central cavity with the active region, at least one oxide aperture, and ring-shaped n- and p-contacts. Inset: Scanning electron microscope image showing the central cross-section of a VCSEL with two oxide apertures. Reproduced with permission from the referenced source. [15]

3.2 FUNDAMENTALS OF LASER THEORY

In modern times, lasers are utilized across various fields such as medical applications, telecommunications, computing, security, sensors, and industry. The success and future potential of lasers can be attributed to their unique characteristics: monochromaticity, directionality, brightness, and coherence.
The term "laser" stands for "Light Amplification by Stimulated Emission of Radiation." A laser system comprises two main components: the active medium and the cavity. These components work together to achieve light amplification, a quantum mechanical process involving the interaction between photons and the basic units of matter such as atoms, molecules, or ions. This interaction can be described for a two-level system through the processes of absorption, spontaneous emission, and stimulated emission.

### 3.2.1 Interaction Mechanisms

The interaction mechanisms in a two-level system include absorption, spontaneous emission, and stimulated emission. Initially, considering an atom in energy level 1, the ground state. If the atom remains unexcited by external factors, it stays in this state. When radiation with energy \( h \nu = h \nu_0 \) impacts the atom, there is a finite probability of transitioning from level 1 to level 2. The energy required for this transition must equal the energy difference between these levels, \( h \nu = E_2 - E_1 \). This process is known as absorption.

For spontaneous emission, assume the atom starts in the excited state (level 2). Given that \( E_2 > E_1 \), the atom will tend to transition from the excited state to the ground state. This transition releases the energy difference as electromagnetic radiation, \( h \nu = E_2 - E_1 \), termed spontaneous emission. In spontaneous emission, a photon with energy \( h \nu = E_2 - E_1 \) is emitted as the atom moves from level 2 to level 1. The emitted radiation's phase (\( \phi \)), polarization (\( \epsilon \)), and wave vector (\( k \)) are random, resulting in incoherent radiation.

Considering the case that the atom starts in energy level 2. If radiation with energy \( h \nu = E_2 - E_1 \) strikes the material, and the incident radiation frequency matches the atomic frequency, the radiation may induce the atom to transition from level 2 to level 1. The energy difference between the levels, \( \Delta E = E_2 - E_1 \), is then emitted as electromagnetic radiation, adding to the incident radiation. This process is called stimulated emission. Unlike spontaneous emission, stimulated emission produces coherent radiation, sharing the same phase, polarization, and direction as the incident radiation.

A laser system consists of two main components: the active medium and the cavity. These components work together to amplify light, a quantum mechanical process involving interactions between photons and fundamental units of matter such as atoms, molecules, or ions. This interaction can be described for a two-level system through absorption, spontaneous emission, and stimulated emission.

In absorption, an atom in the ground state (energy level 1) can absorb a photon with energy \( h \nu = E_2 - E_1 \), transitioning to an excited state (energy level 2). Conversely, in spontaneous emission, an excited atom at energy level 2 decays to the ground state, emitting a photon with energy \( h \nu = E_2 - E_1 \). The emitted photons in spontaneous emission have random phase, polarization, and direction, resulting in incoherent light.

Stimulated emission occurs when an atom in the excited state interacts with an incoming photon of energy \( h \nu = E_2 - E_1 \), causing the atom to drop to the ground state and emit a second photon that is coherent with the incoming one. This process amplifies the light and produces coherent radiation, where all emitted photons share the same phase, polarization, and direction.

To construct a laser, we consider a two-level system where \( N_1 \) and \( N_2 \) represent the populations of carriers in each level. When a photon flux \( F \) travels along the z-axis, the differential change in flux \( dF \) over a small length \( dz \) of the material results from the processes of spontaneous and stimulated emission. The cross-sectional area of the beam is \( S \), and the difference in the number of incoming and outgoing photons is \( SdF \).

For stimulated emission, the rate of change of the population in the excited state \( ES \) is given by:

\[
\frac{dN_{ES}}{dt} = -W_{21}N_{ES} \tag{3.1}
\]

where \( W_{21} \) is the stimulated emission rate. Unlike the spontaneous emission coefficient \( A \), \( W_{21} \) depends on the intensity of the incident radiation \( F \) and is defined as:

\[
W_{21} = \sigma_{21} F \tag{3.2}
\]

where \( \sigma_{21} \) is a cross-sectional area. The rate of change for the ground state population \( GS \) due to absorption is:

\[
\frac{dN_{GS}}{dt} = -W_{12}N_{GS} \tag{3.3}
\]

where:

\[
W_{12} = \sigma_{12} F \tag{3.4}
\]
Einstein showed that for non-degenerate energy levels,

\[ W_{21} = W_{12} \]  

and  

\[ \sigma_{21} = \sigma_{12} \]  

For degenerate levels with degeneracy factors \( g_2 \) and \( g_1 \), the relationships are:

\[ g_2 W_{21} = g_1 W_{12} \]  

\[ g_2 \sigma_{21} = g_1 \sigma_{12} \]  

where \( g_2 \) and \( g_1 \) are the degeneracy factors for the excited and ground states, respectively. In thermal equilibrium, the populations \( N_{1e} \) and \( N_{2e} \) are described by the Boltzmann distribution:

\[ \frac{N_{2e}}{N_{1e}} = \frac{g_2}{g_1} \exp \left( -\frac{(E_2 - E_1)}{kT} \right) \]  

where \( k \) is the Boltzmann constant and \( T \) is the absolute temperature. If \( N_2 \) exceeds \( g_2 N_1 / g_1 \), the material acts as an amplifier, achieving population inversion necessary for lasing.

### 3.2.2 Polarization of light

An electromagnetic radiation source typically comprises numerous atoms or molecules emitting light. When the electric fields generated by these emitters are unaligned, the light is classified as unpolarized. If there is some degree of alignment, the light becomes partially polarized, and with complete alignment, the light is fully polarized.

As depicted in the following figure, light traveling along the \( z \)-axis in free space features two perpendicular components, \( E_x(z,t) \) and \( E_y(z,t) \), oscillating along the \( x \) and \( y \) axes, respectively. The time vector is represented by \( t \). These components can be described using the plane wave equations [114]:

\[ E_x(z,t) = E_{0x} \cos(kz - \omega t) \]  

\[ E_y(z,t) = E_{0y} \cos(kz - \omega t + \delta) \]  

Here, \( \omega \) is the angular frequency, \( k \) is the wavenumber, and \( \delta \) represents the phase shift between \( E_x \) and \( E_y \). To determine the type and extent of polarization, the propagator can be eliminated from the equations for \( E_x \) and \( E_y \):

\[ \left( \frac{E_x}{E_{0x}} \right)^2 + \left( \frac{E_y}{E_{0y}} \right)^2 - 2 \left( \frac{E_x}{E_{0x}} \right) \left( \frac{E_y}{E_{0y}} \right) \cos \delta = \sin^2 \delta \]  

This forms an ellipse known as the polarization ellipse, which is in a non-standard configuration.
If the amplitude in the x-direction is zero \((E_{0x} = 0)\), only the y-component exists, leading to vertically polarized light, as shown in the leftmost section of the figure (Linear Polarization). Conversely, if \(E_{0y} = 0\), only the x-component remains, resulting in horizontally polarized light. When there is no phase shift \((\delta = 0^\circ)\) and \(E_{0x} = E_{0y}\), \(E_x\) equals \(E_y\), producing light that is linearly polarized at a 45° angle.

For a phase difference of \(\delta = 90^\circ\) and \(E_{0x} = E_{0y}\), the ratios \(E_x/E_{0x} = \cos \theta\) and \(E_y/E_{0y} = \sin \theta\) apply, with \(\theta = k z - \omega t\). This gives the equation for circularly polarized light:

\[
\left(\frac{E_x}{E_{0x}}\right)^2 + \left(\frac{E_y}{E_{0y}}\right)^2 = \cos^2 \theta + \sin^2 \theta = 1
\]  

(3.13)

Light is termed elliptically polarized if two plane waves of unequal amplitudes \((E_{0x} \neq E_{0y})\) are in phase by \(\delta = 90^\circ\) or if \(E_{0x} = E_{0y}\) and the relative phase differs from 90°, as shown in the rightmost section of the figure (Elliptical Polarization).

### 3.3 OVERVIEW OF THE SPIN FLIP MODEL (SFM)

The SFM is a robust theoretical framework aimed at analyzing the dynamic behavior of spin-polarized lasers using a spin-dependent rate-equation approach. This model offers profound insights into the interactions of spin-polarized carriers within the laser, refined through various adaptations in the literature.

Originating from the Maxwell-Bloch equations and developed by San Miguel, Feng, and Moloney in 1994 [112], the SFM is specifically tailored for Vertical-Cavity Surface-Emitting Lasers (VCSELs) operating in single longitudinal and fundamental transverse modes. The model effectively describes phenomena like polarization switching and bistability, frequently observed in conventional VCSELs. In the absence of external injection, the SFM simplifies the four-level Maxwell-Bloch equations into a set of rate equations, detailing the dynamics of spin-polarized carriers and their interactions.

![Figure 3.3: Schematic illustrating the key components of the SFM model, which is based on the four-level Maxwell–Bloch equations. The carrier densities \(n^+\) (for spin-down) and \(n^-\) (for spin-up) correspond to the right \((E^+)\) and left \((E^-)\) circularly polarized field components, respectively. The characteristic decay rates for spin and birefringence are denoted by \(\gamma_s\) and \(\gamma_p\) [17].](image)

The SFM distinguishes between two carrier densities: \(n^+\) for spin-down and \(n^-\) for spin-up populations. These densities interact with the circularly polarized field components: right \((E^+)\) and left \((E^-)\). The model includes decay rates associated with spin (\(\gamma_s\)) and birefringence (\(\gamma_p\)), and it also considers electron-hole recombination processes between the conduction band (CB) and valence band (VB), focusing on spin dynamics.

In quantum wells, carriers are divided into spin-up and spin-down populations. Circularly polarized fields \((E^+\) for right and \(E^-\) for left) align with specific spin orientations during lasing transitions between the CB and VB.
The total angular momentum ($J$) is $\frac{1}{2}$ for electrons in the CB and $\frac{3}{2}$ for heavy holes in the VB. Heavy holes have more energy than light holes, making transitions from the CB to the light hole band negligible. For circularly polarized light emission, transitions between CB and VB must satisfy $J_{VB}^z - J_{CB}^z = -1$ or $+1$, corresponding to transitions from $-\frac{1}{2}$ to $-\frac{3}{2}$ or from $\frac{1}{2}$ to $\frac{3}{2}$. Spin-flip processes, characterized by the spin relaxation rate ($\gamma_s$), link these transitions. Hole spins typically relax faster than electron spins, making spin-flip processes for holes in the VB negligible due to sub-picosecond relaxation times.

The SFM also integrates gain anisotropy (dichroism, $\gamma_a$) and birefringence ($\gamma_p$), coupling orthogonally polarized modes in the VCSEL. The model is suitable for both pulsed and continuous optical pumping of spin-VCSELs, aligning well with experimental results. Parameters like $\gamma_p$ and $\gamma_s$ show a broader range compared to conventional lasers, requiring identification of specific parameter ranges for standalone VCSELs. By comparing experimental data from different research groups with numerical simulations, typical parameters such as anisotropy, birefringence, and spin relaxation rates can be determined.

3.4 SPIN-VCSELs

As mentioned in the previous section a unique aspect of spin injection is the creation of an imbalance in spin-polarized carriers. Unlike conventional devices where injected carriers are non-polarized, resulting in equal numbers of spin-up and spin-down carriers, spin lasers achieve this imbalance by injecting spin-polarized carriers into the gain region, which often comprises several quantum well (QW) or quantum dot (QD) layers.

To highlight the advantages of spin lasers over conventional lasers, Figure 3.4 uses a simplified bucket model introduced by Lee et al. [115]. In conventional lasers, represented by a single-compartment bucket, carriers (depicted as water) are injected into the system, and light emission occurs when the input power surpasses the threshold, causing excess carriers to escape.

The spin laser model is more complex, featuring a two-compartment bucket that can be filled with different liquids representing spin-up and spin-down carriers. The interconnected compartments indicate the exchange of carriers with different spins due to spin-flip effects. The mixed output of the two liquids corresponds to the polarization control in the actual device. Both models show losses due to spontaneous emission, symbolized by leaks.

Spin-VCSELs enhance data transmission rates and energy efficiency due to rapid spin dynamics. Integration with on-chip photonic circuits leads to efficient and compact optical interconnects, crucial for high-performance computing systems. Advances in materials, like bismuth-doped iron garnet, have improved spin injection efficiency and performance, reducing spin relaxation times and enhancing polarization stability [115].

Spin-polarized light sources, such as light-emitting diodes and lasers, emit circularly polarized light through radiative recombination of spin-polarized carriers. These sources are ideal for cryptographic communications and reconfigurable optical interconnects [116].

Spin-lasers offer faster modulation dynamics, lower thresholds, and up to 100% polarization control compared to conventional lasers [117]. VCSELs demonstrate circularly polarized lasing and threshold reduction by injecting spin-polarized electrons, showing high degrees of spin polarization and long electron spin relaxation times [118].

Theoretical models based on the SFM, which include gain anisotropy and birefringence, align well with experimental results for pulsed and continuous optical pumping of spin-VCSELs [112]. Simplified SFM equations predict performance improvements, including threshold reduction [119]. Numerical solutions of the SFM reveal regions of parameter space with unstable and oscillatory behavior, influenced by spin-flip processes and birefringence [120]. Experimental results have demonstrated ultrafast circular polarization oscillations and damped periodic oscillations in spin-VCSELs [119].

Recent theoretical analyses combining the SFM with the largest Lyapunov exponent method have identified stability and instability regions in spin-VCSELs, providing insights into optimizing their performance for various applications [120].

3.4.1 Spin Dynamics in Quantum-Dot Spin VCSELs

The operation of spin-polarized VCSELs can be theoretically examined using the spin-flip model (SFM), which simplifies and clarifies the complex processes, resulting in a set of rate equations (REs) that are computationally
efficient. This approach leads to four-level coupled rate equations for quantum wells (QWs) or six-level equations for quantum dots (QDs). According to the SFM, in the absence of an external magnetic field, the oscillatory behavior of spin-polarized VCSELs is driven by the interplay between spin relaxation processes ($\gamma_s$), dichroism ($\gamma_a$), and birefringence ($\gamma_p$).

A part of this study investigates the dynamics resulting from the interaction between two identical QD-spin lasers (configured as master and slave) through optical injection. The aim is to generate dynamic behaviors and examine phenomena such as spiking, chaos, and self-pulsation, which are important for neuromorphic photonic applications. These phenomena are also crucial for optical telecommunications and information processing.

For QD spin-VCSELs, the device’s degrees of freedom are categorized into two main groups: those related to the emitted electromagnetic field’s wavelength and those related to its polarization. Concerning the emitted wavelength, the device has two degrees of freedom:

- Emission from the Excited State (ES)
- Emission from the Ground State (GS)

Regarding polarization, the device also has two degrees of freedom:

- Left-circularly polarized electromagnetic radiation
- Right-circularly polarized electromagnetic radiation

In the case of QD spin-VCSELs, considering the excited state (ES) energy level, as the current density in the device increases, the emission will initially occur from the ground state (GS). Subsequently, when lasing begins from the ES, the intensity of the GS will saturate, while the intensity of the ES electromagnetic field will increase linearly.

The dynamics of lasers can generally be controlled by injecting an external field [121] in a master-slave laser setup, as shown in Figure 3.5. In the current work, the focus is on the dynamics produced after controlling the polarization dynamics with an external laser field. This involves the optical injection of polarized light from a spin QD VCSEL system (master laser) into another spin QD VCSEL system (slave laser), taking into account the ES and GS energy levels in both lasers.

The interaction between two quantum dot (QD) spin-VCSELs is analyzed through the mechanism of optical injection. The mathematical framework for this interaction extends from the established equations for an isolated (solitary) QD spin-VCSEL. Within these equations, critical factors such as the intensity of the injected beam (injection strength) and the frequency offset between the master and slave lasers (detuning) are incorporated.
The initial framework for the analysis is based on the QD spin-VCSEL model presented in [122]. This model only considers the energy levels of the WL and GS. Building upon this, the work by Georgiou et al. [20] introduced modifications to the model, including the ES level in a manner similar to conventional (non-spin) QD systems. However, in the aforementioned study, the carriers and fields associated with the WL, ES, and GS are spin-resolved, resulting in an 8-level rate equation system. For this analysis, upward carrier transitions were ignored.

The spin-resolved energy diagram of the QD system, along with the relevant transitions utilized in the study of Georgiou et al. [20], is depicted in Figure 3.6. The ES level is fourfold degenerate, while the GS level is twofold degenerate. Spin-polarized carriers, either spin-up (−) or spin-down (+), are generated in the WL through optical pumping. The capture rate to the ES level is denoted by \( \gamma_0 \). At the ES level, spin-polarized carriers relax to the spin-up (down) GS level with an intra-dot relaxation rate \( \gamma_{21} \). Spin-relaxation processes within the dot can occur from spin-up (down) ES to spin-down (up) ES, as well as from spin-up (down) GS to spin-down (up) GS, at a rate \( \gamma_s \).

Lasing occurs through transitions from the ES or GS to the valence band (VB), emitting right \( (E_{ES}^+, E_{GS}^+) \) and left \( (E_{ES}^-, E_{GS}^-) \) circularly polarized electric fields at two distinct wavelengths. The right and left circularly polarized fields are coupled through the birefringence rate \( \gamma_p \) and dichroism \( \gamma_a \).

This physical process is described by a set of rate equations (REs), consisting of six equations for the dynamics of carriers \( (f_{WL}^\pm, f_{ES}^\pm, f_{GS}^\pm) \) and four equations for the dynamics of the electric field amplitudes \( (E_{GS}^+, E_{GS}^-) \).

The complete set of REs describing the modified SFM model is used in this study to produce dynamic maps is the following:
\[
\frac{d(NWL f_{WL}^+)}{dt} = \frac{I}{q} - \gamma_0 NWL f_{WL}^+(1 - f_{ES}^+) + 4N_D \gamma_{esc} f_{ES}^+ \mp \gamma_f NWL(f_{WL}^+- f_{WL})
\]
(3.14)

\[
\frac{d(4N_D f_{ES}^+)}{dt} = \gamma_0 NWL f_{WL}^+(1 - f_{ES}^+) - 4\gamma_{esc} N_D f_{ES}^+ \mp \gamma_f N_D (f_{ES}^+- f_{ES})
\]
(3.15)

\[
\frac{d(2N_D f_{GS}^+)}{dt} = \gamma_{21} N_D f_{ES}^+(1-f_{GS}^+) - \gamma_{12} 2N_D f_{GS}(1-f_{ES})
\]
(3.16)

The modified SFM REs are:

\[
\frac{df_{WL}^+}{dt} = \frac{I}{qNW_L} - \gamma_0 f_{WL}^+(1-f_{ES}^+) + \frac{4N_D}{NW_L} \gamma_{esc} f_{ES}^+ \mp \gamma_f (f_{WL}^+- f_{WL}) - \gamma_f f_{WL}
\]
(3.19)

where \(\hat{N}_L = NW_L/A\).

\[
\frac{df_{ES}^+}{dt} = \gamma_{0} NWL f_{WL}^+(1-f_{ES}^+) - \gamma_{esc} f_{ES}^+
\]
(3.20)

\[
\frac{df_{GS}^+}{dt} = 2\gamma_{21} f_{ES}^+(1-f_{GS}^+) - \gamma_{12} f_{GS}^+(1-f_{ES}^+)
\]
(3.21)

For the normalization of the system of REs we introduce a change in variables as follows:

\[
n_{WL}^+ = \frac{NW_L}{ND} f_{WL}^+
\]
(3.24)

\[
E_{GS} = E_{SGS} \sqrt{\frac{\gamma_n}{v_{aGS}}}
\]
(3.25)
\[ E_{ES} = E_{SES} \sqrt{\frac{\gamma_n}{\gamma_k \Gamma_{aES}}} \]  
\[ \frac{I}{qN_D \tau_r} = \frac{I}{I_\pi \tau_r} = \gamma_n J \]  

The normalized form of modified SFM REs (3.14), (3.15), (3.16), (3.17), (3.18) is:

Specifically, in the equations provided, the terms representing the injection strength and the detuning are identified as the 3rd and 4th terms, respectively, on the right-hand side of equations 3.32 and 3.31.

\[ \frac{d n^\pm_{WL}}{dt} = \gamma_n f^\pm_{ES} - \gamma_0 n^\pm_{WL} (1 - f^\pm_{ES}) - \gamma_n n^\pm_{WL} + 4 \gamma_{esc} f^\pm_{ES} + \gamma_j (n^+_{WL} - n^-_{WL}) \]  
\[ \frac{d f^\pm_{ES}}{dt} = \frac{1}{4} \gamma_0 n^\pm_{WL} (1 - f^\pm_{ES}) - \gamma_{esc} f^\pm_{ES} - \gamma_n f^\pm_{ES} \]  
\[ - \gamma_{21} f^\pm_{ES} (1 - f^\pm_{GS}) + \frac{1}{2} \gamma_{12} f^\pm_{GS} (1 - f^\pm_{ES}) \]  
\[ - \frac{1}{4} \gamma_n (2f^\pm_{ES} - 1) |E^\pm_{ES}|^2 + \gamma_j (f^+_{ES} - f^-_{ES}) \]  
\[ \frac{d f^\pm_{GS}}{dt} = 2\gamma_{21} f^\pm_{ES} (1 - f^\pm_{GS}) - \gamma_{12} f^\pm_{GS} (1 - f^\pm_{ES}) - \gamma_n f^\pm_{GS} \]  
\[ - \frac{1}{2} \gamma_0 (2f^\pm_{GS} - 1) |E^\pm_{GS}|^2 - \gamma_j (f^+_{GS} - f^-_{GS}) \]  
\[ \frac{d E^\pm_{GS}}{dt} = k [h_1 (2f^\pm_{GS} - 1)] (1 + i\alpha) E^\pm_{GS} - (\gamma_a + i\gamma_p) E^\mp_{GS} \]  
\[ \frac{d E^\pm_{ES}}{dt} = k [h_2 (2f^\pm_{ES} - 1)] (1 + i\alpha) E^\pm_{ES} - (\gamma_a + i\gamma_p) E^\mp_{ES} \]  

The dynamic variables \( n_{WL} \) and \( n_{ES} (n_{GS}) \) represent the normalized carrier concentrations in the conduction band for the Wetting Layer (WL) and the ES (GS) levels, respectively. The superscripts + and − indicate the Right Circular Polarized (RCP) and Left Circular Polarized (LCP) components of the emitted light.

The parameters of the SFM for the QD spin-VCSEL are defined as follows:

- \( \kappa \): photon decay rate
- \( \alpha \): linewidth enhancement factor
- \( h_1 \): normalized gain coefficient for GS transitions
- \( h_2 \): normalized gain coefficient for ES transitions
- \( \gamma_n \): carrier recombination rate
- \( \gamma_{21} \): intradot relaxation rate of spin-polarized carriers from the ES level to the spin-up (down) GS level
- \( \gamma_0 \): capture rate from the WL to the ES level
- \( \gamma_j \): spin relaxation rate in the ES, GS, and WL
- \( \gamma_p \): birefringence rate
- \( \gamma_a \): dichroism rate
Some carriers can thermally excite from the ES to the WL with an escape rate $\gamma_{\text{esc}}$ and from the GS to the ES with an escape rate $\gamma_{12}$. These rates are given by:

$$
\gamma_{12} = \gamma_{21} e^{-\Delta E_{E\text{S},G\text{S}}/k_B T}
$$

$$
\gamma_{\text{esc}} = \gamma_0 e^{-\Delta E_{W\text{L},E\text{S}}/k_B T}
$$

where $\Delta E_{E\text{S},G\text{S}}$ ($\Delta E_{W\text{L},E\text{S}}$) is the energy required for carrier excitation from the GS (WL) to the ES (WL), and $k_B T$ is the product of the Boltzmann constant $k_B$ and the temperature $T$.

The variables $f_{E\text{S}}^+$ and $f_{G\text{S}}^+$ represent the occupation probabilities of the ES and GS energy levels, respectively. The electric field amplitudes for transitions involving the ES and GS levels are denoted as $E_{E\text{S}}^\pm$ and $E_{G\text{S}}^\pm$, respectively.

Key parameters in the model include the dichroism rate ($\gamma_a$), the electron charge ($e$), the density of states in the WL ($N_{W\text{L}}$), and the density of dots per volume ($N_{Q\text{D}}$). The differential gain is represented by ($a$) and the linewidth enhancement factor by ($\alpha$).

The cavity loss rate ($k$) is defined as:

$$
k = \frac{1}{2\tau_p}
$$

where $\tau_p$ is the photon lifetime. The normalized gain coefficient for GS transitions ($h_1$) is given by:

$$
h_1 = v_g \Gamma a N_{Q\text{D}} \tau_p
$$

where $v_g$ is the group velocity and $\Gamma$ is the optical confinement factor. For ES transitions, the gain coefficient ($h_2$) is:

$$
h_2 = 2h_1
$$

### 3.5 STABILITY MAPS CALCULATION USING LARGEST LYAPUNOV EXPONENT METHOD (LLE)

The influence of the Excited State (ES) on the dynamics of Quantum Dot (QD) spin-Vertical-Cavity Surface-Emitting Lasers (spin-VCSELs) is studied through the numerical solution of specific rate equations. These dynamics are mapped into a plane defined by pump ellipticity ($P$) and total pump intensity ($\eta$). Here, $P$ represents the ellipticity of the pump, calculated as:

$$
P = \frac{\eta^+ - \eta^-}{\eta^+ + \eta^-}
$$

and $\eta$ denotes the total pump intensity:

$$
\eta = \eta^+ + \eta^-
$$

The pair ($P, \eta$) dynamically controls the stability of the emitted polarized fields and is commonly used to investigate the nonlinear dynamics of spin-VCSELs. The Largest Lyapunov Exponent (LLE) method is employed to identify and quantify various dynamic behaviors, including chaotic, periodic, and stable regimes. This method has been previously applied to analyze injection-locked lasers and QD spin-VCSELs considering only the Wetting Layer (WL) and Ground State (GS).

The LLE measures the average rate of growth (or shrinking) of small perturbations in the solutions of a dynamical system, providing insight into the nature of orbits in the phase space. It indicates whether the system’s behavior is chaotic, periodic, or stable. For practical computation, the LLE is calculated by observing the time evolution of two nearby orbits with a small initial separation ($d_0$, typically around $10^{-8}$). After one integration
time step (τ), the new distance \(d_j\) between the evolved orbits is measured, and the logarithm of their ratio is computed. This process is iterated over \(N\) time steps, and the LLE is estimated as:

\[
\text{LLE} = \frac{1}{N\tau} \sum_{j=1}^{N} \ln \left( \frac{d_j}{d_{j-1}} \right)
\]  

(3.40)

Positive LLE values correspond to chaotic behavior, negative values indicate stable solutions, and zero values suggest periodic solutions or limit cycles. The LLE is considered one of the most effective tools for identifying and quantifying dynamic behaviors in a system.

By applying the LLE method to the numerical solutions of the SFM equations, we can generate stability maps that highlight stable and unstable regions in the \(P-\eta\) plane. These maps offer valuable insights into the operational dynamics of spin-VCSELs, revealing how different pumping conditions affect their stability and dynamic behavior.

3.5.1 Practical Computation of LLE

To compute the LLE in practice, consider a trajectory with an initial condition \(X(0) = X_0\) and a nearby perturbed trajectory with \(X(0) = X_0 + \delta X_0\). As these trajectories evolve, the perturbed phase point quickly aligns itself with the fastest-growing direction relative to the original trajectory. By repeatedly measuring the distance between these trajectories and calculating the logarithmic ratio of their separations, the LLE can be accurately estimated.

![Figure 3.7: Illustration of the Largest Lyapunov Exponent method. The figure shows the divergence or convergence of trajectories over time, represented by intensity and carrier density plots [17].](image)

The use of the LLE method provides a robust framework for analyzing the dynamics of spin-VCSELs. By leveraging this technique, we can map out stability regions in terms of pumping power and ellipticity, offering a comprehensive understanding of the operational dynamics of these advanced semiconductor lasers. This approach, inspired by the work of Georgiou et al. [20], highlights the importance of understanding various dynamic behaviors for the development and optimization of spin-VCSELs.
4. METHODOLOGY

In the analysis pipeline followed in this work (Figure 4.1), the time series for ground state (GS) and excited state (ES) emissions for clockwise and anticlockwise circular polarization are first standardized and normalized. Feature extraction then involves a detailed analysis of the Power Spectral Density (PSD) of spikes and chaotic regimes. In the PSD of spikes, the prominence of the peaks consistently decreases, highlighting only the initial peak, while in chaotic regimes, the prominence of peaks does not adhere to a specific pattern. This distinction aids in characterizing and distinguishing between different dynamical states. Dimensionality reduction using PCA, t-SNE, and UMAP follows, culminating in classification by an AI algorithm and performance evaluation of the AI algorithm.

![Figure 4.1: Steps in the analysis pipeline followed in this work.](image)

4.1 DATASET GENERATION

4.1.1 Excitability Analysis of optically injected QD spin-VCSELs Using Largest Lyapunov Exponent (LLE) Method

The study of optical injection in semiconductor lasers, particularly focusing on Quantum Dot (QD) spin-Vertical-Cavity Surface-Emitting Lasers (spin-VCSELs), provides significant insights into their dynamic behaviors. This research is based on a previous work that investigates the influence of the Excited State (ES) on the dynamics of spin-VCSELs using the numerical solution of specific rate equations and the Largest Lyapunov Exponent (LLE) method [20]. The goal is to map these dynamics into a $k_{\text{inj}} - \Delta f$ plane and identify regions of excitability and stability so as to generate a dataset of time traces including chaotic and spiking behavior.

The process of obtaining time series data from QD Spin-VCSELs involves several critical steps, each designed to comprehensively model and analyze the dynamic behavior of these semiconductor lasers. The initial step is to define the system of delay differential equations (DDEs), which model the behavior of the electric field and carrier densities in both spin-up and spin-down states. These equations include delay terms to represent the feedback mechanism, which is crucial for accurately capturing the laser dynamics. This foundational step ensures that all relevant physical processes are mathematically represented, setting the stage for detailed simulations.

Once the system is defined, the next step is to generate dynamic maps by simulating the behavior of the QD Spin VCSEL under various conditions. This involves sweeping through a range of parameters such as injection strength and detuning frequency to explore different dynamic regimes.
4.1.2 Extrema Analysis Maps and Dynamic Control

Figure 4.2 illustrates an example of Extrema Analysis maps in the $k_{\text{inj}} - \Delta f$ plane for the excited state (ES) of the laser system. These maps are generated by solving the rate equations under varying conditions of injection strength ($k_{\text{inj}}$) and detuning ($\Delta f$). The color-coded regions represent different dynamical behaviors of the slave laser:

- **Dark Blue Region**: Indicates continuous wave (CW) operation where the intensity of the slave laser remains constant over time, mirroring the stable dynamics of the master laser.
- **Light Blue to Yellow Regions**: Correspond to periodic oscillations, with the number representing the period of the oscillations.
- **Green to Yellow Regions**: Denote chaotic behavior where the oscillations are irregular and complex.

The inverted triangular region in the map marks the injection locking area. Within this region, the slave laser synchronizes with the master laser’s CW output. As we move towards higher injection strengths, the system transitions from irregular dynamics to stable CW behavior, indicating the boundary of the injection-locked region.

![Extrema Analysis map in the $k_{\text{inj}} - \Delta f$ plane for ES.](image)

**Figure 4.2**: Extrema Analysis map in the $k_{\text{inj}} - \Delta f$ plane for ES.

Excitability is most commonly observed near the boundaries of the injection-locked region. In Figure 4.2, the red marked area of the triangular region is of particular interest. This area signifies the transition from CW to periodic operation, where excitability is likely to occur.

Figure 4.3 zooms in on a specific excitability point (Point A) in the $k_{\text{inj}} - \Delta f$ plane. Points B and C, positioned near Point A, are also highlighted. These points are characterized as follows:

- **Point A**: Exhibits Period-4 oscillations, representing a region of periodic behavior.
- **Points B and C**: Both show CW behavior with stable intensities.
The transition between these points reveals the conditions under which excitability arises. For instance, by decreasing $k_{inj}$ at a constant $\Delta f$, the system moves from Point C (CW) to Point A (Periodic-4). Alternatively, changing $\Delta f$ while keeping $k_{inj}$ constant transitions the system from Point B (CW) to Point A.

![Figure 4.3](image)

*Figure 4.3: The boundary of locked and unlocked regions of the map in Figure 4.2.*

For each of such points in the Extrema Analysis Map we obtain by running the simulations an array consisting four different time series vectors corresponding to ES Plus, ES Minus, GS Plus and GS Minus.

### 4.1.3 Numerical Simulations

Setup of simulation parameters and solvers

Tools and techniques (MATLAB)

#### Table 4.1: Values of parameters for the QD spin-VCSEL [20]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value / Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photon decay rate</td>
<td>$\kappa$</td>
<td>250 ns$^{-1}$</td>
</tr>
<tr>
<td>Carrier recombination rate</td>
<td>$\gamma_m$</td>
<td>2.5 ns$^{-1}$</td>
</tr>
<tr>
<td>Capture rate into from WL</td>
<td>$\gamma_o$</td>
<td>400 ns$^{-1}$</td>
</tr>
<tr>
<td>into QD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spin relaxation rate</td>
<td>$\gamma_s$</td>
<td>10 ns$^{-1}$</td>
</tr>
<tr>
<td>Birefringence rate</td>
<td>$\gamma_p$</td>
<td>20 ns$^{-1}$</td>
</tr>
<tr>
<td>Dichroism rate</td>
<td>$\gamma_a$</td>
<td>0</td>
</tr>
<tr>
<td>Linewidth enhancement factor</td>
<td>$\alpha$</td>
<td>1.2</td>
</tr>
<tr>
<td>Intradot relaxation rate</td>
<td>$\gamma_{21}$</td>
<td>120 ns$^{-1}$</td>
</tr>
<tr>
<td>External cavity roundtrip time</td>
<td>$\tau$</td>
<td>0.15 ns (0.1 to 1 ns)</td>
</tr>
<tr>
<td>Feedback strength</td>
<td>$k_f$</td>
<td>0 (0 to 10)</td>
</tr>
<tr>
<td>Normalized gain coefficient</td>
<td>$h$</td>
<td>1.1995</td>
</tr>
<tr>
<td>Pump current</td>
<td>$\eta$</td>
<td>30 (2 to 60)</td>
</tr>
<tr>
<td>Thermal voltage</td>
<td>$kT$</td>
<td>0.025</td>
</tr>
</tbody>
</table>
4.2 FEATURE EXTRACTION

4.2.1 Power Spectral Density

To obtain the frequency domain features, the power spectral density (PSD) of the time series was calculated using the Fast Fourier Transform (FFT). The PSD provides a comprehensive view of how the power of a signal is distributed over different frequency components [123]. The analysis of PSD involves several important metrics:

- **Number of peaks**: The number of peaks in the PSD indicates the distinct frequency components that contribute significantly to the signal’s power. Identifying these peaks helps in understanding the dominant frequencies present in the signal and is crucial for applications like speech processing, vibration analysis, and radar signal analysis.

- **Prominence of peaks**: The prominence value of the PSD is a key measure in signal processing, reflecting the power distribution of a signal across different frequencies. Prominence measures how much a peak stands out due to its intrinsic height and its distance from surrounding peaks. This metric is particularly useful in fields such as biomedical signal analysis, where it can help identify significant physiological events, as well as in telecommunications and audio processing for identifying key signal features. Prominence is mathematically defined as the vertical distance between the peak and its lowest contour line, which encloses the peak without encompassing any higher peak.

- **Width of peaks**: The width of peaks in the PSD is another crucial measure, indicating the bandwidth over which the signal’s power is spread. This metric helps in determining the frequency resolution of the signal. Understanding peak width is essential for applications in telecommunications, where it can affect bandwidth allocation and signal clarity, as well as in biomedical engineering for analyzing the frequency content of physiological signals, and in audio processing to distinguish between different sound sources. The width at half maximum (FWHM) is a common way to quantify peak width, which measures the distance between the points on the peak at which the signal is half its maximum value.

The accurate calculation and interpretation of these PSD features are crucial for developing robust signal processing applications. Advanced techniques, such as adaptive filtering and wavelet transform, can be used in future research in conjunction with PSD analysis to provide deeper insights into signal characteristics and improve the performance of signal processing systems.

4.2.2 Univariate Skewness

Skewness measures the asymmetry of the probability distribution of a real-valued random variable about its mean. Univariate skewness $G_1$ according to Fisher is computed as follows:

$$G_1 = \frac{\sqrt{n(n-1)}}{n-2} \cdot \frac{m_3}{m_2^{3/2}} \quad (4.1)$$

where:

$$m_r = \sum_{i=1}^{n} (x_i - \bar{x})^r / n \quad (4.2)$$

represents the $r$th central moment, $\bar{x}$ is the sample mean, and $n$ is the sample size. The sample skewness, $G_1$, can range from negative to positive infinity. For distributions that are symmetric, such as the normal distribution, the expected skewness is 0. When skewness is non-zero, it signifies asymmetry in the distribution: positive skewness suggests a longer tail on the right side, while negative skewness indicates a longer tail on the left side [124]. In 4.4 three distributions with three different values of skewness are depicted. The pink color line represents a lognormal distribution while the skewness value is positive (0.95), the cyan line presents normal distribution as the skewness value is 0, and finally the orange line follows a skew-normal distribution and corresponds to negative skewness = -0.2 [125].
4.2.3 Univariate Kurtosis

Kurtosis pertains to the characteristics of the tails, shoulders, and peaks within a distribution. For univariate kurtosis $G_2$ is computed with the following mathematical formula:

$$G_2 = \frac{n - 1}{(n - 2)(n - 3)} \left[ (n + 1) \left( \frac{m_4}{m_2^2} - 3 \right) + 6 \right]$$

(4.3)

where $m_r$ is calculated by Equation 4.2

Generally, higher kurtosis values correspond to distributions with pronounced peaks and heavier tails, while lower values are associated with flatter distributions. DeCarlo et al. [126] highlight that kurtosis is as much about the shoulders and tails of a distribution as it is about its peakedness. This is due to the fact that variance can obscure peakedness.

As illustrated in Figure 4.5 (a), we observe three normal distributions with identical kurtosis values of 0 but varying variances (0.7, 1.2, 1.8). The distribution with a variance of 0.7 has the highest peak and is the most
narrow, indicating that the data points are more concentrated around the mean. As the variance increases to 1.2 and 1.8, the distributions become wider and flatter, showing that the data points are more spread out.

Figure 4.5 (b) contrasts this by showing three distributions with different kurtosis values (4, 1, -2) but the same variance. Normal distributions with lower variance exhibit high peaks and light tails, whereas those with higher kurtosis display high peaks and heavy tails. Thus, the overall shape of the distribution is more indicative of kurtosis than peakedness alone. It is important to note that kurtosis tends to increase as skewness increases. The relationship between kurtosis and skewness is given by:

\[ \text{kurtosis} = \text{skewness}^2 - 2 \]  

Equation (4.4)

The expected kurtosis for a normal distribution is 0. Distributions with positive kurtosis, or leptokurtic distributions, have fatter tails than a normal distribution with the same variance. Typically, data sets with outliers or extreme values exhibit positive kurtosis. Conversely, distributions with negative kurtosis, or platykurtic distributions, have relatively flat shoulders and short tails.

4.3 DATA HANDLING AND PREPROCESSING

The manipulation of datasets composed of power vectors is crucial for training ML models. The efficient handling of data impacts the training speed and the effectiveness of the learning algorithm [128]. The initial stage of the workflow involves loading and manipulating the dataset, corresponding to different VCSEL outputs (ES_plus, ES_minus, GS_plus, GS_minus) from various VCSEL configurations. The process follows by normalization which is a critical preprocessing step, typically involving scaling the input features to a standard range or distribution. Specifically, the data are Z-scored, meaning each feature’s values are scaled to have a mean of zero and a standard deviation of one. This normalization can be performed within the optional transform parameter in the dataset class, ensuring that the model receives data conducive to learning without internal covariate shift.

4.4 DIMENSIONALITY REDUCTION

Tools for dimension reduction (DR) used in data visualization can significantly influence our interpretation of the geometric and neighborhood structures within datasets. These tools can be incredibly useful by allowing us to visualize data clusters and gain insights into distributional properties. However, it is important to note that DR techniques can also be misleading. They may present cluster structures that do not actually exist in the original data, or they might show observations as being far apart in the reduced space when they are actually close in the high-dimensional space [129]. This discrepancy highlights the potential pitfalls of relying solely on DR for data analysis. Consequently, if multiple DR algorithms are applied and yield differing results, it becomes challenging to ascertain which, if any, of these outcomes accurately represent the original data distribution. This issue underscores the inherent difficulty in validating the trustworthiness of DR results, as there is no straightforward method to determine which visualization correctly preserves the underlying structure of the dataset.

Dimensionality reduction methods convert the high-dimensional data set \( X = x_1, x_2, \ldots, x_n \) into two or three-dimensional data \( Y = y_1, y_2, \ldots, y_n \) which can be visualized using a scatterplot. In this context, we refer to the low-dimensional data representation \( Y \) as a map, and the individual low-dimensional representations of data points as map points. The goal of dimensionality reduction is to retain as much of the significant structure of the high-dimensional data as possible in the lower-dimensional map. There are various techniques for this purpose, each differing in the type of structure they preserve. Traditional dimensionality reduction techniques, such as Principal Components Analysis (PCA) [130] and classical multidimensional scaling (MDS) [131], are linear methods that aim to keep the low-dimensional representations of dissimilar data points far apart. However, for high-dimensional data that lies on or near a low-dimensional, non-linear manifold, it is generally more crucial to keep the low-dimensional representations of very similar data points close together, which is typically not achievable with linear mapping.

Numerous nonlinear dimensionality reduction techniques have been developed to preserve the local structure of data, with many of them reviewed by Lee and Verleysen [132]. Notable among these are: (1) Sammon mapping [133], (2) Curvilinear Components Analysis (CCA) [134], (3) Stochastic Neighbor Embedding (SNE) [135], (4) Isomap [136], (5) Maximum Variance Unfolding (MVU) [137], (6) Locally Linear Embedding (LLE) [138], and
Laplacian Eigenmaps [139]. While these methods perform well on synthetic data sets, they often struggle to effectively visualize real, high-dimensional data. Most of these techniques fail to simultaneously preserve both local and global structures within a single map. For example, a recent study indicates that even a semi-supervised version of MVU cannot successfully separate handwritten digits into distinct clusters.

The aim of this study is to differentiate chaotic and spiking data derived from the dynamic maps of optically injected quantum dot spin vertical-cavity surface-emitting lasers (QD spin VCSELs) using various dimension reduction (DR) methods. This will include performing spike sorting and subsequently applying artificial intelligence (AI) algorithms for evaluation. Dimension reduction algorithms are examined and compared, specifically t-SNE [140], UMAP [141], and PCA [130]. Each of these algorithms has unique limitations. For instance, t-SNE is highly sensitive to the perplexity parameter and often generates artificial clusters, while both t-SNE and UMAP excel in preserving local structures but struggle with global structure preservation [129], [142]. Principal Component Analysis, by contrast, is rooted in linear algebra and reduces dimensionality by projecting data onto a subspace that maximizes variance, making it less effective for capturing non-linear structures but more robust in maintaining global data topology.

UMAP is known for maintaining local neighborhood continuity, yet it faces challenges with global data structure. PCA, with its linear projection approach, is more effective for stable global structure representation but lacks the ability to capture complex, non-linear relationships in the data. [143]

Notably, none of t-SNE, UMAP, or PCA can smoothly transition from local to global structure preservation through simple parameter adjustments. Comparing these algorithms is inherently complex due to their differing loss functions and parameter spaces. For instance, t-SNE and UMAP derive their repulsive forces from distinct mechanisms, complicating direct comparisons.

Understanding the specific loss functions and their impact on embeddings is crucial for improving these algorithms. The loss function dictates the attractive and repulsive forces between data points, influencing the resulting embeddings.

This study further involves spike sorting, an essential process in analyzing these type of data. By employing these DR methods, we aim to enhance the accuracy of spike sorting and subsequently use AI algorithms to evaluate and refine the sorted spikes. This comprehensive approach not only helps in distinguishing chaotic from spiking data but also improves the overall analysis through advanced AI techniques.

4.4.1 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimension reduction technique that maps high-dimensional data into a lower-dimensional space by preserving the pairwise similarities between data points. The technique is a variation of Stochastic Neighbor Embedding [135] that is much easier to optimize, and produces significantly better visualizations by reducing the tendency to crowd points together in the center of the map. The following subsections describe the steps involved in the t-SNE algorithm, its mathematical foundations, and the optimization process.

Computation of Pairwise Similarities

First, pairwise similarities between data points in the high-dimensional space are computed using a Gaussian kernel, which measures similarity based on the Euclidean distance between data points. For data points \( x_i \) and \( x_j \), the similarity \( p_{ij} \) is calculated as:

\[
p_{ij} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k\neq i} \exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)}
\]

where \( \sigma_i \) is a variance parameter for the Gaussian kernel centered at \( x_i \).

Construction of Probability Distributions

From the pairwise similarities, probability distributions are constructed for each data point. The distributions represent the similarities between a data point and all other points in the high-dimensional space. The similarities are converted into probabilities using the softmax function.
Initialization of the Embedding

An initial embedding of the data points in the low-dimensional space is randomly generated. Each data point is assigned a position in the low-dimensional space.

Computation of Similarity in Low-Dimensional Space

Similar to the high-dimensional space, pairwise similarities between data points are computed in the low-dimensional space. However, in the low-dimensional space, a Student’s t-distribution is used to model the pairwise similarities. For data points \( y_i \) and \( y_j \) in the low-dimensional space, the similarity \( q_{ij} \) is given by:

\[
q_{ij} = \frac{(1 + \| y_i - y_j \|^2)^{-1}}{\sum_{k \neq l} (1 + \| y_k - y_l \|^2)^{-1}} \tag{4.6}
\]

Optimization of the Embedding

The goal of t-SNE is to minimize the divergence between the pairwise similarities in the high-dimensional space \( p_{ij} \) and the low-dimensional space \( q_{ij} \). This is achieved through an iterative optimization process that minimizes the Kullback-Leibler (KL) divergence between the two distributions:

\[
KL(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{4.7}
\]

The positions of the data points in the low-dimensional space are adjusted iteratively using gradient descent to minimize the KL divergence.

Algorithm 1 Optimizing the embedding in t-SNE

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>function OptimizeEmbedding(P, Q, Y, n_epochs)</td>
</tr>
<tr>
<td>2</td>
<td>Initialize ( \alpha ) (learning rate)</td>
</tr>
<tr>
<td>3</td>
<td>for ( e \leftarrow 1, \ldots, n_epochs ) do</td>
</tr>
<tr>
<td>4</td>
<td>for each data point ( y_i ) do</td>
</tr>
<tr>
<td>5</td>
<td>Compute gradient ( \frac{\partial KL}{\partial y_i} )</td>
</tr>
<tr>
<td>6</td>
<td>Update ( y_i \leftarrow y_i - \alpha \cdot \frac{\partial KL}{\partial y_i} )</td>
</tr>
<tr>
<td>7</td>
<td>end for</td>
</tr>
<tr>
<td>8</td>
<td>( \alpha \leftarrow \alpha \cdot 0.99 ) [Decay learning rate]</td>
</tr>
<tr>
<td>9</td>
<td>end for</td>
</tr>
<tr>
<td>10</td>
<td>return ( Y )</td>
</tr>
</tbody>
</table>

Convergence and Visualization

The optimization process continues until a stopping criterion is met, such as a maximum number of iterations or a threshold for the change in the embedding positions. Once the optimization is complete, the final low-dimensional embedding can be visualized. Data points are plotted in the low-dimensional space, where their positions are determined by the optimized embedding. The visualization aims to represent the similarities and relationships between data points in a way that preserves the local and global structure of the high-dimensional data.

In summary, t-SNE effectively captures both local and global structures within the data through a series of carefully designed steps, integrating mathematical rigor with computational efficiency. The optimization step, particularly through the minimization of KL divergence, ensures that the embedding faithfully represents the data structure.

### 4.4.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a fundamental technique for dimensionality reduction that transforms a dataset into a set of orthogonal components, capturing the maximum variance in the data. The following outlines the key steps in the PCA algorithm:
Algorithm 2: PCA Algorithm [144]

1: **Input:** Data matrix $X \in \mathbb{R}^{n \times p}$, number of principal components $k$
2: **Output:** Transformed data matrix $Y \in \mathbb{R}^{n \times k}$
3: **Step 1: Data Standardization**
4: This step involves centering the data by subtracting the mean of each feature from the data points, resulting in a mean-centered dataset.
5: $X_{\text{centered}} \leftarrow X - \text{mean}(X)$
6: **Step 2: Covariance Matrix Computation**
7: Here, we calculate the covariance matrix to understand the variance and the relationship between different features in the dataset.
8: $\Sigma \leftarrow \frac{1}{n-1}X_{\text{centered}}^T X_{\text{centered}}$
9: **Step 3: Eigen Decomposition**
10: This step involves performing eigen decomposition on the covariance matrix to obtain eigenvalues and eigenvectors, which indicate the principal components of the data.
11: $\Lambda, V \leftarrow \text{eig}(\Sigma)$
12: **Step 4: Sorting Eigenvalues and Eigenvectors**
13: The eigenvalues and their corresponding eigenvectors are sorted in descending order of the eigenvalues to identify the most significant principal components.
14: Sort $\Lambda$ in descending order and arrange $V$ accordingly
15: **Step 5: Selection of Top $k$ Eigenvectors**
16: We select the top $k$ eigenvectors corresponding to the largest $k$ eigenvalues to form the transformation matrix.
17: $W \leftarrow \{v_1, v_2, \ldots, v_k\}$
18: **Step 6: Data Transformation**
19: Finally, the original mean-centered data is projected onto the new subspace using the selected eigenvectors, resulting in a reduced dimensional representation of the data.
20: $Y \leftarrow X_{\text{centered}}W$
21: **return** $Y$

In the PCA algorithm, the initial step involves standardizing the data by centering it around the mean. This is followed by the computation of the covariance matrix, which captures the relationships between different dimensions in the data. The eigen decomposition of the covariance matrix is performed to obtain the eigenvalues and eigenvectors. These eigenvectors are then sorted based on their corresponding eigenvalues in descending order, and the top $k$ eigenvectors are selected to form a transformation matrix. Finally, the original data is projected onto the new subspace defined by the selected eigenvectors, resulting in a reduced dimensional representation of the data.

This procedure efficiently captures the most significant variance in the data while reducing the dimensionality, making it a powerful tool for data analysis and visualization.

**Mathematical Formulation:**

1. **Data Standardization:**
   \[ X_{\text{centered}} = X - \text{mean}(X) \]  \hfill (4.8)

2. **Covariance Matrix Computation:**
   \[ \Sigma = \frac{1}{n-1}X_{\text{centered}}^T X_{\text{centered}} \]  \hfill (4.9)

3. **Eigen Decomposition:**
   \[ \Lambda, V = \text{eig}(\Sigma) \]  \hfill (4.10)

4. **Data Transformation:**
   \[ Y = X_{\text{centered}}W \]  \hfill (4.11)

The PCA algorithm provides a linear transformation that maximizes variance, making it ideal for reducing dimensions while preserving as much variability in the data as possible.
4.4.3 Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP)

UMAP (Uniform Manifold Approximation and Projection) is a straightforward and effective method for dimensionality reduction. The process begins with constructing a weighted k-nearest neighbor graph and is followed by optimizing the graph layout. The algorithms involved in this process are detailed below, including the construction of local fuzzy simplicial sets, spectral embedding, and optimization of the embedding with respect to fuzzy set cross entropy.

Algorithm 3 UMAP algorithm [141]

```plaintext
1: function UMAP(X, n, d, min_dist, n_epochs)
2: # Construct the relevant weighted graph
3: for all x ∈ X do
4:    fs_set[x] ← LocalFuzzySimplicialSet(X, x, n)
5: end for
6: top_rep ← ∪x∈X fs_set[x] (The probabilistic t-conorm is recommended)
7: # Perform optimization of the graph layout
8: Y ← SpectralEmbedding(top_rep, d)
9: Y ← OptimizeEmbedding(top_rep, Y, min_dist, n_epochs)
10: return Y
```

The inputs to Algorithm 3 are the dataset X, the neighborhood size n, the target dimension d, the minimum distance parameter min_dist, and the number of optimization epochs n_epochs.

Construction of a Weighted k-Nearest Neighbor Graph

The first phase of UMAP involves constructing a weighted k-nearest neighbor graph. For each data point x, the algorithm computes a local fuzzy simplicial set by defining a probability distribution over its neighbors. This process is mathematically represented as:

\[ \rho_i = \min \{d(x_i, x_j) \mid 1 \leq j \leq k, d(x_i, x_j) > 0\} \]  \hspace{1cm} (4.12)

The value of \( \sigma_i \) is chosen such that:

\[ \sum_{j=1}^{k} \exp \left( -\max\{0, d(x_i, x_j) - \rho_i\} \right) = \log_2(k). \]  \hspace{1cm} (4.13)

The choice of \( \rho_i \) ensures that \( x_i \) connects to at least one other data point with an edge of weight 1, making the resulting fuzzy simplicial set locally connected at \( x_i \). This substantially enhances the representation of high-dimensional data, where other algorithms like t-SNE suffer from the curse of dimensionality.

Next, a weighted directed graph is defined \( \overrightarrow{G} = (V, E, w) \). The vertices \( V \) of \( \overrightarrow{G} \) are simply the set \( X \). The set of directed edges \( E \) is defined as:

\[ E = \{(x_i, x_j) \mid 1 \leq j \leq k, 1 \leq i \leq N\} \]  \hspace{1cm} (4.14)

The weight function \( w \) is defined by:

\[ w((x_i, x_j)) = \exp \left( -\max\{0, d(x_i, x_j) - \rho_i\} \right) \frac{1}{\sigma_i} \]  \hspace{1cm} (4.15)

For a specific point \( x_i \), an induced graph consisting of \( x_i \) and its outgoing edges is formed. This graph represents the 1-skeleton of the fuzzy simplicial set linked to the local metric space around \( x_i \), which is characterized by \( \rho_i \) and \( \sigma_i \). The edge weight indicates the membership strength of the corresponding 1-simplex within the fuzzy simplicial set.

The local fuzzy simplicial set construction is detailed in Algorithm 4.
Algorithm 4 Constructing a local fuzzy simplicial set [141]

1: function LocalFuzzySimplicialSet(\(X, x, n\))
2: \(knn, knn\text{-}dists \leftarrow \text{ApproxNearestNeighbors}(X, x, n)\)
3: \(\rho \leftarrow \text{knn-dist}[1]\) \{Distance to nearest neighbor\}
4: \(\sigma \leftarrow \text{SmoothKNNDist}(knn\text{-}dists, n, \rho)\) \{Smooth approximator to knn-distance\}
5: \(\text{fs\text{-}set}_0 \leftarrow X\)
6: \(\text{fs\text{-}set}_1 \leftarrow \{([x, y], 0) \mid y \in X\}\)
7: for all \(y \in knn\) do
8: \(d_{x,y} \leftarrow \max(0, \text{dist}(x, y) - \rho)/\sigma\)
9: \(\text{fs\text{-}set}_1 \leftarrow \text{fs\text{-}set}_1 \cup \{([x, y], \exp(-d_{x,y}))\}\)
10: end for
11: return \(\text{fs\text{-}set}\)

The normalizing factor \(\sigma\) is computed to ensure the fuzzy simplicial set’s cardinality is fixed. This is achieved through a smooth approximation of the k-nearest neighbor distance, as shown in Algorithm 5.

Algorithm 5 Compute the normalizing factor for distances \(\sigma\) [141]

1: function SmoothKNNDist(knn\text{-}dists, \(n\), \(\rho\))
2: Binary search for \(\sigma\) such that \(\sum_{i=1}^{n} \exp(-(\text{knn\text{-}dists}_i - \rho)/\sigma) = \log_2(\text{length}(\text{knn\text{-}dists}))\)
3: return \(\sigma\)

Optimization of the Low-Dimensional Representation

After constructing the weighted k-nearest neighbor graph, UMAP optimizes the low-dimensional representation by minimizing a cross-entropy loss function. This loss function includes both attractive and repulsive forces, allowing UMAP to preserve both local and global data structures effectively.

The optimization process involves the following steps:

Spectral Initialization

U Spectral embedding initializes the layout by considering the 1-skeleton of the global fuzzy topological representation as a weighted graph. The symmetric normalized Laplacian is then used for the embedding, as described in Algorithm 6.

Algorithm 6 Spectral embedding for initialization [141]

1: function SpectralEmbedding(top\_rep, \(d\))
2: \(A \leftarrow 1\text{-}skeleton\) of top\_rep expressed as a weighted adjacency matrix
3: \(D \leftarrow \text{degree matrix for the graph } A\)
4: \(L \leftarrow D^{1/2}(D - A)D^{1/2}\)
5: \(\text{evec} \leftarrow \text{Eigenvectors of } L\) (sorted)
6: \(Y \leftarrow \text{evec}[1\ldots d+1]\) \{0-base indexing assumed\}
7: return \(Y\)

Optimization of the Embedding

The final step in the UMAP algorithm involves optimizing the embedding by minimizing the fuzzy set cross entropy. This process uses a sampling-based approach for optimization and employs stochastic gradient descent to accelerate convergence.

The fuzzy set cross entropy, with respect to given membership functions \(\mu\) and \(\nu\), is given by:

\[
C((A, \mu), (A, \nu)) = \sum_{a \in A} \mu(a) \log \left(\frac{\mu(a)}{\nu(a)}\right) + (1 - \mu(a)) \log \left(\frac{1 - \mu(a)}{1 - \nu(a)}\right). \tag{4.16}
\]
The first sum depends only on $\mu$, which takes fixed values during the optimization. Therefore, the minimization of cross entropy depends only on the second sum, leading to:

$$
\min - \sum_{a \in A} (\mu(a) \log(\nu(a)) + (1 - \mu(a)) \log(1 - \nu(a))).
$$  \hspace{1cm} (4.17)

Algorithm 7 outlines the optimization process, which balances the local and global structures within the data by iteratively refining the low-dimensional representation.

**Algorithm 7** Optimizing the embedding [141]

1: function OptimizeEmbedding($top\_rep, Y, min\_dist, n\_epochs$)
2: $\alpha \leftarrow 1.0$
3: Fit $\Phi$ from $\Psi$ defined by $min\_dist$
4: for $e \leftarrow 1, \ldots, n\_epochs$ do
5:   for all $([a, b], p) \in top\_rep_i$ do
6:     if Random() $\leq p$ then
7:       #Sample simplex with probability $p$
8:       $y_a \leftarrow y_a + \alpha \cdot \nabla(\log(\Phi))(y_a, y_b)$
9:     for $i \leftarrow 1, \ldots, n\_neg\_samples \ do$
10:        $c \leftarrow$ random sample from $Y$
11:        $y_a \leftarrow y_a + \alpha \cdot \nabla((1 - \Phi))(y_a, y_c)$
12:   end for
13: end if
14: end for
15: $\alpha \leftarrow 1.0 - e/n\_epochs$
16: end for
17: return $Y$

In this optimization step, stochastic gradient descent is employed to minimize the cross entropy between the fuzzy sets. This approach ensures that the embedding faithfully represents the data structure. The use of stochastic gradient descent accelerates convergence, providing an efficient solution for large datasets.

The optimization balances local and global structures within the data by iteratively refining the low-dimensional representation. Sampling is employed to approximate this optimization process, where 1-simplices are sampled with probability $\mu(a)$ and updated according to the value of $\nu(a)$. Negative sampling is used to handle the term $(1 - \mu(a)) \log(1 - \nu(a))$ by assuming randomly sampled potential 1-simplices as negative examples and updating according to the value of $1 - \nu(a)$.

By iteratively refining the embedding through these steps, UMAP effectively captures both local and global structures within the data, providing a comprehensive representation in a lower-dimensional space.

**Hyper-Parameters of the UMAP algorithm**

The key parameters in the UMAP algorithm are as follows:

- $n$: This parameter specifies the number of neighbors used to approximate the local metric.
- $d$: This denotes the target dimension for the embedding.
- $min\_dist$: This determines the desired minimum distance between points that are close in the embedding space.
- $n\_epochs$: This represents the number of epochs for training to optimize the low-dimensional representation.

While the impacts of parameters $d$ and $n\_epochs$ are quite intuitive and straightforward, the influences of $n$ (number of neighbors) and $min\_dist$ require a more detailed examination.

The parameter $n$ can be viewed as the local scale for approximating the manifold, effectively treating the manifold as approximately flat by averaging over the $n$ neighbors. Features of the manifold that exist at a smaller scale than the $n$ nearest neighbors will be omitted, whereas features that require a larger scale than $n$ to be
discerned may not be effectively detected. Therefore, \( n \) represents a trade-off between capturing fine-grained
details and large-scale manifold features. Smaller values of \( n \) enable the capture of detailed manifold structures,
albeit at the cost of a broader understanding of the manifold. Conversely, larger values capture extensive man-
ifold structures while averaging out finer details. When \( n \) is small, the manifold may break into several small,
connected components, which necessitates careful handling during the spectral embedding initialization.

Conversely, the hyperparameter \( \text{min} - \text{dist} \) directly influences the output by controlling the construction of
the fuzzy simplicial set from the low-dimensional representation. It substitutes for the distance to the nearest
neighbor, which is crucial for maintaining local connectivity. Essentially, this parameter determines the degree
of proximity among points in the low-dimensional space. Lower values of \( \text{min} - \text{dist} \) allow points to be closely
packed, which more accurately preserves the manifold structure. However, increasing the \( \text{min} - \text{dist} \) value
forces the embedding to disperse points further apart, which aids in visualization and helps to avoid potential
overplotting issues. Thus, \( \text{min} - \text{dist} \) is considered primarily an aesthetic parameter, crucial for determining
the visual appearance of the embedding and is particularly significant when using UMAP for visualization
purposes.

In subsequent sections, we will present the results of applying UMAP to our data with various hyperparameter
settings to demonstrate their effects.

### 4.5 CLUSTERING METHODOLOGY

The methodology for clustering chaotic and spiking time series data involves the application of three different
clustering algorithms: K-means, DBSCAN, and Agglomerative Clustering. Each of these algorithms has been
selected based on their unique characteristics and advantages, which are particularly suited for the nature of our
datasets. In this section, we provide a detailed explanation of these algorithms, the rationale for their selection,
and the parameter optimization process employed to enhance clustering performance.

#### 4.5.1 K-means Clustering

K-means clustering is one of the most widely used algorithms due to its simplicity and efficiency. It partitions the
data into \( k \) clusters, where each data point belongs to the cluster with the nearest mean, serving as a prototype of
the cluster. The algorithm aims to minimize the within-cluster variance, making it effective for datasets where
clusters are approximately spherical and equally sized. Given our feature space, which includes skewness,
kurtosis, and power spectral density (PSD) peaks, K-means helps in efficiently grouping the data based on these
statistical properties. The required steps for K-means clustering implementation are demonstrated by Algorithm
8. The selection of the optimal number of clusters, \( k \), is crucial and was determined using the Elbow method
and the Silhouette Score analysis [145, 146].

**Algorithm 8 K-means Clustering Algorithm**

1. **Input:** Data points \( X = \{x_1, x_2, \ldots, x_n\} \), number of clusters \( k \), maximum iterations \( \text{max\_iter} \)
2. **Output:** Cluster assignments \( C = \{c_1, c_2, \ldots, c_n\} \), cluster centroids \( \mu = \{\mu_1, \mu_2, \ldots, \mu_k\} \)
3. Initialize centroids \( \mu_1, \mu_2, \ldots, \mu_k \) randomly from the data points
4. repeat
5. for each data point \( x_i \) do
6. Assign \( x_i \) to the nearest centroid \( \mu_j \)
7. \( c_i \leftarrow \arg \min_j \|x_i - \mu_j\|^2 \)
8. end for
9. for each centroid \( \mu_j \) do
10. Update centroid \( \mu_j \) to the mean of all points assigned to cluster \( j \)
11. \( \mu_j \leftarrow \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i \)
12. end for
13. Compute the total within-cluster variance
14. \( \text{WCV} \leftarrow \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 \)
15. until convergence or maximum iterations reached
16. return Cluster assignments \( C \), cluster centroids \( \mu \)
4.5.2 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN (Algorithm 9) is a density-based clustering algorithm that identifies clusters as areas of high density separated by areas of low density. Unlike K-means, DBSCAN does not require the number of clusters to be specified a priori and is capable of identifying arbitrarily shaped clusters and handling noise effectively. This is particularly advantageous for our dataset, where the chaotic nature of the time series can lead to clusters with irregular shapes and varying densities. The parameters for DBSCAN, namely the minimum number of points (\(\text{minPts}\)) and the neighborhood radius (\(\epsilon\)), were optimized using a grid search approach to maximize the clustering performance based on the silhouette score and noise handling [147, 148].

Algorithm 9 DBSCAN Clustering Algorithm

```plaintext
1: Input: Data points \(X = \{x_1, x_2, \ldots, x_n\}\), neighborhood radius \(\epsilon\), minimum points \(\text{minPts}\)
2: Output: Cluster assignments \(C = \{c_1, c_2, \ldots, c_n\}\)
3: Initialize all points as unvisited
4: Initialize cluster index \(k \leftarrow 0\)
5: for each unvisited point \(x_i\) do
6: Mark \(x_i\) as visited
7: Retrieve \(\epsilon\)-neighborhood \(N_\epsilon(x_i)\)
8: if size of \(N_\epsilon(x_i)\) < \(\text{minPts}\) then
9: Mark \(x_i\) as noise
10: else
11: Increment cluster index \(k \leftarrow k + 1\)
12: Expand cluster \(C_k\) with \(x_i\) and \(N_\epsilon(x_i)\)
13: for each point \(x_j\) in \(N_\epsilon(x_i)\) do
14: if \(x_j\) is unvisited then
15: Mark \(x_j\) as visited
16: Retrieve \(\epsilon\)-neighborhood \(N_\epsilon(x_j)\)
17: if size of \(N_\epsilon(x_j)\) >= \(\text{minPts}\) then
18: Add points in \(N_\epsilon(x_j)\) to \(N_\epsilon(x_i)\)
19: end if
20: end if
21: if \(x_j\) is not yet assigned to a cluster then
22: Assign \(x_j\) to cluster \(C_k\)
23: end if
24: end for
25: end if
26: end for
27: return Cluster assignments \(C\)
```

4.5.3 Agglomerative Clustering

Agglomerative clustering, a type of hierarchical clustering, builds nested clusters by progressively merging or splitting them based on a chosen linkage criterion. For our study, we used Ward’s linkage method, which minimizes the variance of the clusters being merged, thereby producing more compact and spherical clusters. Agglomerative Clustering is particularly useful for its dendrogram, which provides a visual representation of the cluster hierarchy, offering insights into the nested structure of the data. This method complements K-means and DBSCAN by revealing potential sub-clusters within the data [149]. The discrete steps of agglomerative clustering are provided by Algorithm 10.
Algorithm 10 Agglomerative Clustering Algorithm

1: **Input:** Data points $X = \{x_1, x_2, \ldots, x_n\}$
2: **Output:** Dendrogram representing the cluster hierarchy
3: Initialize each data point as a singleton cluster
4: Compute the distance matrix $D$
5: while number of clusters > 1 do
6: Find the pair of clusters $(C_i, C_j)$ with the smallest distance in $D$
7: Merge clusters $C_i$ and $C_j$ to form a new cluster $C_{ij}$
8: Update the distance matrix $D$ using Ward’s linkage criterion
9: Remove entries for $C_i$ and $C_j$ from $D$ and add entry for $C_{ij}$
10: end while
11: Construct dendrogram from the sequence of merge operations
12: return Dendrogram

4.5.4 Parameter Optimization

The effectiveness of clustering algorithms heavily depends on the appropriate setting of their parameters. For K-means, the Elbow method, Silhouette Score, and Davies-Bouldin Index were used to determine the optimal number of clusters. In DBSCAN, $\epsilon$ and $\text{minPts}$ were optimized using grid search, where various combinations of these parameters were evaluated to achieve the highest silhouette score and lowest Davies-Bouldin Index, balancing cluster density and noise. For agglomerative clustering, the linkage criterion was chosen based on its ability to minimize intra-cluster variance, and the dendrogram was analyzed to select a reasonable cut-off point for the number of clusters. Additionally, UMAP parameters such as the number of components, minimum distance, and the number of neighbors were optimized using grid search to achieve the best data visualization and separation.

Evaluation Metrics

- The Elbow Method which is used to determine the optimal number of clusters in K-means by plotting the within-cluster sum of squares (WCSS) against the number of clusters $k$. The 'elbow' point, where the rate of decrease sharply slows down, indicates the optimal $k$. This method helps in balancing the trade-off between reducing WCSS and avoiding overfitting with too many clusters.
- The Silhouette Score which measures how similar an object is to its own cluster compared to other clusters. It ranges from -1 to 1, where a high value indicates well-separated clusters. The silhouette score $s(i)$ for a data point $i$ is calculated as:

$$ s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} $$  \hspace{1cm} (4.18)

where $a(i)$ is the average distance from $i$ to the other points in the same cluster, and $b(i)$ is the minimum average distance from $i$ to points in a different cluster. This score was used to evaluate the clustering performance across different parameter settings for K-means, DBSCAN, and UMAP.
- The Davies-Bouldin Index (DBI) which evaluates the average similarity ratio of each cluster with its most similar cluster. A lower DBI indicates better clustering. It is calculated as:

$$ DBI = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left( \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) $$  \hspace{1cm} (4.19)

where $\sigma_i$ is the average distance of all points in cluster $i$ to its centroid, and $d(c_i, c_j)$ is the distance between centroids $c_i$ and $c_j$. This index was used to optimize parameters for K-means and DBSCAN.
- Dendrogram Analysis which provides a visual representation of the cluster hierarchy. By analyzing the dendrogram, we can select a cut-off point that defines a reasonable number of clusters. The linkage criterion, particularly Ward’s method, ensures that the variance within each cluster is minimized.

4.5.5 Comparative Analysis

The selection of these three clustering algorithms allows for a comprehensive analysis of the data from different perspectives. K-means provides a fast and efficient way to partition the data into a predefined number of clus-
ters. DBSCAN offers robustness against noise and the ability to discover clusters of arbitrary shapes, which is particularly useful for identifying outliers in chaotic time series. Agglomerative Clustering, with its hierarchical approach, helps in understanding the nested structure of the data, making it possible to identify sub-clusters and their relationships. By combining these methods, we ensure a robust and thorough clustering analysis, capturing the complex characteristics of chaotic and spiking time series.

4.6 Evaluation of Clustering Performance

The metrics that where used to evaluate which of the three clustering methods succeeds in capturing the true structure and purity of the clusters of the data are presented in the following points:

- **Adjusted Rand Index (ARI)** that calculates the similarity between the true labels and the predicted clustering, adjusted for chance. It ranges from -1 to 1, where 1 indicates perfect agreement, 0 indicates random clustering, and negative values indicate poor clustering [150].

  \[ \text{ARI} = \frac{\text{Index} - \text{Expected Index}}{\text{Max Index} - \text{Expected Index}} \]  

  (4.20)

- **Normalized Mutual Information (NMI)** which measures the mutual dependence between the true labels and the predicted clustering. It ranges from 0 to 1, where 1 indicates perfect agreement [151].

  \[ \text{NMI}(U,V) = \frac{2 \cdot I(U;V)}{H(U) + H(V)} \]  

  (4.21)

- **Silhouette Score** that measures how similar an object is to its own cluster compared to other clusters. It ranges from -1 to 1, where 1 indicates that the object is well-matched to its own cluster and poorly matched to neighboring clusters [152].

  \[ s(i) = \frac{b(i) - a(i)}{\max(a(i),b(i))} \]  

  (4.22)

- **Confusion Matrix** which is a table used to describe the performance of a classification model. For clustering, it compares the true labels to the predicted labels.

- **Accuracy** which is defined as the ratio of correctly predicted instances to the total instances. For clustering, it can be derived from the confusion matrix.

  \[ \text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \]  

  (4.23)

- **Fowlkes-Mallows Index (FMI)** that measures the similarity between clusters in terms of precision and recall [153].

  \[ \text{FMI} = \sqrt{\frac{\text{TP}}{\text{TP} + \text{FP}}} \cdot \sqrt{\frac{\text{TP}}{\text{TP} + \text{FN}}} \]  

  (4.24)

- **Purity Score** which measures the extent to which clusters contain a single class [154].

  \[ \text{Purity} = \frac{1}{N} \max \sum_{k,j} |C_k \cap L_j| \]  

  (4.25)
<table>
<thead>
<tr>
<th>Technique</th>
<th>Principal Component Analysis (PCA) [130]</th>
<th>t-Distributed Stochastic Neighbor Embedding (t-SNE) [140]</th>
<th>Uniform Manifold Approximation and Projection (UMAP) [141]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Linear dimensionality reduction</td>
<td>Non-linear dimensionality reduction for visualization</td>
<td>Non-linear dimensionality reduction for visualization and manifold learning</td>
</tr>
<tr>
<td>Core Idea</td>
<td>Projects data onto a lower-dimensional linear subspace</td>
<td>Minimizes divergence between pairwise probabilities in high and low dimensions</td>
<td>Preserves both local and global data structure by constructing a fuzzy topological representation</td>
</tr>
<tr>
<td>Mathematics</td>
<td>Eigenvalue decomposition of the covariance matrix</td>
<td>Uses Kullback-Leibler divergence to minimize the difference between joint probabilities</td>
<td>Uses Riemannian geometry and fuzzy simplicial sets to model data relationships</td>
</tr>
<tr>
<td>Scalability</td>
<td>High, efficient for large datasets</td>
<td>Medium, computationally intensive and scales quadratically</td>
<td>High, efficient and scales linearly</td>
</tr>
<tr>
<td>Speed</td>
<td>Fast, can be performed quickly on large datasets</td>
<td>Slow, due to the complexity of the optimization process</td>
<td>Fast, due to efficient algorithms for constructing and optimizing the fuzzy simplicial set</td>
</tr>
<tr>
<td>Parameters</td>
<td>Number of components to retain</td>
<td>Perplexity, learning rate</td>
<td>Number of neighbors, minimum distance between points, number of epochs</td>
</tr>
<tr>
<td>Interpretability</td>
<td>High, provides loadings for each component and the amount of variance explained</td>
<td>Medium, results can be sensitive to parameter choices and may create spurious clusters</td>
<td>Medium to High, good balance of local and global structure, but results can vary based on parameters</td>
</tr>
<tr>
<td>Output</td>
<td>Orthogonal linear components</td>
<td>Low-dimensional embedding where similar objects are modeled by nearby points</td>
<td>Low-dimensional embedding preserving the topological structure of the data</td>
</tr>
<tr>
<td>Loss Function</td>
<td>Maximizes variance in the data</td>
<td>Uses Kullback-Leibler divergence</td>
<td>Uses cross-entropy which includes both attractive and repulsive forces</td>
</tr>
<tr>
<td>Optimization Method</td>
<td>Eigenvalue decomposition</td>
<td>Gradient descent</td>
<td>Stochastic gradient descent</td>
</tr>
<tr>
<td>Initialization</td>
<td>N/A</td>
<td>Random initialization of low-dimensional points</td>
<td>Spectral initialization using Laplacian Eigenmaps, providing faster convergence and greater consistency</td>
</tr>
<tr>
<td>Advantages</td>
<td>Simple, interpretable, fast, retains global structure</td>
<td>Captures complex structures, good for detailed cluster visualization</td>
<td>Preserves local and global structure, scalable, flexible, and fast</td>
</tr>
<tr>
<td>Disadvantages</td>
<td>Only captures linear relationships, may not capture complex structures</td>
<td>Computationally expensive, can create misleading clusters, does not preserve global structure</td>
<td>Parameter-sensitive, may require careful tuning, interpretability can vary</td>
</tr>
</tbody>
</table>
5. EXPERIMENTS AND RESULTS

5.1 ANALYSIS OF THE POWER SPECTRAL DENSITY OF THE DATA

![Figure 5.1: Representation of Power Spectral Density of spiking time series data.]

<table>
<thead>
<tr>
<th>Metrics for Widths</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean difference</td>
<td>-0.00</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.00</td>
</tr>
<tr>
<td>Range</td>
<td>0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics for Prominences</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean difference</td>
<td>-557.94</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>452.74</td>
</tr>
<tr>
<td>Range</td>
<td>1000.15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Found 6 Peaks:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak 1 Location = 0.38 Hz Width = 0.01 Hz Prominence = 1939.32</td>
<td></td>
</tr>
<tr>
<td>Peak 2 Location = 0.75 Hz Width = 0.02 Hz Prominence = 2338.41</td>
<td></td>
</tr>
<tr>
<td>Peak 3 Location = 1.13 Hz Width = 0.01 Hz Prominence = 2162.23</td>
<td></td>
</tr>
<tr>
<td>Peak 4 Location = 1.51 Hz Width = 0.01 Hz Prominence = 1551.17</td>
<td></td>
</tr>
<tr>
<td>Peak 5 Location = 1.89 Hz Width = 0.01 Hz Prominence = 374.84</td>
<td></td>
</tr>
<tr>
<td>Peak 6 Location = 2.26 Hz Width = 0.01 Hz Prominence = 106.66</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Metrics for widths and prominences of the peaks for spiking time series.

In the analysis of the PSD of spikes, it is observed that the prominence of the peaks consistently decreases, highlighting only the initial peak in each experiment. This consistent decrease suggests a structured and predictable behavior in the spiking regime, where the primary peak is the dominant spectral feature and subsequent peaks diminish in prominence.

Conversely, in the PSD of chaotic regimes, the prominence of the peaks does not adhere to a specific pattern. This irregularity is characteristic of chaotic systems, where the energy is distributed across a broad range of
Figure 5.2: Representation of Power Spectral Density of spiking time series data.

<table>
<thead>
<tr>
<th>Metrics for Widths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean difference</td>
</tr>
<tr>
<td>Standard deviation</td>
</tr>
<tr>
<td>Range</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics for Prominences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean difference</td>
</tr>
<tr>
<td>Standard deviation</td>
</tr>
<tr>
<td>Range</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Found 4 Peaks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak 1</td>
</tr>
<tr>
<td>Peak 2</td>
</tr>
<tr>
<td>Peak 3</td>
</tr>
<tr>
<td>Peak 4</td>
</tr>
</tbody>
</table>

Table 5.2: Metrics for widths and prominences of the peaks.

...
prominence can indicate overall trends in peak heights. A small mean difference, as observed in the chaotic regime, suggests that there are no consistently large or small peaks, which aligns with the unpredictable nature of chaos. In spiking regimes, the mean difference is typically more pronounced between the first peak and subsequent peaks, reflecting the dominance of the primary peak.

The analysis of peak prominence, width, and statistical metrics such as standard deviation and mean difference provides a comprehensive understanding of the spectral characteristics of spiking and chaotic regimes. In spiking regimes, the prominence consistently decreases, peaks are narrower, standard deviation is low, and the mean difference is pronounced between the primary and subsequent peaks. This reflects the periodic behavior of the system. In chaotic regimes, the prominence does not follow a specific pattern, peak widths are variable, standard deviation is high, and the mean difference is small, indicating a broad and complex spectral landscape.

Moreover, in the PSD of spikes, the peaks tend to be narrower, reflecting the concentrated energy at specific frequencies. In chaotic regimes, peaks often have varying widths, reflecting the spread of energy over a broader frequency range. This variability in width is consistent with the broader, more complex spectral content of chaotic signals.

In conclusion, the prominence of peaks in PSD analysis serves as a valuable indicator of the underlying dynamics of the system. The consistent decrease in peak prominence in spiking regimes reflects structured periodic behavior, whereas the lack of a specific pattern in chaotic regimes highlights the complex, unpredictable nature of chaos. These findings are corroborated by theoretical insights from nonlinear dynamics, reinforcing the utility of PSD analysis in characterizing and distinguishing different dynamical states in semiconductor lasers and other complex systems.
5.2 DIMENSIONALITY REDUCTION AND CLUSTER EVALUATION RESULTS

In this particular step of the implementation an extensive analysis was conducted using datasets containing spiking, chaotic, periodic, bursting spiking and chaotic spiking time series data obtained by different dynamic maps. The ES plus time series in our study case where the ones which represented the lasing output of the QD spin VCSEL.

In the first part of the dimensionality reduction study, PCA was applied to the data. The results, depicted in Figure 5.4, show that PCA was able to separate the spiking and chaotic data to some extent, though the clusters are not distinctly obvious. This suggests that while PCA provided a reasonable initial separation, it did not create clearly defined clusters. The features used for this analysis were [skewness_value, kurtosis_value, median_prominence, num_pks, range_width]. In general, from all the experiments conducted, these features were found to best divide the data for the ES-plus dataset. However, when PCA was applied to other time series datasets such as ES-minus, GS-plus, and GS-minus, it did not separate the data effectively. This indicates that PCA’s capability to distinguish between different types of data is limited in these cases, necessitating the exploration of more sophisticated dimensionality reduction techniques for better clustering performance.

In the further step of this study, we implemented the second dimensionality reduction method, t-SNE. Initially, we applied t-SNE to the ES-plus time series, using grid search to find the best parameters for t-SNE. The features used in the first figure were skewness value, kurtosis value, median of prominence, number of peaks, and range of peaks width of the ES time series. Grid search was conducted for the best t-SNE parameters, including Perplexity and Learning Rate, based on the silhouette score. The best t-SNE parameters for this analysis were found to be Perplexity = 50 and Learning Rate = 500, with a best silhouette score of 0.90851. The resulting t-SNE projection is shown in Figure 5.5.

In the second experiment using t-SNE, the features were skewness_value, kurtosis_value, std_prominence, num_pks, and range_width, with the best t-SNE parameters found to be Perplexity = 5 and Learning Rate = 10, yielding a best silhouette score of 0.66396. The resulting t-SNE projection is displayed in Figure 5.6.

Next, the GS-minus time series were examined using the features skewness_value, kurtosis_value, std_prominence, num_pks, and range_width. The best t-SNE parameters were Perplexity = 20 and Learning Rate = 100, with a best silhouette score of 0.84681. The t-SNE projection for this analysis is shown in Figure 5.7.
Lastly, the GS-minus sub-dataset was again examined, but this time with features skewness_value, kurtosis_value, median_prominence, num_pks, and range_width. The best t-SNE parameters were Perplexity = 20 and Learning Rate = 500, with a best silhouette score of 0.82758. The resulting t-SNE projection is illustrated in Figure 5.8.
Upon analyzing the resulting figures from applying t-SNE with various conditions each time, it is evident that there are some subtle clusters within the data. Particularly in the case of the GS-minus dataset, the t-SNE method reveals more distinct clusters compared to other datasets, although the clusters are still not highly distinct. This indicates that while t-SNE is effective in visualizing the structure within the data, further refinement and exploration of parameters may be necessary to achieve more distinct and interpretable clustering results.

As aforementioned in Chapter 4, UMAP has demonstrated superior performance compared to PCA and t-SNE in our experiments. Therefore, we continued our experiments with UMAP to leverage its ability to create clearer and more distinct clusters. This decision was driven by UMAP’s capacity to preserve more of the local and global structure of the data, which is crucial for achieving meaningful clustering outcomes.
In general, in the experiments following, the optimal parameters varied between the ES-plus and ES-minus time series sub-datasets. For the ES-plus dataset, after performing a grid search, the best UMAP parameters were found to be: n_neighbors set to 15 and min_dist set to 0.1. The UMAP reduction for ES-plus finished with a cost of 3.00 seconds and completed the basic reduction. In contrast, for the ES-minus dataset, the optimal UMAP parameters were determined to be: n_components set to 2, min_dist set to 0.1, n_neighbors set to 20, and metric set to 'euclidean'. K-means clustering was then applied with num_clusters set to 2 for both datasets in the first place. The clustering performance was evaluated using several metrics.

In following experiment, UMAP was used for with the parameters: n_neighbors set to 15, min_dist set to 0.1, and the metric set to 'euclidean'. K-means clustering was then applied with num_clusters set to 2. The features examined in the first experiment where skewness_value, kurtosis_value, standard_deviation_prominence, num_pks, and range_width. The clustering performance was evaluated using several metrics.
The Elbow method, shown in Figure 5.9, suggested that the optimal number of clusters is around $k = 2$, as adding more clusters beyond this point provides diminishing returns in reducing the within-cluster sum of squares (WCSS).

The UMAP projection of time series data with the best parameters is shown in Figure 5.10. The mean silhouette score for this projection was 0.75, and the Davies-Bouldin Index was 0.45, indicating well-separated and dense clusters.

Figure 5.11 shows the UMAP projection with K-means clustering applied. The clustering performance metrics are presented in Table 5.4.

The next experiment uses the same set of parameters and features as the previous one. However instead of taking into account standard deviation of prominence, the median prominence utilized on the feature vector.
Table 5.4: Clustering Performance Metrics and Parameters (UMAP 2 K-means) for skewness_value, kurtosis_value, std_prominence, num_pks, and range_width.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Silhouette Score</td>
<td>0.75</td>
<td>Good clustering quality with well-separated clusters</td>
</tr>
<tr>
<td>Adjusted Rand Index (ARI)</td>
<td>0.35</td>
<td>Moderate agreement with true labels</td>
</tr>
<tr>
<td>Normalized Mutual Information (NMI)</td>
<td>0.45</td>
<td>Moderate mutual dependence between clustering and true labels</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.60</td>
<td>60% of the data points correctly clustered</td>
</tr>
<tr>
<td>Fowlkes-Mallows Index (FMI)</td>
<td>0.5565</td>
<td>Moderate similarity between clusters and true labels</td>
</tr>
<tr>
<td>Purity</td>
<td>0.6716</td>
<td>Moderate homogeneity within clusters</td>
</tr>
</tbody>
</table>

Figure 5.12: Silhouette Plot for UMAP Reduced Data with Median Prominence (ES Plus)

Figure 5.13: Elbow Method for Optimal k (ES Plus, Median Prominence)
**Table 5.5: Clustering Performance Metrics and Parameters (UMAP 2 K-means, Median Prominence)**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Silhouette Score</td>
<td>0.60</td>
<td>Reasonably well-separated clusters</td>
</tr>
<tr>
<td>Adjusted Rand Index (ARI)</td>
<td>0.40</td>
<td>Moderate agreement with true labels</td>
</tr>
<tr>
<td>Normalized Mutual Information (NMI)</td>
<td>0.45</td>
<td>Moderate mutual dependence between clustering and true labels</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.50</td>
<td>50% of the data points correctly clustered</td>
</tr>
<tr>
<td>Fowlkes-Mallows Index (FMI)</td>
<td>0.55</td>
<td>Moderate similarity between clusters and true labels</td>
</tr>
<tr>
<td>Purity</td>
<td>0.68</td>
<td>Moderate homogeneity within clusters</td>
</tr>
</tbody>
</table>

The clustering performance metrics, presented in Table 5.5, show that using median prominence yields moderate results compared to using the standard deviation of prominence. The initial metrics suggest lower quality
clustering, and the UMAP projection seems less distinct.

The mean silhouette score of 0.60 indicates moderate clustering quality with reasonably well-separated clusters. The Adjusted Rand Index (ARI) of 0.40 suggests a moderate level of agreement with the true labels, while the Normalized Mutual Information (NMI) of 0.45 indicates moderate mutual dependence between the clustering and true labels. The accuracy of 0.50 shows that 50% of the data points were correctly clustered. The Fowlkes-Mallows Index (FMI) of 0.55 suggests moderate similarity between the clusters and the true labels, and a purity score of 0.68 indicates moderate homogeneity within the clusters.

Overall, these results highlight the effectiveness of using standard deviation of prominence as a feature in the UMAP when using and K-means clustering approach with 2 clusters, providing slightly better performance metrics compared to the use of the median of prominence.

Nonetheless, upon further analysis, specifically by examining the ES-minus data, which is more saturated than the ES-plus data, the clustering methods demonstrate improved performance. By analyzing the data top-down and plotting the time traces for each data point, we observed that the study actually reveals three distinct clusters. Additionally, the elbow method, which is used to determine the optimal number of clusters, already when analysing ES-plus 5.13 suggests splitting the data into three clusters.

The analysis revealed effective spike sorting, which was confirmed by the detailed clustering metrics and visualizations.

This detailed approach highlights the importance of focusing on specific subsets of data to uncover more accurate and meaningful clustering structures.

The UMAP projection result demonstrated by Figure 5.16 illustrates the effectiveness of the technique in separating different dynamic states of the VCSEL. The clusters correspond to different operational modes such as low-intensity spiking behaviour (Figure 5.19), spiking with bursting behavior (Figure 5.18), chaotic behavior (Figure 5.17 5.21), dense spiking (Figure 5.20) and chaotic spiking behaviour (Figure 5.21). The features extracted include skewness and kurtosis of the ES minus time series ES minus, median prominence of peaks of the PSD, total number of peaks and range of peak widths. The UMAP parameters used in this experiment were n_neighbors = 20, min_dist = 0.1, n_components = 2, and metric = euclidean.

![Figure 5.16: UMAP Projection results for ES minus time traces of the dataset, (min_dist=0.1, n_neighbors=20)](image-url)
Before implementing K-means clustering on the ES-minus time series for 3 clusters, we initially performed the same analysis on the ES-plus data to determine if the clustering metrics would improve comparing to the experiment where we used 2 clusters. The results showed an significant improvement in the metrics, indicating an average percentage improvement of approximately 45%. However, in the subsequent experiments with the ES-minus data, the clustering evaluation metrics were even better, demonstrating superior differentiation and clustering performance.

Figure 5.23 illustrates the comparison of three clustering methods: K-means, Hierarchical, and DBSCAN, applied to UMAP-reduced data from the ES Minus dataset, focusing on 2 clusters. The purpose of this comparison is to evaluate the effectiveness and performance of these clustering techniques in identifying distinct clusters within the dataset.

Each method has its own strengths and weaknesses, which can be seen in the clustering results.

The results depicted in the figure 5.23 demonstrate how each method groups the data points and highlight their respective performance in terms of cluster separation and noise handling.
Figure 5.19: Low intensity spiking behaviour observed mainly in ES Plus time trace ($k_{inj} = 18.3673, \Delta f = 5.38402$).

Figure 5.20: Dense spiking behaviour observed mainly on the ES plus time trace ($k_{inj} = 30.6122, \Delta f = 13.0769$).

Table 5.6: Clustering Performance Metrics Summary for ES Minus (2 Clusters)

<table>
<thead>
<tr>
<th>Metric</th>
<th>K-means</th>
<th>DBSCAN</th>
<th>Agglomerative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Silhouette Score</td>
<td>0.81</td>
<td>0.75</td>
<td>0.81</td>
</tr>
<tr>
<td>Adjusted Rand Index (ARI)</td>
<td>0.4428</td>
<td>0.1800</td>
<td>0.2945</td>
</tr>
<tr>
<td>Normalized Mutual Information (NMI)</td>
<td>0.3735</td>
<td>0.1900</td>
<td>0.2428</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.3831</td>
<td>0.3500</td>
<td>0.2239</td>
</tr>
<tr>
<td>Fowlkes-Mallows Index (FMI)</td>
<td>0.5665</td>
<td>0.5000</td>
<td>0.6452</td>
</tr>
<tr>
<td>Purity</td>
<td>0.6816</td>
<td>0.5500</td>
<td>0.7761</td>
</tr>
</tbody>
</table>

The clustering performance metrics and confusion matrix for the ES Minus dataset with 2 clusters provide insights into the effectiveness of the clustering methods. The metrics for K-means, DBSCAN, and Agglomerative clustering are summarized in Table 5.6. K-means and Agglomerative methods show a mean silhouette score of 0.81, indicating good clustering quality, whereas DBSCAN shows a slightly lower silhouette score of 0.75. Adjusted Rand Index (ARI) values suggest moderate agreement with true labels for K-means (0.4428) and
Figure 5.21: Chaotic spiking behaviour present mainly in ES plus time trace ($k_{inj} = 14.2851, \Delta f = 6.92308$).

Figure 5.22: K-means Clustering ES plus time series (Mean silhouette score: 0.87 for 3 clusters).

Figure 5.23: Clustering methods comparison: K-means, Hierarchical, and DBSCAN clustering on UMAP-reduced data (ES Minus, 2 Clusters)

Agglomerative (0.2945), while DBSCAN has a lower ARI of 0.1800. Normalized Mutual Information (NMI) scores further reflect this trend, with K-means (0.3735) and Agglomerative (0.2428) showing higher mutual dependence with true labels compared to DBSCAN (0.1900).

Accuracy values indicate that K-means (0.3831) and DBSCAN (0.3500) correctly classify a higher percentage
Table 5.7: Confusion Matrix for K-means Clustering (2 Clusters)

<table>
<thead>
<tr>
<th>True Cluster 1</th>
<th>Predicted Cluster 1</th>
<th>Predicted Cluster 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Cluster 1</td>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>True Cluster 2</td>
<td>15</td>
<td>35</td>
</tr>
</tbody>
</table>

of data points than Agglomerative (0.2239). The Fowlkes-Mallows Index (FMI) suggests that Agglomerative (0.6452) has better similarity between clusters and true labels compared to K-means (0.5665) and DBSCAN (0.5000). Purity scores highlight that Agglomerative clustering (0.7761) has the highest homogeneity within clusters, followed by K-means (0.6816) and DBSCAN (0.5500).

The confusion matrix for our clustering analysis is presented in Table 5.7. Here, "Cluster 1" is identified as Chaos, and "Cluster 2" is Spiking.

Table 5.8: Clustering Performance Metrics Summary for ES Minus (3 Clusters)

<table>
<thead>
<tr>
<th>Metric</th>
<th>K-means</th>
<th>DBSCAN</th>
<th>Agglomerative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Silhouette Score</td>
<td>0.98</td>
<td>0.79</td>
<td>0.95</td>
</tr>
<tr>
<td>Adjusted Rand Index (ARI)</td>
<td>0.85</td>
<td>0.50</td>
<td>0.80</td>
</tr>
<tr>
<td>Normalized Mutual Information (NMI)</td>
<td>0.88</td>
<td>0.55</td>
<td>0.84</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.92</td>
<td>0.65</td>
<td>0.89</td>
</tr>
<tr>
<td>Fowlkes-Mallows Index (FMI)</td>
<td>0.87</td>
<td>0.60</td>
<td>0.85</td>
</tr>
<tr>
<td>Purity</td>
<td>0.95</td>
<td>0.70</td>
<td>0.93</td>
</tr>
</tbody>
</table>

Figure 5.24 illustrates the comparison of three clustering methods: K-means, Hierarchical, and DBSCAN, applied to UMAP-reduced data from the ES Minus dataset, focusing on 3 clusters. Cluster 1 is identified as Chaos, Cluster 2 as Spiking.

Table 5.9: Confusion Matrix for K-means Clustering (3 Clusters)

<table>
<thead>
<tr>
<th>True Cluster 1</th>
<th>Predicted Cluster 1</th>
<th>Predicted Cluster 2</th>
<th>Predicted Cluster 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Cluster 1</td>
<td>28</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>True Cluster 2</td>
<td>2</td>
<td>31</td>
<td>2</td>
</tr>
<tr>
<td>True Cluster 3</td>
<td>0</td>
<td>2</td>
<td>33</td>
</tr>
</tbody>
</table>

The clustering performance metrics in the Table 5.8 and confusion matrix offer valuable insights into the efficacy of K-means, Hierarchical, and DBSCAN clustering methods applied to UMAP-reduced data for the ES Minus dataset with three clusters. The Mean Silhouette Scores indicate that K-means (0.98) and Agglomerative (0.95) methods exhibit excellent clustering quality with well-separated clusters, while DBSCAN (0.79) shows good clustering quality. Adjusted Rand Index (ARI) values suggest high agreement with true labels for K-means (0.85) and Agglomerative (0.80), compared to moderate agreement for DBSCAN (0.50). Normalized Mutual Information (NMI) scores reinforce this trend, with K-means (0.88) and Agglomerative (0.84) demonstrating high mutual information with true labels, and DBSCAN (0.55) showing moderate levels.
Accuracy values reveal that K-means (0.92) and Agglomerative (0.89) correctly classify a higher percentage of data points compared to DBSCAN (0.65). The Fowlkes-Mallows Index (FMI) indicates good similarity between clusters and true labels for K-means (0.87) and Agglomerative (0.85), while DBSCAN (0.60) shows moderate similarity. Purity scores highlight high homogeneity within clusters for K-means (0.95) and Agglomerative (0.93), with DBSCAN (0.70) performing moderately. The confusion matrix for K-means clustering aligns with these metrics, showing minimal misclassifications and high accuracy, thus underscoring the robustness of the K-means method in this context.

The confusion matrix for our clustering analysis is presented in Table 5.9. Here, "Cluster 1" is considered the positive class and "Cluster 2" and "Cluster 3" the negative classes. The matrix provides the following insights:

- True Positives (TP): 28 instances of "Cluster 1" correctly identified as "Cluster 1".
- True Negatives (TN): 31 instances of "Cluster 2" and 28 instances of "Cluster 3" correctly identified.
- False Positives (FP): 2 instances of "Cluster 2" and 2 instances of "Cluster 3" incorrectly identified as "Cluster 1".
- False Negatives (FN): 2 instances of "Cluster 1" incorrectly identified as "Cluster 2" or "Cluster 3".
6. CONCLUSION

The intrinsic physical characteristics of photonics, combined with bio-inspired spike-based processing, offer substantial advancements in speed and efficiency over traditional electronic systems. This integration paves the way for ultrafast cognitive computing with applications in adaptive control, learning, and autonomous systems.

The research undertaken in this study explores the intricate dynamics of optically injected Quantum Dot (QD) spin-Vertical-Cavity Surface-Emitting Lasers (spin-VCSELs) and their potential for neuromorphic photonics applications.

Particularly QD spin-VCSELs exhibit unique characteristics that make them superior for neuromorphic photonics. Their ability to generate and control excitabile spiking signals at sub-nanosecond rates, coupled with their high efficiency and low energy consumption, makes them ideal candidates for next-generation photonic neural network hardware. These devices benefit from mature fabrication processes, high wall-plug efficiencies, and the potential for integration into scalable 3D photonic circuits. Future advancements, such as spin-VCSELs and high-beta VCSELs, promise even greater efficiencies and reduced energy costs, further solidifying their role in neuromorphic computing.

The timeseries for the ground state (GS) and excited state (ES) emissions were obtained through the numerical solution of specific rate equations using the Largest Lyapunov Exponent (LLE) method, mapped into a $k_{\text{inj}} - \Delta f$ plane to identify regions of excitability and stability (spiking and chaotic behaviour). This involved analyzing the Power Spectral Density (PSD) of the laser’s output to extract significant features such as statistical values regarding the number of peaks, the prominence of the peaks and also their width. The PSD analysis revealed structured and predictable behaviors in spiking regimes, with consistent peak prominence, whereas chaotic regimes displayed irregular and complex spectral landscapes, aligning with the theoretical insights from nonlinear dynamics and chaos theory. Additional features examined were the skewness, kurtosis of the time series data. All these features formed the basis for.

UMAP, t-SNE, and PCA were employed for dimensionality reduction, followed by clustering methods including K-means, DBSCAN, and Agglomerative clustering. Among these methods, UMAP provided the most distinct and well-separated clusters, particularly when applied to the ES-minus dataset. This resulted in a more accurate categorization of the dynamic states into spiking and chaotic behaviors, further distinguishing sub-categories within spiking such as low-intensity spiking, chaotic spiking, bursting spiking, and normal spiking behavior.

The UMAP projection results confirmed that UMAP outperformed PCA and t-SNE in preserving the local and global structure of the data, providing clearer and more distinct clusters. The ES plus time series represent the lasing output of the VCSEL. When subjected to UMAP after z-scoring, the data showed differentiation but more sparse clustering. This indicates that while UMAP captures the underlying structure, the lasing dynamics might not be fully separable in the reduced space. However, the ES minus time series showed clear separation into distinct clusters when analyzed with UMAP. The best clustering was achieved when using ES minus with UMAP parameters: min_dist set to 0.1 and n_neighbors set to 20. This configuration not only categorized spiking and chaos but also distinguished various spiking behaviors such as low-intensity spiking, chaotic spiking, bursting spiking, and normal spiking behavior.

The clustering performance metrics used in this analysis included the mean silhouette score, adjusted Rand index (ARI), normalized mutual information (NMI), accuracy, Fowlkes-Mallows index (FMI), and purity. These metrics provided a comprehensive evaluation of clustering quality, indicating that UMAP combined with K-means and Agglomerative clustering effectively identified and visualized distinct clusters in the dataset. The highest performance was observed with UMAP and K-means clustering, achieving a mean silhouette score of 0.98 and accuracy 0.92 for the ES-minus dataset.

The significance of these findings extends beyond the technical realm into practical applications. For instance, understanding and controlling the dynamic states of spin-VCSELs can lead to advances in secure communication systems, where the distinct spiking patterns could be used for encoding information. Additionally, the insights gained from this study contribute to the broader field of neuromorphic engineering, where integrating photonic and electronic systems can overcome the limitations of current microelectronic neural networks. By taking
advantage of the high-speed and low-power characteristics of photonic devices, along with the efficiency of bio-inspired processing, we can achieve sophisticated and high-bandwidth processing capabilities. The integration of advanced dimensionality reduction and clustering techniques further enhances the capability to analyze and interpret complex dynamic behaviors, paving the way for sophisticated applications in adaptive control, learning, perception, motion control, sensory processing, autonomous robotics, and ultrafast cognitive computing of the radio frequency spectrum.
7. FUTURE WORK

7.1 EXPANDING THE DATASET AND EXPERIMENTAL CONDITIONS

The dataset utilized in this study was derived from numerical simulations based on specific rate equations for QD spin-VCSELs. Future research should aim to expand this dataset by incorporating experimental data from actual QD spin-VCSEL systems under a variety of injection conditions and parameter settings. Such an approach would not only validate the simulation results but also provide a more comprehensive understanding of the dynamic behaviors observed in practical scenarios. Empirical data would bridge the gap between theoretical models and real-world applications, ensuring the robustness and applicability of the insights gained. Moreover, exploring a wider range of photonic devices and configurations could reveal new operational states, thereby enhancing the overall analysis and paving the way for more resilient and adaptable photonic neural networks.

7.2 ADVANCED FEATURE EXTRACTION AND CLASSIFICATION METHODS

The feature extraction process in this study was focused on specific characteristics such as skewness, kurtosis, median prominence, number of peaks, and range width. Future work should consider employing more sophisticated feature extraction methods, potentially incorporating machine learning techniques for automated feature selection and extraction. Advanced algorithms can uncover subtle patterns and correlations within the data that might be overlooked by conventional methods. Additionally, integrating cutting-edge classification algorithms, such as deep learning models, could significantly improve the accuracy and robustness of classifying dynamic states. These advancements would enable a more precise and reliable differentiation of various operational states, ultimately enhancing the performance and utility of photonic neural networks.

7.3 EXPLORING REAL-WORLD APPLICATIONS

The potential applications of neuromorphic photonics in adaptive control, learning, and sensory processing are extensive. Future research should focus on translating the theoretical findings of this study into real-world applications. For instance, developing photonic neural networks (PNNs) based on VCSELs for tasks such as autonomous robotics, high-speed signal processing, and real-time data analysis could showcase the practical utility of these technologies. Collaborating with industry partners to integrate these advanced photonic systems into existing computational frameworks could also accelerate the commercial adoption of neuromorphic photonics. Demonstrating the scalability and practical benefits of photonic neural networks would pave the way for the next generation of high-speed, energy-efficient, and adaptive computing systems, revolutionizing fields that demand rapid and precise information processing.
References


ANNEXES

MATLAB Scripts

Analysis of Peak Prominence, Width, and Statistical Metrics in Power Spectral Density (PSD) for Spikes and Chaos

```matlab
close all;
%clear all;

fs = 100; % sampling frequency

%-----------Definition of filter parameters-------------
cutoff_freq = 2; % Hz
filter_length = 101; % filter length must be odd to ensure
% that the filter has a linear phase response
%(all frequencies in the input signal are delayed by the same amount of
time)

%------------Designing low-pass filter------------
nyquist_freq = fs/2;
normalized_cutoff_freq = cutoff_freq/nyquist_freq;
filter_coeffs = fir1(filter_length-1, normalized_cutoff_freq, 'low'); % the filter length argument in the fir1 function specifies the number of
taps % which is one less than the actual filter length

%------------Applying filter to signal------------
filtered_signal = filter(filter_coeffs, 1, ES_minus(end-10000:end));

%------------Computing power spectrum of filtered signal------------
y = fft(detrend(filtered_signal));
n = length(filtered_signal);
f = (0:n/2-1)*(fs/n);
power = abs(y(1:n/2)).^2/n;

%------------Plotting power spectrum------------
plot(f,power)
xlabel('Frequency (Hz)')
ylabel('Power')

%------------Finding peaks in power spectrum------------
[pk, locs, widths, proms] = findpeaks(power, f, 'MinPeakHeight', 0.03*max
(power), 'WidthReference', 'halfprom'); % peak widths should be measured at half-prominence
% the minimum peak height to 5% of the maximum power in the spectrum
diff_widths = diff(widths(2:end));
diff_proms = diff(proms(2:end));

mean_diff_widths = mean(diff_widths, 'all');
std_diff_widths = std(diff_widths, 0, 'all');
range_diff_widths = range(diff_widths);
```
mean_diff_proms = mean(diff_proms, 'all');
std_diff_proms = std(diff_proms, 0, 'all');
rise_diff_proms = range(diff_proms);

% Printing the results
fprintf('Metrics for widths:
');
fprintf('Mean difference: %.2f
', mean_diff_widths);
fprintf('Standard deviation: %.2f
', std_diff_widths);
fprintf('Range: %.2f
', range_diff_widths);

fprintf('Metrics for proms:
');
fprintf('Mean difference: %.2f
', mean_diff_proms);
fprintf('Standard deviation: %.2f
', std_diff_proms);
fprintf('Range: %.2f
', range_diff_proms);

%-----------Plotting power spectrum with peaks and widths----------
hold on
plot(locs, pk, 'ro')
plot([locs-widths/2; locs+widths/2], [pk; pk], 'r')
%------------Printing peak information ------------
fprintf('Found %d peaks:
', length(pk));
for i = 1:length(pk)
    fprintf('Peak %d: Location = %.2f Hz, Width = %.2f Hz, Prominence = %.2f
', i, locs(i), widths(i), proms(i));
end

Dimensionality Reduction using UMAP and clustering with kmeans, Agglomerative Clustering, DBSCAN

clear all;
close all;

% directories and files
base_dir = pwd; % Current working directory
data_files = dir(fullfile(base_dir, 'vector_data*.mat'));

%categories and their respective ranges
spiking_files = 1:43; % Indices for spiking data
chaotic_files = 44:75; % Indices for chaotic data

% Define labels and categories
categories = {'Spiking', 'Chaos'};
category_colors = containers.Map({'Spiking', 'Chaos'}, {[1 0 0], [0 1 0]});
category_markers = containers.Map({'Spiking', 'Chaos'}, {'o', 's'});

% Initialization of structure to hold all processed data, labels, and file info
all_processed_data = [];
all_labels = {};
file_info = {};

77
% Processing files
for idx = 1:length(data_files)
    file = data_files(idx);
    % Extraction of the file number from the file name
    file_number = sscanf(file.name, 'vector_data%d.mat');
    try
        % Load the file
        D = load(fullfile(file.folder, file.name));
        % Determine the category based on the file number
        if ismember(file_number, spiking_files)
            category = 'Spiking';
        elseif ismember(file_number, chaotic_files)
            category = 'Chaos';
        else
            continue;
        end

        fprintf('Processing file: %s (Category: %s)
', file.name, category);
        % Process only ES_plus_storage field if it exists
        if isfield(D, 'ES_plus_storage')
            var_name = 'ES_plus_storage';
            unique_name = sprintf('%s_%s', var_name, category);
            processed_data = process_time_series(D.(var_name),
                unique_name);
            % Extraction and combination of features into a single row
            features = extract_features(processed_data);
            all_processed_data = [all_processed_data; features];
            % Determination of the label and file info
            all_labels{end+1} = category;
            file_info{end+1} = file.name;  % Storing of the file name

            fprintf('Processed %s
', unique_name);
        end
        catch ME
            fprintf('Error processing file: %s
', file.name);
            fprintf('Error message: %s
', ME.message);
        end
    end
end

% Check if lengths match
disp(['Length of all_processed_data: ', num2str(size(all_processed_data,1))]);
disp(['Length of all_labels: ', num2str(length(all_labels))]);
disp(['Length of file_info: ', num2str(length(file_info))]);

% Performing UMAP
[reduction, umap, clusterIdentifiers, extras] = run_umap(
    all_processed_data, 'n_components', 2, 'min_dist', 0.1, 'n_neighbors', 20, 'metric', 'euclidean');
% Check if reduction length matches all_labels length
disp(['Length of reduction: ', num2str(size(reduction, 1))]);

if size(reduction, 1) == length(all_labels)
  %% K-means Clustering
  num_clusters = 2;
  [idx_kmeans, C_kmeans] = kmeans(reduction, num_clusters);
  silhouette_kmeans = silhouette(reduction, idx_kmeans);
  mean_silhouette_kmeans = mean(silhouette_kmeans);
  fprintf('Mean silhouette score for k-means: %.2f
',
          mean_silhouette_kmeans);

  %% Agglomerative Hierarchical Clustering
  Z = linkage(reduction, 'ward');
  idx_hierarchical = cluster(Z, 'maxclust', num_clusters);
  silhouette_hierarchical = silhouette(reduction, idx_hierarchical);
  mean_silhouette_hierarchical = mean(silhouette_hierarchical);
  fprintf('Mean silhouette score for hierarchical: %.2f
',
          mean_silhouette_hierarchical);

  %% DBSCAN Clustering
  epsilon = 0.5;
  minpts = 5;
  idx_dbscan = dbscan(reduction, epsilon, minpts);
  silhouette_dbscan = silhouette(reduction, idx_dbscan);
  mean_silhouette_dbscan = mean(silhouette_dbscan);
  fprintf('Mean silhouette score for DBSCAN: %.2f
',
          mean_silhouette_dbscan);

  %% Visualization
  figure;
  subplot(1, 3, 1);
  gscatter(reduction(:, 1), reduction(:, 2), idx_kmeans, 'rgb', 'o');
  hold on;
  plot(C_kmeans(:, 1), C_kmeans(:, 2), 'kx', 'MarkerSize', 15, 'LineWidth', 3);
  hold off;
  title('K-means Clustering');
  xlabel('UMAP 1');
  ylabel('UMAP 2');
  legend('Cluster 1', 'Cluster 2', 'Centroids', 'Location', 'best');
  grid on;

  subplot(1, 3, 2);
  gscatter(reduction(:, 1), reduction(:, 2), idx_hierarchical, 'rgb', 'o');
  title('Hierarchical Clustering');
  xlabel('UMAP 1');
  ylabel('UMAP 2');
  legend('Cluster 1', 'Cluster 2', 'Location', 'best');
  grid on;

  subplot(1, 3, 3);
  gscatter(reduction(:, 1), reduction(:, 2), idx_dbscan, 'rgb', 'o');
  title('DBSCAN Clustering');
  xlabel('UMAP 1');
ylabel('UMAP 2');
legend('Cluster 1', 'Cluster 2', 'Noise', 'Location', 'best');
grid on;
else
    error('Mismatch between number of data points and number of labels.');
end

% Function to extract features from processed data
function features = extract_features(processed_data)
    % Extracting relevant features from processed data
    peaks = processed_data.peaks;
    % Ignore the first peak for prominence calculations
    if length(peaks.proms) > 1
        prods = peaks.proms(2:end);
        widths = peaks.widths(2:end);
    else
        prods = [];
        widths = [];
    end
    num_pks = length(prods); % Number of peaks (excluding the first)
    % Calculation of skewness and kurtosis from the original time series
    skewness_value = skewness(processed_data.time_series);
    kurtosis_value = kurtosis(processed_data.time_series);
    skewness_value_p = skewness(processed_data.power);
    kurtosis_value_p = kurtosis(processed_data.power);
    std_widths = std(widths);
    % Metric for prominence: median of prominences
    if isempty(prods)
        median_prominence = 0;
    else
        median_prominence = median(prods);
    end
    % Metric for widths: range of widths
    if isempty(widths)
        range_width = 0;
    else
        range_width = max(widths) - min(widths);
    end
    % Aggregate features into a single row vector
    features = [skewness_value, kurtosis_value, median_prominence, num_pks, range_width];
end

% Function to perform all calculation processes for each time series
function processed_data = process_time_series(time_series, var_name)
    try
        % Z-score the time series
        zscored_time_series = zscore_time_series(time_series);
        % Define sampling frequency

```matlab
fs = 100; % Sampling frequency

% Computation of the power spectrum PSD
y = fft(detrend(zscored_time_series));
n = length(zscored_time_series);
f = (0:n/2-1) * (fs/n);
power = abs(y(1:floor(n/2))).^2 / n;

% Find Peaks in the PSD
[pk, locs, widths, proms] = findpeaks(power, f, 'MinPeakHeight', 0.03 * max(power), 'WidthReference', 'halfprom');

% Combination of all processed data into a structure
processed_data = struct(...
    'time_series', zscored_time_series, ...
    'power', power, ...
    'peaks', struct('pk', pk, 'locs', locs, 'widths', widths, 'proms', proms) ...
);

catch ME
    fprintf('Error processing time series: %s\n', var_name);
    fprintf('Error message: %s\n', ME.message);
    processed_data = struct();
end

% Z-score function
function zscored_ts = zscore_time_series(time_series)
    mean_ts = mean(time_series);
    std_ts = std(time_series);
    zscored_ts = (time_series - mean_ts) / std_ts;
end
```

Dimensionality reduction with PCA and kmeans clustering
```
clear all;
close all;

% Definition of directories and files
base_dir = pwd; % Current working directory
data_files = dir(fullfile(base_dir, 'vector_data*.mat'));

% Definition of categories and their respective ranges
spiking_files = 1:43; % Indices for spiking data
chaotic_files = 44:75; % Indices for chaotic data

categories = {'Spiking', 'Chaos'};
category_colors = containers.Map({'Spiking', 'Chaos'}, {{1 0 0}, [0 1 0]});
category_markers = containers.Map({'Spiking', 'Chaos'}, {'o', 's'});

% Initialization of the structure to hold all processed data, labels, and file info
all_processed_data = [];
all_labels = {};
```
file_info = {};  

% Processing files
for idx = 1:length(data_files)
    file = data_files(idx);
    % Extraction the file number from the file name
    file_number = sscanf(file.name, 'vector_data%d.mat');
    try
        % Load the file
        D = load(fullfile(file.folder, file.name));
        % Determination the category based on the file number
        if ismember(file_number, spiking_files)
            category = 'Spiking';
        elseif ismember(file_number, chaotic_files)
            category = 'Chaos';
        else
            continue;
        end
        fprintf('Processing file: %s (Category: %s)
', file.name, category);  
        % Debugging statement
        % Process only ES_plus_storage field if it exists
        if isfield(D, 'ES_minus_storage')
            var_name = 'ES_minus_storage';
            unique_name = sprintf('%s_%s', var_name, category);
            % Generation of the processed data for each time series
            processed_data = process_time_series(D.(var_name),
                                                  unique_name);
            % Extraction and combination of features into a single row
            features = extract_features(processed_data);
            all_processed_data = [all_processed_data; features];
            % Determination label and file info
            all_labels{end+1} = category;
            file_info{end+1} = file.name;  
            % Storing of the file name
            % Debugging print statement
            fprintf('Processed %s
', unique_name);
        end
        catch ME
            fprintf('Error processing file: %s
', file.name);
            fprintf('Error message: %s
', ME.message);
        end
    end
    disp(['Length of all_processed_data: ', num2str(size(all_processed_data, 1))]);
    disp(['Length of all_labels: ', num2str(length(all_labels))]);
    disp(['Length of file_info: ', num2str(length(file_info))]);
% Performing PCA
[coeff, pca_reduction, latent] = pca(all_processed_data);
% Check if reduction length matches all_labels length
disp(['Length of pca_reduction: ', num2str(size(pca_reduction, 1))]);

if size(pca_reduction, 1) == length(all_labels)
    % Performing k-means clustering
    num_clusters = 2; % Set the number of clusters
    [idx, C] = kmeans(pca_reduction(:, 1:2), num_clusters); % Using only first 2 principal components

    % Compute silhouette values
    silhouette_values = silhouette(pca_reduction(:, 1:2), idx);

    % Calculate mean silhouette score
    mean_silhouette_score = mean(silhouette_values);

    % Display mean silhouette score
    fprintf('Mean silhouette score: %.2f
', mean_silhouette_score);

    % Create a new figure for silhouette plot
    figure;
    silhouette(pca_reduction(:, 1:2), idx);
    title('Silhouette Plot for PCA Reduced Data');
    xlabel('Silhouette Value');
    ylabel('Cluster');

    % Create a new figure for clustered PCA projection
    figure;
    hold on;
    % Plot each cluster with a different color and marker
    scatter(pca_reduction(idx == 1, 1), pca_reduction(idx == 1, 2), 50, 'r', 'o');
    scatter(pca_reduction(idx == 2, 1), pca_reduction(idx == 2, 2), 50, 'g', 's'); % Plot centroids
    plot(C(:, 1), C(:, 2), 'kx', 'MarkerSize', 15, 'LineWidth', 3);
    hold off;
    title('PCA Projection with K-means Clustering');
    xlabel('PCA 1');
    ylabel('PCA 2');
    legend({'Cluster 1', 'Cluster 2', 'Centroids'}, 'Location', 'best');
    grid on;
else
    error('Mismatch between number of data points and number of labels.');
end

% Function to extract features from processed data
function features = extract_features(processed_data)
    % Extract relevant features from processed data
    peaks = processed_data.peaks;

    if length(peaks.proms) > 1
        proms = peaks.proms(2:end); % Ignore first peak
        widths = peaks.widths(2:end); % Ignore first peak
    else

83
proms = []; widths = []; end

num_pks = length(proms); % Number of peaks (excluding the first)

% Calculation of skewness and kurtosis from the original time series
skewness_value = skewness(processed_data.time_series);
kurtosis_value = kurtosis(processed_data.time_series);
skewness_value_p = skewness(processed_data.power);
kurtosis_value_p = kurtosis(processed_data.power);

std_widths = std(widths);

% Metric for prominence: median of prominences
if isempty(proms)
    median_prominence = 0;
else
    median_prominence = median(proms);
end

% Metric for widths: range of widths
if isempty(widths)
    range_width = 0;
else
    range_width = max(widths) - min(widths);
end

% Aggregation of features into a single row vector
features = [skewness_value, kurtosis_value, median_prominence, num_pks, range_width];
end

% Function to perform all calculation processes for each time series
function processed_data = process_time_series(time_series, var_name)
try
% Definition sampling frequency
fs = 100; % Sampling frequency

% Computation of the power spectrum PSD
y = fft(detrend(time_series));
n = length(time_series);
f = (0:n/2-1) * (fs/n);
power = abs(y(1:floor(n/2))).^2 / n;

% Find Peaks in the PSD
[pk, locs, widths, proms] = findpeaks(power, f, 'MinPeakHeight', 0.03*max(power), 'WidthReference', 'halfprom');

% Combination of all processed data into a structure
processed_data = struct(...
    'time_series', time_series, ...
    'power', power, ...
    'peaks', struct('pk', pk, 'locs', locs, 'widths', widths, 'proms', proms) ...);
catch ME
    fprintf('Error processing time series: %s\n', var_name);
    fprintf('Error message: %s\n', ME.message);
end
Grid search to define the optimal UMAP parameters

```matlab
# Grid search to define the pest parameters for UMAP
clear all;
close all;

% Define directories
base_dir = '..', ...
categories = {'Chaos', 'Period1', 'Quasi-Periodic', 'Spikes'};
save_dir = '..', ...
if ~exist(save_dir, 'dir')
    mkdir(save_dir);
end

% Define sampling frequency
fs = 100; % Sampling frequency

% Initialize structure to hold all processed data and labels
all_processed_data = [];
all_labels = {};

% Define colors for each category
category_colors = containers.Map({'Chaos', 'Period1', 'Quasi-Periodic', 'Spikes'}, ...
    {[1 0 0], [0 1 0], [0 0 1], [1 1 0]});
category_markers = containers.Map({'Chaos', 'Period1', 'Quasi-Periodic', 'Spikes'}, ...
    {'o', 's', 'd', '^'});

% Define files to exclude
files_to_exclude = {'bursting_spike.mat', 'bursting_spike2.mat', 'bursting_spike3.mat', 'bursting_spike4.mat', ...
    'chaotic_spike.mat', 'chaotic_spike2.mat'};

% Processing files
for category_idx = 1:length(categories)
    category = categories(category_idx);
    category_dir = fullfile(base_dir, category);
    data_files = dir(fullfile(category_dir, '*.mat'));
    for file = data_files'
        % Handle specific exclusions and reclassifications
        if ismember(file.name, files_to_exclude)
            fprintf('Skipping file: %s\n', file.name);
            continue;
        elseif contains(file.name, 'chaotic_spike')
            fprintf('Reclassifying file: %s to Chaos\n', file.name);
            category = 'Chaos';
        end
        try
            % Load the file
```

85
D = load(fullfile(file.folder, file.name));

fprintf('Processing file: %s (Category: %s)\n', file.name, category); % Debugging statement

% Process only ES_plus and GS_plus fields if they exist
fields_to_process = {'GS_plus', 'GS_minus', 'ES_plus', 'ES_minus'};
for i = 1:length(fields_to_process)
    if isfield(D, fields_to_process{i})
        var_name = fields_to_process{i};
        unique_name = sprintf('%s_%s', var_name, category);

        % Generate the processed data for each time series
        processed_data = process_time_series(D.(var_name), fs, unique_name);

        % Extract and combine features into a single row
        features = extract_features(processed_data);
        all_processed_data = [all_processed_data; features];

        % Determine label
        all_labels{end+1} = category;

        % Debugging print statement
        fprintf('Processed %s\n', unique_name);
    end
end

% Check if lengths match
disp(['Length of all_processed_data: ', num2str(size(all_processed_data, 1))]);
disp(['Length of all_labels: ', num2str(length(all_labels))]);

% Define parameter ranges for grid search
n_neighbors_values = [5, 10, 15, 30, 50];
min_dist_values = [0.1, 0.5, 0.9];
best_score = Inf;
best_params = struct('n_neighbors', [], 'min_dist', []);
for n_neighbors = n_neighbors_values
    for min_dist = min_dist_values
        fprintf('Testing n_neighbors: %d, min_dist: %.2f\n', n_neighbors, min_dist);

        % Perform UMAP
        [reduction, umap, clusterIdentifiers, extras] = run_umap(
            all_processed_data, 'n_components', 2, 'min_dist', min_dist, 'n_neighbors', n_neighbors, 'metric', 'euclidean');

        % Check if reduction length matches all_labels length

if size(reduction, 1) == length(all_labels)
    % Calculate skewness and kurtosis for UMAP reduction
    umap_skewness = skewness(reduction);
    umap_kurtosis = kurtosis(reduction);

    % Calculate a score (e.g., sum of variances or other metric)
    score = sum(var(reduction));

    % If this score is the best, remember the parameters
    if score < best_score
        best_score = score;
        best_params.n_neighbors = n_neighbors;
        best_params.min_dist = min_dist;
    end

    % Display skewness and kurtosis
    disp(['UMAP Skewness: ', num2str(umap_skewness)]);
    disp(['UMAP Kurtosis: ', num2str(umap_kurtosis)]);
    disp(['Score: ', num2str(score)]);
    else
        error('Mismatch between number of data points and number of labels.');
    end
end
end

fprintf('Best parameters: n_neighbors = %d, min_dist = %.2f\n',
    best_params.n_neighbors, best_params.min_dist);

% Function to extract features from processed data
function features = extract_features(processed_data)
    % Extract relevant features from processed data
    peaks = processed_data.peaks;

    num_pks = length(peaks.pk);  % Number of peaks
    proms = peaks.proms;  % Prominences

    % Calculate skewness and kurtosis from the power spectrum
    skewness_value = skewness(processed_data.power);
    kurtosis_value = kurtosis(processed_data.power);

    % Aggregate features into a single row vector
    features = [skewness_value, kurtosis_value, num_pks, std(proms)];
end

% Function to perform all calculation processes for each time series
function processed_data = process_time_series(time_series, fs, var_name)
    try
        % Computation of the power spectrum PSD
        y = fft(detrend(time_series));
        n = length(time_series);
        f = (0:n/2-1) * (fs/n);
        power = abs(y(1:floor(n/2))).^2 / n;

        % Find Peaks in the PSD
        [pk, locs, widths, proms] = findpeaks(power, f, 'MinPeakHeight',
            0.03*max(power), 'WidthReference', 'halfprom');
% Combine all processed data into a structure
processed_data = struct(
    'time_series', time_series, ...
    'power', power, ...
    'peaks', struct('pk', pk, 'locs', locs, 'widths', widths, 'proms', proms) ...
);

% Error handling
catch ME
    fprintf('Error processing time series: %s\n', var_name);
    fprintf('Error message: %s\n', ME.message);
    processed_data = struct();
end

% Dimensionality reduction with t-SNE and Grid search to determine the best parameters

clear all;
close all;

% Define directories
base_dir = '..';
categories = {'Chaos', 'Period1', 'Quasi-Periodic', 'Spikes'};
save_dir = '...';
if ~exist(save_dir, 'dir')
    mkdir(save_dir);
end

fs = 100; % Sampling frequency

% Initialization of structure to hold all processed data and labels
all_processed_data = [];
all_labels = {};

% Definition colors for each category
category_colors = containers.Map({'Chaos', 'Period1', 'Quasi-Periodic', 'Spikes'}, ...)
    {[1 0 0], [0 1 0], [0 0 1], [1 1 0]});
category_markers = containers.Map({'Chaos', 'Period1', 'Quasi-Periodic', 'Spikes'}, ...)
    {'o', 's', 'd', '^'})

% Processing files
for category_idx = 1:length(categories)
    category = categories(category_idx);
category_dir = fullfile(base_dir, category);
data_files = dir(fullfile(category_dir, '*.mat'));
    for file = data_files'
        try % Loading the file
            D = load(fullfile(file.folder, file.name));
        fprintf('Processing file: %s (Category: %s)\n', file.name, category); % Debugging statement
        end
    end
end
```matlab
% Processing only ES_plus and GS_plus fields if they exist
fields_to_process = {'ES_plus', 'GS_plus'};
for i = 1:length(fields_to_process)
    if isfield(D, fields_to_process(i))
        var_name = fields_to_process(i);
        unique_name = sprintf('%s_%s', var_name, category);
        
        % Generating the processed data for each time series
        processed_data = process_time_series(D.(var_name), fs, unique_name);
        
        % Extracting and combining features into a single row
        features = extract_features(processed_data);
        all_processed_data = [all_processed_data; features];
        
        % Determination label
        all_labels(end+1) = category;
        
        % Debugging print statement
        fprintf('Processed %s
', unique_name);
    end
end

catch ME
    fprintf('Error processing file: %s
', file.name);
    fprintf('Error message: %s
', ME.message);
end

disp(['Length of all_processed_data: ', num2str(size(all_processed_data, 1))]);
disp(['Length of all_labels: ', num2str(length(all_labels))]);

% Definition of parameter ranges for grid search
perplexity_values = [5, 10, 30, 50];
learning_rate_values = [10, 100, 200, 500];
best_score = Inf;
best_params = struct('perplexity', [], 'learning_rate', []);
best_mappedX = [];

for perplexity = perplexity_values
    for learning_rate = learning_rate_values
        fprintf('Testing perplexity: %d, learning_rate: %.2f
', perplexity, learning_rate);
        
        % Performing t-SNE
        mappedX = tsne(all_processed_data, 'Perplexity', perplexity, 'LearnRate', learning_rate);
        
        % Checking if reduction length matches all_labels length
        if size(mappedX, 1) == length(all_labels)
            % Calculating a score (sum of variances or other metric)
            score = sum(var(mappedX));
```
% If this score is the best, remember the parameters and the result
if score < best_score
    best_score = score;
    best_params.perplexity = perplexity;
    best_params.learning_rate = learning_rate;
    best_mappedX = mappedX;
end

% Displaying the score
disp(['Score: ', num2str(score)]);
else
    error('Mismatch between number of data points and number of labels.');</end

fprintf('Best parameters: Perplexity = %d, Learning Rate = %.2f
',
    best_params.perplexity, best_params.learning_rate);

% Plotting the best t-SNE result
if ~isempty(best_mappedX)
    figure;
    hold on;

    % Plotting each category with a different color and marker
    for k = 1:length(categories)
        category = categories{k};
        category_idx = strcmp(all_labels, category);
        scatter(best_mappedX(category_idx, 1), best_mappedX(category_idx, 2),
            50, 'MarkerFaceColor', category_colors(category), 'Marker',
            category_markers(category));
    end

    % Finalizing the plot
    hold off;
    title('t-SNE Projection of Different Time Series Categories');
    xlabel('t-SNE 1');
    ylabel('t-SNE 2');
    legend(categories, 'Location', 'best');
    grid on;
end

% Function to extract features from processed data
function features = extract_features(processed_data)
    % Extracting relevant features from processed data
    peaks = processed_data.peaks;
    num_pks = length(peaks.pk); % Number of peaks
    proms = peaks.proms; % Prominences

    % Calculating skewness and kurtosis from the original time series
    skewness_value = skewness(processed_data.time_series);
    kurtosis_value = kurtosis(processed_data.time_series);
% Aggregating features into a single row vector
features = [skewness_value, kurtosis_value, num_pks, std(proms)];
end

% Function to perform all calculation processes for each time series
function processed_data = process_time_series(time_series, fs, var_name)
    try
        % Computation of the power spectrum PSD
        y = fft(detrend(time_series));
        n = length(time_series);
        f = (0:n/2-1) * (fs/n);
        power = abs(y(1:floor(n/2))).^2 / n;

        % Finding Peaks in the PSD
        [pk, locs, widths, proms] = findpeaks(power, f, 'MinPeakHeight', 0.03*max(power), 'WidthReference', 'halfprom');

        % Combining all processed data into a structure
        processed_data = struct('time_series', time_series,
                                'power', power,
                                'peaks', struct('pk', pk, 'locs', locs, 'widths', widths, 'proms', proms));
    catch ME
        fprintf('Error processing time series: %s\n', var_name);
        fprintf('Error message: %s\n', ME.message);
        processed_data = struct();
    end
end

Dimensionality reduction with UMAP, clustering with k-means (3 clusters) and calculation of evaluation metrics

clear all;
close all;

% Defining directories and files
base_dir = pwd; % Current working directory
data_files = dir(fullfile(base_dir, 'vector_data*.mat'));

% Defining categories and their respective ranges
spiking_files = 1:43; % Indices for spiking data
chaotic_files = 44:75; % Indices for chaotic data

categories = {'Spiking', 'Chaos'};
category_colors = containers.Map({'Spiking', 'Chaos'}, {[1 0 0], [0 1 0]});
category_markers = containers.Map({'Spiking', 'Chaos'}, {'o', 's'});

% Initializing structure to hold all processed data, labels, and file info
all_processed_data = [];
all_labels = {};
file_info = {};

91
% Processing files
for idx = 1:length(data_files)
    file = data_files(idx);
    % Extracting the file number from the file name
    file_number = sscanf(file.name, 'vector_data%d.mat');
    try
        % Loading the file
        D = load(fullfile(file.folder, file.name));
        % Determining the category based on the file number
        if ismember(file_number, spiking_files)
            category = 'Spiking';
        elseif ismember(file_number, chaotic_files)
            category = 'Chaos';
        else
            continue;
        end
        fprintf('Processing file: %s (Category: %s)\n', file.name, category);  % Debugging statement
        % Processing only ES_plus_storage field if it exists
        if isfield(D, 'ES_plus_storage')
            var_name = 'ES_plus_storage';
            unique_name = sprintf('%s_%s', var_name, category);

            % Generating the processed data for each time series
            processed_data = process_time_series(D.(var_name), unique_name);

            % Extracting and combining features into a single row
            features = extract_features(processed_data);
            all_processed_data = [all_processed_data; features];
        end
        % Determining label and file info
        all_labels{end+1} = category;
        file_info{end+1} = file.name;  % Store the file name
        % Debugging print statement
        fprintf('Processed %s\n', unique_name);
    catch ME
        fprintf('Error processing file: %s\n', file.name);
        fprintf('Error message: %s\n', ME.message);
    end
end

% Converting labels to numerical form
numeric_labels = grp2idx(all_labels);

% Checking if lengths match
disp(['Length of all_processed_data: ', num2str(size(all_processed_data, 1))]);
disp(['Length of all_labels: ', num2str(length(all_labels))]);
disp(['Length of file_info: ', num2str(length(file_info))]);
Performing UMAP

```matlab
[reduction, umap, clusterIdentifiers, extras] = run_umap(
    all_processed_data, 'n_components', 2, 'min_dist', 0.5, 'n_neighbors',
    30, 'metric', 'euclidean');
```

Checking if reduction length matches all_labels length

```matlab
disp(['Length of reduction: ', num2str(size(reduction, 1))]);
```

Ensureing lengths match

```matlab
if size(reduction, 1) == length(all_labels)
    % Performing k-means clustering
    num_clusters = 3; % Setting the number of clusters
    [idx, C] = kmeans(reduction, num_clusters);

    % Computing silhouette values
    silhouette_values = silhouette(reduction, idx);

    % Calculating the mean silhouette score
    mean_silhouette_score = mean(silhouette_values);

    % Displaying the mean silhouette score
    fprintf('Mean silhouette score: %.2f
', mean_silhouette_score);

    % Creating a new figure for silhouette plot
    figure;
    title('Silhouette Plot for UMAP Reduced Data');
    xlabel('Silhouette Value');
    ylabel('Cluster');
    saveas(gcf, 'silhouette_kmeans3.png');

    % Creating a new figure for k-means clustered UMAP projection
    figure;
    title('UMAP Projection with K-means Clustering');
    xlabel('UMAP 1');
    ylabel('UMAP 2');
    legend({'Cluster 1', 'Cluster 2', 'Cluster 3'}, 'Location', 'best');
    grid on;
    saveas(gcf, 'umap_kmeans3.png');

    % Computation of evaluation metrics
    fprintf('Evaluating clustering performance:
');
    ari = adjusted_rand_index(numeric_labels, idx);
    fprintf('Adjusted Rand Index: %.4f
', ari);

    % Normalized Mutual Information (NMI)
    nmi = normalized_mutual_info(numeric_labels, idx);
```
fprintf('Normalized Mutual Information: %.4f
', nmi);

% Confusion Matrix and Accuracy
conf_matrix = confusionmat(numeric_labels, idx);
disp('Confusion Matrix:');
disp(conf_matrix);
accuracy = sum(diag(conf_matrix)) / sum(conf_matrix(:));
fprintf('Accuracy: %.4f
', accuracy);

% Fowlkes-Mallows Index (FMI)
fmi = fowlkes_mallows(numeric_labels, idx);
fprintf('Fowlkes-Mallows Index: %.4f
', fmi);

% Purity Score
purity = sum(max(conf_matrix, [], 2)) / sum(conf_matrix(:));
fprintf('Purity: %.4f
', purity);

else
    error('Mismatch between number of data points and number of labels.');
end

% Function to extract features from processed data
function features = extract_features(processed_data)
% Extract relevant features from processed data
peaks = processed_data.peaks;

% Ignoring the first peak for prominence calculations
if length(peaks.proms) > 1
    proms = peaks.proms(2:end);
    widths = peaks.widths(2:end);
else
    proms = [];
    widths = [];
end
num_pks = length(proms); % Number of peaks (excluding the first)

% Calculation of skewness and kurtosis from the original time series
skewness_value = skewness(processed_data.time_series);
kurtosis_value = kurtosis(processed_data.time_series);
skewness_value_p = skewness(processed_data.power);
kurtosis_value_p = kurtosis(processed_data.power);
std_widths = std(widths);
% Metric for prominence: median of prominences
if isempty(proms)
    median_prominence = 0;
else
    median_prominence = median(proms);
end
% Metric for widths: range of widths
if isempty(widths)
    range_width = 0;
else
    range_width = max(widths) - min(widths);
end
% Aggregated features into a single row vector
features = [skewness_value, kurtosis_value, median_prominence, num_pks, range_width];
end

% Function to perform all calculation processes for each time series
function processed_data = process_time_series(time_series, var_name)
    try
        % Z-score the time series
        zscored_time_series = zscore_time_series(time_series);

        % Definition of sampling frequency
        fs = 100; % Sampling frequency

        % Computation of the power spectrum PSD
        y = fft(detrend(zscored_time_series));
        n = length(zscored_time_series);
        f = (0:n/2-1) * (fs/n);
        power = abs(y(1:floor(n/2))).^2 / n;

        % Find Peaks in the PSD
        [pk, locs, widths, proms] = findpeaks(power, f, 'MinPeakHeight', 0.03 * max(power), 'WidthReference', 'halfprom');

        % Combination of all processed data into a structure
        processed_data = struct(
            'time_series', zscored_time_series,
            'power', power,
            'peaks', struct('pk', pk, 'locs', locs, 'widths', widths, 'proms', proms)
        );
    catch ME
        fprintf('Error processing time series: %s\n', var_name);
        fprintf('Error message: %s\n', ME.message);
        processed_data = struct();
    end
end

% Z-score function
function zscored_ts = zscore_time_series(time_series)
    mean_ts = mean(time_series);
    std_ts = std(time_series);
    zscored_ts = (time_series - mean_ts) / std_ts;
end

% Function to calculate Adjusted Rand Index (ARI)
function ari = adjusted_rand_index(true_labels, predicted_labels)
    ari = rand_index(true_labels, predicted_labels, 'adjusted');
end

% Function to calculate Normalized Mutual Information (NMI)
function nmi = normalized_mutual_info(true_labels, predicted_labels)
    nmi = mutualinfo(true_labels, predicted_labels);
end

% Function to calculate Fowlkes-Mallows Index (FMI)
function fmi = fowlkes_mallows(true_labels, predicted_labels)
    contingency = crosstab(true_labels, predicted_labels);
    TP = sum(arrayfun(@(x) nchoosek(x, 2), contingency), 'all');
    FPa = sum(arrayfun(@(x) nchoosek(x, 2), sum(contingency, 2))) - TP;
    FPb = sum(arrayfun(@(x) nchoosek(x, 2), sum(contingency, 1))) - TP;
    fmi = TP / sqrt((TP + FPa) * (TP + FPb));
end

% Function to calculate Adjusted Rand Index
function ari = rand_index(true_labels, predicted_labels, type)
    n = length(true_labels);
    contingency = crosstab(true_labels, predicted_labels);

    % Computation of the sum of combinations for the contingency matrix
    sum_comb_c = sum(arrayfun(@(x) nchoosek(x, 2), contingency), 'all');

    % Computation of the sum of combinations for the rows and columns
    sum_comb_a = sum(arrayfun(@(x) nchoosek(x, 2), sum(contingency, 2)));
    sum_comb_b = sum(arrayfun(@(x) nchoosek(x, 2), sum(contingency, 1)));

    if strcmp(type, 'adjusted')
        expected_index = sum_comb_a * sum_comb_b / nchoosek(n, 2);
        max_index = (sum_comb_a + sum_comb_b) / 2;
        ari = (sum_comb_c - expected_index) / (max_index - expected_index);
    else
        ari = sum_comb_c / nchoosek(n, 2);
    end
end

% Helper function for entropy calculation
function H = entropy(labels)
    p = labels / sum(labels);
    H = -sum(p .* log2(p + eps));
end

% Helper function for mutual information calculation
function MI = mutualinfo_matrix(contingency)
    n = sum(contingency, 'all');
    pi = sum(contingency, 2);
    pj = sum(contingency, 1);
    MI = 0;
    for i = 1:size(contingency, 1)
        for j = 1:size(contingency, 2)
            if contingency(i, j) > 0
                MI = MI + (contingency(i, j) / n) * log2((contingency(i, j) * n) / (pi(i) * pj(j)));
            end
        end
    end
end

% Function to calculate mutual information
function MI = mutualinfo(true_labels, predicted_labels)
    contingency = crosstab(true_labels, predicted_labels);
    H_true = entropy(sum(contingency, 2));
    H_pred = entropy(sum(contingency, 1));
\text{MI} = \text{mutualinfo_matrix(contingency)};
\text{MI} = \text{MI} / \sqrt{H_{\text{true}} \times H_{\text{pred}}};
\textbf{end}
import os
import glob
import scipy.io
import numpy as np
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Definition of the base directory
base_dir = r'..

# Definition of the range for spiking and chaos data
spiking_range = range(1, 44)
chaos_range = range(44, 73)

# Initialization of the lists to store the data and labels
data_list = []
labels_list = []

# Loading spiking data
for idx in spiking_range:
    file = os.path.join(base_dir, f'vector_data{idx}.mat')
    if os.path.isfile(file):
        mat = scipy.io.loadmat(file)
        for key in ['GS_plus_storage', 'GS_minus_storage',
                    'ES_plus_storage', 'ES_minus_storage']:
            if key in mat:
                data = mat[key].flatten()
                data_list.append(data)
                labels_list.append('spiking')

# Loading chaos data
for idx in chaos_range:
    file = os.path.join(base_dir, f'vector_data{idx}.mat')
    if os.path.isfile(file):
        mat = scipy.io.loadmat(file)
        for key in ['GS_plus_storage',
                    'ES_plus_storage']:
            if key in mat:
                data = mat[key].flatten()
                data_list.append(data)
                labels_list.append('chaos')

# Preparing the data for t-SNE
all_data = np.array(data_list)
all_labels = np.array(labels_list)

# Applying t-SNE
tsne = TSNE(n_components=2, perplexity=30, n_iter=300, random_state=42)
tsne_results = tsne.fit_transform(all_data)

# Plotting
plt.figure(figsize=(20, 12))
for label in ['spiking', 'chaos']:
    indices = all_labels == label
    plt.scatter(tsne_results[indices, 0], tsne_results[indices, 1],
                label=label, alpha=0.7, edgecolors='w', s=100)
plt.title('t-SNE of Spiking and Chaos Data')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.legend()
plt.show()
# Plotting
plt.figure(figsize=(20, 12))
for cls in classes:
    indices = all_labels == cls
    plt.scatter(pca_results[indices, 0], pca_results[indices, 1],
                label=cls, alpha=0.7, edgecolors='w', s=100)
plt.title('PCA of Chris Data')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()

Dimesionality Reduction with UMAP AND kNN graph

# ------------UMAP & KNN for the time series-----------
import os
import glob
import pandas as pd
import numpy as np
import umap
import networkx as nx
import matplotlib.pyplot as plt
from sklearn.metrics import pairwise_distances
from sklearn.neighbors import kneighbors_graph

# Definition of the base directory
base_dir = r'..

# Definition of the subdirectories
directories = ['ES_plus_storage', 'GS_plus_storage']
types = ['spike', 'chaos']

data_dict = {directory: [] for directory in directories}
labels_dict = {directory: [] for directory in directories}

# Loading of data from CSV files
for directory in directories:
    for type_ in types:
        path = os.path.join(base_dir, directory, type_, '*.csv')
        files = glob.glob(path)
        for file in files:
            data = pd.read_csv(file, header=None).values.flatten()
            data_dict[directory].append(data)
            labels_dict[directory].append(type_)

# Converting lists to dataframes
for key in data_dict:
    data_dict[key] = pd.DataFrame(data_dict[key])
    labels_dict[key] = pd.Series(labels_dict[key])

# Function to create a mutual k-NN graph
def mutual_knn_graph(data, k=5):
    knn_graph = kneighbors_graph(data, k, mode='connectivity',

100
adj_matrix = knn_graph.toarray()
mutual_adj_matrix = adj_matrix * adj_matrix.T  # Ensure mutual connectivity
return mutual_adj_matrix

# Creation subplots
fig, axes = plt.subplots(2, 2, figsize=(20, 12))
axes = axes.flatten()

# Application of UMAP with mutual k-NN graph and plot with different colors for spiking and chaos data
for i, key in enumerate(data_dict):
    data = data_dict[key]
    labels = labels_dict[key]

    # Creation mutual k-NN graph
    mutual_knn_adj = mutual_knn_graph(data, k=10)

    # Using mutual k-NN graph to compute distances
    shortest_path_lengths = dict(nx.all_pairs_shortest_path_length(graph))
    distances = np.zeros((len(data), len(data)))
    for i, target_dict in shortest_path_lengths.items():
        for j, length in target_dict.items():
            distances[i, j] = length

    # Application UMAP with custom distance metric
    reducer = umap.UMAP(n_components=2, metric='precomputed', n_neighbors=15,
                        min_dist=0.1)
    umap_results = reducer.fit_transform(distances)

    ax = axes[i]
    for label in types:
        indices = labels == label
        ax.scatter(umap_results[indices, 0], umap_results[indices, 1],
                   label=label, alpha=0.7, edgecolors='w', s=100)

    ax.set_title(f'UMAP with Mutual k-NN Graph of {key}')
    ax.set_xlabel('Component 1')
    ax.set_ylabel('Component 2')
    ax.legend()

plt.tight_layout()
plt.show()