Enhancing Glitch Identification in Gravitational Wave Analysis: A Continual Learning Approach

by

Brian Baert

in partial fulfillment of the requirements for the degree of

Master of Science
in Artificial Intelligence

at the Open University, Faculty of Science
Master Artificial Intelligence
to be defended publicly on Wednesday June 26, 2024 at 14:00 PM.

Student number:	850794710	
Course code:	IM9506	
Thesis committee:	Dr. Stefano Bromuri (chairman),	Open University
	Dr. Daniel Tan (supervisor),	Open University

ACKNOWLEDGEMENT

Embarking on this journey into the fascinating field of Continual Learning and Glitch Detection would not have been possible without the support of many incredible people.

First and foremost, I want to express my gratitude to my supervisors, Dr. Stefano Bromuri and Dr. Daniel Tan as well as Tom Dooney. Your guidance and insightful discussions, and unwavering patience throughout this process where invaluable.

My heartfelt thanks go to the entire faculty and staff at Howest (Howest University of Applied Sciences). My homebase and supportive environment. A special shout-out to the Applied Computer Science education program for their tireless efforts to aid me wherever they could, especially the program coordinator, William Schokkelé, Director of Education, Isabel Uitdebroeck and my dear colleague Dr. Ines Devlieger for her critical eye on proofreading the results section.

To my amazing husband, Cédric, my unwavering rock throughout this endeavor, thank you for your belief in me. Your understanding during late nights and weekends spent buried in research has been a source of immense strenght. I would not be here without your support and love.

Who would have thought that at the age of 41 I would finally succeed in achieving one of my long aspired goals of achieving a Master of Science degree, a journey that started over three years ago with a Postgraduate has finally come to an end. An open ending because life starts at 41 right...

CONTENTS

List of Figures

List of Tables

ACRONYMS

A-GEM Averaged Gradient Episodic Memory. x, 15, 38, 48, 57, 58, 59

ANAM Averaged Normalized Autocorrelation Method. 6

BBH Binary Black Hole. 1, 3

BNS Binary Neutron Star. 1, 3, 5, 6

BWT Backward Transfer. 58

CBC Compact Binary Coalescence. 3, 5, 6

CCSNe Core-Collapse Supernovae. 1, 3

CL Continual Learning. x, 1, 2, 9, 11, 12, 14, 15, 16, 17, 18, 20, 21, 23, 24, 30, 31, 35, 38, 40, 45, 56, 57, 58, 59, 60

CNN Convolutional Neural Network. 2, 4, 5, 8, 16, 21, 22

DER Dark Experience Replay. x, 17, 41, 42, 43, 44, 56, 57, 59

DER++ Dark Experience Replay ++. x, 17, 43

DL Deep Learning. x, 2, 3, 4, 5, 11, 19, 56, 59

EWC Elastic Weight Consolidation. x, 2, 11, 12, 13, 14, 40, 59

FD Fractal Dimension. v, 2, 6, 7, 57

FROMP Functional Regularization Of the Memorable Past. 14

FWT Forward Transfer. 58

GEM Gradient Episodic Memory. 15

GW Gravitational Waves. x, 1, 2, 3, 5, 6, 26, 59, 60

iDQ inferential Data Quality. 2

IL Incremental Learning. 2, 9, 11, 18, 20

LIGO Laser Interferometer Gravitational-wave observatory. v, 1, 2, 4, 6, 26

LR Logistic Regression. 5

LwF Learning without Forgetting. x, 12, 14, 35, 41, 57, 59

mAP mean Average Precision. 6

ML Machine Learning. 2, 3, 5, 12

NCM Nearest-Class-Mean. 16

NS Neutron Star. 1

NSBH Neutron Star-Black Hole. 1, 3

O1 First observing run. 1

O2 Second observing run. 1

O3 Third observing run. 1, 5, 6

OD Object Detection. 5

PCA Principal Component Analysis. 5

SCR Supervised Contrastive Replay. x, 16, 44, 52, 57, 59

SGD Stochastic Gradient Descent. 14

SI Synaptic Intelligence. 12, 14

SNR Signal-to-Noise Ratio. 4, 6, 25

SVM Support Vector Machine. 5

t-SNE t-Distributed Stochastic Neighbor Embedding. 24, 32, 36, 39, 42, 46, 48, 52, 54, 56, 57

UMAP Uniform Manifold Approximation and Projection. 24, 32, 36, 39, 42, 46, 48, 52, 56

VAR Variation method. 6

XdG Context-dependent Gating. 11

XGBoost Extreme Gradient Boosting. 5

YOLO You Only Look Once. 5

ABSTRACT

This thesis explores the incorporation of Continual Learning (CL) into traditional Deep Learning (DL) frameworks for classifying glitches in Gravitational Waves (GW) data analysis. The research utilizes strain data and auxiliary channel data in a multimodal fusion approach to enhance the performance of the glitch classification. Various Continual Learning strategies, such as Naive CL, Learning without Forgetting (LwF), Averaged Gradient Episodic Memory (A-GEM), Elastic Weight Consolidation (EWC), Dark Experience Replay (DER), Dark Experience Replay ++ (DER++), and Supervised Contrastive Replay (SCR), were assessed. The results reveal that although naive strategies exhibited the highest individual performance, multimodal fusion significantly boosts overall classification accuracy, reaching up to 100% test set accuracy when combining spectrograms (Q-scans) and fractal dimension matrices. Furthermore, the thesis underscores the limitations and potential of the Avalanche framework for managing multimodal data, indicating the need for further research into CL applications in this area. Future research should aim to broaden the range of glitch types and explore advanced architectures like Transformers to further enhance the robustness and applicability of the proposed methods.

1

INTRODUCTION

1.1. INTRODUCTION

The following thesis investigates Continual Learning as a framework for robust and efficient Gravitational Waves analysis. CL algorithms excel at incrementally learning from new data streams while retaining knowledge of previously encountered classes [De Lange et al., 2021; Qu et al., 2021; van de Ven et al., 2022].
GW are ripples in space-time caused by violent disruptions in the universe. Massive accelerating objects like Neutron Star (NS) and Binary Black Hole (BBH) cause such strong waves propagating in all directions. Other cataclysmic events, responsible for GW, are supernovae and obviously residues of the Big Bang [Caltech, 2024b].
Although the generation of GW is a frenzy of energy, by the time the waves (invisible for the human eye) reach earth they are imperceptibly small.
The birth of LIGO (Laser Interferometer Gravitational-wave observatory) offered a way to make very small measurements, equivalent to fractional changes in distance of 10^{-21} (smaller than an atom), microscopic measurements needed to detect GW [Glanzer et al., 2023; Zevin et al., 2017].
LIGO has already completed three observing runs, abbreviated O1 (First observing run), O2 (Second observing run) and O3 (Third observing run). At this very moment of writing the fourth observing run started on May the 24th of 2023 and the first period, 04a has ended on January the 16th of 2024 [IGWN, 2024].
Up to this point the types of uncovered GW signals are limited to BBH, Binary Neutron Star (BNS) and Neutron Star-Black Hole (NSBH) incidents. Researchers hope to unravel previously unseen cosmological cataclysms such as CCSNe (Core-Collapse Supernovae), magnetars or probabilistic background waves from the Big Bang [Cuoco et al., 2020].
GW Analysis is a challenging field where scientists with varying backgrounds work together. With each observing run of LIGO resulting in more detected gravitational events (the majority were BBH mergers) [Glanzer et al., 2023; Zevin et al., 2017], [Caltech, 2024a].

The accurate identification of GW signals amidst noisy data remains a significant challenge. One of the primary sources of noise in GW detectors are glitches. Glitches are short-lived non-Gaussian noise features that can be instrumental or environmental in nature. Glitches impact the analysis of wave candidates in three ways. Firstly, glitches increase the background loudness and thus reduce the candidate significance. Secondly glitches increase the uncertainty in the categorisation process because they operate with the same

amplitude of astrophysical signals. Thirdly, the amount of usable data is even decreased more due to glitch contamination [Cuoco et al., 2020; Zevin et al., 2017].
Deep Learning (DL) and especially Convolutional Neural Network (CNN) have proven to be suitable architectures for extracting glitch features [Fernandes et al., 2023; George et al., 2017; Glanzer et al., 2023]. Together with the GravitySpy citizen science project [Bahaadini et al., 2018; Zevin et al., 2017], the images can then be classified into their corresponding glitch morphologies, as seen in Figure 1.1.

GW detectors host hundreds of thousands of auxiliary channels that monitor the state of the instruments and the environment. They have shown promise in characterizing glitches, such as inferential Data Quality (iDQ) [Davis and Walker, 2022; Essick et al., 2020] or Elastic-net based ML for Understanding (EMU) [Colgan et al., 2020] have been studied. The fusion of Fractal Dimension (FD) analysis and ML offers a robust framework for glitch detection in GW analysis. Moreover, the scalability of FD analysis across auxiliary channels holds promise for real-time glitch detection [Benedetto et al., 2023; Colgan et al., 2020].

The limitations of the current classical approaches of offline training a neural network are evident due to the new data expected to be delivered in the fourth observing run [Wu et al., 2024]. In this context, the field of Incremental Learning (IL) or CL emerges as a compelling area. The IL approach offers a fresh framework that allows models to learn from a continuously expanding dataset while still maintaining performance on previously encountered data [De Lange et al., 2021; Qu et al., 2021; van de Ven et al., 2022]. Unlike batch learning, where models are trained on static datasets, CL enables models to adapt incrementally to new data while retaining knowledge of previously encountered classes [Kirkpatrick et al., 2017].

In the context of GW analysis, CL strategies allow us to accommodate new glitch classes (as LIGO continues to operate and collect data, new glitch classes may emerge where CL ensures that the model can seamlessly incorporate these novel classes without compromising performance on existing ones). It mitigates catastrophic forgetting (traditional ML models tend to forget previously learned information when exposed to new data. CL methods such as Elastic Weight Consolidation (EWC) or memory-augmented architectures address this issue by preserving knowledge of existing glitch morphologies [Abbott et al., 2023; Kirkpatrick et al., 2017].

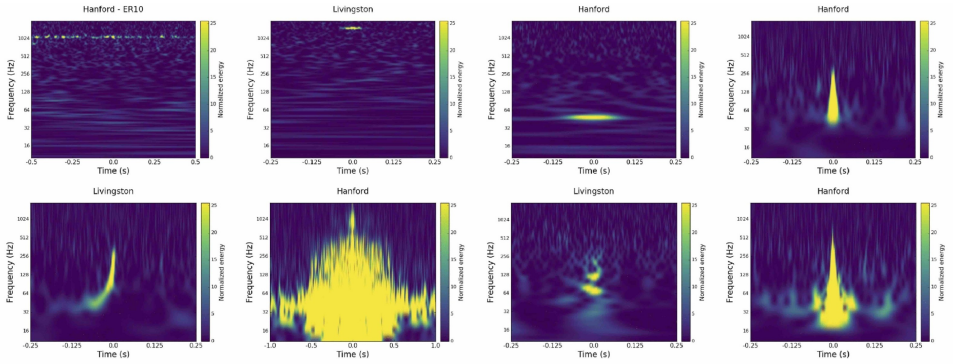


Figure 1.1: Some glitch class morphologies. From top left to bottom right: 1080Lines, 1400Ripples, Air Compressor, Blip, row two: Chirp, Extremely Loud, Helix, Koi Fish. from [Glanzer et al., 2023]

In addition to identification and characterization of glitches, noise subtraction is another problem relating to signal processing [Benedetto et al., 2023; Cornish et al., 2021;

Davis and Walker, 2022; Ormiston et al., 2020]. BayesWave is a tool that can be used to split the GW signal from the noise [Cornish et al., 2021].

Glitches are only one side of the spectrum, it is evenly important that Compact Binary Coalescence (CBC) parameters can be correctly estimated. Researchers typically use waveform templates to compute the necessary parameter posteriors based on numerical solutions to the two body problem, but this remains computationally inefficient [Ajith, 2011; Coogan et al., 2022; Isoyama et al., 2020].

Searching GW signals is commonly split up in four separate categories. The first one being the CBC's (BBH, BNS, NSBH), the second type are called GW bursts these can be the product of CCSNe or pulsar glitches. The third type are called long continuous GW. The last category are residue waves from distant CBC's or even the Big Bang [Abbott et al., 2020a; Caltech, 2024c]. The use of Machine Learning (ML), especially Random Forests (RF) was able to increase the detection speed of CBC's [Kapadia et al., 2017]. The Deep Learning (DL) approach on the other hand has had mixed results [Chatterjee et al., 2021; Gebhard et al., 2019; Ruan et al., 2023]. Searching for other types of signals via ML or DL remains difficult because there is much uncertainty about morphology (due to lack of training data) [Cuoco et al., 2020].

A last application area where DL is being investigated is the estimation of astrophysical parameters. Here Bayesian approaches have the ability to proof usefull, but the computational cost of calculating all posterior probabilities is an obstacle [Cuoco et al., 2020].

1.2. THESIS STRUCTURE

The thesis is structured as follows:

1. Introduction
2. Related Work and Background
3. Research Methodology
4. Models and Results
5. Discussion
6. Conclusions and Recommendations
7. Appendices

2

RELATED WORK AND BACKGROUND

2.1. RELATED WORK

2.1.1. GRAVITY SPY

The construction of one of the main datasets for the classification and detection of glitches is the Gravity Spy dataset [Glanzer et al., 2023; Zevin et al., 2017]. The Gravity Spy dataset consists of a large set of Qscans (also called spectrograms). These Qscans are obtained via the Omicron algorithm pipeline [Robinet et al., 2020]. Omicron originates from the Q-transformation [Chatterji et al., 2004]. The colour scale, or normalized energy, used in these images, as in Figure 2.1, is the square of the SNR in a given Q-tile, divided by the mean squared magnitude for stationary white noise.

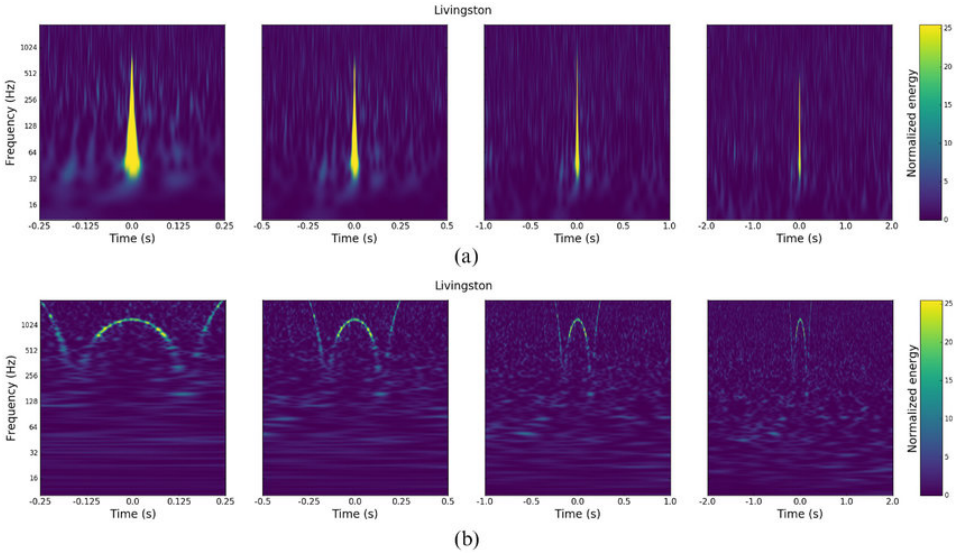


Figure 2.1: LIGO Qscan images of 'blip' (first row) and 'whistle' (second row) over four different durations [Zevin et al., 2017]

The researchers use visual inspection by citizen scientists as well as DL classification using a CNN. The CNN outputs a confidence score p for each glitch class by using a softmax layer.

2.1.2. GLITCH CLASSIFICATION

Because the GW signal strength is weaker than the background noise, the preferred architecture is often a CNN's with wavelet and Q-transforms Cuoco et al. [2020].

The proposed DL methodology presented in [Bişkin et al., 2023] uses a combination of ML and DL algorithms for classification. It consists of two phases, first, they utilise feature extraction possibilities of the pre-trained DL models Inception-v3 [Szegedy et al., 2016] and ResNet-50 [He et al., 2016] on the Q-transformed glitch images. Secondly, they apply PCA to select 100 components, these serve as input for SVM, LR and XGBoost algorithms. They opted to use only 10 glitch classes from the Gravity Spy dataset and no up- or downsampling techniques to issue the imbalancedness of the dataset. All samples are Q-transformed before entering the training phase. The results showed that Inception-v3-based architectures perform faster and the LR algorithm delivers the highest accuracy on the test set (96.06%). Certain glitch morphologies like '1400 ripples', 'light modulation', 'Tomte' and 'None of the above' had substantially lower scores.

The usage of CNN for glitch classification is described in [Fernandes et al., 2023]. They opt for a ResNet [He et al., 2016] architecture baseline (ResNet18 and ResNet34) and the `fastai` `fit_one_cycle` route.

2.1.3. GW DETECTION

The updated Gravity Spy classification model [Jarov et al., 2023] can differentiate CBC signals from detector glitches. To achieve this accuracy improvement, they used a two-step approach. First, a baseline segment is selected from O3 and the `pycbc` package [Nitz et al., 2020] is used to inject waveform simulations originating from masses between $3M_{\odot}$ and $350M_{\odot}$. Secondly, two classifiers (one for low-mass signals and one for high-mass CBC signals) are implemented. Because certain glitch morphologies have similar structures, this type of noise must be inserted in the training set as well. Finally, the Gravity Spy CNN architecture is retrained on the extended training sets resulting in a signal-versus-glitch classifier. The relationship between the total mass of the inspiraling objects and the GW accuracy was the main reason for splitting the classifiers into low and high masses.

2.1.4. OBJECT DETECTION FOR CBC

Researchers [Aveiro et al., 2022] have investigated the use of a classical single shot Object Detection method, You Only Look Once (YOLO) [Redmon et al., 2016], to detect a subset of CBC events, namely BNS mergers. Single shot detector methods are cost- and time-efficient [Redmon and Farhadi, 2017] and hence are suitable candidates to integrate in a CBC object detection method. They use generated BNS signals via the `PyCBC` package [Nitz et al., 2020] and the generation pipeline from [Gebhard et al., 2019]. The generated signals are then randomly cut to the required 8s duration, with one condition, the generated waveform must be in the cut. Just as in the Gravity Spy generation process [Zevin et al., 2017], the Q-transform [Chatterji et al., 2004] is used to generate the Qscans (here they opt for the `librosa` package [McFee et al., 2015]).

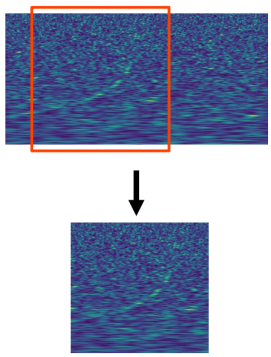


Figure 2.2: Qscan generation and final 8-bit PNG image [Aveiro et al., 2022]

The Qscans are converted to 8-bit PNG images as seen in Figure 2.2

The bounding box coordinates of the GW class are included in a separate text file per image. Labelling is automated as part of the generation pipeline. Initial tests were promising with acceptable mean Average Precision (mAP) results. The research team experimented with varying SNR parameters (between 10 and 20). Lower SNR degraded the resulting metrics. Although no real training data were used, exploratory tests on GW170817 [Abbott et al., 2017] and GW190425 [Abbott et al., 2020b] BNS coalescence events indicate the model's ability to perform GW detections.

2.1.5. Fractal based noise characterisation

Two relevant studies were [Cavaglia, 2022] and [Laguarta et al., 2023]. The noise found in GW detectors usually is non-Gaussian and non-stationary [Abbott et al., 2020a]. Distinguishing noise from spectrograms and auxiliary channels can be done straightforwardly, but determining the CBC parameters is not [Powell, 2018]. Researchers [Cavaglia, 2022] propose utilising Fractal Dimension (FD) on auxiliary channels like instrumental control data to identify transient origins of noise. FD calculation approximations are done in a time-efficient way. Fractals are basically subsets of n-dimensional Euclidean spaces [Mandelbrot, 1989] where $D_F < n$ and $D_F \notin \mathbf{Z}$. FD measures the degree of complexity of the set. The Minkowski-Bouligand or box-counting definition is used [Weisstein, 2019]. The definition used in [Cavaglia, 2022] is taken from [Dubuc et al., 1989]. Because exact measurements are not uniformly defined, approximations like Variation method (VAR) or Averaged Normalized Autocorrelation Method (ANAM) are used.[Cavaglia, 2022] prefers the VAR approximation and applies it to two stretches of LIGO data from O3 where whistle glitches occur. Whistle glitches are short-duration noise (around 1 second) and have a frequency between 100Hz and several kHz. In the analysis, the focus lies on the strain data and only one auxiliary channel `L1:LSC-PRCL_OUT_DQ` (interferometer PRC (Power Recycling Cavity) monitor).

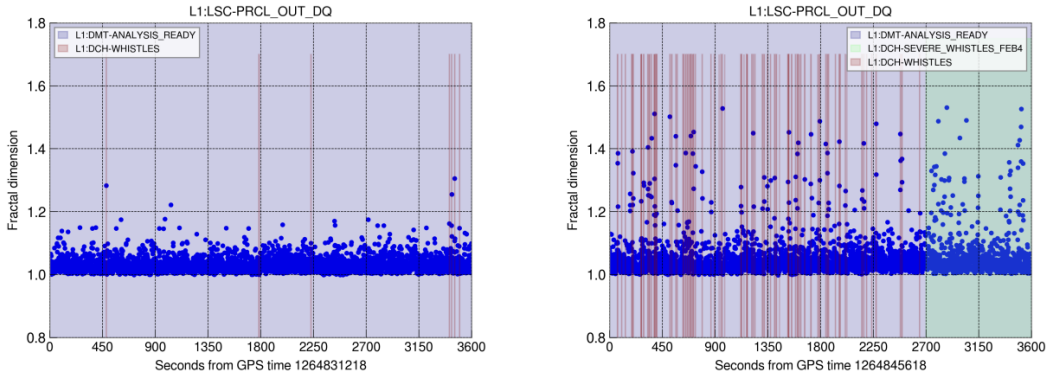


Figure 2.3: FD plot for the L1:LSC-PRCL_OUT_DQ auxiliary witness channel. On the left plot nominal interferometer observing mode. On the right plot data with a growing rate of glitches. [Cavaglia, 2022]

In the left plot of Figure 2.3 it is clear that the nominal FD is static and D_F ranges between 1 and 1.2. Investigating the Q-scans of the noise points with $D_F > 1.2$ shows, at least for three out of four points, a clear whistle signal is diagnosed. In the right plot a big amount of noisy glitches is shown, this renders the data unusable for GW analysis. Their experiment was done on another data strain, showing similar results. They conclude that

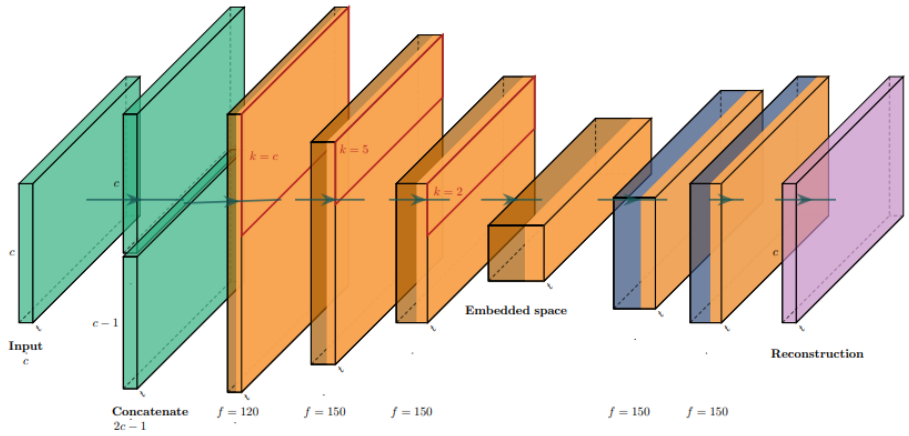


Figure 2.4: Autoencoder architecture with periodic convolutions implementation. [Laguarta et al., 2023]

FD can be used to investigate the stationarity (or non-) of the interferometer in the company of noise. Periods with a lot of (high) frequency noise show an increase in FD variability [Cavaglia, 2022].

Researchers [Laguarta et al., 2023] use a convolutional autoencoder approach to cope with the absence of a ground truth and the absence of absolute order (variety of magnitudes). Although there are many studies on glitch morphologies, we are not sure whether the proposed class is the real class. Autoencoders have the ability to discover new structures without prior knowledge [Bank et al., 2023]. The absence of absolute ordering can be tackled via the application of periodic convolutions. The autoencoder architecture is seen in Figure 2.4, each convolution layer is accompanied by a ReLU (Rectified Linear Unit) activation function. Despite the lowered dimensionality in the autoencoder, it is still very high. The t-SNE method can be applied for further dimensionality reduction. The results indicated that the autoencoder reconstruction error was highest in instances of the `Scattered_Light` class. Safe auxiliary channel FD analysis can be beneficial for noise detection and classification. Extending the proposed technique to other interferometers and investigating possible safe auxiliary channels - data strain glitch correlations is recommended.

2.1.6. GLITCH IDENTIFICATION FROM ENVIRONMENTAL DATA

A study by [Colgan et al., 2020] extracts useful statistical information from auxiliary channels in the vicinity of a glitch (in a 2.5 second timewindow), the acquired data is then used in a logistic regressor classifier. To limit the amount of weights, elastic net regularization [Zou and Hastie, 2005] is applied for penalisation. The results vary around 80 % accuracy, but it does not predict the glitch class, only if it is a possible glitch or not.

2.1.7. LIGO'S FOURTH OBSERVING RUN

The initial evaluation of the fourth observing run (04) highlighted constraints in the architecture of the Gravity Spy machine learning model. Its simplicity led to challenges in generalization and the incapacity to effectively process inputs from multiple time windows [Wu et al., 2024]. The drawbacks of the Gravity Spy classifier, as indicated in [Alvarez-Lopez et al., 2023; Jarov et al., 2023], primarily include the significance of the location within the

images. A line positioned at the top or bottom of the images is crucial as it represents various glitches at different frequencies. This characteristic poses difficulties for the traditional CNN architecture in capturing distinctive features in glitch classes. Additionally, the shallow network structure necessitates substantial resizing of input dimensions, leading to a loss of information. Finally, the classifier struggles with an issue of confidence overfitting. In their study, researchers [Wu et al., 2024] suggest a new architecture and compare three fusion strategies.

2.1.8. INCREMENTAL LEARNING

2.1.8.1. INTRODUCTION

In IL or also called CL, we feed the data progressively to the model and thus acquire new knowledge over time (just as humans do). It is especially useful and efficient in learning from sequential, non-stationary data, such as wave signals. [van de Ven et al., 2022] illustrates the three scenarios of IL in terms of input X, context C and within-context output space Y.

Scenario	**Description**	**Mapping**
Task-incremental learning	sequential learning of distinct task solutions.	$f : X \times C \mapsto Y$
Domain-incremental learning	Problem-solving in different contexts.	$f : X \mapsto Y$
Class-incremental learning	Incrementally observed class discrimination	$f : X \mapsto C \times Y$

Table 2.1: Incremental learning scenarios at computational level from [van de Ven et al., 2022]

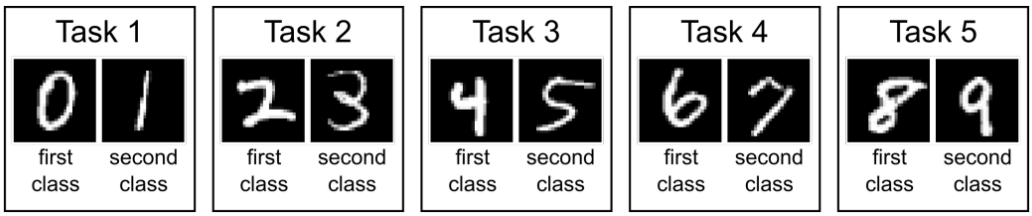


Figure 2.5: Schematic view of an MNIST task protocol, from [Van de Ven and Tolias, 2019]

Looking at figure 2.5 we can split MNIST according to the three scenarios as follows:

Scenario	**Description**
Task-incremental learning	With task given, is it the first or second class? (e.g. 0 or 1)
Domain-incremental learning	With task unknown, is it a first or second class? (e.g. in [0,2,4,6,8] or in [1,3,5,7,9]?
Class-incremental learning	With task unknown, which digit is it? (e.g. choice from 0 to 9)

Table 2.2: Split MNIST according to three scenarios from [Van de Ven and Tolias, 2019]

The first scenario, Task-incremental learning (or Task-IL), involves an algorithm learning a set of distinct tasks sequentially, where the task identity is explicitly provided or clearly distinguishable. Models can be trained with task-specific components, or even use separate networks for each task. This approach avoids the problem of catastrophic forgetting (adapting to a different distribution usually leads to a significant decrease in the ability to capture the original ones [Wang et al., 2023]) altogether [Masse et al., 2018; Ruvolo

and Eaton, 2013]. But the main challenge is to effectively share the learnt representations while optimising the performance-complexity trade-off [Lopez-Paz and Ranzato, 2017; Vogelstein et al., 2020].

The second scenario, Domain-incremental learning (or Domain-IL) encompasses an algorithm's ability to learn a sequence of tasks characterised by the same underlying problem structure (same classes) but varying contextual and in input distributions [Ke et al., 2021; Mirza et al., 2022]. Unlike the first scenario, the algorithm is not explicitly informed about the task at the test time. However, the output space remains consistent between tasks, indicating generalisability to unseen tasks [Aljundi et al., 2017]. As explained in [Mirza et al., 2022] domain-incremental learning can be used to learn the same set of objects (classes) over a variety of weather conditions (domains), while having access to the task-ID (current weather), this is illustrated in Figure 2.6.

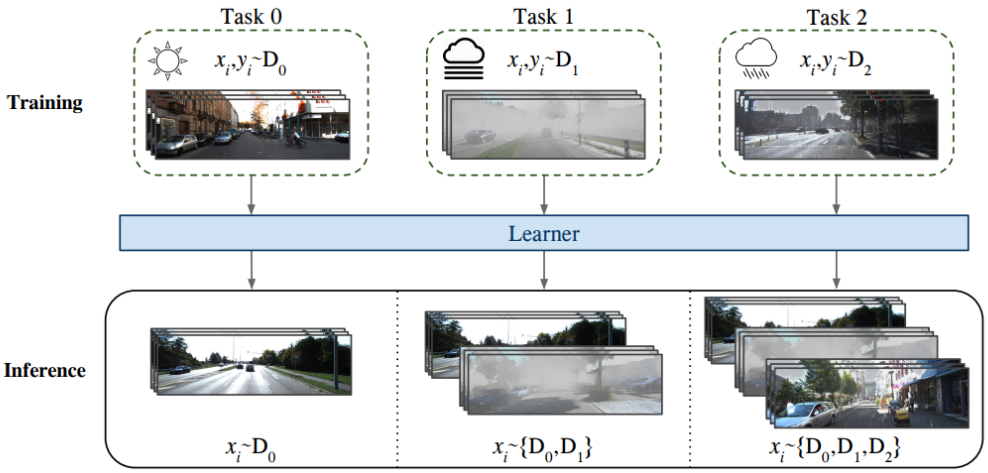


Figure 2.6: Domain-incremental learning setting. Different tasks correspond to different weather conditions which are learned in the training phase. During inference, the performance of the current task is evaluated along with all the previously seen tasks. [Mirza et al., 2022]

The final scenario, Class-incremental learning (Class-IL), involves an algorithm learning to distinguish between a sequentially expanding set of classes or objects. The common setup involves encountering a sequence of classification tasks, where each task introduces new classes, and the algorithm must learn to classify samples across all accumulated classes [Van de Ven et al., 2020; Von Oswald et al., 2019]. Task identification is crucial here as it determines the set of potential classes for a given sample. In essence, the algorithm must master both within-episode classification (distinguishing between classes in the same episode) and between-episode classification (identifying which episode a sample belongs to). This scenario presents a significant challenge to overcome catastrophic forgetting (see 2.2.2), as the algorithm must not only maintain previously learned classes, but also adapt to new ones without degrading performance on previously learned classes [Belouadah et al., 2021; Masana et al., 2022].

As illustrated in Figure 2.7, Task-IL involves having a task identifier available during inference, eliminating the need for methods to differentiate between classes from various tasks. On the other hand, in Class-IL, the learner does not have access to the task identifier during inference, requiring it to differentiate between all classes across tasks.

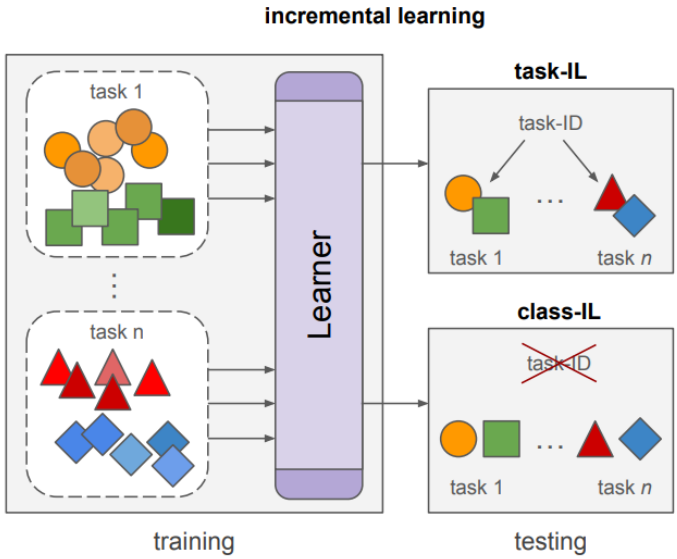


Figure 2.7: In incremental learning, disjoint tasks are learned sequentially. Task-IL has access to the task identifier during evaluation, while class-IL does not. [Masana et al., 2022]

Empirical comparison of some DL training strategies was executed on the MNIST [Deng, 2012] and CIFAR-100 [Krizhevsky et al., 2009] datasets. A baseline network with two fully connected hidden layers with a softmax output layer was used to compare performance. The XdG and the separate network methodology [Masse et al., 2018] require context knowledge of the training record, so they can only be applied in Task-IL scenarios. Regularisation methods such as EWC [Kirkpatrick et al., 2017] employ a quadratic penalty term for each previously learnt context, effectively preventing parameter changes that would significantly deviate from the values acquired during context-specific training. Task-IL scenarios performed equally well as opposed to the baseline method, domain-IL and especially class-IL had a considerable drop in performance. The parameter regularisation implied a significant reduction. Replay-based algorithms had the best results for all scenarios of IL. [Khare et al., 2021] found that the use of class imbalance improves detection in class-IL approaches.

2.2. Background

2.2.1. CL Concepts and Formulation

We follow the framework proposed by [Lesort et al., 2020]. In the context of continual learning, the data can be considered as coming from a (potentially infinite) sequence $\mathbf{D}$ of distributions that are not known in advance. This sequence, denoted as $\mathbf{D} = \{D_1, ..., D_N\}$, is defined over the random variables X and Y, where X represents the input and Y represents the output. At each time step i, the algorithm receives a training set Tr_i consisting of one or more observations drawn from the distribution D_i. A learning task is marked by a distinct task label $t \in \{1, \ldots, T\}$ and its intended goal or objective function h^*.

An algorithm with the signature A^{CL}, also known as a Continual Learning algorithm, is defined as follows:

$$A_i^{CL} : \langle h_{i-1}, Tr_i, M_{i-1}, t_i \rangle \leftarrow \langle h_i, M_i \rangle \quad , \forall D_i \in \mathbf{D} \tag{2.1}$$

In this equation, h_i represents the current model or hypothesis that is continuously learned at time i. M_i is an external memory buffer that stores previous training examples. t_i is a task label associated with the training examples, and Tr_i represents the training set of examples.

CL methodologies are specific settings where the sequence of N task labels follows a defined pattern over time. There are three commonly observed situations:

- Single Incremental Task (SIT): the task remains the same throughout.
- Multi Task (MT): each new task is different from the previous one.
- Multi Incremental Task (MIT): the model may encounter a combination of new and old tasks, which can be presented in an interleaved manner.

Task incremental scenarios assume that task labels are available for each sample. However, it is challenging to obtain explicit task labels for each sample in real-world applications [Cossu et al., 2021].

The classification of CL strategies can be categorised into three distinguishable groups: regularisation strategies, architectural strategies, and replay strategies [Parisi et al., 2019].

Regularisation strategies aim to achieve a balance between plasticity and stability by incorporating regularisation terms into the loss function. Example regularisation strategies are EWC (see 2.2.3.1), SI (see 2.2.3.2) and LwF (see 2.2.3.3).

The goal of a CL algorithm is to learn how to accurately classify, at any specific training stage, examples from all of the observed tasks up until the current one $t \in \{1, \ldots, t_c\}$:

$$\underset{\theta}{\operatorname{argmin}} \sum_{t=1}^{t_c} \mathbf{L}_t$$

where

$$\mathbf{L}_t = \mathbb{E}\left(L\left(y, f_\theta(x)\right)\right)$$

and $\mathbb{E}$ is the expectation over data sampled from a distribution $\mathbf{D}_t$.

2.2.2. Catastrophic forgetting

While ML models often achieve or even exceed human-level performance on specific tasks, they lack the ability to adapt dynamically to new data. Unlike humans, who learn concepts sequentially and reinforce their understanding by observing new examples, neural networks suffer from a phenomenon known as catastrophic forgetting [French, 1999]. This means that they need to be retrained from scratch every time new data becomes available. The dilemma (or trade-off) of Stability-Plasticity pertains to the capacity to incorporate new information while preserving past knowledge during the process of recoding [Grossberg, 2012]. Neural networks often tend to be skewed towards too much plasticity (integrating new knowledge) [De Lange et al., 2021]. Forgetting is, not only related to machines, it is part of the very nature of learning (e.g. forgetting biased information). The decay of forgetting can be described by a power law. When a person is taught a specific concept, if there is no repetition or emotional significance attached to it, the information is quickly forgotten but not entirely erased. Over time, forgetting follows a declining pattern [Pashler et al., 2009].

2.2.3. CL TRAINING METHODS

In the subsequent section, we will outline a few of the frequently employed training techniques. Most of the training strategies follow the guidelines of General Continual Learning (GCL) as proposed by [Farquhar and Gal, 2018].

- Do not rely on task boundaries.
- No oracle with task identifiers during testing.
- A bounded constant memory throughout training.

2.2.3.1. ELASTIC WEIGHT CONSOLIDATION

The study by [Kirkpatrick et al., 2017] and [De Lange et al., 2021] explains EWC is a regularization method designed to address the issue of catastrophic forgetting in neural networks. It achieves this by constraining the learning process. The main concept behind EWC involves incorporating a quadratic penalty term into the standard loss function. This penalty term takes into account the difference between the current weight values and the optimal weights that were obtained during the previous task learning phase. By doing so, EWC minimises interference between tasks and ensures a balance between learning new tasks and retaining knowledge from previous tasks. The concept of Elastic Weight Consolidation

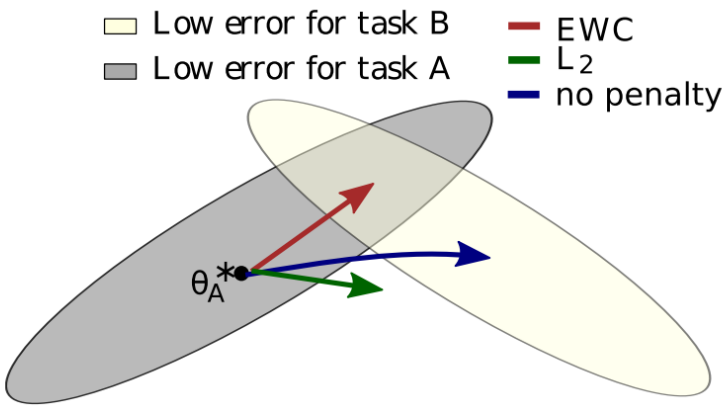


Figure 2.8: EWC guarantees task A is remembered while training on B. [Kirkpatrick et al., 2017]

(EWC) allows for the retention of knowledge from task A while training on task B. The training process is represented in a conceptual parameter space (Figure 2.8), where regions of parameters resulting in good performance on task A are depicted in grey, and those for task B are depicted in cream color. Once task A is learned, the parameters are located at θ_A^* (see Figure 2.8). However, when considering the gradient steps of task B (blue arrow in Figure 2.8), the loss of task B is minimized but at the expense of compromising the knowledge acquired from task A, this is equivalent to "no penalty":

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathbf{L}_B(\theta)$$

To prevent forgetting the learned knowledge from task A, a distance minimization technique is applied between θ and θ_A^* (L_2 in Figure 2.8):

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathbf{L_B}(\theta) + \frac{1}{2}\alpha(\theta - \theta_A^*)^2$$

Here, α is a scalar that determines the relative importance of the old task compared to the new one. The constraint of L_2 is highly restrictive and has the capacity to hinder the learning process of task B. In the realm of deep learning, it is common practice to excessively parameterize models. Some parameters may be less beneficial while others hold greater significance.

EWC discovers a solution for task B without significantly impairing task A's performance (red arrow in Figure 2.8) by explicitly calculating the importance of each weight for task A. This importance value, also called the Fisher Information matrix[1], quantifies the weight's contribution to the performance on previously learned tasks. Weights with higher importance values have a greater impact on the performance of the previous tasks. The learning process is formulated as follows:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathbf{L_B}(\theta) + \frac{1}{2}\lambda I_{\theta^*_A,i}(\theta_i - \theta^*_{A,i})^2$$

In the above equation, $L_{\theta^*_A,i}$ represents the diagonal elements of the Fischer information matrix, and λ is the hyperparameter that determines the strength of the regularization penalty. The implementation operates by computing the Fisher information matrix I_i for each batch B_i. This is achieved by conducting a forward and backward propagation of the patterns. Subsequently, the implementation stores the obtained I and the optimal weights θ.

2.2.3.2. Synaptic Intelligence

According to [Zenke et al., 2017] and [Maltoni and Lomonaco, 2019], the introduction of SI as a modification to EWC involved performing the computationally expensive Fisher matrix computations online during SGD throughout the learning process. The implementation of this approach involves calculating the weight importance matrix I_i for each batch of data samples B_i using the information already available during SGD. The weight importance matrix I and the optimal weights θ are then stored.

2.2.3.3. Learning without Forgetting

The aim of the LwF approach is to manage the process of forgetting by enforcing output stability (Maltoni et al., 2019). It follows a similar approach to parameter regularization methods like EWC (see Section 2.2.3.1) and SI (see Section 2.2.3.2), where a regularization term is added to the classification loss function **L**. Although originally designed for classification tasks, it has also been successfully utilized in object detection [De Lange et al., 2021].

2.2.3.4. FROMP

The FROMP training strategy is a novel approach in CL addressing the catastrophic forgetting problem. It regularises the network outputs at a few memorable past examples that are crucial to avoid forgetting. By using a Gaussian process formulation of deep networks, the approach enables training in weight-space while identifying both the memorable past and a functional prior. It involves a slight modification of the Adam optimizer and a minor increase in computational cost. [Pan et al., 2020]. A schematic representation of FROMP is depicted in figure 2.9.

[1] The Fisher Information Matrix provides a curvature approximation of the loss function [Ly et al., 2017]

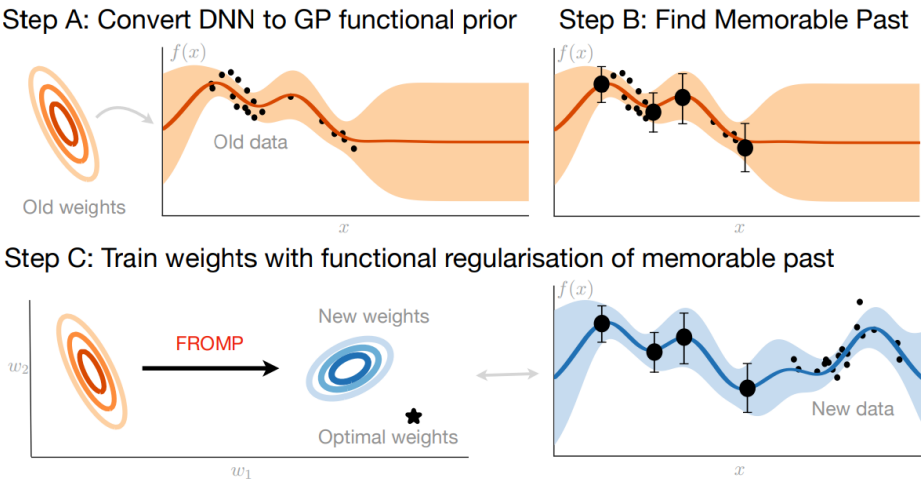


Figure 2.9: The FROMP method consists of three main steps where they first convert a DNN (Deep Neural Network) to a GP (Gaussian-Process), find memorable examples, and train weights with functional regularisation of those examples. [Pan et al., 2020]

2.2.3.5. GEM

The Continual Learning (CL) strategy, proposed by [Lopez-Paz and Ranzato, 2017], is a model that overcomes the limitations imposed by Empirical Risk Minimization [Vapnik, 1991] on other supervised learning techniques. To bridge this gap, the Gradient Episodic Memory (GEM) model incorporates an episodic memory M_t that selectively stores a subset of observed examples from a specific task t. The memory capacity is limited to a total of M locations, with each task having $m = \frac{M}{T}$ memories. The examples stored in M are utilized to determine predictor functions f_θ by minimising the corresponding loss function

$$\mathbf{L}(f_\theta, M_k) = \frac{1}{|M_k|} \sum_{(x_i,k,y_i)\in M_k} \mathbf{L}(f_\theta(x_i,k), y_i)$$

Overfitting to the stored examples in M_k is a common issue when minimizing the loss at the current example along with the above loss function. To address this problem, they solve the following optimization problem:

$$\underset{\theta}{\operatorname{argmin}}\, \mathbf{L}(f_\theta(x,t),y) \text{ subject to } \mathbf{L}(f_\theta, M_k) \le \mathbf{L}(f_\theta^{t-1}, m_k), \forall k < t$$

Here, f_θ^{t-1} represents the previous predictor state at the end of learning task $t-1$. The detailed solution can be found in the original paper by [Lopez-Paz and Ranzato, 2017].

2.2.3.6. A-GEM

[Chaudhry et al., 2018b] proposes a more efficient version of GEM where it tries to ensure that at every training step the average episodic memory loss over the previous tasks does not increase. Formally the objective of A-GEM is

$$\underset{\theta}{\operatorname{argmin}}\, \mathbf{L}(f_\theta D_t) \text{ such that } \mathbf{L}(f_\theta, M) \le \mathbf{L}(f_\theta^{t-1}, M), \text{ where } M = \bigcup_{k<t} M_k$$

The detailed solution and update rule can be found in the original paper by [Chaudhry et al., 2018b].

2.2.3.7. iCaRL

As described in [Rebuffi et al., 2017] iCaRL (Incremental Classifier and Representation Learning) allows learning in a class incremental way as depicted in figure 2.10. It learns strong classifiers and a data representation simultaneously. In iCaRL, the model learns to recognize new classes while retaining its ability to recognize old ones. This is achieved by storing examples of each class, which are selected based on their closeness to the mean feature of that class. The model is then trained on both the new data and these examples. This approach ensures that the model does not forget the old classes when learning new ones.

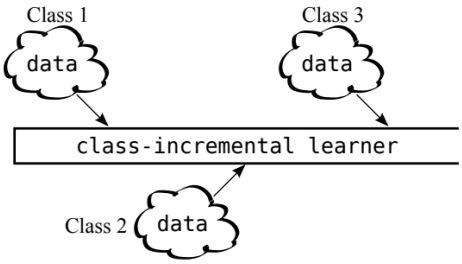


Figure 2.10: Class-incremental learning. [Rebuffi et al., 2017]

iCaRL utilizes a CNN as its underlying mechanism. The CNN functions as a feature extractor that can be trained, denoted as $\phi : X \to \mathbf{R}^d$, followed by a single classification layer containing sigmoid output nodes equal to the number of classes encountered up to that point. All feature vectors undergo L^2 normalization. The classification approach employed by iCaRL is based on a nearest-mean-of-exemplars strategy.

In order to predict a label, y^*, for a new input, x, the system calculates a prototype vector for each class seen so far, represented as $\mu_1, ..., \mu_t$, where μ_y represents the average feature vector of all examples belonging to class y. It also computes the feature vector of the image that should be classified and assigns the class label with the most similar prototype:

$$y^* = \underset{y=1,...,t}{\operatorname{argmin}} \|\phi(x) - \mu_y\|$$

2.2.3.8. SCR

Supervised Contrastive Replay (SCR) is a technique introduced by [Mai et al., 2021] to address some challenges common in CL. Because the softmax classifier is not the best choice for CL approaches, it aims to leverage the Nearest-Class-Mean (NCM) classifier instead to effectively mitigate catastrophic forgetting (or also called task-recency bias).

The SCR process (see Figure 2.11) involves the creation of an input batch during training by combining a minibatch B_n from the datastream with another minibatch B_M from the memory buffer M_i.

These input batches, along with their augmented views, are encoded by a shared encoder and projection head. Followed by an evaluation using supervised contrastive loss. In the testing phase, the projection head is removed. Supervised Contrastive Loss is an alternative loss function to cross entropy that can leverage label information more effectively. [Khosla et al., 2020].

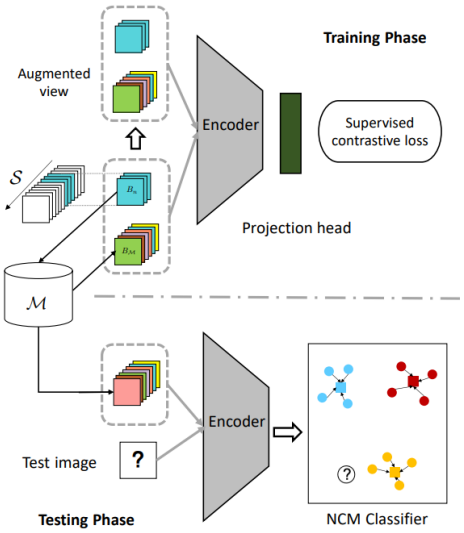


Figure 2.11: A schematic view of SCR. [Mai et al., 2021]

2.2.3.9. DER AND DER++

Dark Experience Replay (DER) is proposed by [Buzzega et al., 2020] as an improvement on Experience Replay (ER) while maintaining a simple formulation. It relies on dark knowledge for distilling past experiences, sampled over the training trajectory.
Formally, given a CL classification split in T tasks

It is problematic that the data from earlier tasks $\mathbf{D}_t$ for $t \in \{1, \ldots, t_c - 1\}$ is unavailable. To preserve the knowledge acquired previously, the subsequent objective function, which mirrors the teacher-student method, is minimized:

$$\mathbf{L}_{t_c} + \alpha \sum_{t=1}^{t_c-1} \mathbb{E}\left[D_{KL}\left(f_{\theta_t^*}(x) \parallel f_\theta(x)\right)\right]$$

where θ_t^* is the optimal set of parameters at the end of task t, α is a hyperparameter balancing the trade-off between the terms. D_{KL} stands for the Kullback-Leibler divergence, a measure of how one probability distribution diverges from a second, expected probability distribution. It is commonly used to quantify how much knowledge from previous tasks is retained or lost when learning new tasks [Goodfellow et al., 2016].
The above objective function requires having access to $\mathbf{D}_t$ for earlier tasks. To address this issue, a replay buffer M_t is utilized to store previous experiences related to task t. Instead of preserving the actual labels y, the network logits (pre-activation outputs) $z \doteq h_{\theta_t}(x)$ are maintained.

$$\mathbf{L}_{t_c} + \alpha \sum_{t=1}^{t_c-1} \mathbb{E}\left[D_{KL}\left(\sigma(z) \parallel f_\theta(x)\right)\right]$$

where σ is the softmax function and (x, z) are drawn from the replay buffer M_t.
DER uses reservoir sampling as defined by [Vitter, 1985] to select $|M|$ random samples from the input stream, this way guaranteeing the same probability of being stored in the buffer without knowing the length in advance. This way the previous equation (with $\mathbb{E}$ drawn from M) becomes:

$$\mathbf{L}_{t_c} + \alpha\, \mathbb{E}\left[D_{KL}\left(\sigma(z) \parallel f_\theta(x)\right)\right]$$

Given some basic assumptions, as mentioned in [Hinton et al., 2015], maximizing the KL divergence corresponds to minimizing the Euclidean distance among the logits. Consequently, the target function for DER is defined as

$$\mathbf{L}_{t_c} + \alpha\, \mathbb{E}\left[\|z - h_\theta(x)\|_2^2\right]$$

with $\mathbb{E}$ being approximated by gradient computation on batches sampled from the replay buffer [Boschini et al., 2022; Buzzega et al., 2020].
DER++ adjusts the objective function by adding an additional balancing term β on alternative 'buffer' data points

$$\mathbf{L}_{t_c} + \alpha\, \mathbb{E}\left[\|z_1 - h_\theta(x_1)\|_2^2\right] + \beta \mathbb{E}\left[\mathbf{L}\left(y_2, f_\theta(x_2)\right)\right]$$

DER++ collapses to DER when the coefficient $\beta = 0$.
Experiment results, as described in [Buzzega et al., 2020] conclude that DER converges to more flat minima and to more calibrated networks.

2.2.4. CL METRICS

Evaluation of learning tasks is a major part of research. As [Qu et al., 2021] indicates, not only the performance on new tasks, but also the amount of forgotten wisdom must be considered. We follow the notation used in [Chaudhry et al., 2018a; Díaz-Rodríguez et al., 2018; Lopez-Paz and Ranzato, 2017] to define the most popular metrics.
We denote $a_{q,p}$ as the accuracy on the separate test set of the task p after IL of q previous tasks, where $p \leq q$.

2.2.4.1. AVERAGE ACCURACY (A_q)

The metric A_q, as explained in [Wang et al., 2023], is utilized to assess the effectiveness of the IL model. It is calculated by averaging the accuracies obtained after learning a total of q tasks.

$$A_q = \frac{1}{q} \sum_{p=1}^{q} a_{q,p} \tag{2.2}$$

This method is employed to assess the capacity to retain information from previous tasks while acquiring new knowledge.

2.2.4.2. AVERAGE FORGETTING (F_q)

The metric F_q, which is described in [Qu et al., 2021], quantifies the amount of knowledge that is forgotten during the first $q-1$ tasks. It is calculated using the formula

$$F_q = \frac{1}{q-1} \sum_{p=1}^{q-1} f_p^q \tag{2.3}$$

where f_p^q represents the knowledge forgotten for task p. This is determined by subtracting the remaining knowledge after learning q tasks from the maximum knowledge obtained during the IL process. In eq. 2.3, f_p^q is calculated as follows:

$$f_p^q = \max_{o \in \{1,\ldots,q-1\}} a_{o,p} - a_{q,p}, \forall p < q. \tag{2.4}$$

2.2.4.3. INTRANSIGENCE (I_q)

The metric called Intransigence (or learning plasticity) quantifies the extent to which CL hinders a model from learning a new task in comparison to traditional batch learning. It is calculated as follows:

$$I_q = a_q^* - q_{q,q} \tag{2.5}$$

In eq. 2.5, a_q^* represents the accuracy achieved on the test set of the q^{th} task when batch learning is employed for all q tasks [Chaudhry et al., 2018a].

2.2.4.4. BACKWARD TRANSFER (BWT_q)

The concept of backward transfer, as described in [Lopez-Paz and Ranzato, 2017], is how much the learning process on task q affects the performance of previously learned tasks. This can be quantified using the following equation:

$$BWT_q = \frac{1}{q-1} \sum_{p=1}^{q-1} (a_{q,p} - a_{p,p}) \tag{2.6}$$

2.2.4.5. FORWARD TRANSFER (FWT_q)

Forward transfer refers to the degree to which the continuous learning of the q^{th} task has the potential to influence the performance of future tasks. In [Lopez-Paz and Ranzato, 2017], the accuracy on the test set of the p^{th} task at random initialization is represented as b_p. The forward transfer metric, denoted as FWT_q, is defined as follows:

$$FWT_q = \frac{1}{q-1} \sum_{p=2}^{q} (a_{p-1,p} - b_p) \tag{2.7}$$

[Wang et al., 2023]

2.2.5. MULTIMODAL FUSION

DL models increasingly interact through multiple modalities, such as vision, text, touch, ... Each modality provides a unique perspective on the environment. Multimodal Fusion is the technique for integrating information from different modalities, in case of glitch detection this could be the spectrogram image and the calculated fractal dimensions. By effectively fusing information, models can achieve superior performance in various tasks. [Gao et al., 2020; Ngiam et al., 2011; Zhang et al., 2021] Fusing different modalities is one of the challenges in multimodal learning. There are two approaches (see Figure 2.12 commonly used, 'Early Fusion' and 'Late Fusion'. Depending on the location of the fuse operation.

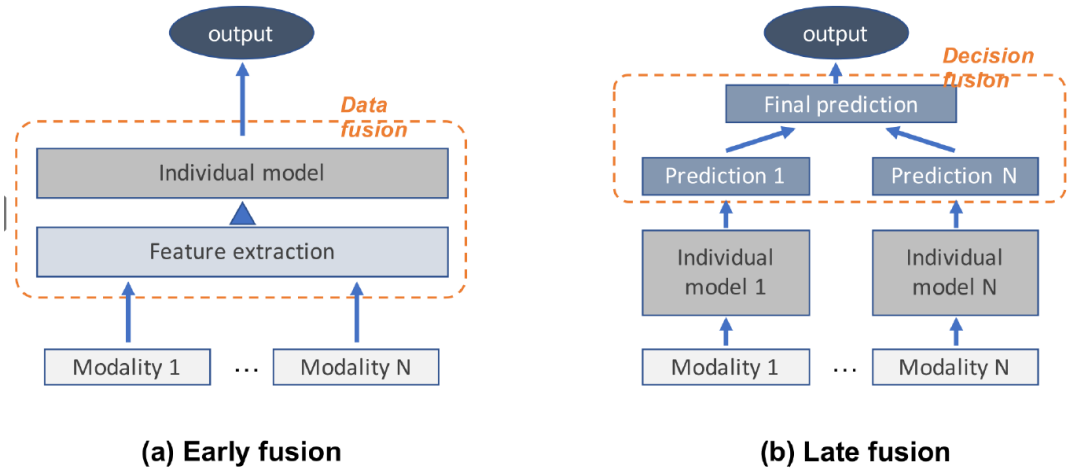


Figure 2.12: Early vs Late Fusion [Kim et al., 2021]

To define Early and Late Fusion, let M represent the total number of modalities. Each modality input is represented as a vector $\mathbf{v}_m$. We assume a K-class classification scenario where y represents the labels. The prediction probability of the k-th class from the m-th modality is denoted by p_m^k, and the final prediction probability of the k-th class by the model is denoted by p^k. [Liu et al., 2018]

With Early Fusion, features from different modalities are combined at the input level. If for instance we have a text and image, early fusion merges the raw text and image features before feeding them into the model. This assumes that all modalities have compatible representations.

In Early Fusion, a unified model h is trained to capture the relationships and interactions among low-level features. The resulting prediction is given by

$$p = h([\mathbf{v}_1, ..., \mathbf{v}_m])$$

Early Fusion has a simple pipeline because only one model is needed for training.

With Late Fusion, each modality is processed independently and their features are combined at a later stage. This allows each modality to learn its own representations and then combines them.
Late Fusion combines unimodal decision values through a fusion mechanism F (such as averaging, voting, etc.). Given that h_i is applied to modality i, the final prediction is

$$p = F(h_1(\mathbf{v}_1), ..., h_m(\mathbf{v}_m))$$

This approach permits the utilization of various models, providing greater adaptability.

Usually, specialized networks are employed for various modalities to produce their respective representations, which are then combined or aggregated. Subsequently, a prediction is made based on the aggregated representation, typically using another neural network to capture the interactions between the different modalities. Summation (or averaging) and concatenation are two frequently used aggregation techniques.

$$\mathbf{u} = \left[f_1(\mathbf{v}_1), ..., f_m(\mathbf{v}_m)\right]$$

where f_m is a domain specific neural network. Given the combined vector output $\mathbf{u}$, another network g computes the final output

$$p = g(\mathbf{u})$$

[Gadzicki et al., 2020; Liu et al., 2018; Wang et al., 2022]

2.2.6. PYTHON PACKAGES

A recent toolbox called `PyCIL` [Zhou et al., 2023a, b] implements several state-of-the-art IL algorithms. The package `Avalanche` [Lomonaco et al., 2021] provides a library for benchmarking, training, prototyping, training and evaluation of CL algorithms. Continuum [Douillard and Lesort, 2021] is a library that provides data loading functionalities for CL.

2.2.6.1. AVALANCHE

The Avalanche package came into existence with a clear goal in mind, *"Pushing Continual Learning to the next level, providing a shared and collaborative library for fast prototyping, training and reproducible evaluation of continual learning algorithms."* [ContinualAI, 2024]

Avalanche is structured based on a series of core design principles: 1) minimizing code length for rapid prototyping and error minimization, 2) ensuring reproducibility, 3) promoting modularity and reusability, 4) enhancing code efficiency, scalability, and portability and 5) prioritizing usability.

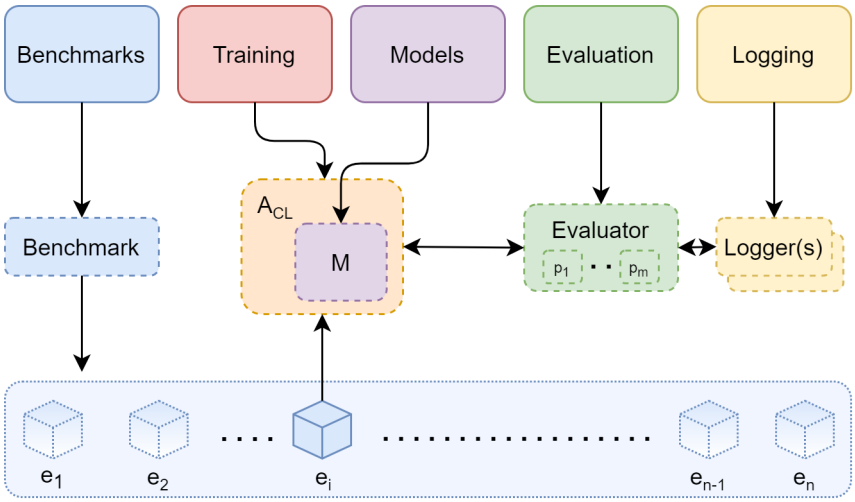


Figure 2.13: The main Avalanche modules and how they interact with each other. [ContinualAI, 2024]

Avalanche is organized in five main modules denoted in figure 2.13.

1) **Benchmarks**: This component preserves a consistent application programming interface (API) for managing data, primarily by producing a sequence of data from multiple datasets. It contains the primary CL performance metrics (similar to the approach taken in `torchvision`).

2) **Training**: This module offers a comprehensive range of tools related to model training. It encompasses straightforward and effective methods for incorporating new continual learning strategies, along with a collection of pre-implemented CL baselines and cutting-edge algorithms that can be utilized for comparison purposes.

3) **Evaluation**: This module offers a comprehensive set of tools and measurements that can help evaluate a CL algorithm in relation to the various factors that we consider crucial for a system that learns continuously.

4) **Models**: In this module, you will discover various model structures and pre-trained models suitable for your ongoing learning experiment (similar to what has been done in `torchvision.models`).

5) **Logging**: It contains sophisticated logging and visualization functionalities, such as built-in support for `stdout`, file, and `Tensorboard` (Effortlessly monitor your experiment metrics in real-time using just one line of code on a comprehensive interactive dashboard).

[Carta et al., 2023; Lomonaco et al., 2021]

2.2.7. Saliency Mapping

The Image-Specific Class Saliency Visualisation is a method described in [Simonyan et al., 2013] for understanding a CNN decision-making process by visualizing the spatial support of a particular class within a given image. The goal is to rank the pixels of an input image based on their influence on the score function, which indicates the likelihood of the presence of a specific class. Given an image I_0, a class label c and a CNN classifying score

function $S_c(I_0)$, we would like to rank the pixels of I based on their influence on the score $S_c(I)$. The linear score model, where the score S_c is determined by the dot product between the weight vector w_c and the image I, and adding a bias term b_c.

$$S_c(I) = w_c^T I + b_c$$

In this model, the importance of individual pixels in the image for the class c is directly proportional to the magnitude of the corresponding elements in the weight vector w_c. However, the class score function is non-linear, therfore the linear model cannot be applied. The purpose of the Class Saliency Visualisation is to address this non-linearity in CNN's. Given an image I_0, the class score $S_c(I)$ can be approximated with a first-order Taylor expansion:

$$S_c(I) \approx w^T I + b$$

where w is the partial derivative of S_c w.r.t. the image I at the point I_0:

$$w = \left.\frac{\partial S_c}{\partial I}\right|_{I_0}$$

The above approach aims to infer the importance of pixels in an image for a specific class despite the non-linear nature of the classifier. This visualization method helps in understanding how the model makes its predictions by highlighting which parts of the image contribute most significantly to the classification of a particular class.

The class saliency map $M \in \mathbf{R}^{m \times n}$ for an image I_0 (of size $m \times n$) and a class c requires a single back-propagation pass through the classification CNN. The elements of the derivative w are then rearranged.

In case of a grey-scale image, the mapping is computed as $M_{ij} = \left|w_{h(i,j)}\right|$ where $h(i,j)$ is the corresponding pixel-index of w in row i and column j.

In case the image is an RGB-image, the color channel c must be taken into account. The authors [Simonyan et al., 2013] opt for the maximum value of w over the different color channels: $M_{ij} = \max_c \left|w_{h(i,j,c)}\right|$

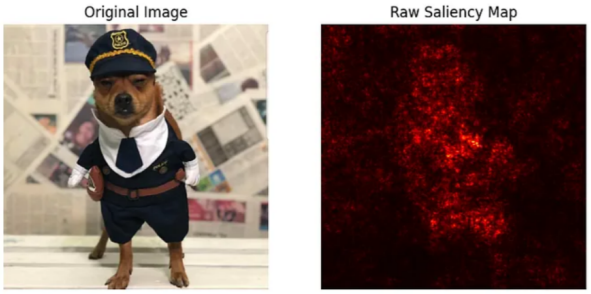


Figure 2.14: Example saliency map and original image. Taken from [Newgin, 2024]

3

RESEARCH METHODOLOGY

3.1. RESEARCH

3.1.1. RESEARCH QUESTIONS

The upcoming research will investigate the following questions:

> RQ1: Do classes learned from a continual learning approach align with existing gravity spy classes?
> RQ2: How effective are continual learning approaches when incorporating auxiliary channel data for detecting and classifying glitch morphologies?
> RQ3: What if strain data (spectrograms) are combined with auxiliary channel data in a multimodal approach? Does this learn a robust classification?

3.1.2. RESEARCH METHOD

3.1.2.1. RQ1

In order to evaluate the alignment between classes acquired from a Continual Learning approach and existing gravity spy classes, we proceed with the following steps:

- Data preprocessing is required for the utilization of a CL approach, involving the selection of either the O3a or O3b dataset that includes labeled data with various glitch categories. This dataset will be segmented into sequential chunks to represent a continuous data stream.

- Model selection and Training: A proper CL algorithm will be chosen, taking into account aspects such as memory usage and its ability to reduce catastrophic forgetting. The model will be trained gradually on the segmented data, mimicking a continual learning scenario.

- In order to assess the alignment, we will compare the classes acquired from the CL model with the existing gravity spy classes. This assessment could include traditional measures like accuracy and confusion matrix, as well as the specific metrics for CL outlined in Section 2.2.4.

3.1.2.2. RQ2

In order to assess how well auxiliary channel-based strategies work for detecting and categorizing glitch shapes, we will utilize the following approach:

- The dataset preparation will be done by gathering glitch-affected and clean signal samples. Useful information needs to be extracted from a selection of auxiliary channels in a limited time window around glitch transients. The approach of [Colgan et al., 2020] will be used.
- A Convolutional-based architecture will be used, together with the same selection of CL algorithms, used in RQ1.
- Evaluating the model's performance is done by using metrics like precision, recall and F1-score. Additionally visualization techniques can be employed to understand the relative feature importance of selected auxiliary channels for glitch classification.

3.1.2.3. RQ3

To investigate if strain data (spectrogram images) are combined with auxiliary channel information gives a more robust classifier, the following steps are used:

- The dataset used in RQ2, together with the spectrograms provided by Melissa Lopez, will be used.
- A multimodal fusion architecture will be used to incorporate auxiliary channel data alongside the spectrogram dataset. The model will be trained to learn robust glitch detection and classification.
- Evaluating the model's performance is done by using metrics like precision, recall and F1-score. Additional visualization techniques such as t-SNE, UMAP and Saliency mapping will be used.

4

MODELS AND RESULTS

4.1. TECHNICAL ASPECTS

The code repository containing all notebooks, models and results can be found via
https://github.com/brianbaert/MscThesis.git

All the experiments were done on a single GPU server with the following specifications:

Processors	4x Intel(R) Xeon(R) 3-2286G CPU @ 4.00GHz
Operating System	Microsoft Windows Server 2022 Datacenter
GPU	NVIDIA RTX A4500 (20GB)
RAM	32 GB

Table 4.1: Server specifications

4.2. DATA DESCRIPTION

4.2.1. GRAVITY SPY DATASET

The dataset includes 23 different types of glitches represented by time-frequency spectrograms. These glitches have been detected using the Omicron trigger pipeline [Robinet et al., 2020], which applies specific criteria such as identifying real-time events with a SNR above 7.5 and a peak frequency ranging from $10Hz$ to $2048Hz$. Subsequently, the strain data is transformed into time-frequency spectrograms using the Q-transform process [Chatterji et al., 2004], these spectrograms are also known as Omega scans. Gravity Spy produces four spectrograms for every detected glitch, utilizing different time windows: 0.5s, 1.0s, 2.0s, and 4.0s. An illustration of a "Koi Fish" anomaly is depicted in figure 4.1, while an instance of a "Whistle" anomaly is displayed in figure 4.2.

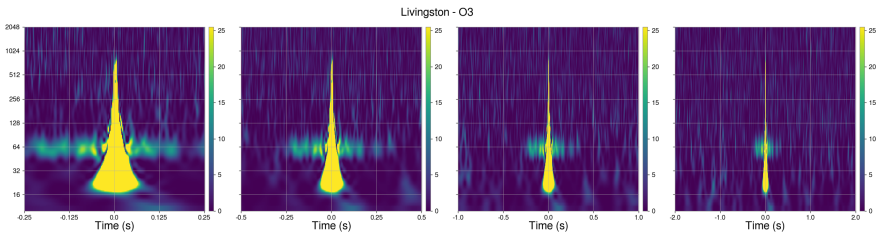


Figure 4.1: Spectrogram example of a Koi Fish glitch with four time windows (0.5s, 1.0s, 2.0s an 4.0s

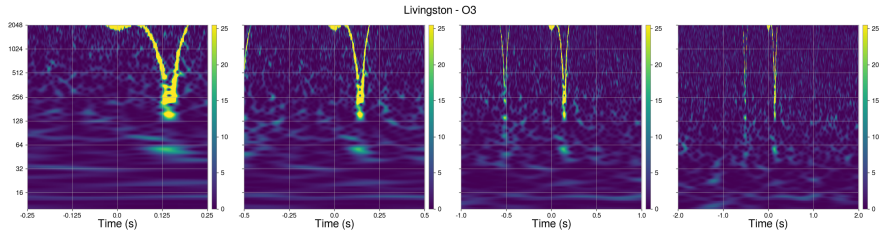


Figure 4.2: Spectrogram example of a Whistle glitch with four time windows (0.5s, 1.0s, 2.0s an 4.0s

The class names were established collaboratively by LIGO and citizen scientists. However, there may be a noisy-label problem in the ground truth, as some misclassifications of glitches occurred by both the classifier and volunteers. Hence, it is essential to focus on glitches with a confidence score exceeding 0.9. The dataset shows an imbalance, with O3a having mostly Fast scattering and Tomte glitches, whereas O3b contains mainly Scattered light or Fast scattering glitches. Preliminary results (obtained from LigoDV-Web[1]) of O4a are mainly Low Frequency burst (45,8%) and Low Frequency lines (33,1%) .

4.2.2. O3 Auxiliary Channel data

As described on the GWOSC[2] website, the O3 auxiliary channel dataset includes channels (more than 200 000 per detector) that were used to create data quality flags, subtract instrumental noise from the strain data and measure the amount of correlated noise between the GW detectors. Data quality flags play a crucial role in the O3 auxiliary channel data, serving to eliminate time periods where data integrity may be compromised by instrument-related or environmental interference. At times, an auxiliary channel may capture extraneous noise sources such as scattered light or power frequency variations, enabling LIGO to enhance the accuracy of strain data by removing such noise components. Conversely, an auxiliary channel can also serve as an indicator of detector glitches, signaling instances of malfunction to analysts for data reliability assessment. Nonetheless, this approach may be flawed if auxiliary channels inadvertently detect disturbances in the gravitational wave channel, leading to the simultaneous appearance of an astrophysical signal in both channels. Channels that detect heightened power levels from the gravitational wave channel are labeled as "unsafe" for vetoing purposes, while those unaffected by this interference are deemed "safe." Vetoing of gravitational wave candidates relies solely on data from "safe" channels.

Each channel is identified by a name following the format `[IFO]:[SYSTEM]-[NAME]` , where `[IFO]` denotes the instrument (L1, H1, ...), `[SYSTEM]` specifies the subsystem (PSL, IMC, TCS, PEM, ...), and `[NAME]` provides a description of the channel. The dataset available on the GWOSC website comprises 40 channels from the collective LIGO detectors (Hanford, Livingston) that were utilized in the O3 analysis. The dataset has a total size of 13 TB, the full channel list can be found via the LIGO git repository[3] [project, 2024]

[1] https://ldvw.ligo.caltech.edu/ldvw/gspySearch
[2] https://gwosc.org/O3/auxiliary/
[3] https://git.ligo.org/gwosc/

4.3. MODEL

4.3.1. MODEL DESIGN AND PARAMETER CHOICES FOR RQ1

The `MultiViewColorNet_resnet18` architecture is a PyTorch model extending the `nn.Module` class. It is designed for a multi-class classification task.

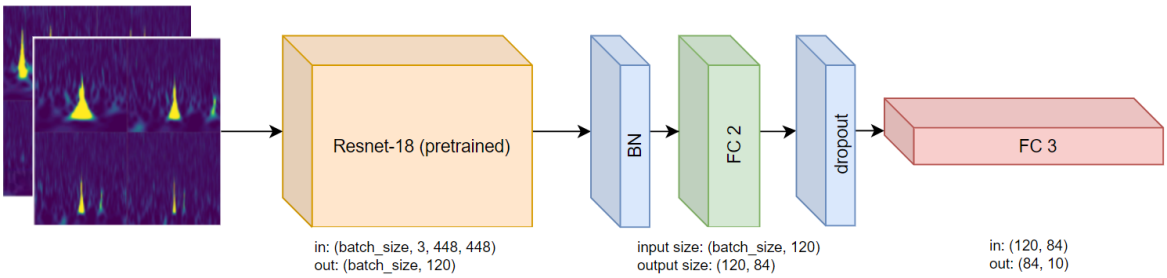


Figure 4.3: MultiviewColorNet_SlimResnet18

The full PyTorch implementation can be found in Appendix 6.2, in short the implementation consists of:

- A pre-trained ResNet-18 model is loaded into `self.resnet`. The weights are set to 'DEFAULT' (pre-trained on ImageNet).
- The kernel size of the first convolutional layer in the ResNet model adapted to (7x7), this is a common practice in ResNet-based architectures [Tomen and van Gemert, 2021] and has been succesfull in medical image classification on imbalanced datasets [Mursalim and Kurniawan, 2021].
- All parameters in the ResNet model are unfrozen, allowing them to be updated during training.
- The last layer of the ResNet model, a fully connected layer, is replaced with a new linear layer that has 120 output features.
- Two additional fully connected layers (`self.fc2` and `self.fc3`) are added, with the final layer outputting a number of features equal to `num_classes`.
- A dropout layer (`self.dropout`) with a dropout probability of 0.3 is added to help prevent overfitting (according to [Park and Kwak, 2017] 0.3 is a good value for more complex CNN's like ResNet).
- A batch normalization layer (`self.bn`) is added to normalize the activations of the neurons in the network.

The train dataset contains a balanced random selection of spectrograms from 10 glitch classes ('Blip', 'Blip_Low_Frequency', 'Extremely_Loud', 'Fast_Scattering', 'Koi_Fish', 'Low_Frequency_Burst', 'Low_Frequency_Lines', 'Scattered_Light', 'Tomte', 'Whistle') (each datapoint consists of a 0.5, 1.0, 2.0 and 4.0 second view of the glitch).

The spectrograms containing glitches are fused via a custom PyTorch DataLoader extending the ImageFolder class. It concatenates the views into a single image of size 448 by

448 (see Figure 4.4 for an example), allowing a model to learn from all views simultaneously. This is particularly useful because different views of a glitch can provide different insights. The final image and its label are returned.

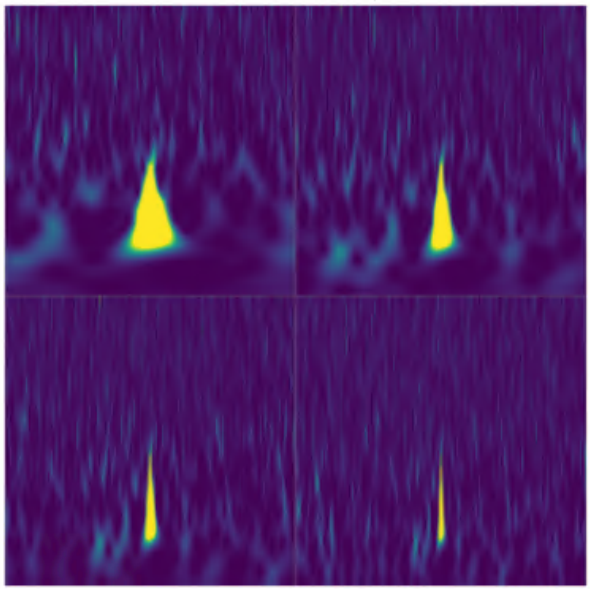


Figure 4.4: Example of a fused image (glitch type 'Tomte').

The Avalanche 2.2.6.1 New Classes benchmark (`nc_benchmark`) is used because it is specifically designed for creating benchmarks in a "New Classes" continual learning scenario, where each experience introduces new classes that were not present in the previous experiences. Here we opt for 5 experiences (2 classes per experience).

The optimizer for the majority of experiments is the `AdamW` optimizer, a variant of the Adam optimizer including weight decay. It is often preferred over standard Adam because it handles weight decay in a more principled manner, leading to better generalization on some tasks [Loshchilov and Hutter, 2019]. The learning rate is set to 0.001 and weight decay to 0.00001. These are common hyperparameter choices for AdamW.

The criterion used for the experiments is CrossEntropyLoss, a common choice for multi-class classification tasks. This loss function combines LogSoftmax[4] and NLLLoss[5] in one single class, making it suitable for training a classification problem with more than 2 classes.

[4]Applies the log(Softmax(x)) function to an input Tensor. [Miranda, 2017]
[5]Negative Log Likelihood Loss [Miranda, 2017]

4.3.2. Model design and parameter choices for RQ2

The `FractalDimensionConvNet` architecture is a PyTorch model extending the `nn.Module` class and uses a combination of convolutional layers, batch normalization, dropout and fully connected layers. It is designed for a multi-class classification task.

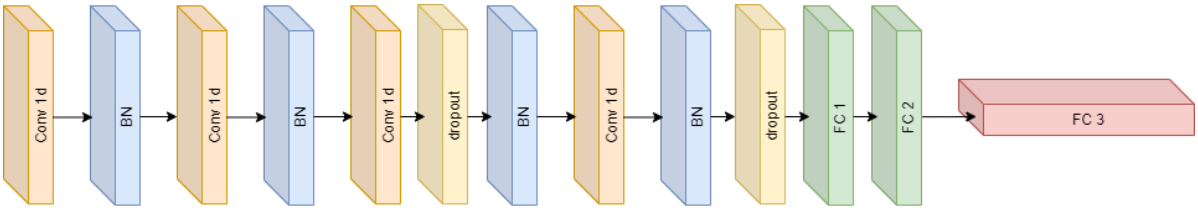


Figure 4.5: FractalDimensionConvNet

The full PyTorch implementation can be found in Appendix 6.2, in short the implementation consists of:

- Four pairs of 1-dimensional convolutional layers (`nn.Conv1d`) are used to process the fractal dimension data into a 1-dimensional format.
- Fully connected Layers are added to flatten the output and output a three-class prediction.
- Two dropout layers (`self.dropout`) with a dropout probability of 0.1 and 0.3 are added to help prevent overfitting.
- Four batch normalization layers (`self.bn`) are added to normalize the activations of the neurons in the network.
- The model uses the SELU (Scaled Exponential Linear Unit) activation function after each convolutional and the first fully connected layer, which can lead to self-normalizing neural networks [Kılıçarslan et al., 2021; Rasamoelina et al., 2020]. The second fully connected layer users the ReLU activation function.

The train dataset contains a balanced (896 entries for each of the three glitch categories) selection of 2688 records where each record contains fractal dimension calculations on 50 auxiliary channels from 3 glitch classes ('Whistle', 'Tomte', 'Scattered_Light'). Each channel has 56 fractal dimension calculations. The dataset selection is based on the study by [Laguarta et al., 2023].

An example visualisation is show in Figure 4.6

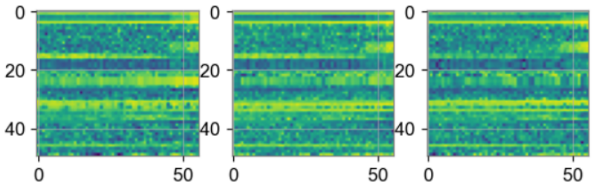


Figure 4.6: Example visualisations of FD matrices with 50 channels. Left = 'Whistle', Middle = 'Tomte' and right = 'Scattered_Light'

The optimizer for the majority of experiments is the `AdamW` optimizer. The learning rate is set to 0.001 and weight decay to 0.00001. Finally, the criterion used for the experiments is CrossEntropyLoss just as in 4.4.1.1.

4.3.3. Model design and parameter choices for RQ3

The `MultiModalNet` architecture is a PyTorch model extending the `nn.Module` class and uses a combination of the `MultiViewColorNet` architecture of RQ1 4.3.1 and the `FractalDimensionConvNet` of RQ2 4.3.2. The outputs of both models are concatenated and send through a fully connected linear layer mapping the outputs to the final output classes.

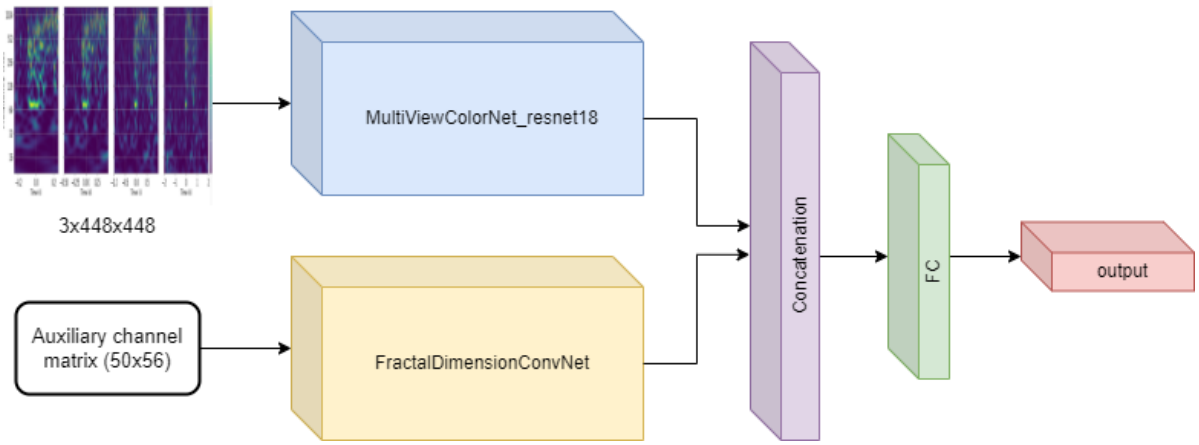


Figure 4.7: MultiModalNet architecture

The model uses the `CrossEntropyLoss` function designed for multiclass classification and the `AdamW` optimizer. The learning rate is set to the default value 0.001 and weight decay to 0.00001.

Due to the fact that the CL package `Avalanche` [Lomonaco et al., 2021] does not provide native Multimodal support, the experimentation is limited to a basic Multimodal pytorch training loop.

4.4. Results

4.4.1. RQ1

Each of the strategies are trained for 100 epochs on every task (of distinguishing between two glitch categories). The order in which the classes are fed is kept the same over all strategies (1 = {Koi_Fish; Low_Frequency_Burst}, 2 = { Tomte; Extremely_Loud }, 3 = { Whistle; Fast_Scattering }, 4 = { Blip_Low_Frequency ; Scattered_Light } and 5 = { Blip; Fast_Scattering })

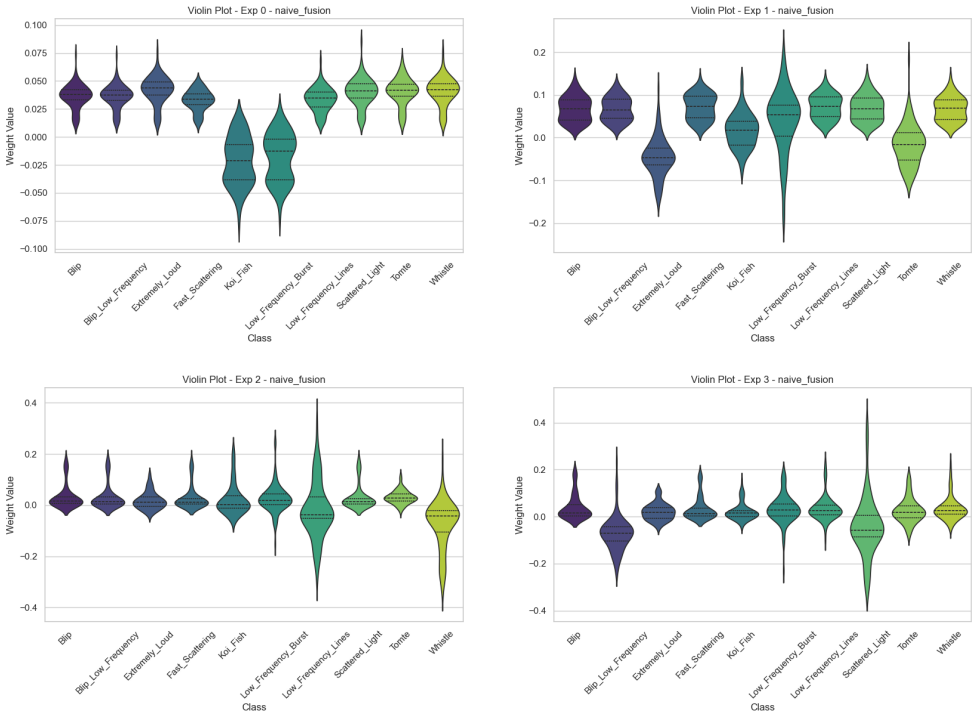
4.4.1.1. Naive CL strategy (baseline)

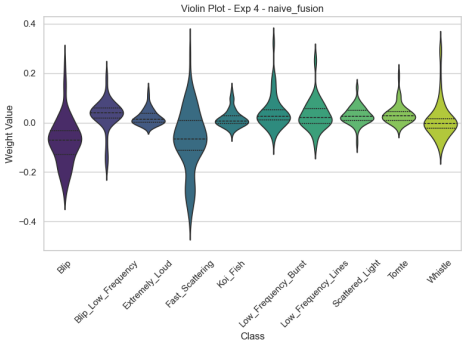
As baseline we use the CL stategy 'Naive'. This is the simplest form of CL, where the model is trained on each task (in our setting we have 5 tasks (experiences) consisting of two classes per task) in sequence without any explicit mechanism to mitigate catastrophic forgetting. However, because early runs of this model suffered from extreme recency bias, the strategy is augmented with a couple of plugins to improve its performance:

- `ReplayPlugin`: this plugin implements experience replay, a common technique to mitigate catastrophic forgetting. It stores a subset of the training data and interleaves training on the current task with training on the stored data. The memory size is set to twice the size of the training set, meaning it can store two complete copies of the training set.

- `EarlyStoppingPlugin`: This plugin implements early stopping, and thus prevents overfitting.

During training we keep track of the weight distribution changes via violin plots. After the first 100 epochs training on differentiating the classes from experience 0 to 4 we get:





For each experience, the weights for the Softmax classification corresponding to the classes in that experience in the fully connected layer are adjusted more than for the other classes. The median shift and distribution shape of the violin body illustrate this.

From the Confusion Matrix as shown in Figure 4.8-(a) we find that the overall accuracy is 81.8%. Some glitch classes like 'Whistle' and 'Frequency_Lines' have perfect accuracy, whilst others like 'Blip_Low_Frequency' and 'Tomte' suffer from a lot of False Negatives, on the other hand 'Blip' suffers from a lot of False Positives.

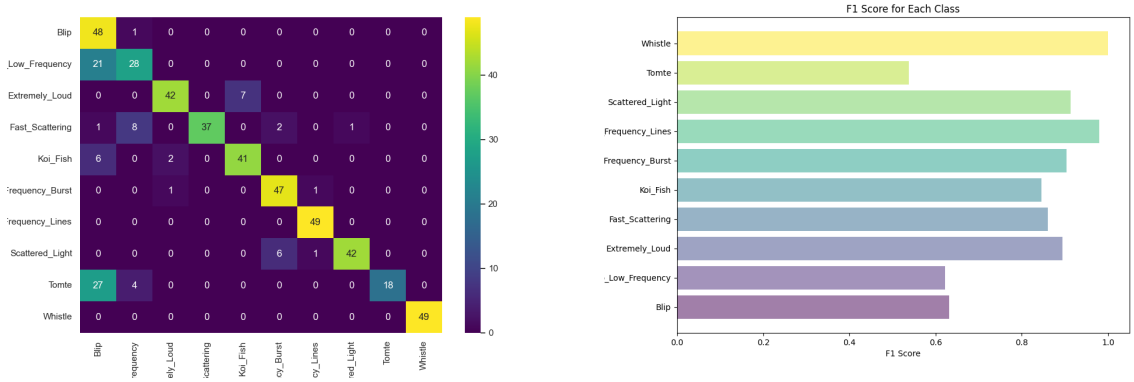


Figure 4.8: Confusion matrix and f1 plot for Naive baseline

We investigate the t-SNE and UMAP representations[6], The t-SNE visualization as shown in Figure 4.9 illustrates a clustering of glitch classes in 2D space. The separation of most glitch categories like 'Whistle' and 'Scattered_Light' is apparent. But the overlap of the Glitch categories 'Blip', 'Blip_Low_Frequency' and 'Tomte' could proof problematic in producing False positives for one of the three classes. A similar interpretation can be made from Figure 4.10. Here 'Koi_Fish' and 'Extremely_Loud' as well as 'Blip', 'Blip_Low_Frequency' and 'Tomte' are relatively close to eachother, suggesting shared global features.

[6]UMAP tends to better preserve global structure compared to t-SNE, it strikes a balance between global and local structure. [McInnes et al., 2018]

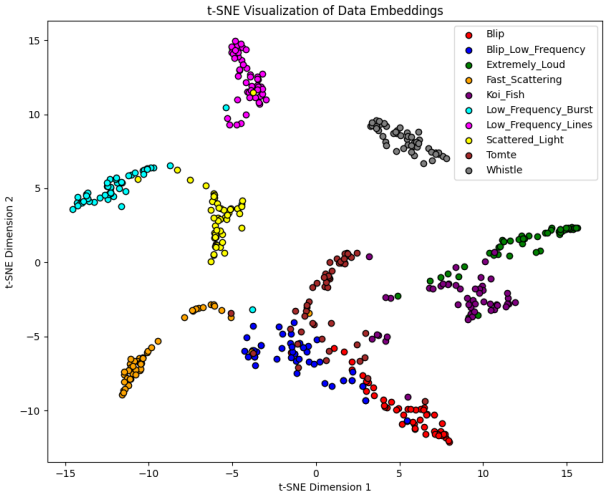


Figure 4.9: t-SNE visualization Naive strategy

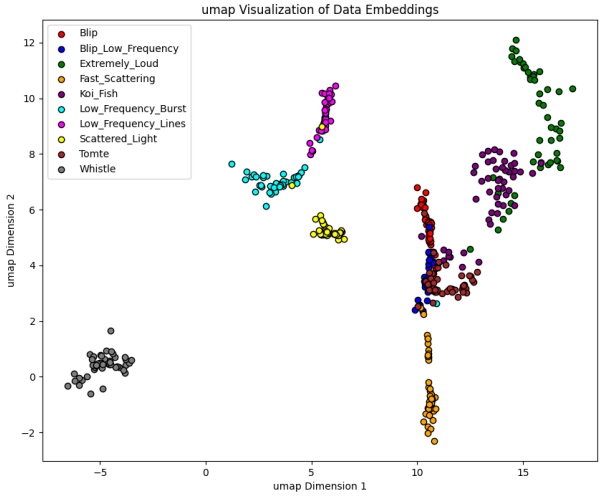


Figure 4.10: UMAP visualization Naive strategy

The images of the Saliency Mapping in Figure 4.11 confirm some of the misclassifications we already concluded from the Confusion Matrix. Specifically on row 2 and row 3 we see the 'Blip_Low_Frequency' misclassified as a 'Blip'. It is clear that the shape of these two are very similar. Because 'Blip' is provided in the last experience, the recency bias favors this category. Presumably if the 'Blip_Low_Frequency' was used in the last experience, the results would be mirrored.

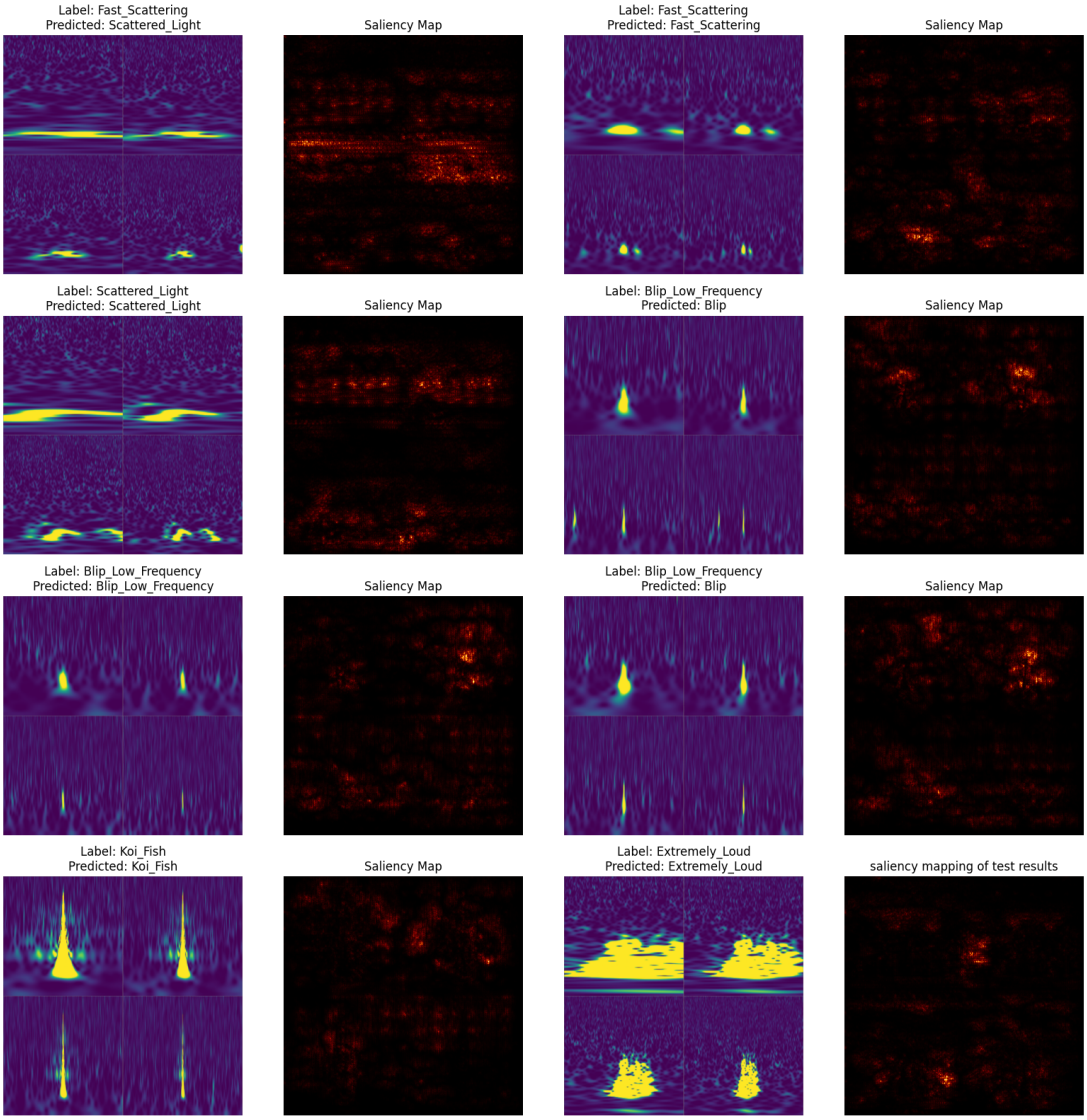


Figure 4.11: Saliency Mapping of Naive baseline

4.4.1.2. LwF STRATEGY

A first CL strategy addressing catastrophic forgetting is called Learning without Forgetting (LwF). The model is trained, just as in the naive baseline, on five distinct tasks that contain two classes per experience. The hyperparameter choices are:

- `alpha=0.25`: α controls the balance between the loss on the new task and the distillation loss used to retain knowledge from old tasks. A value of 0.25 is a common choice based on the paper by [Oren and Wolf, 2021].
- `temperature=2.0`: τ controls the "softness" of the logits during knowledge distillation. A higher τ leads to softer targets, encouraging the network to learn a more generalizable representation that benefits both old and new tasks.

The strategy is augmented with the same `ReplayPlugin`. Early stopping is not inherently supported by LwF and thus not applied.

Changes in weight distribution are also tracked. A similar behavior can be found. For illustrative purposes we only show the violin plots after the first and the last experience.

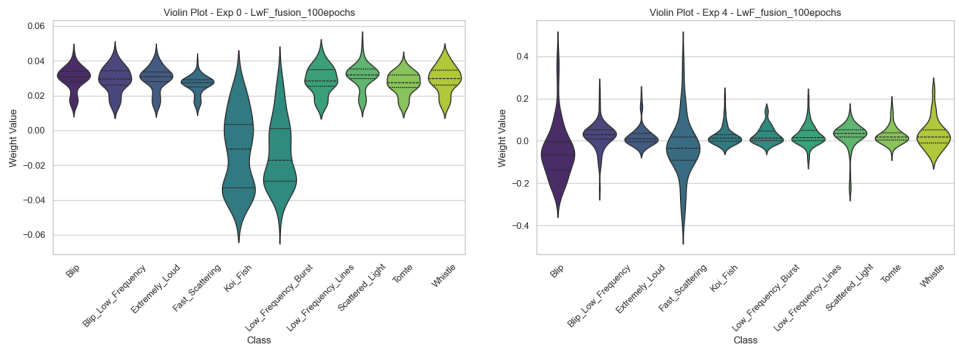


Figure 4.12: Weight distribution changes

From the Confusion Matrix, f1 plot and Classification report as shown in Figure 4.13 and Table 4.2 we find that the overall accuracy is 88.2%, higher than the naive baseline. But 'Blip_Low_Frequency' and 'Tomte' still suffer from a lot of False Negatives, on the other hand 'Blip' suffers from a lot of False Positives.

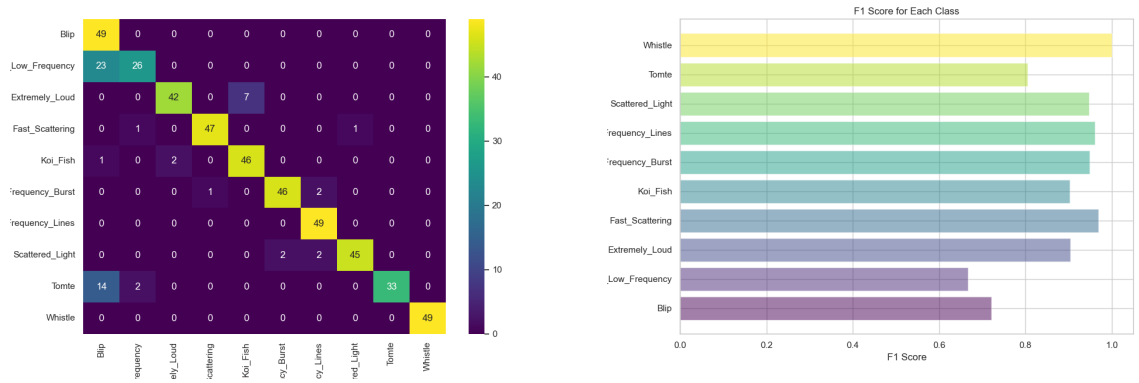


Figure 4.13: Confusion matrix and f1 plot for LwF strategy

Glitch type	Precision	Recall	f1-score
Blip	0.56	1.00	0.72
Blip_Low_Frequency	0.90	0.53	0.67
Extremely_Loud	0.95	0.86	0.90
Fast_Scattering	0.98	0.96	0.97
Koi_Fish	0.87	0.94	0.90
Low_Frequency_Burst	0.96	0.94	0.95
Low_Frequency_Lines	0.92	1.00	0.96
Scattered_Light	0.98	0.92	0.95
Tomte	1.00	0.67	0.80
Whistle	1.00	1.00	1.00

Table 4.2: Classification report of the LwF strategy.

Due to these findings, further exploration of the embedding using t-SNE and UMAP plots (in two dimensions) is necessary.
The t-SNE and UMAP representation, as shown in Figure 4.14, shows a clustering of glitch categories in 2D space.

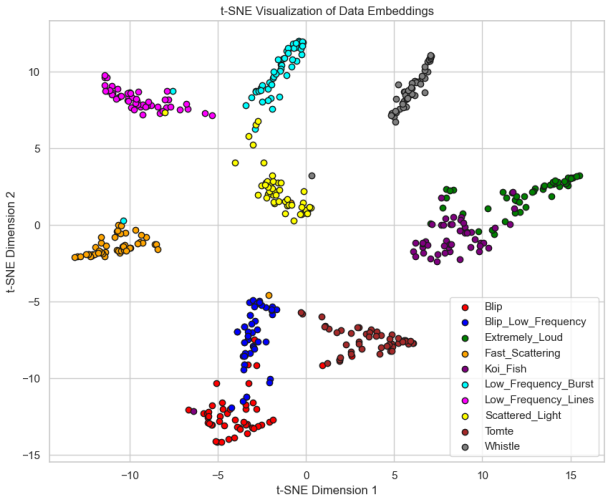


Figure 4.14: t-SNE visualization LwF

Some relevant observations that can be made:

- There seems to be a rough separation between glitches along t-SNE dimension 1. This is visible in both the t-SNE and UMAP visualization.
- Potential similarities among certain glitch classes like "Koi_Fish" and "Tomte" appear close together in the plot. This indicates that these classes share some underlying features.

The images of the Saliency Mapping in Figure 4.16 show that the model attends to different regions of the spectrogram to classify different glitch types. The model makes a mistake

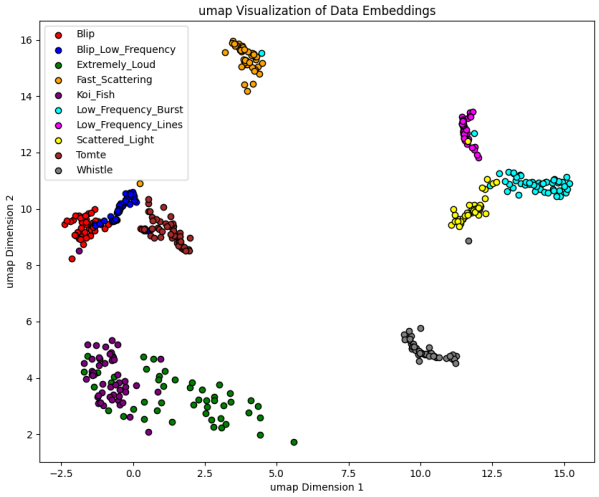


Figure 4.15: UMAP visualization LwF

in the top left image. This is predicted to be a 'Blip', but actually this is a 'Tomte'. When investigating the 'Koi_Fish' pattern next to the 'Blip' and 'Tomte', we see that similar features are retrieved. This confirms the observations made by the t-SNE and UMAP visualizations.

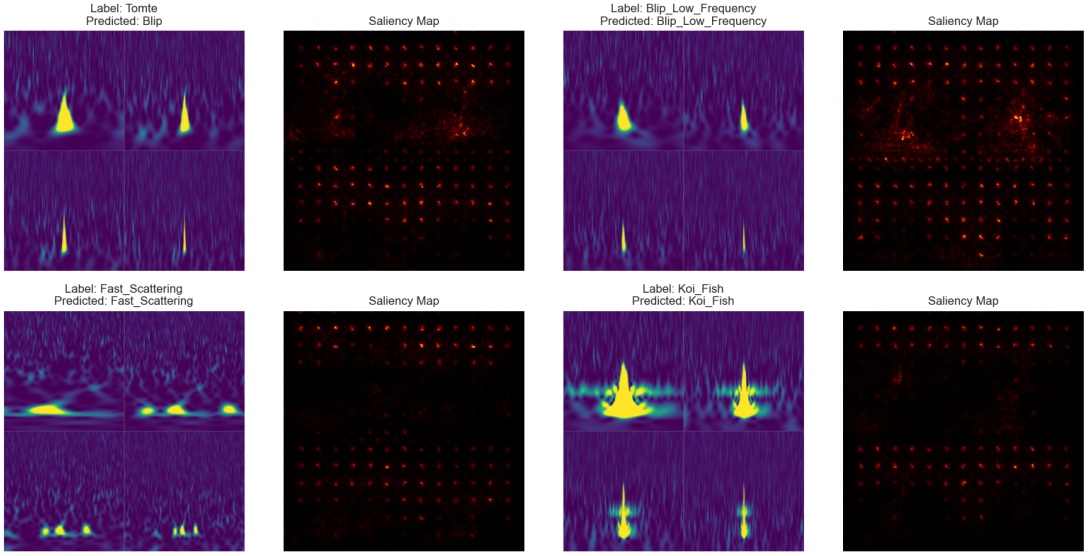


Figure 4.16: Saliency Mapping of LwF

4.4.1.3. AGEM STRATEGY

A second CL strategy addressing catastrophic forgetting is called Averaged Gradient Episodic Memory (A-GEM). A-GEM utilises an episodic memory of gradients computed on past tasks. During training, it computes the gradient of the current model with respect to the new task's loss. It then compares the gradient with the average gradient computed on the past tasks. If the new gradient deviates significantly from the average, A-GEM adjusts the update direction to prevent catastrophic forgetting. The model is trained, just as in the naive baseline, on five distinct tasks that contain two classes per experience.

The strategy is augmented with the `ReplayPlugin` used in the previous strategy, but also an `AGEMPlugin`. The `AGEMPlugin` is specifically designed for A-GEM it maintains a set of past examples.

The hyperparameter choices are:

- `patterns_per_exp`: The number of patterns (samples) stored for each past experience. Based on the papers by [Chaudhry et al., 2018b; Lopez-Paz and Ranzato, 2017] and the fact there are 10 patterns in the benchmark, here we opt for a value of '10'.

- `sample_size`: The number of patterns sampled from the memory buffer during training. Here we opt for the default same sample size equal to the dataloader batch size.

Changes in weight distribution are tracked for illustrative purposes and shown in Figure 4.17

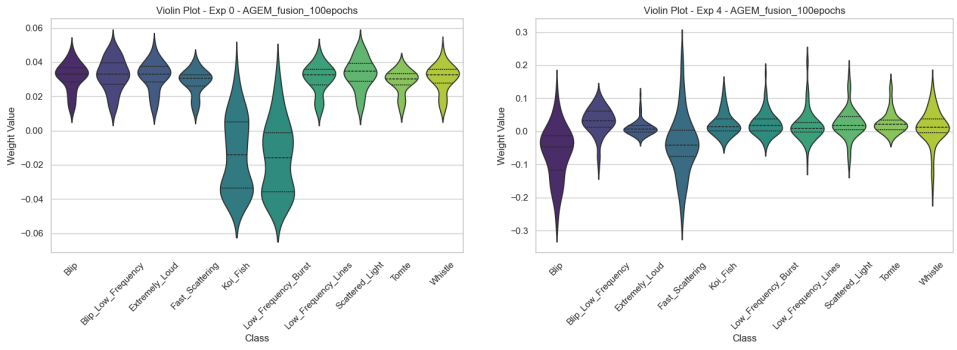


Figure 4.17: Weight distribution changes

From the Confusion Matrix, f1 plot and Classification report as shown in Figure 4.18 and Table 4.3 we find that the overall accuracy is 87%, one percent point lower than the LwF results as depicted in 4.4.1.2. The model performs well on the majority of classes, but struggles with 'Tomte' and 'Blip_Low_Frequency'. Especially the recall on 'Tomte' is very low, indicating a high number of False Negatives. On the other hand, if the 'Tomte' glitch class is detected, the model is very reliable (high precision).

Glitch type	Precision	Recall	f1-score
Blip	0.90	0.76	0.82
Blip_Low_Frequency	0.58	0.96	0.72
Extremely_Loud	0.1.00	0.84	0.91
Fast_Scattering	1.00	0.90	0.95
Koi_Fish	0.73	1.00	0.84
Low_Frequency_Burst	0.96	0.94	0.95
Low_Frequency_Lines	0.96	0.98	0.97
Scattered_Light	0.96	0.94	0.95
Tomte	0.95	0.39	0.55
Whistle	0.98	1.00	0.99

Table 4.3: Classification report of the AGEM strategy.

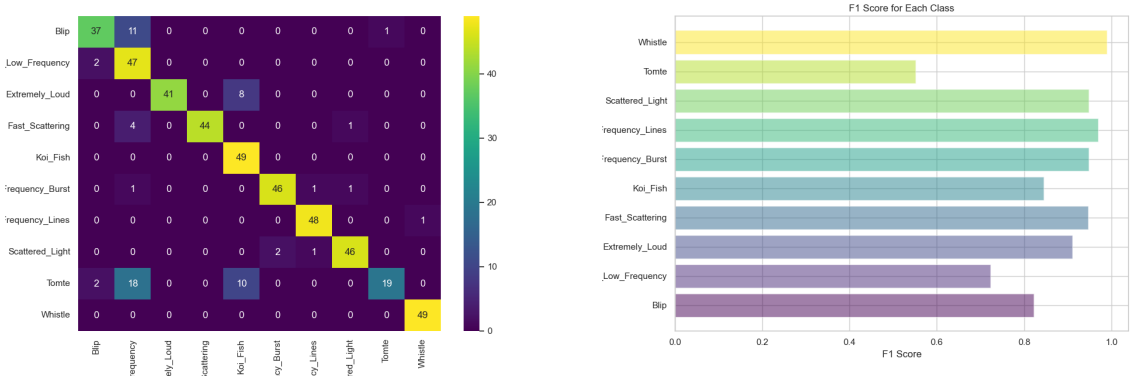


Figure 4.18: Confusion matrix and f1 plot for AGEM strategy

Similar findings can be found from the investigation of the embedding space using t-SNE and UMAP plots in Figure 4.19,.

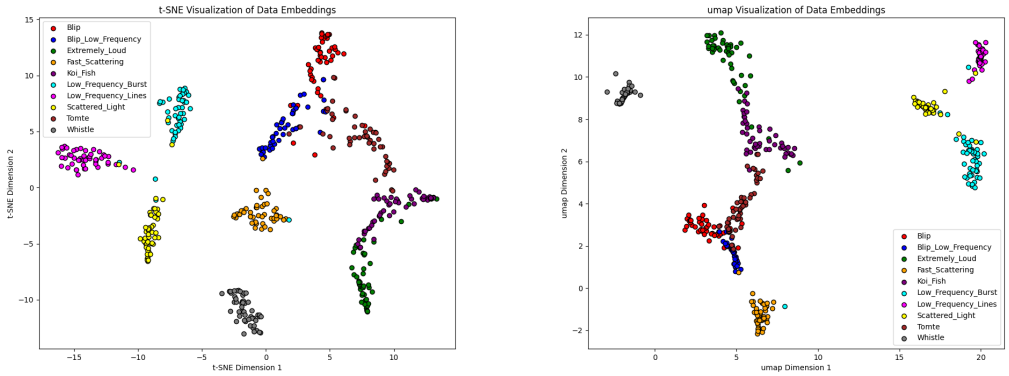


Figure 4.19: t-SNE and UMAP visualisation of the AGEM strategy.

4.4.1.4. EWC STRATEGY

A third CL strategy addressing catastrophic forgetting is called Elastic Weight Consolidation (EWC). EWC acts like a memory preserver in neural networks. During training on new experiences, EWC penalizes large updates to important weights used in previous tasks. This gentles the learning process, allowing the network to adapt to new information while retaining previously learned knowledge, hence avoiding recency bias.

The strategy is augmented with the `ReplayPlugin` used in the previous strategies. An extra hyperparameter, `ewc_lambda`, must be selected. We choose a value of $\lambda = 0.1$, signifying a moderate penalty for weight updates when training on new tasks. Typically, a greater value (approaching 1) emphasizes the preservation of previous knowledge, although it may hinder the acquisition of new skills.

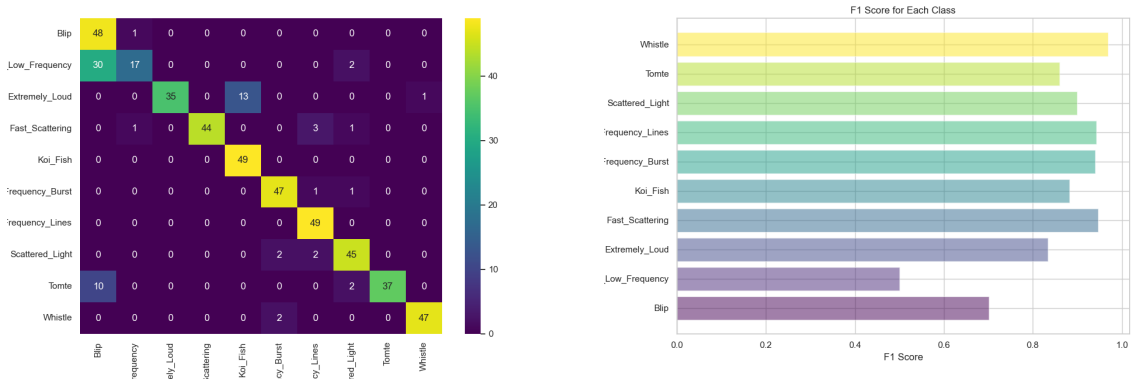


Figure 4.20: Confusion matrix and f1 plot for EWC strategy

Glitch type	Precision	Recall	f1-score
Blip	0.55	0.98	0.70
Blip_Low_Frequency	0.89	0.35	0.50
Extremely_Loud	1.00	0.71	0.83
Fast_Scattering	1.00	0.90	0.95
Koi_Fish	0.79	1.00	0.88
Low_Frequency_Burst	0.92	0.96	0.94
Low_Frequency_Lines	0.89	1.00	0.94
Scattered_Light	0.88	0.92	0.90
Tomte	1.00	0.76	0.86
Whistle	0.98	0.96	0.97

Table 4.4: Classification report of the EWC strategy.

From the Confusion Matrix, f1 plot and Classification report as shown in Figure 4.20 and Table 4.4 we find that the overall accuracy is 85%, several percent points lower than the previous best LwF results as depicted in 4.4.1.2. The model performs well on the majority of classes, but struggles with 'Blip' and 'Blip_Low_Frequency'. Especially the recall on 'Blip_Low_Frequency' is very low.

4.4.1.5. DER STRATEGY

Dark Experience Replay (DER) combats recency bias by replaying past predictions, acting like a teacher. It stores the model's past outputs (logits) from old tasks alongside the original data. During training on new experiences (new classes), DER forces the model's current predictions on old data to align with its past predictions, reminding it of what it already knew, thus hindering catastrophic forgetting.
The hyperparameter `alpha` controls the weight given to the replay loss (from past predictions) compared to the current task loss. A value of $\alpha = 1.0$ means that the replay loss is given equal importance to the current task loss during training on new experiences. Basically the model must optimize two things equally: 1) Fitting the new data and 2) Staying consistent with past predictions [Buzzega et al., 2020].

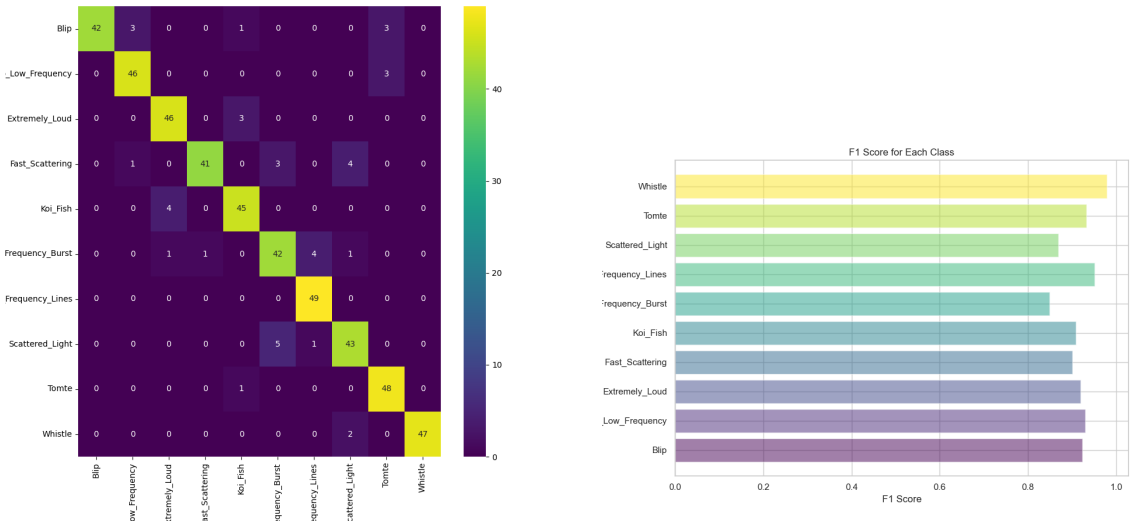


Figure 4.21: Confusion matrix and f1 plot for DER strategy

Glitch type	Precision	Recall	f1-score
Blip	1.00	0.86	0.92
Blip_Low_Frequency	0.92	0.94	0.93
Extremely_Loud	0.90	0.94	0.92
Fast_Scattering	0.98	0.84	0.90
Koi_Fish	0.90	0.92	0.91
Low_Frequency_Burst	0.84	0.86	0.85
Low_Frequency_Lines	0.91	1.00	0.95
Scattered_Light	0.86	0.88	0.87
Tomte	0.89	0.98	0.93
Whistle	1.00	0.96	0.98

Table 4.5: Classification report of the DER strategy.

From the Confusion Matrix, f1 plot and Classification report as shown in Figure 4.21 and Table 4.5 we find that the overall accuracy is 92%, more than the previous best LwF

strategy from 4.4.1.2.

Because of the good results, it is noteworthy to further investigate the t-SNE and UMAP visualisations in Figure 4.22.

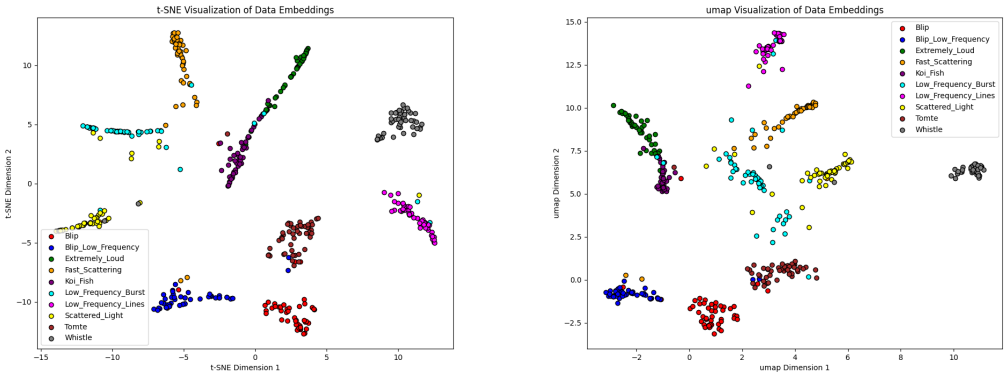


Figure 4.22: t-SNE and UMAP visualisation of the DER strategy.

Some relevant observations that can be made:

- The separation between 'Blip', 'Blip_Low_Frequency' and 'Tomte' is more defined as opposed to the previous best results.
- DER seems to have more troubles with distinguishing 'Extremely_Loud' and 'Koi_Fish' glitch types.

The images of the Saliency Mapping in Figure 4.23 visualise the above made observations more clearly.

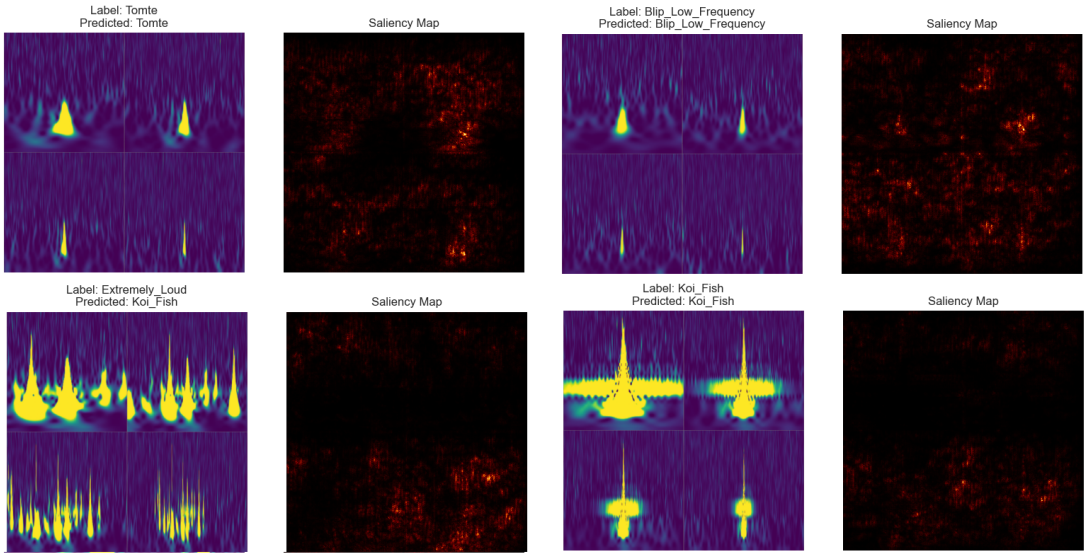


Figure 4.23: Saliency Mapping of DER

4.4.1.6. DER++ STRATEGY

Dark Experience Replay ++ (DER++) is an advanced version of DER from 4.4.1.5. It extends the replay buffer by also storing the ground truth logits and replay examples [Buzzega et al., 2020]. DER++ mitigates, in this way, the shortcoming of DER that weakens when the logits are highly biased due to a sudden distribution shift [Ling et al., 2022].

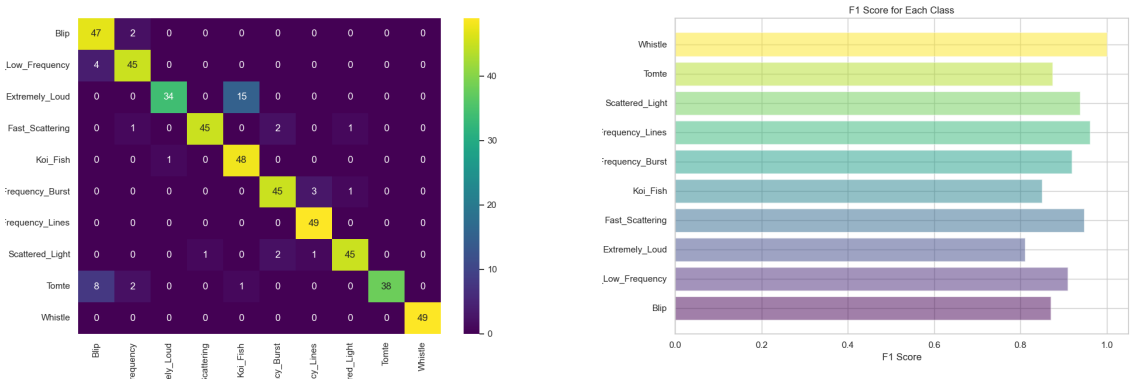


Figure 4.24: Confusion matrix and f1 plot for DER++ strategy

Glitch type	Precision	Recall	f1-score
Blip	0.80	0.96	0.87
Blip_Low_Frequency	0.90	0.92	0.91
Extremely_Loud	0.97	0.69	0.81
Fast_Scattering	0.98	0.92	0.95
Koi_Fish	0.75	0.98	0.85
Low_Frequency_Burst	0.92	0.92	0.92
Low_Frequency_Lines	0.92	1.00	0.96
Scattered_Light	0.96	0.92	0.94
Tomte	1.00	0.78	0.87
Whistle	1.00	1.00	1.00

Table 4.6: Classification report of the DER++ strategy.

From the Confusion Matrix, f1 plot and Classification report as shown in Figure 4.24 and Table 4.6 we find that the overall accuracy is 91%, one percent point less than the previous best DER strategy from 4.4.1.5. It has perfect results on 'Whistle' and 'Low_Frequency_Lines' glitches.

4.4.1.7. SCR STRATEGY

As described in 2.2.3.8, SCR leverages the Nearest-Class-Mean classifier as a substitute for the Softmax classifier by addressing recency bias and avoiding structural changes in the fully-connected layer for new classes. There is however a risk involved, the effectiveness of SCR might depend on the quality of the data embeddings and the between-class separation. It also has the tendency of introducing additional bias [Aleixo et al., 2023; Mai et al., 2021].

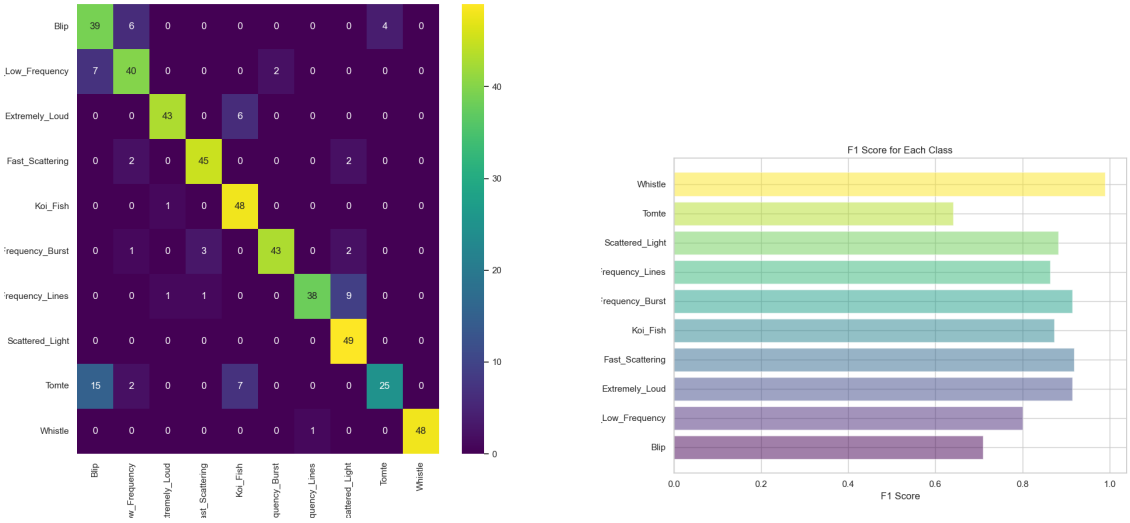


Figure 4.25: Confusion matrix and f1 plot for SCR strategy

Glitch type	Precision	Recall	f1-score
Blip	0.64	0.80	0.71
Blip_Low_Frequency	0.78	0.82	0.80
Extremely_Loud	0.96	0.88	0.91
Fast_Scattering	0.92	0.92	0.92
Koi_Fish	0.79	0.98	0.87
Low_Frequency_Burst	0.96	0.88	0.91
Low_Frequency_Lines	0.97	0.78	0.86
Scattered_Light	0.79	1.00	0.88
Tomte	0.86	0.51	0.64
Whistle	1.00	0.98	0.99

Table 4.7: Classification report of the SCR strategy.

From the Confusion Matrix, f1 plot and Classification report as shown in Figure 4.25 and Table 4.7 we find that the overall accuracy is 85%, worse than the best performing DER strategy from 4.4.1.5. It has good results on 'Whistle' and Fast_Scattering' glitches. But fails greatly on 'Tomte' and 'Blip' glitches. The training time of the implementation used in the `Avalanche` package[7] takes very long, almost three times (2.5 days) longer than 4.4.1.5.

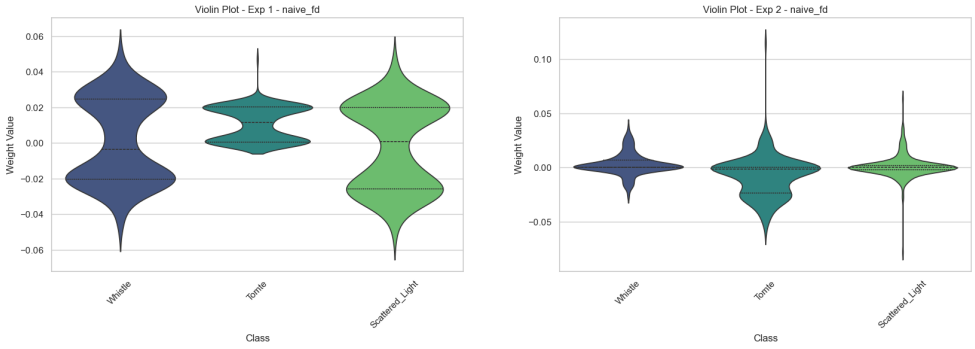
[7] avalanche.training.SCR at `https://shorturl.at/ewOU3`

4.4.2. RQ2

Each of the strategies are trained for 15 epochs on every task (of learning the weights on the three classes). More training epochs resulted in overfitting behavior. Between 10 and 15 epochs of training the results converge. The order in which the classes are fed is kept the same over all strategies (1 = {Whistle}, 2 = {Scattered_Light }, 3 = {Tomte }).

4.4.2.1. NAIVE CL STRATEGY (BASELINE)

As baseline we use the CL stategy 'Naive', the hyperparameter choices and additional plugins are similar to those in 4.4.1.1. One difference, the benchmark consists of 3 experiences where each experience contains instances of 1 class only. During training we track the weight distribution changes via violin distribution plots. In the following table, 4.4.2.1, the violin plots for the second and last experience are shown. After the first 15 epochs training on differentiating the classes from experience 0 to 2 we get:



The model has an overall accuracy of 94% and a Backward Transfer (BWT) metric of 0.0523, indicating that the model was successful in mitigating forgetting on the old task while the model learned the new task. A similar conclusion can be made from the Confusion matrix in Figure 4.26 and classification report. The model performs well on the 'Whistle' and 'Scattered_Light' class, but struggles some more with 'Tomte'. The F1 scores for two of the classes 'Whistle' and 'Scattered_Light' are above 0.9, the lowest being the 'Tomte' result with F1= 0.89, scores above 0.9 are considered very good.

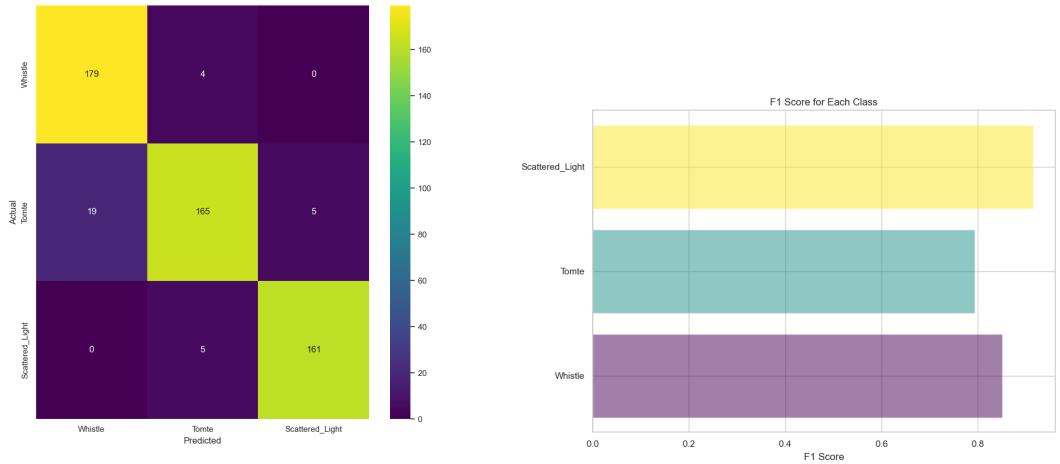


Figure 4.26: Confusion matrix and f1 plot for Naive baseline

Glitch type	Precision	Recall	f1-score
Whistle	0.92	0.91	0.92
Tomte	0.87	0.92	0.89
Scattered_Light	0.98	0.95	0.96

Table 4.8: Classification report of the FD Naive strategy.

Just as in our experiments for 4.4.1 we investigate the t-SNE and UMAP representations. As depicted in Figure 4.27, the t-SNE visualization demonstrates the clustering of three glitch classes in a 2D layout. The plot shows clear separation among the three classes, implying that the model successfully differentiates these glitch types. The compactness of the point clouds within each class suggests high similarity among the data samples of a class. Nonetheless, there is some intersection involving the 'Tomte' class and the other two classes.

A similar interpretation can be made from Figure 4.28. The 'Whistle' class forms a distinct elongated cluster. This indicates that the instances are quite alike in the high-dimensional space. The 'Tomte' cluster appears as a dense group, yet it is proximate to other clusters. It is evident that there is overlap among different classes. This raises the concern whether some instances might have been incorrectly classified during manual labeling.

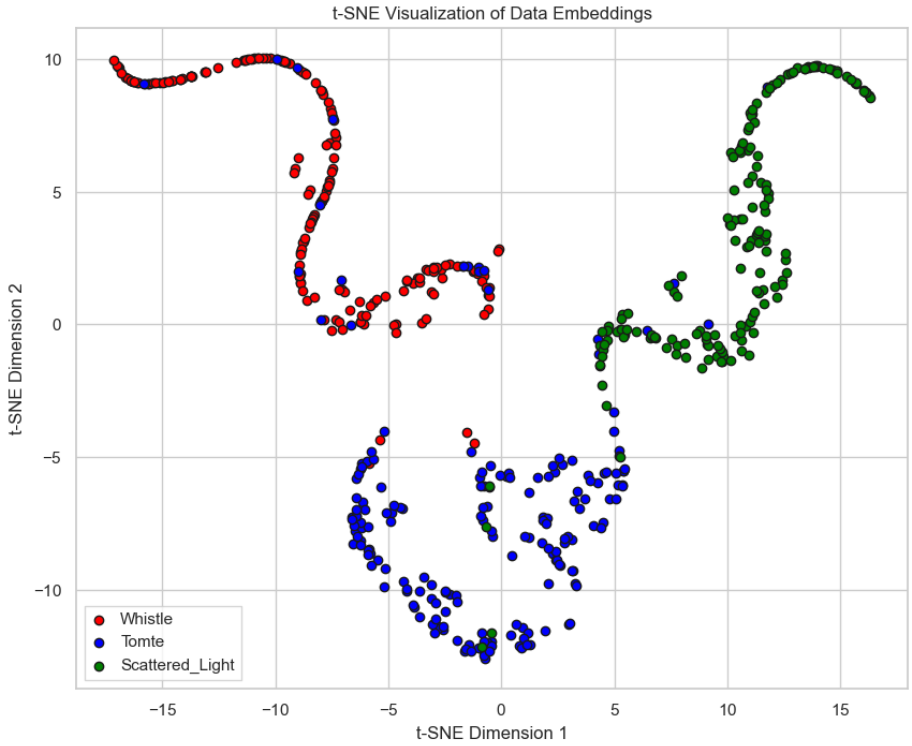


Figure 4.27: t-SNE visualization Naive FD strategy

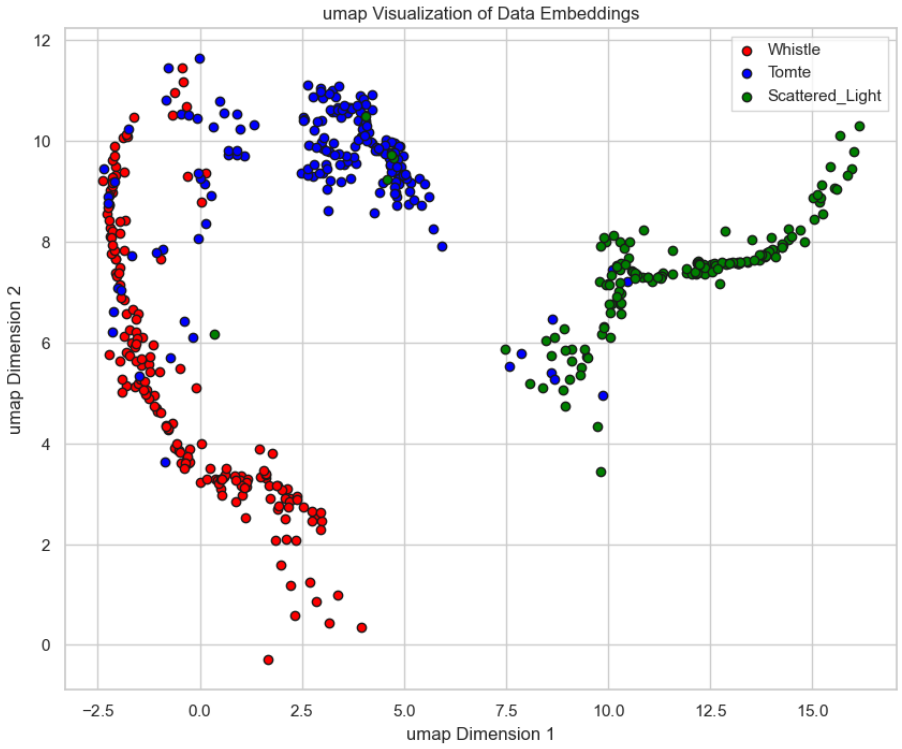


Figure 4.28: UMAP visualization Naive FD strategy

4.4.2.2. LwF STRATEGY

The strategy is implemented with the same hyperparameter choices as described in 4.4.1.2. The benchmark as described in 4.4.2.1 is used.
The trained model has an overall accuracy of 92%, less than the Naive baseline. The F1 scores (see Table 4.9) are all less than those of the Naive baseline.

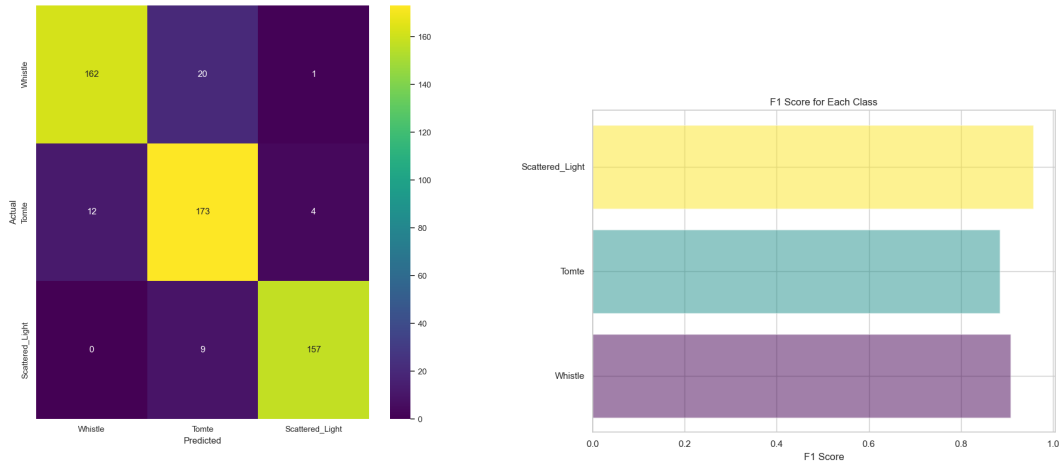


Figure 4.29: Confusion matrix and f1 plot for LwF FD strategy

Glitch type	Precision	Recall	f1-score
Whistle	0.93	0.91	0.92
Tomte	0.87	0.92	0.89
Scattered_Light	0.98	0.95	0.96

Table 4.9: Classification report of the FD LwF strategy.

The Backward Transfer (BWT) metric is −0.0654, suggesting the model had a slight decline in performance on the old task after learning the new task. However, the magnitude of forgetting is relatively small.

The 'Scattered_Light' glitch category has the highest overall F1 score, and from the t-SNE and UMAP representations shown in Figure 4.30 and 4.31 it is clear that the cluster containing the 'Scattered_Light' test instances is much more dense, as opposed to the 'Whistle' and 'Tomte' clusters showing quite some overlap. This indicates the model has more trouble distinguishing between those two categories.

4.4.2.3. AGEM STRATEGY

The strategy is augmented, just as in 4.4.1.3 with the `ReplayPlugin`, and also an `AGEMPlugin`. The `AGEMPlugin` is set to accept `patterns_per_experience=20`, based on the study by [Chaudhry et al., 2019]. The `sample_size` is set to the dataloader size as previously suggested. The results, even after retraining with different memory size and patterns_per_experience, are very bad. A-GEM clearly suffers from catastrophic forgetting as depicted in Figure 4.32 and it is impossible to plot the t-SNE and UMAP plots because of `nan` values occurring in the loss calculation. The model outputs the last encountered class 'Whistle' in all test cases.

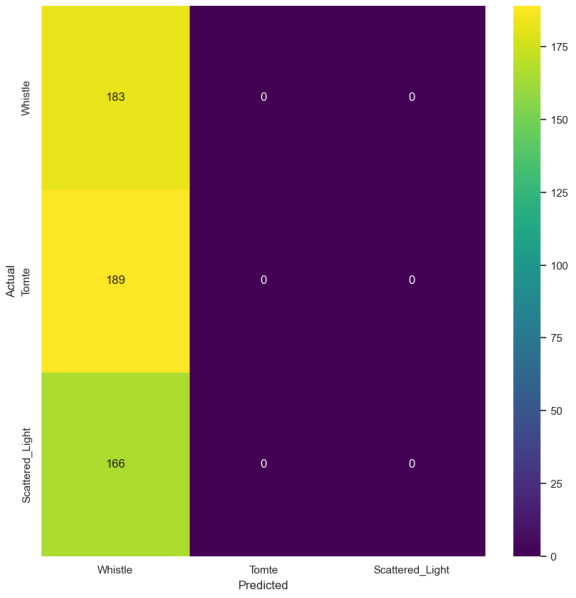


Figure 4.32: Confusion matrix for AGEM strategy

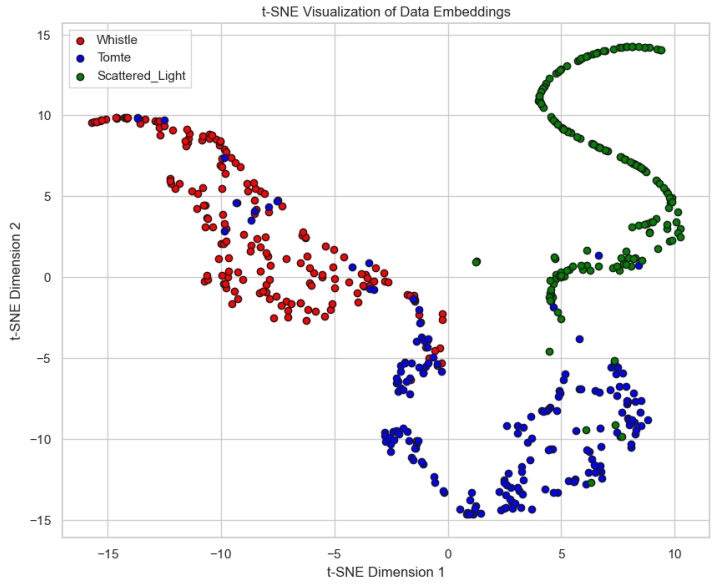


Figure 4.30: t-SNE visualization LwF FD strategy

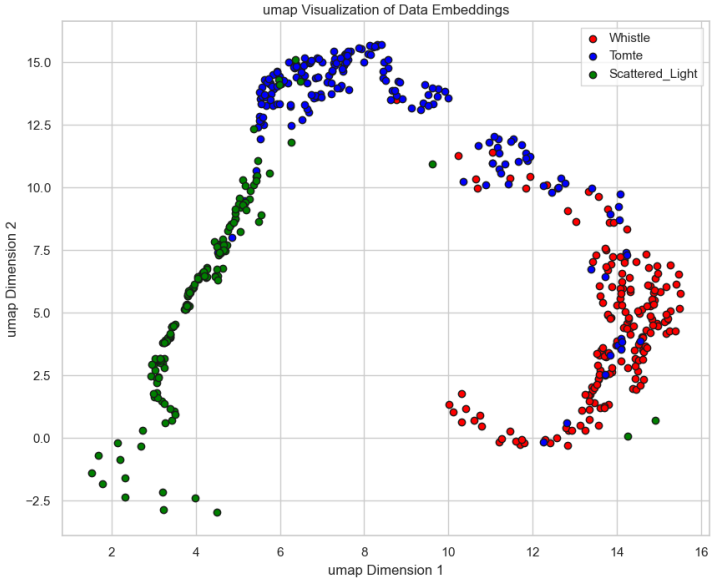


Figure 4.31: UMAP visualization LwF FD strategy

4.4.2.4. EWC STRATEGY

The strategy is implemented with the same hyperparameter choices as described in 4.4.1.4. The benchmark as described in 4.4.2.1 is used.
The trained model has an overall accuracy of 90%, less than the Naive baseline. The F1 scores (see Table 4.10) are all less than those of the Naive baseline.

The Backward Transfer (BWT) metric is −0.1176, suggesting a moderate level of catastrophic forgetting on the previous experiences after learning the new experience.

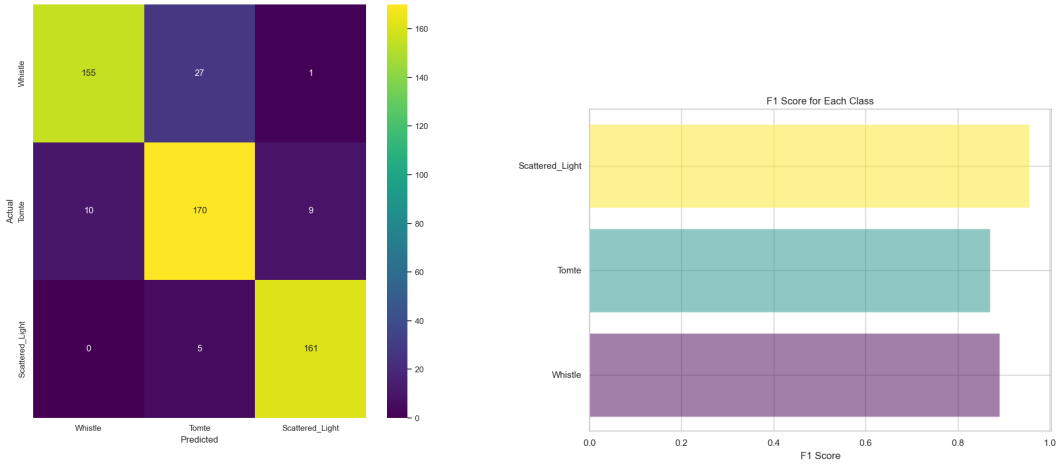


Figure 4.33: Confusion matrix and f1 plot for EWC FD strategy

Glitch type	Precision	Recall	f1-score
Whistle	0.89	0.93	0.91
Tomte	0.85	0.87	0.86
Scattered_Light	0.97	0.90	0.94

Table 4.10: Classification report of the FD EWC strategy.

4.4.2.5. DER STRATEGY

The strategy is implemented with the same hyperparameter choices as described in 4.4.1.5 with the benchmark as described in 4.4.2.1.
The trained model has an overall accuracy of 90%, less than the Naive baseline. The F1 scores (see Table 4.10) are all less than those of the Naive baseline.

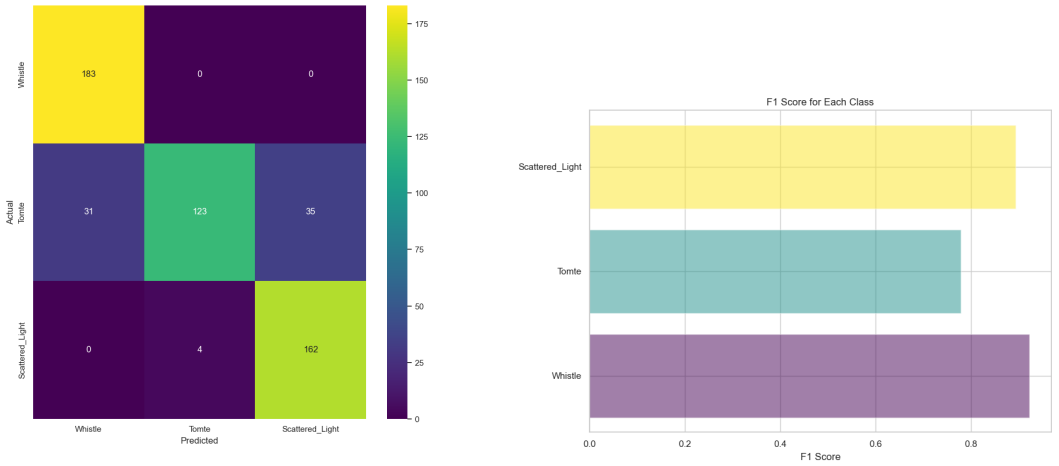


Figure 4.34: Confusion matrix and f1 plot for DER FD strategy

Glitch type	Precision	Recall	f1-score
Whistle	0.89	0.93	0.91
Tomte	0.85	0.87	0.86
Scattered_Light	0.97	0.90	0.94

Table 4.11: Classification report of the FD DER strategy.

The Backward Transfer (BWT) metric is 0.0131, indicating that the model was successful in mitigating forgetting on the old task while the model learned the new task. A BWT close to zero (ideal scenario) indicates that the model is able to maintain its performance on old tasks while learning new ones with no degradation in performance on old tasks.

4.4.2.6. DER++ STRATEGY

The strategy is implemented with the identical hyperparameter values as described in 4.4.1.6 with the benchmark as described in 4.4.2.1.
The trained model has an overall accuracy of 77%, this is the worst performing model. The F1 scores (see Table 4.12) are all less than those of the Naive baseline. Especially very bad performance metrics on the 'Tomte' glitch.

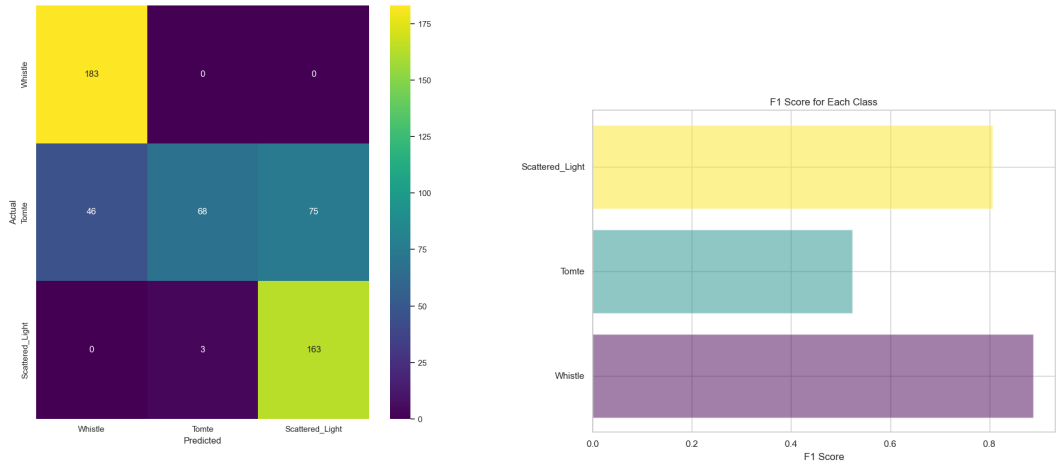


Figure 4.35: Confusion matrix and f1 plot for DER++ FD strategy

Glitch type	Precision	Recall	f1-score
Whistle	0.80	1.00	0.89
Tomte	0.96	0.36	0.52
Scattered_Light	0.68	0.98	0.81

Table 4.12: Classification report of the FD DER++ strategy.

The Backward Transfer (BWT) metric is −0.0458, suggesting the model had a slight decline in performance on the old task after learning the new task. However, the magnitude of forgetting is relatively small.

The t-SNE plot in 4.36 illustrates the difficulty of differentiating between 'Tomte' and the other two classes.

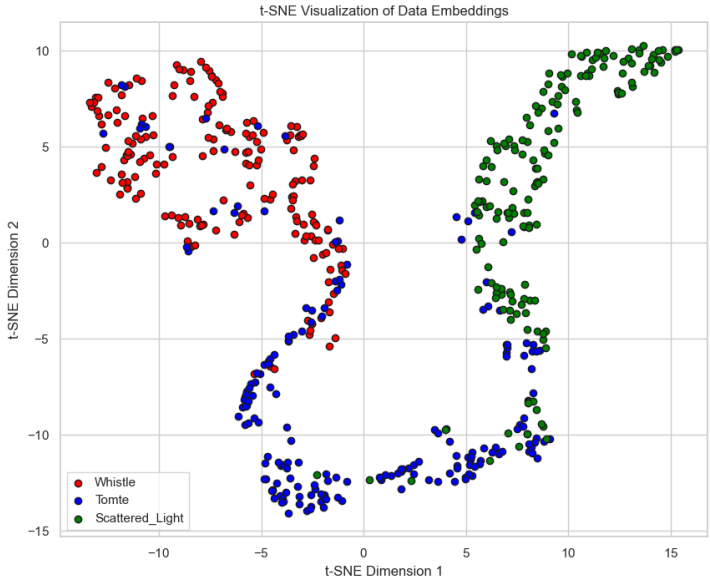
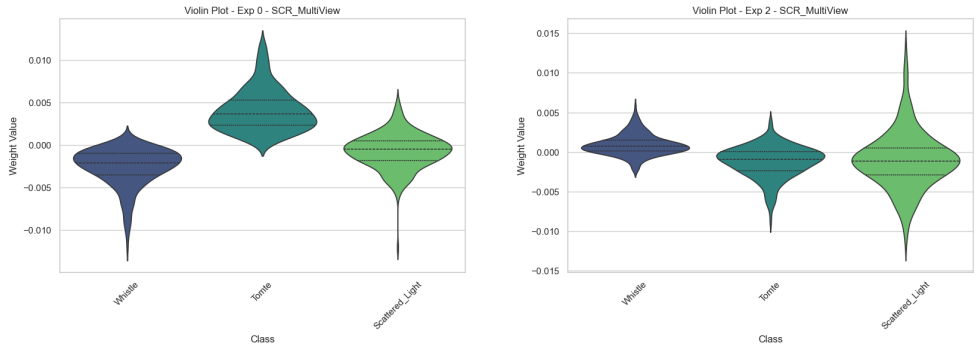


Figure 4.36: t-SNE visualization DER++ FD strategy

4.4.2.7. SCR STRATEGY

The SCR stategy on the Fractal Dimension data has a similar implementation as in 4.4.1.7. The model has an overall accuracy of 93% and scores second just after the Naive strategy used in 4.4.2.1. The weight distributions shown in Figure 4.4.2.7 indicate the difficulty the model has with distinguishing 'Whistle' and 'Tomte' glitches. Behavior that is confirmed by the t-SNE and UMAP plots in Figures 4.37 and 4.38.



The Backward Transfer (BWT) metric is 0.3595, a high value, suggesting that the model suffers from moderate catastrophic forgetting on the old task after learning the new task.

Glitch type	Precision	Recall	f1-score
Whistle	0.94	0.96	0.95
Tomte	0.93	0.85	0.89
Scattered_Light	0.91	0.96	0.94

Table 4.13: Classification report of the FD SCR strategy.

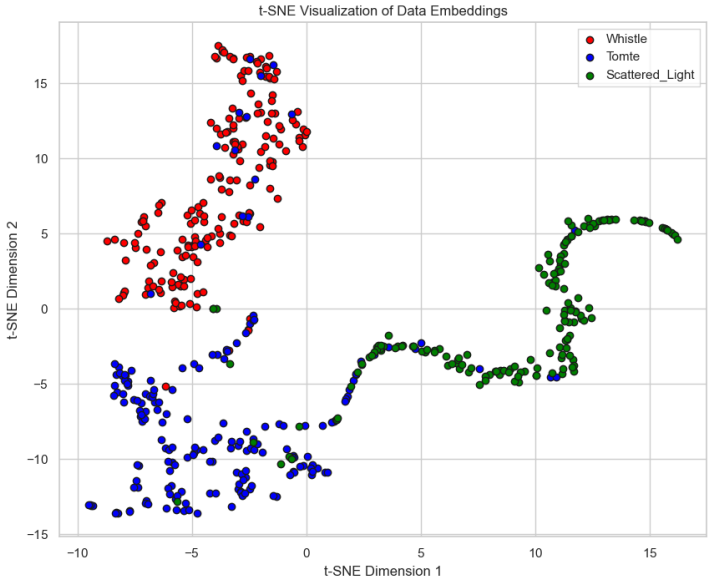


Figure 4.37: t-SNE visualization SCR FD strategy

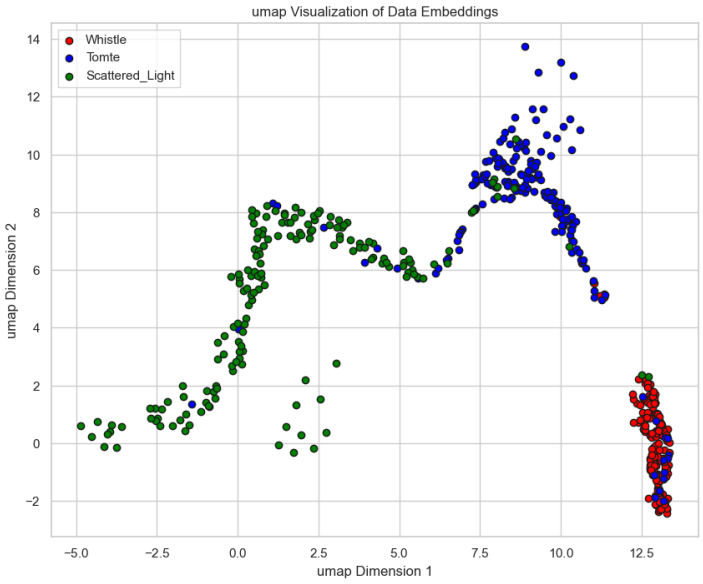


Figure 4.38: UMAP visualization SCR FD strategy

4.4.3. RQ3

The training is done on the dataset provided by Melissa Lopez, used in the research by [Dooney et al., 2022, 2024; Laguarta et al., 2023]. The training is done for 30 epochs on the classes {Whistle, Scattered_Light, Tomte}.

The ultimate version of the model achieved a Cross Entropy Loss of 0.0122 and a training accuracy of 99.61%. The outcomes detailed in Table 4.14 for the unseen test samples were remarkably impressive, demonstrating flawless precision, recall, and f1-score across all three categories. Notably, the performance on the Tomte glitch significantly surpassed that of earlier 'best' models as discussed in 4.4.1.5 and 4.4.2.1.

Glitch type	Precision	Recall	f1-score
Whistle	1.00	1.00	1.00
Tomte	1.00	1.00	1.00
Scattered_Light	1.00	1.00	1.00

Table 4.14: Classification report of the Multimodal architecture.

The resulting t-SNE plot confirms the perfect test set results. There are two points that may be outliers or incorrectly classified by the Gravity Spy project [Zevin et al., 2017].

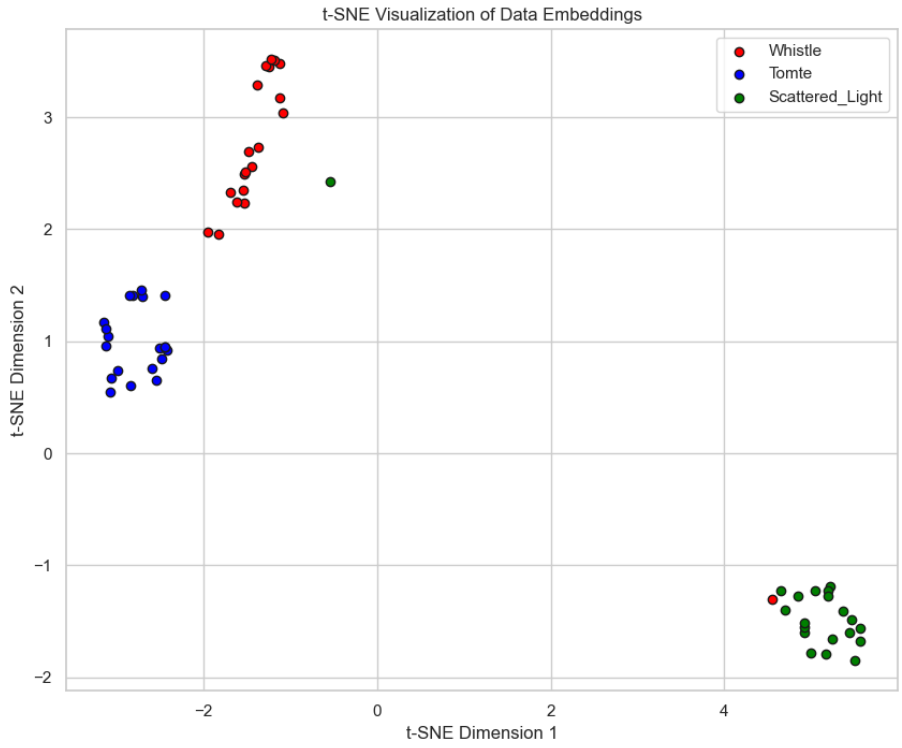


Figure 4.39: t-SNE visualization of Multimodal architecture.

The Saliency mapping in Figure 4.40 illustrates the distinctive capabilities of the model. There are clear boundaries between the three glitch categories.

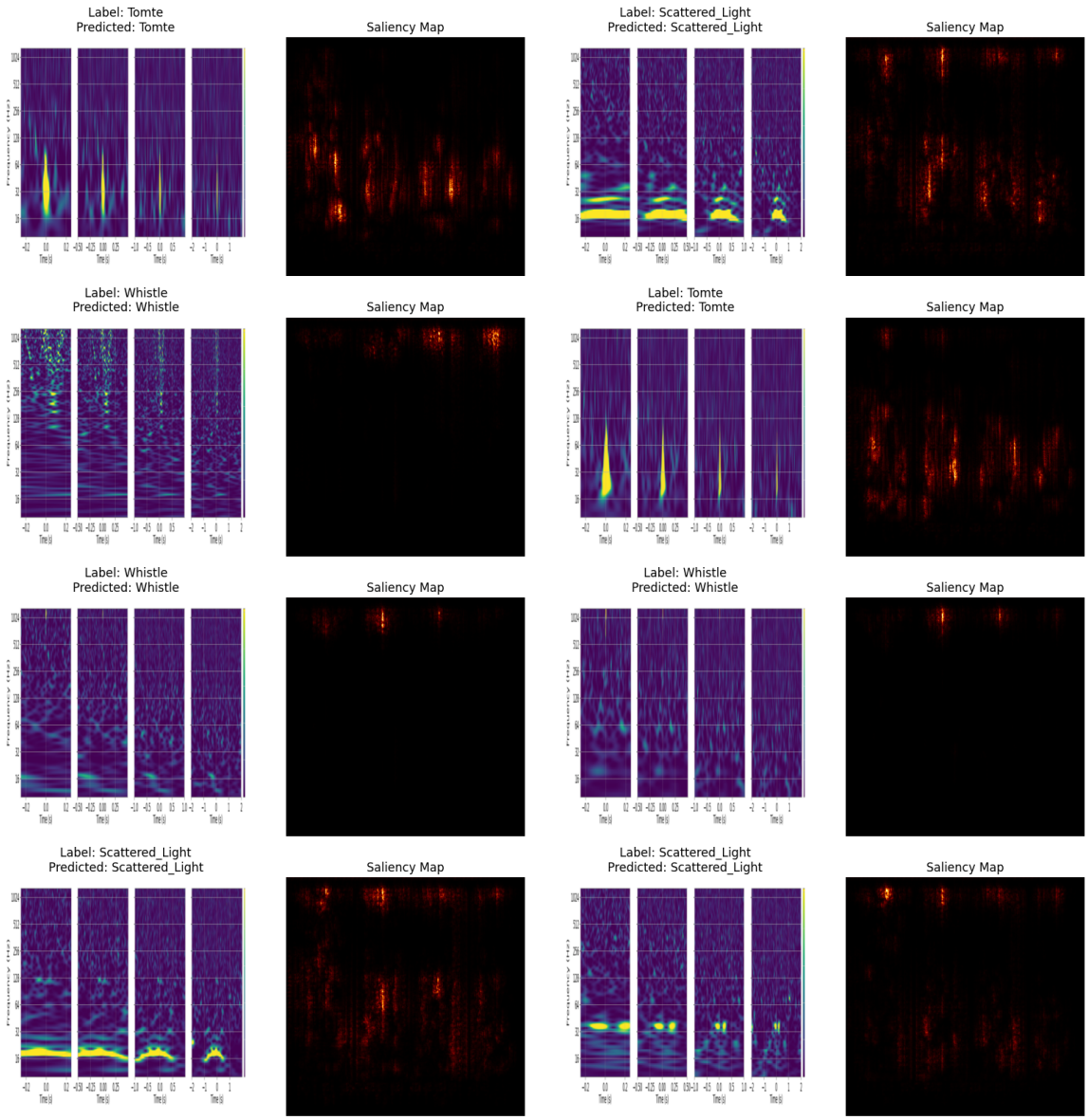


Figure 4.40: Saliency mapping of the Multimodal architecture.

5

DISCUSSION

5.1. GENERAL

5.1.1. RQ1

Most of the CL approaches on the spectrogram images generated class predictions that aligned with the existing Gravity Spy classes. An overview of the corresponding F1 scores can be found in Table 5.1. There were, however, some oddities and difficulties.
From the t-SNE and UMAP plots it seemed like the majority of the strategies have difficulty with distinguishing 'Blip', 'Blip_Low_Frequency' and 'Tomte' from one another. Only DER was able to achieve high precision and recall on these glitches. The question arises whether the three classes represent some form of similar short-duration anomaly in the data. Further research could prove useful.

Glitch type	Naive	LwF	AGEM	EWC	DER	DER++	SCR
Blip	0.63	0.72	0.82	0.70	0.92	0.87	0.71
Blip_Low_Frequency	0.62	0.67	0.72	0.50	0.93	0.91	0.80
Extremely_Loud	0.89	0.90	0.91	0.83	0.92	0.81	0.91
Fast_Scattering	0.86	0.97	0.95	0.95	0.90	0.95	0.92
Koi_Fish	0.85	0.90	0.84	0.88	0.91	0.85	0.87
Low_Frequency_Burst	0.90	0.95	0.95	0.94	0.85	0.92	0.91
Low_Frequency_Lines	0.98	0.96	0.97	0.94	0.95	0.96	0.86
Scattered_Light	0.91	0.95	0.95	0.90	0.87	0.94	0.88
Tomte	0.54	0.80	0.55	0.86	0.93	0.87	0.64
Whistle	1.00	1.00	0.99	0.97	0.98	1.00	0.99

Table 5.1: f1-score table for each strategy.

Although the results are acceptable to good and correspond with the original Gravity Spy classes, not all glitch types have been incorporated in the experiment.
Just as described by [Wu et al., 2024] the existing DL classifiers face limitations. With the Fourth Observing Run ongoing. They propose multi-view fusion attention based approaches.

5.1.2. RQ2

The effectiveness of CL approaches in incorporating auxiliary channel data for detecting and classifying glitch morphologies has been explored. The architecture employed, consisting of convolutional, dropout and batch normalization layers, was able to effectively learn to differentiate glitch types. Due to the limited (and difficult) availability of auxiliary data for only three glitch classes, the findings for the effectiveness of CL based on the calculated FD of a selection of auxiliary channels should, however, be treated with caution. From Table 5.2 it is noticeable that the Naive strategy was able to produce the best results together with the provided convolutional architecture. The SCR, LwF and DER strategies also have acceptable results.
The results follow the conclusions made by [Laguarta et al., 2024], the information from auxiliary channels serves as witnesses to detect glitches.

Glitch type	Naive	LwF	AGEM	EWC	DER	DER++	SCR
Scattered_Light	0.96	0.95	0.00	0.94	0.94	0.81	0.94
Tomte	0.89	0.89	0.00	0.86	0.86	0.52	0.89
Whistle	0.92	0.92	0.51	0.93	0.91	0.89	0.95

Table 5.2: f1-score table for each strategy.

A-GEM suffered from extreme catastrophic forgetting. This could be due to an error in the implementation or an incorrect choice of hyperparameter values.

5.1.3. RQ3

In adressing the final research question, we explored the potential of combining the strain data (in the form of spectrograms) with the auxiliary channel data (fractal dimension matrix). The findings gave outstanding results. These results, as documented in 4.4.3, gave 99.61% training accuracy as well as 100% test set accuracy.
Strain data and auxiliary channel data each capture different aspects of the underlying physical processes. By fusing these modalities, this lead to a more comprehensive representation of the system. The spectrograms provide temporal and frequency-domain features, while auxiliary channels offer context-specific information (e.g. environmental conditions, instrument status, ...). The joint representation enabled the model to learn discriminative features that are not apparent in either modality alone. This result was visualised via the t-SNE plot in Figure 4.39.

The Late Fusion implementation in Pytorch was used because this was the most straightforward and less time consuming. The outputs of both the MultiViewColorNet and FractalDimensionConvNet were concatenated and sent through a fully connected layer. The experiments, analogous to RQ2, were carried out on a selection of three glitches due to the limited and difficult availability of the auxiliary channel information.

5.2. THE AVALANCHE PACKAGE

While Avalanche [Lomonaco et al., 2021] offers a robust framework for handling a variety of data streams, the current version does have its limitations. It does not seamlessly integrate different modalities like images and text. Customization for multimodal data requires significant effort as the core framework lacks support. This lack of inherent support posed an additional risk and is the reason why the proposed solution for RQ3 in section 4.4.3 feels like not in line with the CL paradigm.

Another issue we encountered were the evaluation mechanisms, calculating the BWT and FWT metrics gave quite some errors.

These CL specific metrics were omitted during the time-consuming training of the RQ1 experiments.

Avalanche's performance can vary depending on the specific task and data at hand. It might not always provide the optimal solution for certain types of tasks, an example was the failure of the A-GEM strategy in the experiments of RQ2.

Possible alternatives are e.g. Continuum and Sequoia. Continuum [Douillard and Lesort, 2021], offers a high-level interface for CL with a variety of strategies and benchmarks. It is known for its simplicity and ease of use.

Sequoia [Normandin et al., 2021] is a flexible research framework specifically designed to make development easy and understandable. It provides a unified interface for a wide range of CL strategies and benchmarks.

6

CONCLUSIONS AND FUTURE WORK

6.1. CONCLUSIONS

This thesis effectively connects CL and traditional DL methods for classifying glitches in GW data. By combining strain data with auxiliary channel data using a multimodal fusion framework, the research shows notable enhancements in classification accuracy and provides a more profound understanding of glitch properties. The study emphasizes several important discoveries:

- **Effectiveness of CL strategies**: The implementation of various CL strategies such as Naive CL, Learning without Forgetting (LwF), Averaged Gradient Episodic Memory (A-GEM), Elastic Weight Consolidation (EWC), Dark Experience Replay (DER), DER++, and Supervised Contrastive Replay (SCR) revealed that while the Naive approach exhibited best performance, other methods also showed promise. A-GEM suffered from extreme catastrophic forgetting, indicating potential implementation issues or suboptimal hyperparameter choices.

- **Multimodal Fusion**: Integrating strain data (Q scans) with auxiliary channel data (fractal dimension matrices) resulted in exceptional outcomes, reaching up to 100% accuracy on the test set. This combination offered a more holistic view of the system, encapsulating various facets of the underlying physical processes and allowing the model to identify distinctive features not visible in individual modalities.

- **The Avalanche Framework**: Although the Avalanche Framework was useful for managing multiple datastreams, it faced difficulties in smoothly integrating diverse modalities and computing CL-specific metrics. Future research should explore alternative frameworks like Continuum and Sequoia for improved support and efficiency in multimodal environments.

- **Research Methodology**: The research approach, based on thorough experimentation and analysis, confirmed the efficacy of multimodal fusion in classifying glitches. Employing convolutional networks with dropout and batch normalization layers was successful in distinguishing between glitch types, even with the limited auxiliary data available.

6.2. FUTURE WORK

The research done as described in 3.1 and 4 is promising about the potential of investigating multi-modal fusion architectures for GW analysis, the combination of Spectrogram Image Analysis combined with Fractal Dimensions of Auxiliary Channels gave outstanding results 4.4.3.

Further research, especially the integration of attention [Niu et al., 2021; Vaswani et al., 2017] should be investigated on additional fractal dimension data, preferably with more than 3 glitch categories.

Another topic of interest and research is how to succesfully apply CL in a Multimodal setting. There is some research ongoing, one of the most notable solutions is the Climb benchmark [Srinivasan et al., 2022]. This benchmark is designed to study CL for multimodal tasks where it evaluates models in two phases. An upstream phase where the model is trained on a sequence of tasks and evaluated on its ability to retain knowledge from previous tasks while learning new ones. And a downstream phase where the model is evaluated on the ability to transfer knowledge to new tasks with minimal data.

BIBLIOGRAPHY

Benjamin P Abbott, Rich Abbott, TDea Abbott, Fausto Acernese, Kendall Ackley, Carl Adams, Thomas Adams, Paolo Addesso, RX Adhikari, Vaishali B Adya, et al. Gw170817: observation of gravitational waves from a binary neutron star inspiral. *Physical review letters*, 119(16):161101, 2017. 6

Benjamin P Abbott, Rich Abbott, Thomas D Abbott, Sheelu Abraham, Fausto Acernese, Kendall Ackley, Carl Adams, Vaishali B Adya, Christoph Affeldt, Michalis Agathos, et al. A guide to ligo–virgo detector noise and extraction of transient gravitational-wave signals. *Classical and Quantum Gravity*, 37(5):055002, 2020a. 3, 6

BP Abbott, R Abbott, TD Abbott, S Abraham, F Acernese, K Ackley, C Adams, RX Adhikari, VB Adya, Christoph Affeldt, et al. Gw190425: Observation of a compact binary coalescence with total mass $3.4 m_{\odot}$. *The Astrophysical Journal*, 892(1):L3, 2020b. 6

R Abbott, H Abe, F Acernese, K Ackley, S Adhicary, N Adhikari, RX Adhikari, VK Adkins, VB Adya, C Affeldt, et al. Open data from the third observing run of ligo, virgo, kagra and geo. *arXiv preprint arXiv:2302.03676*, 2023. 2

P Ajith. Addressing the spin question in gravitational-wave searches: Waveform templates for inspiralling compact binaries with nonprecessing spins. *Physical Review D*, 84(8): 084037, 2011. 3

Everton L. Aleixo, Juan G. Colonna, Marco Cristo, and Everlandio Fernandes. Catastrophic forgetting in deep learning: A comprehensive taxonomy, 2023. 44

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3366–3375, 2017. 10

Sofia Alvarez-Lopez, Annudesh Liyanage, Julian Ding, Raymond Ng, and Jess McIver. Gspynettree: A signal-vs-glitch classifier for gravitational-wave event candidates. *arXiv preprint arXiv:2304.09977*, 2023. 7

João Aveiro, Felipe F Freitas, Márcio Ferreira, Antonio Onofre, Constança Providência, Gonçalo Gonçalves, and José A Font. Identification of binary neutron star mergers in gravitational-wave data using yolo one-shot object detection. *arXiv preprint arXiv:2207.00591*, 2022. v, 5

Sara Bahaadini, Vahid Noroozi, Neda Rohani, Scott Coughlin, Michael Zevin, Joshua R Smith, Vicky Kalogera, and A Katsaggelos. Machine learning for gravity spy: Glitch classification and dataset. *Information Sciences*, 444:172–186, 2018. 2

Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pages 353–374, 2023. 7

Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 135:38–54, 2021. 10

Vincenzo Benedetto, Francesco Gissi, Gioele Ciaparrone, and Luigi Troiano. Ai in gravitational wave analysis, an overview. *Applied Sciences*, 13(17):9886, 2023. 2

OT Bişkin, İ Kırbaş, and A Çelik. A fast and time-efficient glitch classification method: A deep learning-based visual feature extractor for machine learning algorithms. *Astronomy and Computing*, 42:100683, 2023. 5

Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5497–5512, 2022. 17

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020. 17, 41, 43

LIGO Caltech. LIGO Lab | Caltech - faq. https://www.ligo.caltech.edu/page/faq, 2024a. Accessed: 2024-01-11. 1

LIGO Caltech. LIGO Lab | Caltech - what are gravitational waves? https://www.ligo.caltech.edu/page/what-are-gw, 2024b. Accessed: 2024-01-11. 1

LIGO Caltech. Introduction to LIGO & GWs - continuous gw. https://www.ligo.org/science/GW-Continuous.php, 2024c. Accessed: 2024-01-15. 3

Antonio Carta, Lorenzo Pellegrini, Andrea Cossu, Hamed Hemati, and Vincenzo Lomonaco. Avalanche: A pytorch library for deep continual learning. *Journal of Machine Learning Research*, 24(363):1–6, 2023. 21

Marco Cavaglia. Characterization of gravitational-wave detector noise with fractals. *Classical and Quantum Gravity*, 39(13):135012, 2022. v, 6, 7

Chayan Chatterjee, Linqing Wen, Foivos Diakogiannis, and Kevin Vinsen. Extraction of binary black hole gravitational wave signals from detector data using deep learning. *Physical Review D*, 104(6):064046, 2021. 3

Shourov Chatterji, Lindy Blackburn, Gregory Martin, and Erik Katsavounidis. Multiresolution techniques for the detection of gravitational-wave bursts. *Classical and Quantum Gravity*, 21(20):S1809, 2004. 4, 5, 25

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547, 2018a. 18

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018b. 15, 38

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, P Dokania, P Torr, and M Ranzato. Continual learning with tiny episodic memories. In *Workshop on Multi-Task and Lifelong Reinforcement Learning*, 2019. 48

Robert E Colgan, K Rainer Corley, Yenson Lau, Imre Bartos, John N Wright, Zsuzsa Márka, and Szabolcs Márka. Efficient gravitational-wave glitch identification from environmental data through machine learning. *Physical Review D*, 101(10):102003, 2020. 2, 7, 24

ContinualAI. Learn Avalanche in 5 minutes. https://avalanche.continualai.org/getting-started/learn-avalanche-in-5-minutes, 2024. Accessed: 2024-02-22. v, 20, 21

Adam Coogan, Thomas DP Edwards, Horng Sheng Chia, Richard N George, Katherine Freese, Cody Messick, Christian N Setzer, Christoph Weniger, and Aaron Zimmerman. Efficient gravitational wave template bank generation with differentiable waveforms. *Physical Review D*, 106(12):122001, 2022. 3

Neil J Cornish, Tyson B Littenberg, Bence Bécsy, Katerina Chatziioannou, James A Clark, Sudarshan Ghonge, and Margaret Millhouse. Bayeswave analysis pipeline in the era of gravitational wave observations. *Physical Review D*, 103(4):044006, 2021. 2, 3

Andrea Cossu, Antonio Carta, Vincenzo Lomonaco, and Davide Bacciu. Continual learning for recurrent neural networks: an empirical evaluation. *Neural Networks*, 143:607–627, 2021. 12

Elena Cuoco, Jade Powell, Marco Cavaglià, Kendall Ackley, Michał Bejger, Chayan Chatterjee, Michael Coughlin, Scott Coughlin, Paul Easter, Reed Essick, et al. Enhancing gravitational-wave science with machine learning. *Machine Learning: Science and Technology*, 2(1):011002, 2020. 1, 2, 3, 5

Derek Davis and Marissa Walker. Detector characterization and mitigation of noise in ground-based gravitational-wave interferometers. *Galaxies*, 10(1):12, 2022. 2, 3

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44 (7):3366–3385, 2021. 1, 2, 12, 13, 14

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 11

Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018. 18

Tom Dooney, Stefano Bromuri, and Lyana Curier. Dvgan: Stabilize wasserstein gan training for time-domain gravitational wave physics. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 5468–5477. IEEE, 2022. 54

Tom Dooney, Lyana Curier, Daniel Tan, Melissa Lopez, Chris Van Den Broeck, and Stefano Bromuri. cdvgan: One flexible model for multi-class gravitational wave signal and glitch generation. *arXiv preprint arXiv:2401.16356*, 2024. 54

Arthur Douillard and Timothée Lesort. Continuum: Simple management of complex continual learning scenarios. *arXiv preprint arXiv:2102.06253*, 2021. 20, 58

B Dubuc, SW Zucker, C Tricot, JF Quiniou, and D Wehbi. Evaluating the fractal dimension of surfaces. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):113–127, 1989. 6

Reed Essick, Patrick Godwin, Chad Hanna, Lindy Blackburn, and Erik Katsavounidis. idq: Statistical inference of non-gaussian noise with auxiliary degrees of freedom in gravitational-wave detectors. *Machine Learning: Science and Technology*, 2(1):015004, 2020. 2

Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018. 13

Tiago Fernandes, Samuel Vieira, Antonio Onofre, Juan Calderón Bustillo, Alejandro Torres-Forné, and José A Font. Convolutional neural networks for the classification of glitches in gravitational-wave data streams. *Classical and Quantum Gravity*, 40(19):195018, 2023. 2, 5

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 12

Konrad Gadzicki, Razieh Khamsehashari, and Christoph Zetzsche. Early vs late fusion in multimodal convolutional neural networks. In *2020 IEEE 23rd international conference on information fusion (FUSION)*, pages 1–6. IEEE, 2020. 20

Jing Gao, Peng Li, Zhikui Chen, and Jianing Zhang. A survey on deep learning for multimodal data fusion. *Neural Computation*, 32(5):829–864, 2020. 19

Timothy D Gebhard, Niki Kilbertus, Ian Harry, and Bernhard Schölkopf. Convolutional neural networks: A magic bullet for gravitational-wave detection? *Physical Review D*, 100(6):063015, 2019. 3, 5

Daniel George, Hongyu Shen, and EA Huerta. Deep transfer learning: A new deep learning glitch classification method for advanced ligo. *arXiv preprint arXiv:1706.07446*, 2017. 2

J Glanzer, S Banagiri, SB Coughlin, S Soni, M Zevin, Christopher Philip Luke Berry, O Patane, S Bahaadini, N Rohani, K Crowston, et al. Data quality up to the third observing run of advanced ligo: Gravity spy glitch classifications. *Classical and Quantum Gravity*, 40(6): 065004, 2023. v, 1, 2, 4

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 17

Stephen T Grossberg. *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, volume 70. Springer Science & Business Media, 2012. 12

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 17

IGWN. LIGO Run Plans. https://observing.docs.ligo.org/plan, 2024. Accessed: 2024-01-13. 1

Soichiro Isoyama, Riccardo Sturani, and Hiroyuki Nakano. Post-newtonian templates for gravitational waves from compact binary inspirals. *Handbook of Gravitational Wave Astronomy*, pages 1–49, 2020. 3

Seraphim Jarov, Sarah Thiele, Siddharth Soni, Julian Ding, Jess McIver, Raymond Ng, Rikako Hatoya, and Derek Davis. A new method to distinguish gravitational-wave signals from detector noise transients with gravity spy. *arXiv preprint arXiv:2307.15867*, 2023. 5, 7

Shasvath J Kapadia, Thomas Dent, and Tito Dal Canton. Classifier for gravitational-wave inspiral signals in nonideal single-detector data. *Physical Review D*, 96(10):104015, 2017. 3

Zixuan Ke, Bing Liu, Hu Xu, and Lei Shu. Classic: Continual and contrastive learning of aspect sentiment classification tasks. *arXiv preprint arXiv:2112.02714*, 2021. 10

Shivam Khare, Kun Cao, and James Rehg. Unsupervised class-incremental learning through confusion. *arXiv preprint arXiv:2104.04450*, 2021. 11

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020. 16

Serhat Kılıçarslan, Kemal Adem, and Mete Çelik. An overview of the activation functions used in deep learning algorithms. *Journal of New Results in Science*, 10(3):75–88, 2021. 29

Gyeongho Kim, Jae Gyeong Choi, Minjoo Ku, Hyewon Cho, and Sunghoon Lim. A multimodal deep learning-based fault detection model for a plastic injection molding process. *IEEE Access*, 9:132455–132467, 2021. v, 19

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. v, 2, 11, 13

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research), 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html. 11

Paloma Laguarta, Robin van der Laag, Melissa Lopez, Tom Dooney, Andrew L Miller, Stefano Schmidt, Marco Cavaglia, Sarah Caudill, Kurt Driessens, Jöel Karel, et al. Detection of anomalies amongst ligo's glitch populations with autoencoders. *arXiv preprint arXiv:2310.03453*, 2023. v, 6, 7, 29, 54

Paloma Laguarta, Robin van der Laag, Melissa Lopez, Tom Dooney, Andrew L Miller, Stefano Schmidt, Marco Cavaglia, Sarah Caudill, Kurt Driessens, Joël Karel, et al. Detection of anomalies amongst ligo's glitch populations with autoencoders. *Classical and Quantum Gravity*, 41(5):055004, 2024. 57

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion*, 58:52–68, 2020. 11

Yu-Kai Ling, Ren Chieh Yang, and Sheng-De Wang. Difficulty-aware mixup for replay-based continual learning. In *International Computer Symposium*, pages 305–316. Springer, 2022. 43

Kuan Liu, Yanen Li, Ning Xu, and Prem Natarajan. Learn to combine modalities in multimodal deep learning. *arXiv preprint arXiv:1805.11730*, 2018. 19, 20

Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido van de Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German I. Parisi, Fabio Cuzzolin, Andreas Tolias, Simone Scardapane, Luca Antiga, Subutai Amhad, Adrian Popescu, Christopher Kanan, Joost van de Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. Avalanche: an end-to-end library for continual learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2nd Continual Learning in Computer Vision Workshop, 2021. 20, 21, 30, 58

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017. 10, 15, 18, 19, 38

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 28

Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul PPP Grasman, and Eric-Jan Wagenmakers. A tutorial on fisher information. *Journal of Mathematical Psychology*, 80: 40–55, 2017. 14

Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3589–3599, 2021. v, 16, 44

Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019. 14

Benoit B Mandelbrot. Fractal geometry: what is it, and what does it do? *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 423(1864):3–16, 1989. 6

Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (5):5513–5533, 2022. v, 10, 11

Nicolas Y Masse, Gregory D Grant, and David J Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018. 9, 11

Brian McFee, Colin Raffel, Dawen Liang, Daniel P Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25, 2015. 5

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 32

Lester James Miranda. Understanding softmax and the negative log-likelihood". *ljvmiranda921.github.io*, 2017. URL https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/. 28

M Jehanzeb Mirza, Marc Masana, Horst Possegger, and Horst Bischof. An efficient domain-incremental learning approach to drive in all weather conditions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3001–3011, 2022. v, 10

M Mursalim and Ade Kurniawan. Multi-kernel cnn block-based detection for covid-19 with imbalance dataset. *International Journal of Electrical and Computer Engineering (IJECE)*, 2021. 27

Sam Ebin Sam Dhas Newgin. Enhancing ai interpretability: Saliency maps with histogram equalization, 2024. URL https://link.medium.com/HK190vk1bJb. [Blog post] Retrieved April 27, 2024, from Medium. v, 22

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011. 19

Alex Nitz, Ian Harry, Duncan Brown, Christopher M Biwer, Josh Willis, Tito Dal Canton, Collin Capano, Larne Pekowsky, Thomas Dent, Andrew R Williamson, et al. gwastro/pycbc: Pycbc release v1. 16.11. *Zenodo*, 2020. 5

Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021. 60

Fabrice Normandin, Florian Golemo, Oleksiy Ostapenko, Pau Rodriguez, Matthew D Riemer, Julio Hurtado, Khimya Khetarpal, Ryan Lindeborg, Lucas Cecchi, Timothée Lesort, et al. Sequoia: A software framework to unify continual learning research. *arXiv preprint arXiv:2108.01005*, 2021. 58

Guy Oren and Lior Wolf. In defense of the learning without forgetting for task incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2209–2218, 2021. 35

Rich Ormiston, Tri Nguyen, Michael Coughlin, Rana X Adhikari, and Erik Katsavounidis. Noise reduction in gravitational-wave data via deep learning. *Physical Review Research*, 2(3):033066, 2020. 3

Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. Continual deep learning by functional regularisation of memorable past. *Advances in Neural Information Processing Systems*, 33:4453–4464, 2020. v, 14, 15

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019. 12

Sungheon Park and Nojun Kwak. Analysis on the dropout effect in convolutional neural networks. In *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II 13*, pages 189–204. Springer, 2017. 27

Harold Pashler, Nicholas Cepeda, Robert V Lindsey, Ed Vul, and Michael C Mozer. Predicting the optimal spacing of study: A multiscale context model of memory. *Advances in neural information processing systems*, 22, 2009. 12

Jade Powell. Parameter estimation and model selection of gravitational wave signals contaminated by transient detector noise glitches. *Classical and Quantum Gravity*, 35(15): 155017, 2018. 6

Gravity Spy 2.0 project. Gravity Spy 2.0 wiki. `https://gswiki.ischool.syr.edu/`, 2024. Accessed: 2024-02-23. 26

Haoxuan Qu, Hossein Rahmani, Li Xu, Bryan Williams, and Jun Liu. Recent advances of continual learning in computer vision: An overview. *arXiv preprint arXiv:2109.11369*, 2021. 1, 2, 18

Andrinandrasana David Rasamoelina, Fouzia Adjailia, and Peter Sinčák. A review of activation function for artificial neural network. In *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 281–286. IEEE, 2020. 29

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. v, 16

Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 5

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 5

Florent Robinet, Nicolas Arnaud, Nicolas Leroy, Andrew Lundgren, Duncan Macleod, and Jessica McIver. Omicron: a tool to characterize transient noise in gravitational-wave detectors. *SoftwareX*, 12:100620, 2020. 4, 25

Wen-Hong Ruan, He Wang, Chang Liu, and Zong-Kuan Guo. Rapid search for massive black hole binary coalescences using deep learning. *Physics Letters B*, 841:137904, 2023. 3

Paul Ruvolo and Eric Eaton. Ella: An efficient lifelong learning algorithm. In *International conference on machine learning*, pages 507–515. PMLR, 2013. 9

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 21, 22

Tejas Srinivasan, Ting-Yun Chang, Leticia Pinto Alva, Georgios Chochlakis, Mohammad Rostami, and Jesse Thomason. Climb: A continual learning benchmark for vision-and-language tasks. *Advances in Neural Information Processing Systems*, 35:29440–29453, 2022. 60

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 5

Nergis Tomen and Jan C van Gemert. Spectral leakage and rethinking the kernel size in cnns. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5138–5147, 2021. 27

Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. v, vii, 9

Gido M Van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):4069, 2020. 10

Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022. vii, 1, 2, 9

Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991. 15

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 60

Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985. 17

Joshua T Vogelstein, Jayanta Dey, Hayden S Helm, Will LeVine, Ronak D Mehta, Tyler M Tomita, Haoyin Xu, Ali Geisa, Qingyang Wang, Gido M van de Ven, et al. Representation ensembling for synergistic lifelong learning with quasilinear complexity. *arXiv preprint arXiv:2004.12908*, 2020. 10

Johannes Von Oswald, Christian Henning, Benjamin F Grewe, and João Sacramento. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019. 10

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2023. 9, 18, 19

Yifan Wang, Xing Xu, Wei Yu, Ruicong Xu, Zuo Cao, and Heng Tao Shen. Comprehensive framework of early and late fusion for image–sentence retrieval. *IEEE MultiMedia*, 29(3): 38–47, 2022. 20

Eric W Weisstein. Minkowski-bouligand dimension. *MathWorld-A Wolfram Web Resource*, 2019. 6

Yunan Wu, Michael Zevin, Christopher PL Berry, Kevin Crowston, Carsten Østerlund, Zoheyr Doctor, Sharan Banagiri, Corey B Jackson, Vicky Kalogera, and Aggelos K Katsaggelos. Advancing glitch classification in gravity spy: Multi-view fusion with attention-based machine learning for advanced ligo's fourth observing run. *arXiv preprint arXiv:2401.12913*, 2024. 2, 7, 8, 56

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017. 14

Michael Zevin, Scott Coughlin, Sara Bahaadini, Emre Besler, Neda Rohani, Sarah Allen, Miriam Cabero, Kevin Crowston, Aggelos K Katsaggelos, Shane L Larson, et al. Gravity spy: integrating advanced ligo detector characterization, machine learning, and citizen science. *Classical and quantum gravity*, 34(6):064003, 2017. v, 1, 2, 4, 5, 54

Duoyi Zhang, Richi Nayak, and Md Abul Bashar. Exploring fusion strategies in deep learning models for multi-modal classification. In *Australasian Conference on Data Mining*, pages 102–117. Springer, 2021. 19

Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: a python toolbox for class-incremental learning. *SCIENCE CHINA Information Sciences*, 66(9):197101–, 2023a. doi: https://doi.org/10.1007/s11432-022-3600-y. 20

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Deep class-incremental learning: A survey. *arXiv preprint arXiv:2302.03648*, 2023b. 20

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 2005. 7

MULTIVIEWCOLORNET_RESNET18

```
class MultiViewColorNet_resnet18(nn.Module):
  def __init__(self, num_classes=10):
    # Call the parent class's constructor
    super(MultiViewColorNet_resnet18, self).__init__()
    # Initialize a pretrained ResNet-18 model with adjusted input size
    self.resnet = models.resnet18(weights='DEFAULT')
    # Access the first convolutional layer
    first_conv = self.resnet.conv1
    # Modify the kernel size of the first convolution
    first_conv.kernel_size = (7, 7)
    # Unfreeze all parameters in the model for training
    for param in self.resnet.parameters():
      param.requires_grad = True
    # Get the number of features in the last layer of the model
    num_features_in = self.resnet.fc.in_features
    # Replace the last layer with a new linear layer
    self.resnet.fc = nn.Linear(num_features_in, 120)
    # Add a second fully connected layer
    self.fc2 = nn.Linear(120, 84)
    # Add a third fully connected layer for the num_classes classes
    in the GravitySpy dataset
    self.fc3 = nn.Linear(84, num_classes)
    # Add a dropout layer to prevent overfitting
    self.dropout = nn.Dropout(p=0.3)
    # Add a batch normalization layer
    self.bn = nn.BatchNorm1d(120)

  def forward(self, x):
    # Forward pass: compute the output of the model by passing
    the input through the model
    x = self.resnet(x)
    # Apply batch normalization
    x = self.bn(x)
    # Apply the ReLU activation function
    x = F.relu(self.fc2(x))
    # Apply dropout
    x = self.dropout(x)
    # Pass the result through the final fully connected layer
    x = self.fc3(x)
    # Return the model's output
    return x
```

MULTIVIEWGRAVITYSPYDATASET DATALOADER

```
class MultiViewGravitySpyDataset(ImageFolder):
    def __init__(self, root, cls, transform=None):
        self.data_dir = root
        self.classes = cls
        self.class_to_indx = {c: i for i, c in enumerate(self.classes)}
        self.image_paths = []
        self.labels = []
        self.versions = ["0.5.png", "1.0.png", "2.0.png", "4.0.png"]
        for class_name in self.classes:
            class_path = os.path.join(root, class_name)
            for filename in os.listdir(class_path):
              if any(filename.endswith(version) for version in self.versions):
                self.image_paths.append(os.path.join(class_path, filename))
                self.labels.append(self.class_to_indx[class_name])
        self.transform = transform

    def __len__(self):
        return len(self.image_paths) // 4  # Assuming 4 versions for each image

    def __getitem__(self, idx):
        image_paths = self.image_paths[idx*4: (idx+1)*4]
        images = [Image.open(image_path).convert('RGB') for image_path
        in image_paths]
        if self.transform:
            images = [self.transform(img) for img in images]
        top_row = Image.fromarray(np.concatenate([np.array(images[0]),
        np.array(images[1])], axis=1))
        bottom_row = Image.fromarray(np.concatenate([np.array(images[2]),
        np.array(images[3])], axis=1))
        final_image = Image.fromarray(np.concatenate([np.array(top_row),
        np.array(bottom_row)], axis=0))
        temp = np.array(final_image)
        temp = temp.astype(np.float32)
        temp /= 255.0
        fused_image = torch.from_numpy(temp.transpose((2, 0, 1)))
        label = self.labels[idx*4]
        return fused_image, label
```

FRACTALDIMENSIONCONVNET

```
class FractalDimensionConvNet(nn.Module):
    def __init__(self):
        super(FractalDimensionConvNet, self).__init__()
        self.conv1 = nn.Conv1d(in_channels=50, out_channels=128,
        kernel_size=3, stride=1, padding=1) #adjust from 347 to 50
        self.bn1 = nn.BatchNorm1d(128)
        self.conv2 = nn.Conv1d(in_channels=128, out_channels=64,
        kernel_size=3, stride=1, padding=1)
        self.bn2 = nn.BatchNorm1d(64)
        self.conv3 = nn.Conv1d(in_channels=64, out_channels=32,
        kernel_size=3, stride=1, padding=1)
        self.bn3 = nn.BatchNorm1d(32)
        self.conv4 = nn.ConvTranspose1d(in_channels=32, out_channels=64,
        kernel_size=3, stride=1, padding=1)
        self.bn4 = nn.BatchNorm1d(64)
        self.fc1 = nn.Linear(64*56, 1024)
        #self.fc1 = nn.Linear(64*5, 1024) #adjust input size for fd statistics
        self.fc2 = nn.Linear(1024,256)
        self.dropout1 = nn.Dropout(0.1)
        self.dropout2 = nn.Dropout(0.3)
        self.fc3 = nn.Linear(256,3)

    def forward(self, x):
        x = nn.functional.selu(self.conv1(x))
        x = self.bn1(x)
        x = nn.functional.selu(self.conv2(x))
        x = self.bn2(x)
        x = nn.functional.selu(self.conv3(x))
        x = self.bn3(x)
        x = self.dropout1(x)
        x = nn.functional.selu(self.conv4(x))
        x = self.bn4(x)
        x = self.dropout2(x)
        x = x.view(x.size(0), -1)
        x = nn.functional.selu(self.fc1(x))
        x = nn.functional.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```