UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Evaluating Balanced Domain Regularizations for Multi-Domain Learning in Audio Classification Tasks

Alexandre Thurow Bender

Pelotas, 2023

Alexandre Thurow Bender

Evaluating Balanced Domain Regularizations for Multi-Domain Learning in Audio Classification Tasks

Dissertação apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Advisor: Prof. Dr. Ricardo Matsumura Araujo
Co-Advisor: Prof. Dr. Ulisses Brisolara Corrêa

Pelotas, 2023

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

B458e Bender, Alexandre Thurow

Evaluating balanced domain regularizations for multi-domain learning in audio classification tasks / Alexandre Thurow Bender ; Ricardo Matsumura Araujo, orientador ; Ulisses Brisolara Corrêa, coorientador. — Pelotas, 2023.

82 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2023.

1. Multi-domain learning. 2. Audio classification. 3. Machine learning. 4. Batch regularization. 5. Neural networks. I. Araujo, Ricardo Matsumura, orient. II. Corrêa, Ulisses Brisolara, coorient. III. Título.

CDD : 005

Elaborada por Simone Godinho Maisonave CRB: 10/1733

Alexandre Thurow Bender

Evaluating Balanced Domain Regularizations for Multi-Domain Learning in Audio Classification Tasks

Dissertação aprovada, como requisito parcial, para obtenção do grau de Mestre em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 07 de julho de 2023

Banca Examinadora:

Prof. Dr. Ricardo Matsumura Araujo (Orientador)
Doutor em Computação pela Universidade Federal do Rio Grande do Sul (UFRGS).

Prof. Dr. Ulisses Brisolara Corrêa (Coorientador)
Doutor em Computação pela Universidade Federal de Pelotas (UFPel).

Prof. Dr. Frederico Schmitt Kremer
Doutor em Biotecnologia pela Universidade Federal de Pelotas (UFPel).

Prof. Dr. Cristian Cechinel
Doutor em Ingenieria de la Información y del Conocimiento pela Universidad de Alcalá (UAH).

“I penetrated the outer cell membrane with a nanosyringe."
"You poked it with a stick?"
"No!" I said. "Well. Yes. But it was a scientific poke with a very scientific stick.”
— ANDY WEIR, PROJECT HAIL MARY

ABSTRACT

Collections of data obtained or generated under similar conditions are called domains or data sources. The distinct data acquisition or generation conditions are often neglected, but understanding them is vital to address any phenomena emerging from these differences that might hinder model generalization. Multi-domain learning seeks the best way to train a model to perform adequately in all domains used during training. This work explores multi-domain learning techniques that use explicit information about the domain of an example in addition to its class. This study evaluates a general approach (Stew) by mixing all available data and also two batch domain-regularization methods: Balanced Domains and Loss Sum. We train machine learning models with the listed approaches using datasets with multiple data sources for audio classification tasks. The results suggest that training a model using the Loss Sum method improves the performance of models otherwise trained in a mix of all available data.

Keywords: Multi-Domain Learning. Batch Domain Regularizations. Classification Tasks. Audio Processing.

RESUMO

BENDER, Alexandre Thurow. **Evaluating Balanced Domain Regularizations for Multi-Domain Learning in Audio Classification Tasks**. Advisor: Ricardo Matsumura Araujo. 2023. 82 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2023.

Coleções de dados obtidos ou gerados em condições semelhantes são chamadas de domínios ou fontes de dados. As condições distintas de aquisição ou geração de dados são muitas vezes negligenciadas, mas compreendê-las é vital para abordar quaisquer fenômenos emergentes dessas diferenças que possam impedir a generalização de modelos. O aprendizado multidomínio busca a melhor forma de treinar um modelo para que ele tenha um desempenho adequado em todos os domínios utilizados durante o treinamento. Este trabalho explora técnicas de aprendizado multidomínio que usam informações explícitas sobre o domínio de um exemplo, além de sua classe. Este estudo avalia uma abordagem geral (*Stew*) misturando todos os dados disponíveis e também dois métodos de regularização de domínios: *Balanced Domains* e *Loss Sum*. Treinamos modelos de aprendizado de máquina com as abordagens listadas usando conjuntos de dados com múltiplas fontes para tarefas de classificação de áudio. Os resultados sugerem que treinar um modelo usando o método Loss Sum melhora a performance de modelos anteriormente treinados em uma mistura de todos os dados disponíveis.

Palavras-chave: Aprendizado Multidomínio. Regularizações de Domínio em Batch. Tarefas de Classificação. Processamento de Áudio.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

MDL	Multi-Domain Learning
DNN	Deep Neural Network
CNN	Convolutional Neural Network
ResNet	Residual Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
DAPS	Device and Production Speech
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit

CONTENTS

1 INTRODUCTION

Limited data quantity is already a well-established concern when training machine learning models since few examples to learn from may not be sufficient for a model to generalize well to new data (LECUN; BENGIO; HINTON, 2015; DOMINGOS, 2012). However, recent years have shifted the attention of researchers towards the importance of data quality in achieving high-performance models (JAIN et al., 2020; SCHWEIGHOFER, 2022; SAMBASIVAN et al., 2021). Creating datasets representative of real-world information while maintaining a large number of labeled examples is challenging. This is because collecting and annotating samples in the wild is significantly more costly than automatically capturing or generating examples and their labels in a controlled environment. Additionally, even when striving for real-world conditions, data will often be collected with specific devices (e.g. using the same camera for capturing images) or environmental conditions (e.g. recording audio clips indoors). Collections of data obtained or generated under similar conditions are referred to as domains or data sources.

The distinct conditions of data acquisition or generation are often neglected, but understanding them is vital to address any phenomena emerging from these differences that might hinder model generalization. Domain shift is one such phenomenon and refers to when the distribution of the data used during training and the distribution of the data encountered during deployment is different (QUINONERO-CANDELA et al., 2008). There might be several reasons for domain shift to occur, such as changes in the data generation process, variations in the data capturing devices, or even due to the presence of different types of noise or structure in the data. This inconsistency can cause a significant reduction in performance for models trained on one domain but deployed on another, further evidencing the importance of considering the potential for domain shift when designing and deploying machine learning models in real-world applications.

There are several types of domain shift, and despite being discreet, covariate shift is arguably the most common (QUINONERO-CANDELA et al., 2008). It occurs when the input distribution of training and test is different, but the underlying task remains the same. For example, a model trained on images taken during daylight hours may

perform poorly when tested on images taken at night, irrelevant to the task.

One of the main challenges in dealing with domain shift is the lack of labeled data from the target domain (GANIN; LEMPITSKY, 2015), which makes it difficult to adapt the model to the new distribution. Several methods have been proposed to tackle this problem, including domain adaptation techniques such as transfer learning (WEISS; KHOSHGOFTAAR; WANG, 2016), adversarial training (GANIN et al., 2016), and meta-learning (VANSCHOREN, 2018). These techniques aim to align the distributions of the source and target domains or to learn domain-invariant representations.

Other notorious approaches in addressing the challenge of limited data are pre-training and fine-tuning. Machine learning models are commonly trained and evaluated using examples from the same domain. However, whenever there is limited data available for a specific task, a popular solution is to pre-train a model using out-of-domain information (usually in the form of a different, more extensive dataset) and then fine-tune it to the target domain. This technique has become favored over the past years (NIU et al., 2020), as it is accessible to use while also allowing a faster training process. Another advantage of pre-training is the reduced risk of overfitting, notably when working with smaller datasets.

Fine-tuning a pre-trained model on a different dataset is a potential solution whenever there is a single target domain and performance in the pre-trained domain is not necessarily a concern. But whenever performance in the pre-trained domain becomes desirable, this approach might encounter difficulties. In fact, this is a major problem when training models on multiple domains (RIBEIRO; MELO; DIAS, 2019). Maintaining performance in an already trained domain while adding new knowledge to the model is a challenge of its own, as the model is prone to forgetting its previous knowledge (GOODFELLOW et al., 2014). This particular issue is called catastrophic forgetting (FRENCH, 1999), and to overcome it when learning multiple domains at the same time, other techniques must be used.

Traditionally, the standard approach is to mix all training data without any particular concern for their pertaining domains. While doing this might be enough given sufficient data, significantly large datasets and the computational power to train models using them are not easily attainable. One of the reasons for this approach to be acceptable in these conditions is the high difference across examples and domains: if the data does not have a prominent domain, the model is pushed towards domain-agnostic representations. In other words, the domain-specific characteristics in data samples are diluted for not holding a common structure, and as such, they are discarded as noise.

In the vast majority of cases, data does not display this richness of domains. Datasets often have no more than two or three sources of data acquisition. In light of these limiting factors, the potential benefits of using domain information from samples ex-

plicitly remain largely uninvestigated. This study explores techniques that incorporate domain knowledge during the training process of machine learning models when using datasets with multiple sources of data. As such, this work proposes injecting domain information by guaranteeing balanced representations of each domain in a batch, building upon the work of Bender (2022). We investigate the effects of previously proposed approaches and expand them for further comparison. Differently from previous works, mostly which used image classification tasks as a basis for evaluation, we assess the training methods using audio classification tasks. To the best of our knowledge, there are no other batch domain regularization evaluations or proposals using audio data at the present date.

This research aims to understand the best way to learn from multi-domain datasets at once, dismissing the need to train multiple models for different situations. For this, we explore batch domain regularizations, which are usually overlooked in multi-domain learning. Additionally, we expect any potential gains in this regard will directly benefit smaller organizations and individuals with limited access to extremely large and varied datasets that overcome multi-domain issues.

The current study is motivated by the hypothesis that, during the training of machine learning models on multi-domain tasks, the training process takes advantage of using this data and its domain explicitly. We evaluate a general approach of mixing data from different domains together for training a machine learning model and also two new methods previously only tested for images: the first method, Balanced Domains, adapts the general approach by balancing the number of samples from every domain in each batch during training, the second method, Loss Sum, calculates the loss of each domain using the cross entropy loss function separately applied to each batch and sums them together before finally running backpropagation.

The present work is divided into 6 chapters: Chapter 1 provides context and formulates the motivation for this study, as well as its objective; Chapter 2 gives a brief overview of the history of the area and reviews important concepts useful for understanding this work. Chapter 3 discusses notable previous studies in this area. Chapter 4 describes the decision-making process guiding the study, the performed experiments, and their configuration. Chapter 5 presents and discusses the results of the experiments. Finally, Chapter 6 concludes the work.

2 THEORETICAL BACKGROUND

This Chapter provides key concepts relevant to understanding this work. It goes over a brief history of the area, the perceptron, feedforward neural networks, convolutional neural networks, residual neural networks, loss functions, audio processing in neural networks, and multi-domain learning.

2.1 Brief History

The current rampant attention towards artificial neural networks can be traced back to the perceptron algorithm (ROSENBLATT, 1958) in 1958, its biological neuron inspiration and modeling is often credited as one of the most striking early results in connectionist computational models. In contrast to symbolic artificial intelligence expressing knowledge as sets of rules, connectionism approaches to cognitive science are based on modeling the structure of the biological brain, commonly in the form of neural networks (GARSON, 1997).

Most of the research into perceptrons came to an abrupt end in 1969 when Minsky published the book Perceptrons (MINSKY; PAPERT, 2017) outlining the mathematical limits of what the artificial neuron could accomplish. In addition to this, the significant success of symbolic reasoning in applications like the General Problem Solver (NEWELL; SHAW; SIMON, 1959) directed the attention of researchers towards the exploration of symbolic systems as the essence of intelligence. Thus connectionist approaches were vastly abandoned for the next decade, and funding for these projects was scarce until the early 1980s (CREVIER, 1993).

Since then, two major events reignited interest in the connectionist paradigm. One of them was the backpropagation algorithm (RUMELHART; HINTON; WILLIAMS, 1986) back in 1986, allowing for the efficient training of multilayer neural networks. While the other significant circumstance pushing connectionism was the notorious success of deep learning in the tasks of image classification (KRIZHEVSKY; SUTSKEVER; HINTON, 2017) and speech recognition (HINTON et al., 2012) in 2012. In fact, deep learning gained a rich variety of network architectures, including deep feed-

forward neural networks (MOHAMED et al., 2009) (HINTON et al., 2012), convolutional neural networks (CNNs) (LECUN et al., 1989), residual neural networks (ResNets) (HE et al., 2016), long short-term memories (LSTMs) (HOCHREITER; SCHMIDHUBER, 1997), other recurrent neural networks (RNNs) (RUMELHART; HINTON; WILLIAMS, 1985), and recently transformers (VASWANI et al., 2017).

Deep learning is a relatively new archetype in machine learning, where neural architectures are endowed with a huge number of parameters. To better grasp the scale of this: earlier in 2022, Google Research trained a 540-Billion parameter language model (CHOWDHERY et al., 2022). Assuming we represent the trainable weights of the network as 32-bits floating point data types, this neural model would use approximately 16TB of memory. Surely this particular instance is an exceptionally large network, but the fact remains that neural network models have been getting significantly bigger (BROWN et al., 2020; THOPPILAN et al., 2022; RADFORD et al., 2022). Training these models is a non-negligible challenge of its own, and takes equally massive amounts of data and processing power. To enable such a task, models leverage the advances in parallelism and cloud computing, using technologies like graphics processing units (GPUs) (DALLY; KECKLER; KIRK, 2021), tensor processing units (TPUs) (JOUPPI et al., 2017), and datacenter network topologies (SINGH et al., 2015).

Despite the recent boon in deep learning originating with image processing (KRIZHEVSKY; SUTSKEVER; HINTON, 2017), the technology was quickly adopted for a wide variety of tasks in different domains. Whenever there is sufficient available data, it is not unusual for deep learning methods to outperform the more traditional approaches by a significant margin (KRIZHEVSKY; SUTSKEVER; HINTON, 2017; MNIH et al., 2013; DO et al., 2019).

2.2 The Perceptron

The perceptron (ROSENBLATT, 1958; MINSKY; PAPERT, 2017) is an important concept in the field of machine learning. It is usually described as a simple model of a biological neuron, and it serves as the fundamental building block for neural networks, more complex models. The perceptron takes multiple input values, multiplies them by their corresponding weights, and sums the results. An activation function is used to check whether the summed values achieve a certain threshold.

A graphical representation of the perceptron can be seen in Figure 1. Furthermore, Equation 1 depicts the perceptron mathematically: $\hat{y}$ is the perception output, $g(\cdot)$ is the activation function, x denotes the inputs, and w represents the weights, with w_0 being the bias component. The purpose of the bias is to displace the activation function regardless of the inputs, and it can be seen as analogous to the role of a constant in a linear function.

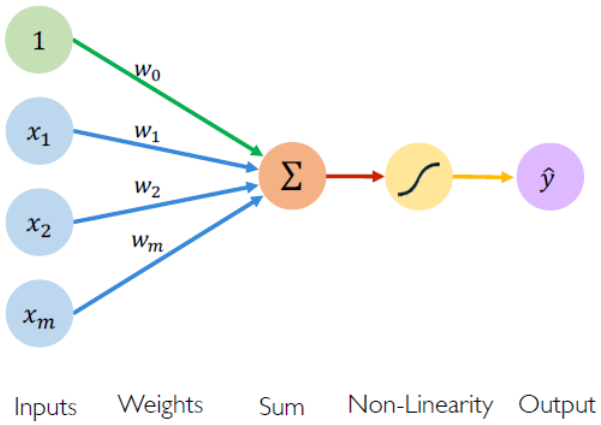

Figure 1 – Graphical representation of the perceptron. The nodes in blue and green denote the input layer (the green one being the bias component), the red node depicts their sum by weight elements, the yellow node is the activation function (depicted as a sigmoid), and the purple node is the model output. Source: Amini (2020).

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{n} w_i x_i\right) \tag{1}$$

During its training process, the perceptron adjusts its weights to match the input and outputs seen in the training data. The algorithm used to update the weights is commonly known as the Perceptron Learning Rule, which minimizes the error between the predictions and the expected value. Its formula can be seen in Equation 2, in which: w_n is the new weight, w_0 is the old weight, α is the learning rate, t is the expected output, and x_i is the input. The learning rate controls how much the weight is updated in each iteration, typically serving as an attenuator to mitigate drastic parameter alterations that hinder the optimization process. It is a core hyperparameter, not only for perceptrons but also for artificial neural networks in general.

$$w_n = w_0 + \alpha t x_i \tag{2}$$

The perceptron learning rule guarantees convergence should a hyperplane exist so as to linearly separate the input vectors (Figure 2). However, if the input vectors are not linearly separable, it is not easy to classify them correctly. In fact, this is a major limitation of the perceptron model, further addressed by following model architectures.

2.3 Feedforward Neural Networks

Feedforward neural networks (GOODFELLOW; BENGIO; COURVILLE, 2016) are the natural evolution from the perceptron algorithm, and they consist of layer association of elementary neuron units very similar to the perception. Each neuron performs a

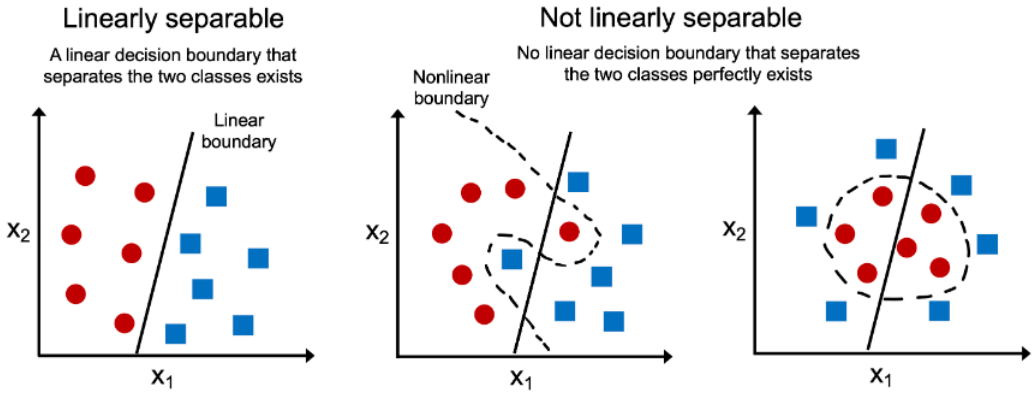

Figure 2 – 2-Dimensional Scatterplot visualization of two different classes (red circles and blue squares) representing linearly and non-linearly separable datasets. Source: Kumar (2022).

weighted sum of the inputs it receives, applies an activation function to produce its output, and finally passes its result as input to the neurons in the subsequent layer. Hence, the term "feedforward" alludes to the flow of information across the network, starting at the input layer, passing through hidden layers, and finally arriving in the output layer (Figure 3).

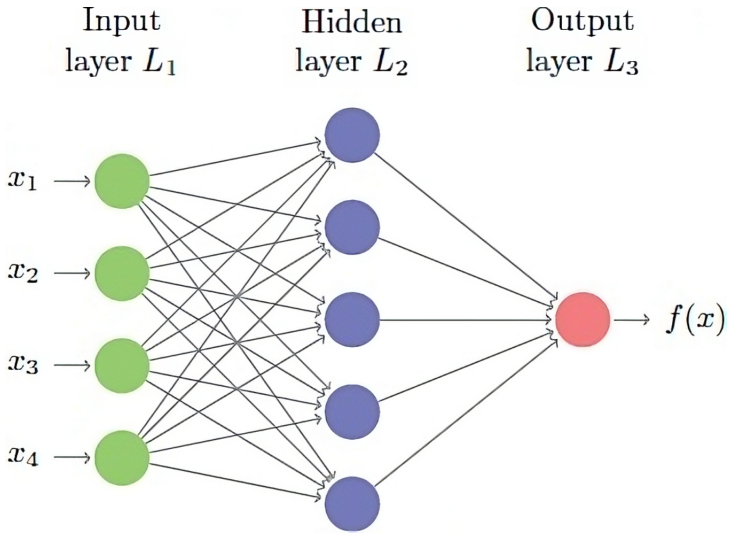

Figure 3 – Diagram of a feedforward neural network with a single hidden layer. Each processing layer derives dimensional transformations of the previous layers. Source: Boehmke (2020).

The neurons in the initial layer represent specific input features from the entry vector. The network learns intermediate representations of these features in its hidden layers. The hidden layers are formed of multiple neurons, and their quantity in each layer denotes the dimensionality of the learned representation at that point in the network topology. The first layers of a neural network learn to detect very simple patterns, but the further toward the output layer, the more complex and abstract these patterns are (Figure 4).

The final layer of the neural network produces the predictions of the model, and

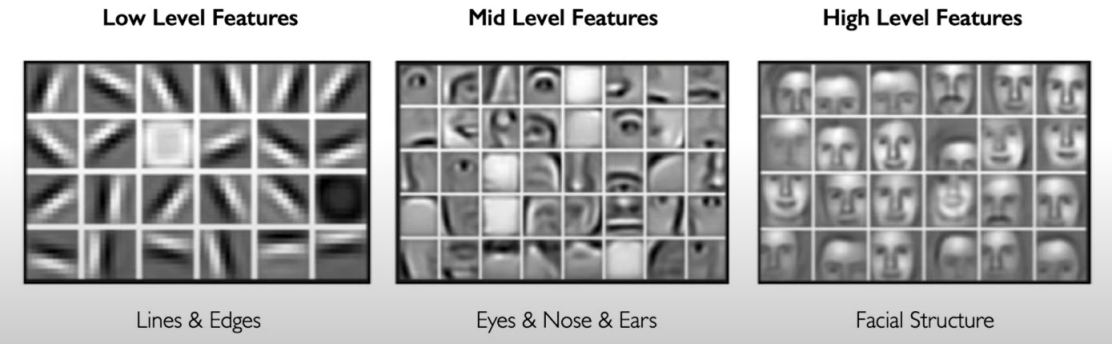

Figure 4 – Layers near the input typically learn simple elementary features like lines and curves in various orientations. The middle layers then use them as building blocks to distinguish more complex shapes like eyes, noses, and ears. As such, layers near the output are able to use this information to recognize specific faces using facial structure configuration, for example. Source: Amini (2020).

its number of neurons will depend on the nature of the problem being solved. For example, in classification problems, it is common to have one neuron for each class, their values often being interpreted as the class probabilities. In regression tasks, the output may consist of a single neuron generating a continuous value.

The training phase of a neural network is significantly more complex when compared to the perceptron. Adjusting the weights and biases of the network is an optimation task, thus optimation algorithms like gradient descent are used. If we understand the parameter adjustment process as an optimization problem, then the loss function is analogous to the evaluation function. Loss functions are further detailed in Section 2.6. For now, it suffices to understand them as a cost function to evaluate how well a model performs. The major challenge in training DNNs lies in estimating the error responsibility for each layer, particularly when calculating the error in layers further away from the output and calculating. The backpropagation algorithm (RUMELHART; HINTON; WILLIAMS, 1986) addresses this by propagating the error from the output layer back to the input layer, in order to adjust the weights and biases in each layer appropriately.

Furthermore, the training algorithm enabled the training of neural networks with multiple layers (Figure 5), heralding the deep learning paradigm. Deep Neural Networks (DNNs) are more complex architectures, able to model increasingly challenging problems. However, the ramping amount of layers and parameters leads to additional training complications, particularly regarding the preservation of the error gradient during the backpropagation algorithm. This phenomenon is known as vanishing gradient (HOCHREITER, 1998), where the gradient component that carries the weight update information diminishes the more it travels toward the initial layers of the network. In practice, this enormously increases the time it takes to train such models.

Several methods were proposed to mitigate complications arising from the vanishing gradient, including ReLU-based activation functions (AGARAP, 2018) and skip

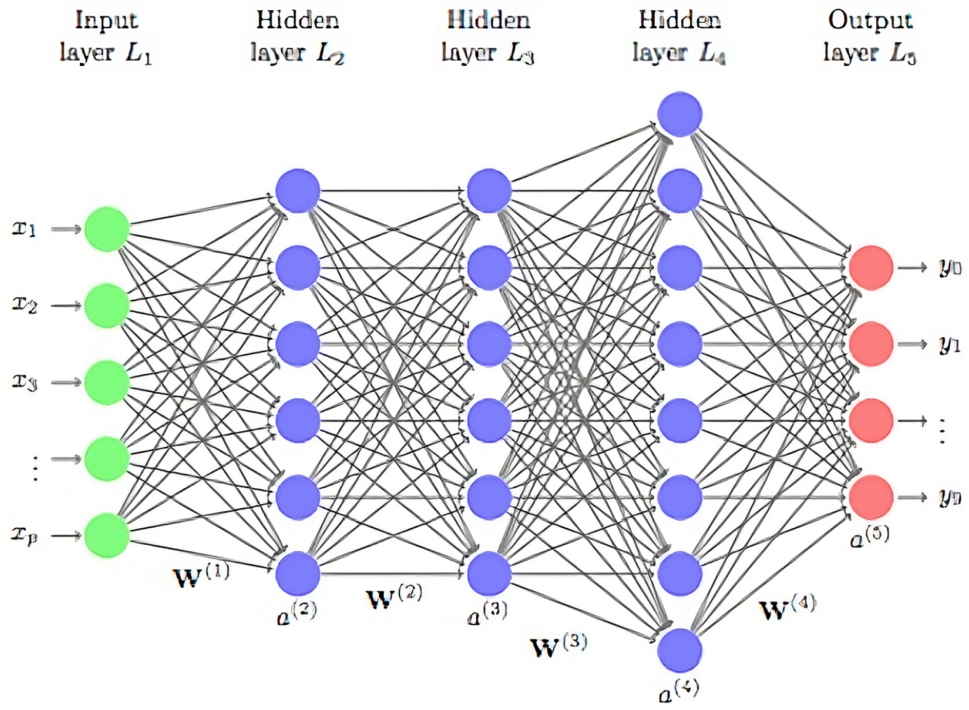

Figure 5 – Diagram of a deep neural network with multiple hidden layers. Deeper neural network models are more representative and capable of capturing more complex patterns. Source: Boehmke (2020).

connections (HE et al., 2016). The ReLU activation function only saturates on the negative side, avoiding the saturation behavior seen in Sigmoid activation functions (Figure 6). This saturation can cause gradients to diminish rapidly during backpropagation, making it difficult for earlier layers to learn. Skip connections (Figure 6) bypass one or more neural network layers, and by doing so they allow the uninterrupted flow of information from earlier to subsequent layers, thus alleviating the diminishing of the gradient throughout the network. Skip connections are an important element in Residual Neural Networks (ResNets), further elaborated in Section 2.5.

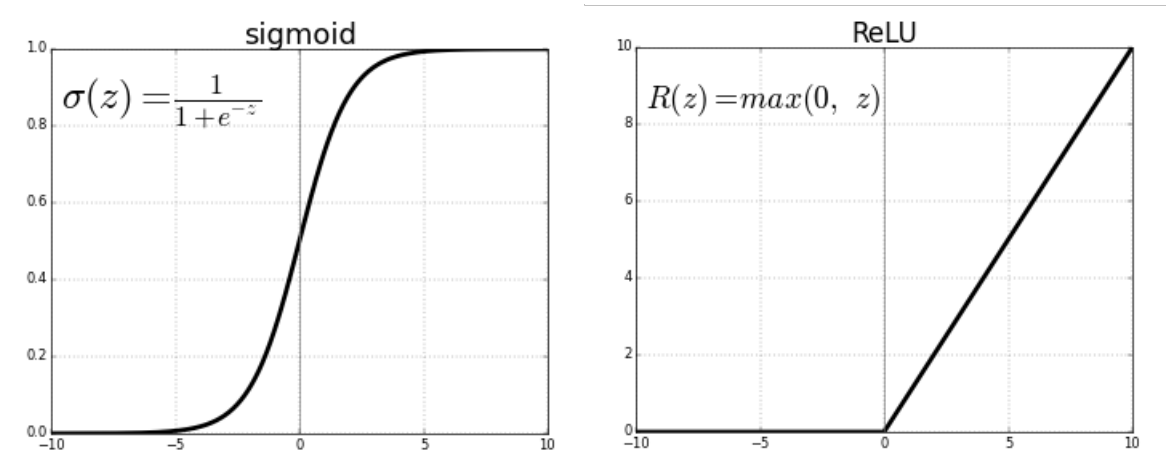

Figure 6 – Sigmoid and ReLU activation functions. In contrast to the saturating sigmoid activation function, function, the ReLU activation function does not saturate. Source: Becker (2020).

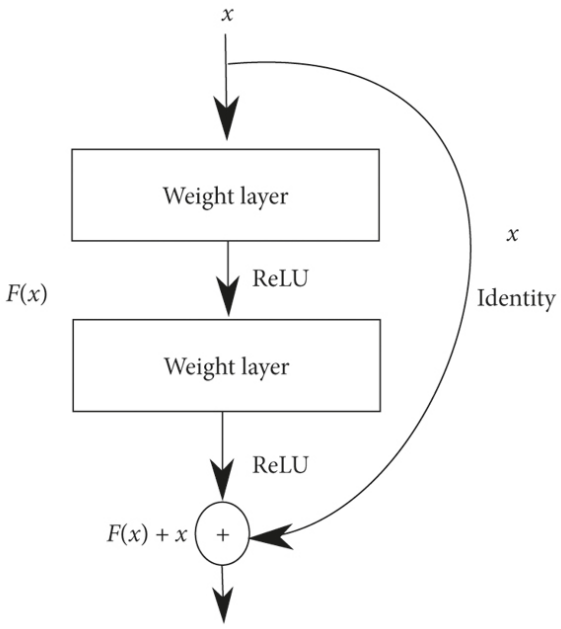

Figure 7 – Residual building blocks depicting the skip-connection. Notice how they essentially enable a shortcut of information across network layers. Source: He et al. (2016).

Neural networks are generalist architectures, and have successfully been applied to a variety of tasks, including image classification, speech recognition, time series forecasting, and natural language processing. Despite their lack of domain-specific dependency modeling (i.e. temporal dynamics or spacial locality), they are regarded as the structural foundation for more complex network architectures that address these limitations.

2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) (LECUN et al., 1989) are deep neural network variations specialized in processing data with a grid-like structure, typically used to address computer vision tasks. Referring back to Figure 5 in the previous section: notice the vast amount of neuron connections — in vanilla DNNs, each neuron is used as input for every neuron in the next layer. The core idea of CNNs is to exploit spatial locality inherently present in the structure of the input data. They limit the number of connections between neurons by introducing convolutional layers. Convolution layers apply convolution operators to the data using learnable filters called kernels. These structures slide across the input data summing element-wise multiplications to produce feature maps. The overall purpose of convolutional layers is to capture local dependencies of spacial patterns explicitly.

Similarly to DNNs, the feature maps increase in complexity the more they approach the end of the network. Suppose an animal image classification task: initial kernels learn to detect edges, lines, and other basic geometric shapes; while kernels further

up on the network build up on previous features and are capable of detecting ears, paws, whiskers, eyes, and common fur texture patterns. Therefore the final layer is the most semantically abstract layer, with its output returning which animal is in the initial image.

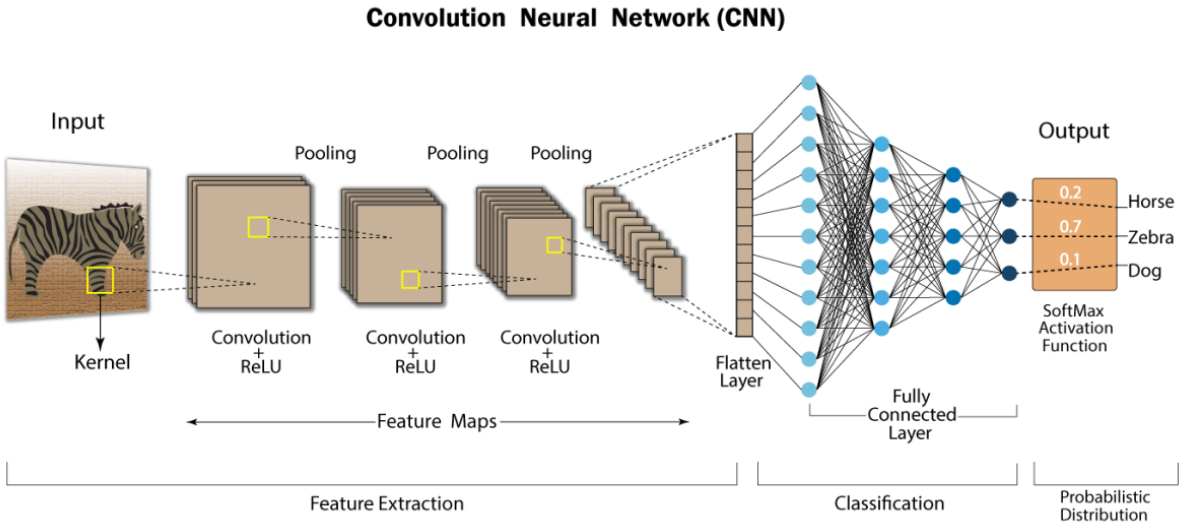

Figure 8 – Deep Convolutional Neural Network illustration. In addition to deep feedforward neural networks fully connected layer, CNNs use convolutional operators to build feature maps. Source: Swapna (2020).

Structurally, CNNs are formed by convolutional layers, pooling layers, and the fully-connected layer (Figure 8). Whereas the previously described convolution layers are responsible for capturing local patterns, the pooling layers are intended to reduce the spatial dimensions of the resulting feature maps, while also retaining the relevant information. Often, multiple convolution and pooling layers are applied in sequence. This organization of layers results in a constant reduction of feature map sizes across the network that gives it a "bottleneck" appearance (Contrasting to DNNs). As such they are often referred to as pyramidal architectures. The feature maps are then flattened and sent to a fully-connected layer (DNN), where the final output is decided using the same process previously described in Section 3.

2.5 Residual Neural Networks

Initially proposed to mitigate the vanishing gradient problem, Residual Neural Networks (ResNets — not to be confused with RNNs, which refer to Recurrent Neural Networks) (HE et al., 2016) are a specialized type of deep neural network that introduces the concept of skip connections or residual connections (Figure 7). Their key idea is to, instead of learning direct mappings, learn residual functions. The latter refers to the difference between the desired output and input to one or more layers (i.e. rather than learning the complete transformation from input to output, learning the residual

information that can be added to the input to obtain the output). This is reminiscent of the mathematically simpler idea of, instead of directly modeling a clean signal from a noisy signal, representing its noise, intending to subtract it from the original signal.

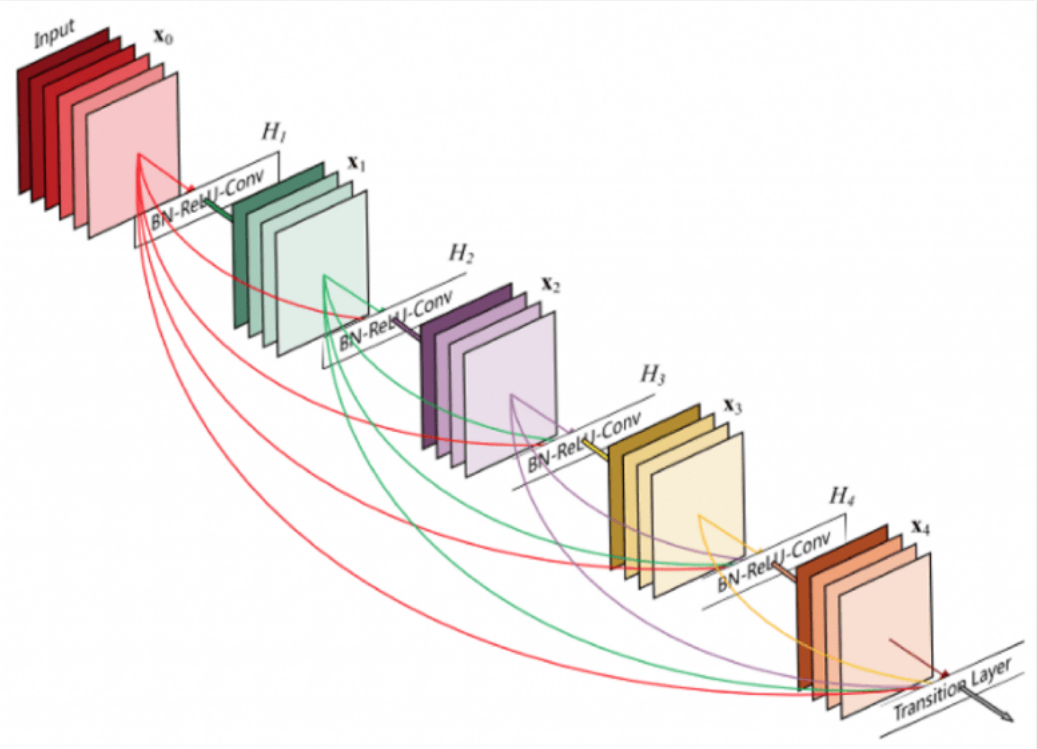

Figure 9 – Residual Neural Network illustration. In this residual block example (from the DenseNet architecture), all layers take all preceding feature maps as input. Source: HUANG, Densely Connected Convolutional Networks.

The residual connections can be seen as literal shortcuts in the ResNet topology, thus enabling the network to propagate the gradient more effectively across layers. Expectedly, this alleviates the vanishing gradient problem and allows for the training of deeper neural networks. This depth allows the networks to learn more complex and abstract representations, leading to improved performance.

2.6 Loss Functions

Loss functions are mathematical cost functions commonly used to quantify discrepancies between predicted values and expected values in supervised machine learning algorithms. They can be understood as error functions intended to measure how well a model performs. They play a vital part in supervised learning algorithms by providing quantifiable values to be minimized during their training loop when the weight parameters are adjusted. Because loss functions are an important aspect of the present work, it is pertinent to review them.

There are several loss function types and variations depending on the task being performed. Even though classification tasks are commonly seen as intuitive examples

whenever discussing supervised machine learning problems, whenever the discussion leans toward loss metrics to update model weights, it is actually regression tasks that become the more didactic examples, as their loss functions are fairly straightforward. The most common ones are Mean Absolute Error (Equation 3), Mean Squared Error (Equation 4), and Root Mean Squared Error (Equation 5). In these equations, M is the number of examples, y_i is the model output, and $\hat{y}_i$ is the expected output.

$$L_{MAE} = \frac{1}{M}\sum_{i=1}^{M}|y_i - \hat{y}_i| \tag{3}$$

$$L_{MSE} = \frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 \tag{4}$$

$$L_{RMSE} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2} \tag{5}$$

Despite their numerous variations, loss functions for regression tasks have a common structure: a distance comparison between two scalar values aggregated using some mathematical mechanism. In practice, we see a subtraction between the predicted and expected outputs, followed by a mathematical artifice to account for signal variations. This is because it is possible to either overshoot or undershoot the expected output when making predictions, and both of these situations are errors regardless of the signal. Therefore, squaring the value (as seen in Equations 4 and 5) or using the modulus operator (seen in Equation 3) are algebraic means to stop them from negating each other when we sum errors across examples. The aggregation mechanism itself is typically a simple arithmetic mean operation, as seen in all three equations.

Loss functions for classification tasks are fundamentally different in the sense that classes are usually depicted as discrete values. As such, simply subtracting the predicted and expected classes would not yield comprehensive results. In fact, doing this would imply creating an ordinal relationship between classes (i.e. mistaking class 1 for class 5 would penalize the model more than mistaking class 1 for class 2). This is not desirable, as classification problems usually do not present ordinal characteristics, and classes tend to be similarly distant and different from each other.

An alternative would be to treat errors using a binary approach: all mistakes committed by the model would be treated equally. As a matter of fact, this is exactly how the perceptron training rule works. However, this approach is severely limited due to its lack of granularity. To account for this, instead of applying classification loss functions using the predicted class, they are applied using model logits. Logits refer to the model raw class values in the last layer prior to the activation function. They can be seen as the model degree of certainty for each class. It is a common (sometimes necessary)

practice to apply a Softmax Function to the logits, which fits them into the $[0, 1]$ interval (Equation 6). Here z is an input vector of K real numbers. One of the benefits of doing so is that it becomes convenient to interpret the bounded logit values as class probability scores. Normalizing the output logits using a softmax function is a requirement for Cross-Entropy loss. Comparing the logits with the expected class improves model training by providing higher mistake granularity.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{6}$$

Classification loss functions like Cross-Entropy perform the comparison by applying a dot product operator between the expected output and the logarithm of the logit value of that class. This also ensures the error is not linear but instead decreases logarithmically with respect to how far it is to the desired class output.

The general Cross-Entropy loss form for multi-class problems can be seen in Equation 7. In the following equations, N refers to the number of classes in a multi-class problem. Differently from previous equations, the summation in Equation 7 does not refer to different examples but to classes, as each class has its logit evaluated and compared. For binary classification problems, Equation 7 can have its summation expanded to accommodate two classes, as seen in Equation 8, depicting Binary Cross Entropy.

These two equations can be further expanded to support a reduction mechanism in order to be applied to a collection of examples (similar to the previous regression loss functions). Equations 9 and 10 show the reduction format for Equations 7 and 8, respectively. It is relevant to note the logarithm operator is negative between zero and one. To account for this, all Cross-Entropy formulas have the need for a leading signal inversion.

$$L_{CE} = -\frac{1}{N} \sum_{i=1}^{N} \hat{y}_i \cdot \log(y_i) \tag{7}$$

$$L_{BCE} = -\Big[\hat{y}_i \cdot \log(y_i) + (1 - \hat{y}_i) \cdot \log(1 - y_i)\Big] \tag{8}$$

$$L_{CE-Reduction} = -\frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{N} \hat{y}_{ij} \cdot \log(y_{ij}) \tag{9}$$

$$L_{BCE-Reduction} = -\frac{1}{M} \sum_{i=1}^{M} \Big[\hat{y}_i \cdot \log(y_i) + (1 - \hat{y}_i) \cdot \log(1 - y_i)\Big] \tag{10}$$

Therefore, while regression loss functions are scalar distance comparison metrics, classification loss functions lean towards being distribution comparison metrics. They

use the dot product operator to perform this comparison, and also use a logarithmic scale for the model class logit output.

Another relevant concept associated with loss functions is that of loss landscape (Figure 10) It refers to the visual representation of model performance (depicted using loss value scores) of model parameters in the weight space. In other words, it denotes how different weight parameter combinations perform. It is a key factor for supervised learning optimization algorithms, like gradient descent as it directly impacts the effectiveness of model training and their resulting performance.

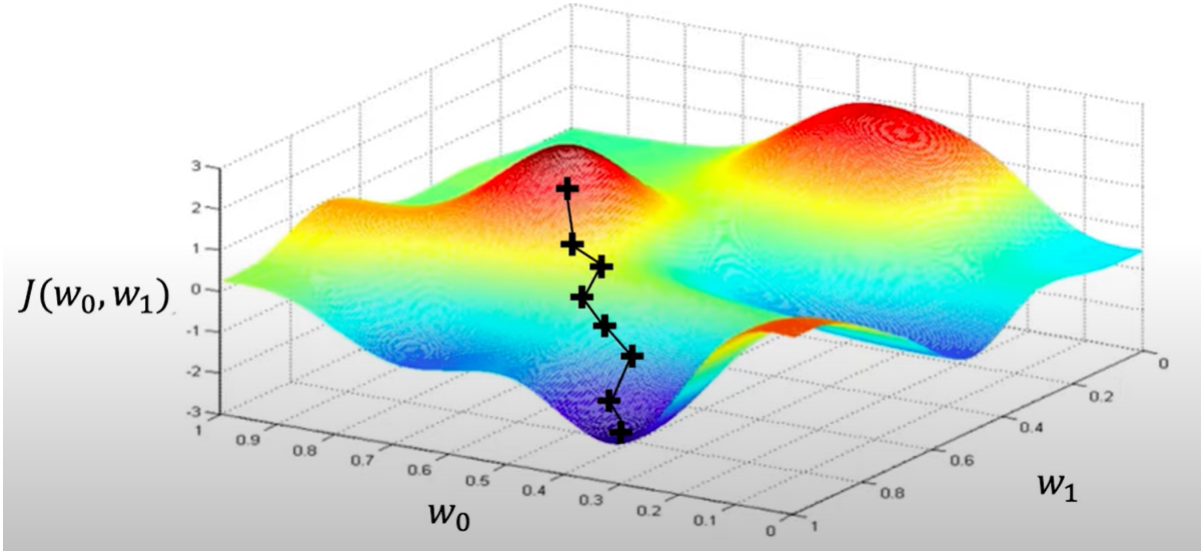

Figure 10 – Gradient Descent optimization algorithm traversal in an arbitrary loss landscape (2-dimensional objective function). Source: Amini (2020).

Admittedly, weight parameter dimensionality rarely presents itself as 2-dimensional, and in practice the parameters require dimensionality reduction techniques to enable this form of visualization. As a result, the visualization itself can be misleading because of this abstraction. Yet, despite the problems of high-dimensionality visualization, the concept of loss landscape remains an important aspect to take into account when designing or analyzing objective functions and their optimizers.

2.7 Audio Processing

Audio is often defined as a form of sound that is limited within the acoustic range humans are biologically capable of hearing. Beyond that, audio is a signal. A signal can be understood as a quantity that changes over time. For audio, the quantity in question is air pressure. To store this information digitally, it is necessary to sample it. Sampling is the process of measuring a signal at discrete points in time. The most common sampling rate for audio is 44.1kHz, which means that 44,100 samples are taken per second. By digitizing audio in this way, it becomes possible to manipulate it in a variety of ways, including editing, processing, and transmitting it to other devices.

It is an essential aspect to many technologies.

Once an audio signal is sampled and digitized it becomes possible to visualize its audio wave (Figure 11). The audio for the example plots is the first 12 seconds of the song "Playing God", by Tim Henson[1]. While raw audio waves contain valuable information about the sound signal, most of this knowledge remains concealed in the frequency domain. One of the reasons for this is that audio waves are rich, complex signals that contain multiple frequency components that typically overlap and interfere with each other. As a result, extracting and analyzing the complex behavior of the audio signal typically requires additional techniques.

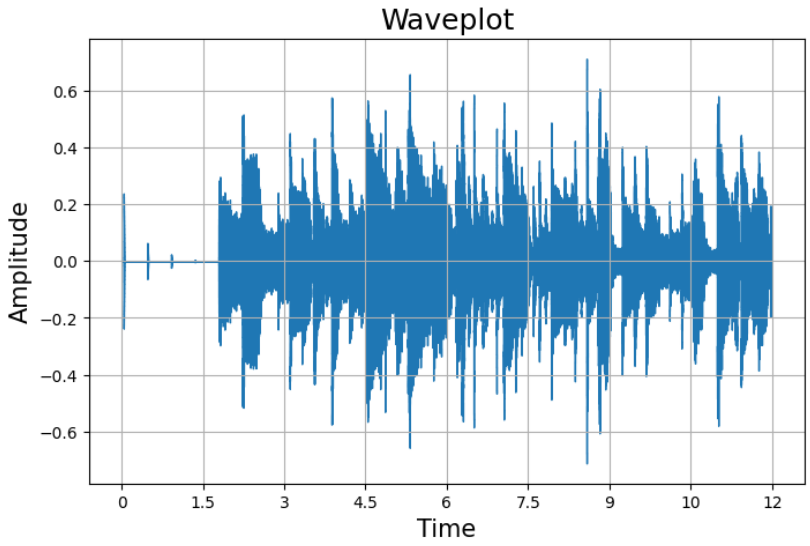

Figure 11 – Raw audio waveplot depicting the first 12 seconds of the song Playing God, by Tim Henson. This same excerpt is used for forthcoming example plots. Source: Author.

Unlike the time domain of a signal, which shows variations in the signal over time, the frequency domain decomposes a signal across its constituent frequencies and shows how much of each frequency component is present. This is desirable because most types of audio signals, such as speech, music, or environmental sounds, have distinct characteristics in terms of frequency patterns and structures.

Digital signal processing techniques are indispensable tools for analyzing and manipulating signals in many fields, such as communication systems, image processing, and audio processing. One of the most commonly used techniques in digital signal processing is the Fourier transform. It allows us to traverse between the time and frequency domains of a signal, which is essential for analyzing signals with complex frequency components.

The Fourier transform is a mathematical technique that converts a signal from the time domain to the frequency domain (Figure 12). It decomposes a signal into a sum of sine and cosine waves of different frequencies, each with its own amplitude and phase.

[1] https://youtu.be/DSBBEDAGOTc

By analyzing the frequency components of a signal, we can extract valuable information about its behavior and characteristics, such as its dominant frequency, harmonics, and noise. In audio processing, the Fourier transform is used for a wide range of tasks, such as speech recognition, music analysis, and noise reduction.

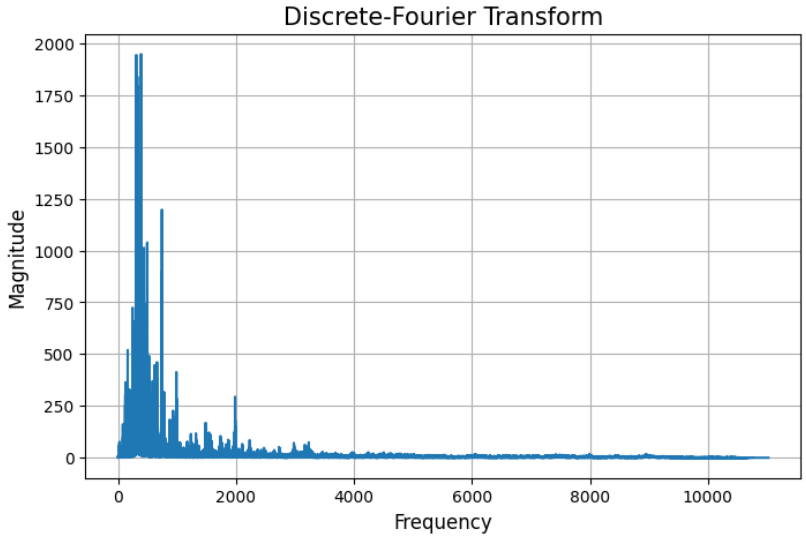

Figure 12 – Discrete-Fourier Transform depicting the frequency domain of the example song (time component information is lost). Notice how certain frequencies dominate the signal. Source: Author.

Despite being a powerful tool for signal analysis, applying the Fourier transform to the signal in its entirety may yield uninformative results, making it difficult to understand the properties of the signal. The main limitation of this approach is that it assumes the signal is stationary. This means it considers the frequency content of the signal to remain constant over time. Expectedly, real-world signals are commonly non-stationary, i.e. their frequency content changes over time. Spectrograms provide an alternative approach to deal with non-periodic signals, addressing the issue by breaking the signal into small segments before computing the Fourier transform for each part. Being so, this approach allows us to track how the frequency components in the signal change over time. Therefore spectrogram representations allow for the capture of frequency domain properties that would otherwise be missed should we apply the transform on the entire signal all at once.

Most frequencies contribute very little to the overall amplitude of the sound, as seen in Figure 13. For this reason, spectrograms commonly use the logarithmic scale to represent the frequency content of signals. Their values are usually expressed using decibels (dB), the same logarithmic scale used to measure the power of sound waves. In spectrograms, the intensity of the sound at each frequency and time point is commonly represented with different colors representing the intensity levels. Because their values are in the decibel scale, it becomes simple to compare the relative loudness of different parts in the signal (Figure 14).

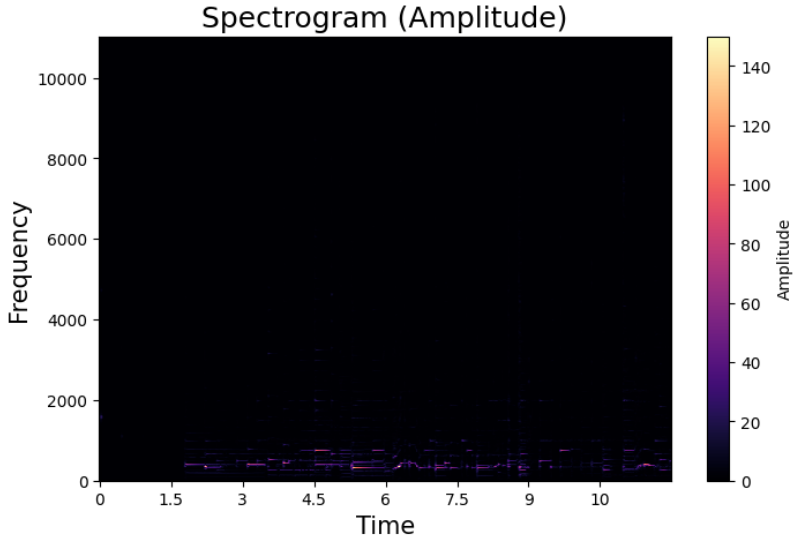

Figure 13 – Audio spectrogram using amplitude intensity values across the time-frequency axis of the example song. Information here is hard to visualize, as most frequencies do not contribute much to the overall amplitude of the song. Despite of this, observe how spectrograms depict information in 3 axis (time, frequency, and amplitude). Source: Author.

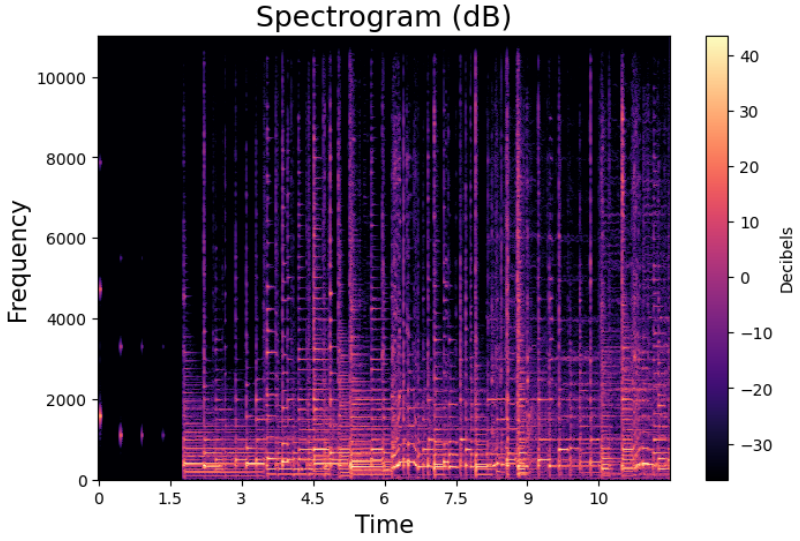

Figure 14 – Audio spectrogram of the example song using the decibel scale. Information using a logarithmic scale is easier to visualize in audio spectrograms. Source: Author.

In addition to using the logarithmic scale to present the amplitude of the frequency components (color axis), spectrograms typically also use a logarithmic scale on the frequency axis (y-axis) as well. They do so to represent signal information in a manner that is consistent with human auditory perception, for the relationship between the frequency of sounds and how we perceive their pitch is logarithmic as well. Consider a sound with a frequency of 100Hz being doubled to 200Hz: we perceive this difference in pitch as being the same as if we had doubled the frequency again to 400Hz. In fact, humans generally perceive an octave (a doubling in frequency) as being the same change in pitch, regardless of the starting frequency. Ultimately, this also implies we are better at detecting differences in lower frequencies than higher ones. It is trivial to tell the difference between 500Hz and 1,000Hz, but it can be hard to distinguish between 10,000Hz and 10,500Hz, despite their distance being the same. For this reason, spectrograms commonly use a specific logarithmic scale called the Mel scale. It is in fact a perceptual scale of pitches judged by its listeners to be equal in distance from one another. Historically, there have been several proposals to define a psychophysical pitch scale dating back to 1937. Since then, the curves depicting the conversion of f hertz to m mels have evolved to the now-popular version with the 700Hz corner frequency published in 1976 (MAKHOUL; COSELL, 1976), which can be seen in Equation 11.

$$m = 2595 \log_{10} \left(1 + \frac{f}{700}\right) \tag{11}$$

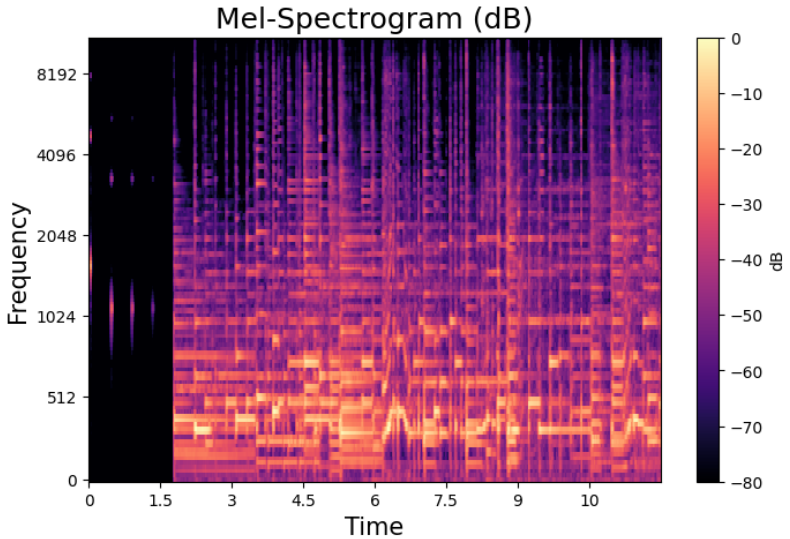

Figure 15 – Audio mel-spectrogram plot of the same example song. Notice how structural patterns in the signal are more evident in the mel scale. Source: Author.

The overall result of using the mel scale is a better visualization of the low-frequency components of the signal (Figure 15), which can be difficult to see on an otherwise linear scale (refer back to Figure 13). Another relevant reason to use such a scale on

the frequency axis is to compress the dynamic range of the signal. This type of scale compresses large values, making it easier to see small frequency changes throughout the signal. Notably, this is useful when working with signals with a wide dynamic range, including music or environmental sounds, while also offering a scale conformable to human auditory perception.

2.8 Multi-Domain Learning

Despite the significant amount of data available nowadays, current training paradigms are restricted in terms of the variety of data they can handle. Typically, models are trained and work with a single data source, usually from a narrow domain. Inevitably, models learn their structural patterns and become biased toward that particular domain, performing well only when working within it. This is a major limitation in terms of generalization when models are expected to perform well in multiple scenarios. Multi-domain learning is concerned with learning multiple domains simultaneously. This paradigm allows models to learn from a variety of domains without harming their ability to learn more nuanced features structurally inherent in each domain.

To better understand the significance of domains conceptually, it is useful to view them through the lens of a task. Mathematically, whenever models are being trained on a task, they are learning a mapping function from the domain (the data) and the image (model output), visualized in Figure 16. Even though we commonly refer to the task in a more abstract manner (e.g. animal classification using images), in reality, the task being learned is much more strict. The learned task could potentially be "differentiating very specific animals using photos taken using a DSLR camera with a particular sensor during an exact time of day with determined weather conditions". In fact, the learned task is very specific to the input data, and much expectation is placed on the ability of models to generalize *ad infinitum*. This often creates a dissonance between the task machine learning specialists are trying to solve and the task the model is being trained on. Not rarely do models fail to generalize to the data distribution in the actually intended task, a very literal instance of solving the wrong problem.

Domain differences lead to errors in a number of ways (BEN-DAVID et al., 2006, 2010). Domain-specific distributions often differ in favoring different features. As such, some features may only appear in one domain. Additionally, features may behave distinctly regarding the label distribution in each domain.

Even though the present work addresses audio data, examples using image data are useful to understand how domain differences manifest. One of the widely used datasets in multi-domain research for images is the Office-31 (XU et al., 2021; NA et al., 2021; KANG et al., 2019; XU et al., 2019). The Office-31 dataset is suitable for multi-domain learning studies since it has different domains but maintains the same classes

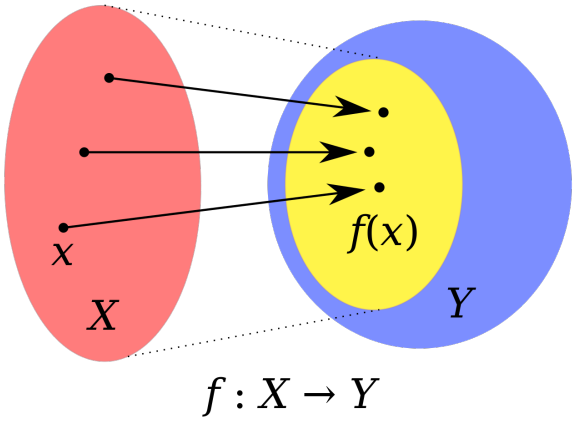

Figure 16 – Evaluating a function f at each element of a subset X of its domain produces the image set of X. Here, f is a function from domain X to codomain Y. The yellow subset of Y is the image of f. Source: Nguyen (2008).

across domains (this is an important characteristic and will be further elaborated in Chapter 4). Its three domains are: Amazon, Webcam, and DSLR (SAENKO et al., 2010).

Figure 17 – Examples from the Office-31 dataset. Each line presents examples of the classes Bike, Headphone, and Scissors for a domain. The domains are Amazon, DSLR, and Webcam, from top to bottom. Source: Bender (2022).

The images in the Amazon domain are provided by an online sales store, therefore all images have a white background where their objects are in a unified color scale. The Webcam domain has low-resolution images (640x480) with significant noise. Finally, the DSLR domain is composed of high-resolution images with low noise (4288×2848). Figure 17 demonstrates examples of images from the bike, headphone, and scissor classes.

Domain differences in audio are more subtle and are very difficult for humans to

perceive in spectrograms. Figure 18 depicts a comparison between the same song from Figure 15 and an artificially mixed version of it, where city ambiance noises were introduced to simulate a noisy domain. Despite being the exact same song, the spectrograms are only vaguely similar on first inspection. In fact, without the information about what exactly is contained in each audio clip, it would not be trivial to point out any potential domain differences just by looking at the spectrograms (contrasting to Figure 17).

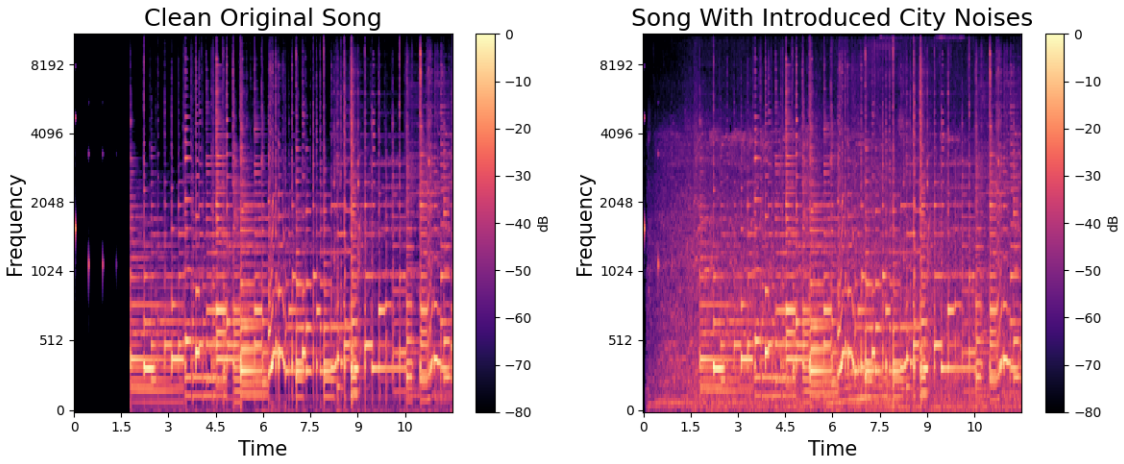

Figure 18 – Audio mel-spectrograms of the original song example and an alternative version of it, mixed with city background noises. Manually investigating domain differences and their properties via mel-spectrogram visualization is not a reasonable task for humans. Source: Author.

3 RELATED WORKS

Within machine learning research areas there are several subfields dedicated to finding and analyzing the best way to train a model in multi-domains at the same time. Among these, domain generalization aims to make a model perform well when using as a test a domain that was not used during its training (GULRAJANI; LOPEZ-PAZ, 2020). Another task, called multi-domain learning seeks to find the best way to train a model so that it performs adequately in all domains used during training (LIU et al., 2019).

During the last few years there has been a growing research interest that addresses the training of models with multi-domains, with research focusing on domain generalization (GULRAJANI; LOPEZ-PAZ, 2020; ARPIT et al., 2021; LAPARRA; BETHARD; MILLER, 2020; XIE et al., 2018; LI et al., 2018, 2017) but few studies have been developed in multi-domain learning. Although many of the single-model multi-domain learning contributions end up proposing domain-specific architectural changes (SICILIA et al., 2021) and although these solutions theoretically consist of the use of a single model, it is still necessary to create a different architecture for different datasets or if new domains are added to the original, making it difficult to scale and adding complexity. Typically, multi-domains are manipulated by creating multiple branches within the neural network, one for each domain to be learned, which shares the initial part of the network as the feature extractor and domain decider.

Nam; Han (2016) proposed MDNet, referred to as Multi-Domain Network. Their approach separates domain-independent information from domain-specific one and learns generic feature representations for video-tracking. To enable this, each domain in MDNet is trained individually while the shared layers of the network are updated in every iteration. In their study, each video sequence in their task of visual tracking is referred to as a domain. Therefore, the proposed MDCNN requires retraining.

In Chen et al. (2018), Chen advocated for BAMDCNN, a Branch-Activated Multi-Domain Convolutional Neural Network for the task of visual tracking. In addition to the main convolutional layers of the CNN, the network has additional branch layers, each specializing in handling a particular group. They extract key frames from the

sequence dataset and group them using a clustering algorithm. During inference, they compare the similarity of the initial frame of test sequences across known groups to identify which inputs should be processed by which branches. As such, they achieve substantial effectiveness when compared to various other state-of-the-art methods for visual tracking.

Liu et al. (2019) argued redundant common features often exist in the intersection of multiple domains, and models frequently learn those features as it is a simple and efficient way to address tasks. However, the existence of this redundancy in the network implies the model does not make full use of features and their learning spaces. In practice, this reduces the discriminability of the features and therefore increases the difficulty of the task. As such, the work addressed the undesirable mixture of features from different classes across domains by proposing an end-to-end network and orthogonality regularizations to separate domain-specific and domain-invariant features. The learning of what they refer to as compact-features (domain-specific features) significantly improves general classification performance.

One of the main contributions over the last recent years of learning multi-domains with the same model comes from speech recognition. SpeechStew (CHAN et al., 2021) performs state-of-the-art speech recognition tasks just by mixing all the data and training the same model using different domains, such as the Stew method we evaluate in this study. Other previous studies trained multi-domain models by mixing all available data (CHOJNACKA et al., 2021; NARAYANAN et al., 2018; LIKHOMANENKO et al., 2020), the main difference being that SpeechStew scales to larger models.

Batch-level domain regularizations were previously evaluated in the context of image classification by Bender (2022). They obtain competitive performance using the Loss Sum approach, where the loss is calculated individually for domains and summed before backpropagation.

Notably, Tetteh et al. (2021) uses multi-domain balanced batch sampling techniques to address X-ray pathology classification tasks in the biomedical domain. They denote performance gains using a balanced batch sampling technique which is analogous to Loss Sum, previously proposed by Bender (2022).

While most studies in multi-domain learning propose architectural changes in models, we propose batch-level regularizations to guarantee appropriate domain representation in examples during the training of models. In fact, this is an architecture-agnostic approach and can be utilized without major alterations in the classical training loop of machine learning models.

4 METHODOLOGY

This Chapter provides a detailed description of the procedures and techniques employed to conduct the study. It describes the systematic approach and methods used to address the research questions and objectives of the study. Additionally, we describe the experimental setup configuration.

4.1 Datasets

In order to evaluate the proposed multi-domain learning training methods, we need datasets that contain explicit domain characteristics. Additionally, the examples must have annotations depicting the domain they are a part of. Furthermore, we are interested in datasets containing an additional, distinct feature to use as the target of classification tasks. It is important to avoid direct relationships between the class target and the domain, as such interactions would hinder the evaluation of domain regularizations by confusing them with class regularization. In fact, when the domain has a direct relationship with the class label, figuring out the domain of an example is often reducible to discovering its class; being at least as complicated as correctly classifying samples (i.e. solving domain classification would imply solving target classification). Ultimately this means developers in this scenario do not have the domain information annotated or easily attainable. For this study, we select three datasets with these characteristics to perform the experiments: DAPS (Device and Produced Speech) containing book excerpt readings, and two bird call recording datasets, FF1010BIRD and WARBLRB10K.

4.1.1 Speaker Identification — DAPS

One of the audio datasets is called Device and Produced Speech (DAPS) (MYSORE, 2014) and contains speech segments of 20 different readers (10 male and 10 female readers) in various recording device types and environmental conditions (15 different domains). The recording process can be seen in Figure 19. Each speaker read 5 public domain book excerpts under different conditions (about 14 minutes of duration per speaker). In its entirety, the dataset consists of about 4 1/2 hours of audio recor-

dings. DAPS was initially used as a speech recognition dataset, but we decide to use it for the task of speaker recognition, classifying the 20 different speakers. We focus on classification problems in this study because, typically, they are more straightforward, thus reliable alternatives to testing new multi-domain train paradigms.

It is expected to encounter domain shift regarding the difference in data recording conditions, i.e. audio clips recorded using an iPhone in a conference room will likely differ in characteristics from those recorded by an iPad in a balcony prone to street noise. Despite noise being a more intuitive cause of domain shift, the differences in recording devices and room acoustic conditions likely also play an important role.

Each domain is split into train and test folds. We do not use a validation fold as we are not optimizing hyperparameters or performing optimization tasks in the model configuration. How each domain is split is important and requires attention to a few details. Note this is a classification task with 20 different classes (the speakers), thus it is important to guarantee a balanced representation of these classes in training and test sets. We use class stratification to address this issue, while also guaranteeing a somewhat even distribution of text scripts and speaker gender.

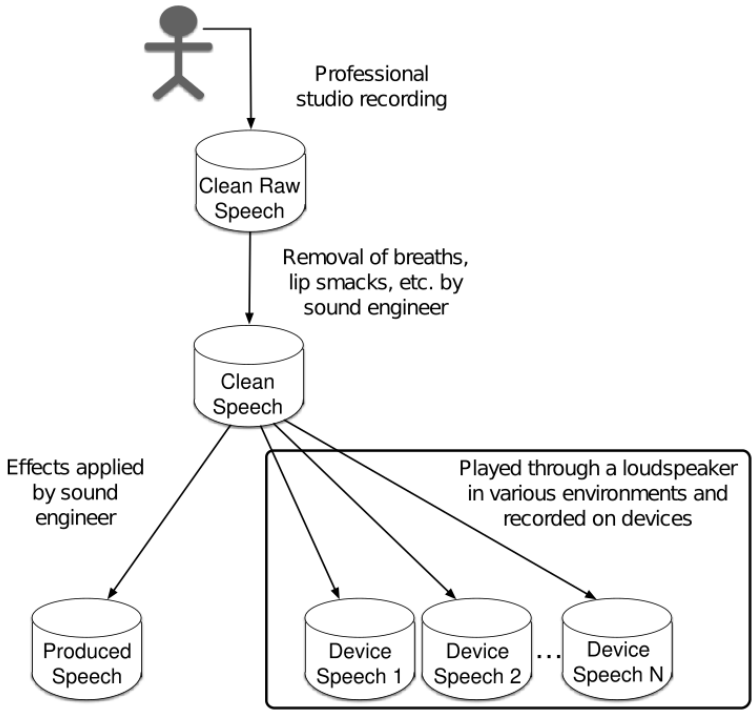

Figure 19 – DAPS dataset domain creation process. Source: Mysore (2014).

Each audio clip is processed to handle trailing silence at the beginning and the end, as some speakers take significant time before they start talking. The former pre-processing is relevant as the clip is then split into 5-second segments, further avoiding examples without speech. Audio clips are then converted from waveform to mel-spectrograms. This representation visually represents the signal amplitude across different frequencies over time. Ultimately, spectrograms can be understood as the ap-

plication of Fourier transforms on overlapping windowed segments of the signal. The mel scale is a unit of pitch to approximate the human perceived frequencies. The use of mel-spectrograms is common in audio processing because humans do not perceive frequencies on a linear scale.

However, we need to divide training and test sets before splitting the audio into 5-second recordings and then use that same fold scheme for the other domains. This is relevant because clips have a different duration from their counterparts in other sources, and would otherwise be unaligned across domains. Performing the train-test separation after splitting the audio tracks into 5-second segments would enable unaligned segments containing the same reader and script content but in distinct domains to be included in train and test sets. Some domain instances in audio are incredibly similar. Consider the signal differences of the same sentence being uttered in a conference room and living room. Detecting said differences could be hard even for a human listener. Therefore having the same (or very similar) audio content included in both train and test sets can cause data leakage leading to an over-optimistic result, even if said data originates from different domains.

4.1.2 Bird Detection — FF1010BIRD and WARBLRB10K

Freefield (FF1010BIRD) (STOWELL; PLUMBLEY, 2013) and Warblr (WARBLRB10K) are both bird detection datasets, but they do not have any domain semantics attributed to example classes. For this reason, we use them together, each behaving as a domain. Despite both being bird presence detection datasets, they are very different. In fact, Freefield is a dataset of professional recordings of on-site observations of birds (collected from the FreeSound online database[1]). It is very diverse in terms of location and environment. Expectedly, they use better recording equipment and usually there is not much background noise. Additionally, it has some label imbalance towards the negative class, supposedly because once the equipment is set on-site, it remains recording audio most of the time.

In contrast to FF1010BIRD, WARBLRB10K contains crowdsourced recordings of birds using the bird-watching smartphone app Warblr[2]. Its label imbalance is towards the positive class, as most users use their devices to record bird calls when in the presence of said birds. This dataset, however, has heavy background noise, including city sounds and even users imitating bird calls, allegedly to coax birds to answer. The recordings vary heavily in terms of audio quality, depending on the smartphone used.

The significant difference between the elected bird datasets is by design and desirable for this study, as domains too similar in nature would entail a difficult multi-domain analysis. The FF1010BIRD dataset contains only 25% bird presence, while the Warblr

[1] https://freesound.org/
[2] https://www.warblr.co.uk/

dataset contains 75% bird presence.

Table 1 – Audio Experiment Results Across Domains

Dataset	Not Bird	Bird	Total
FF1010BIRD	5755	1935	7690
WARBLRB10K	1951	6045	7996

4.2 Evaluated Methods

Reiterating, our hypothesis is that, during the training of machine learning models on multi-domain tasks, the training process takes advantage of explicitly using this data and its domain. In order to evaluate it, we evaluate three methods for training models in multi-domain tasks, including the traditional method that does not explicitly considers the different domains. This section describes Stew, Balanced Domains, and Loss Sum.

4.2.1 Stew

The more intuitive approach to using data from multiple sources at the same time is the Stew method (named due to the SpeechStew method (CHAN et al., 2021)). The method consists of simply mixing data from multi-domains together homogeneously, without any special processing or distinction. This method is already in use for various multi-domain tasks in areas such as speech recognition.

To compose large datasets, it is common to use different sources containing the same data classes, so that the data come from different domains. Commonly these domains are not explicit, which makes the Stew method the only possible option without the need to perform complex analyses to infer domains. In this way, the Stew method is inherently present in most models trained using such datasets. Regardless of its simplicity, the Stew approach yields competitive results in multi-domain learning tasks, in particular whenever there are large amounts of data available.

Whenever a dataset has numerous well-represented domains, it is speculated to encourage the model toward domain-agnostic representations. Even if generic representations are desirable, the datasets containing the information necessary for a model to be capable of achieving such knowledge are rare. Not only is the creation of datasets a challenging endeavor, but the verification of it is often neglected. ImageNet has been the standard pre-train dataset for image-related tasks for the past years, and researchers frequently stumble on annotation errors and report them even at present.

4.2.2 Balanced Domains

Historically, there have been significant improvements in image classification using simple data processing methods and regularizations, such as dealing with label imba-

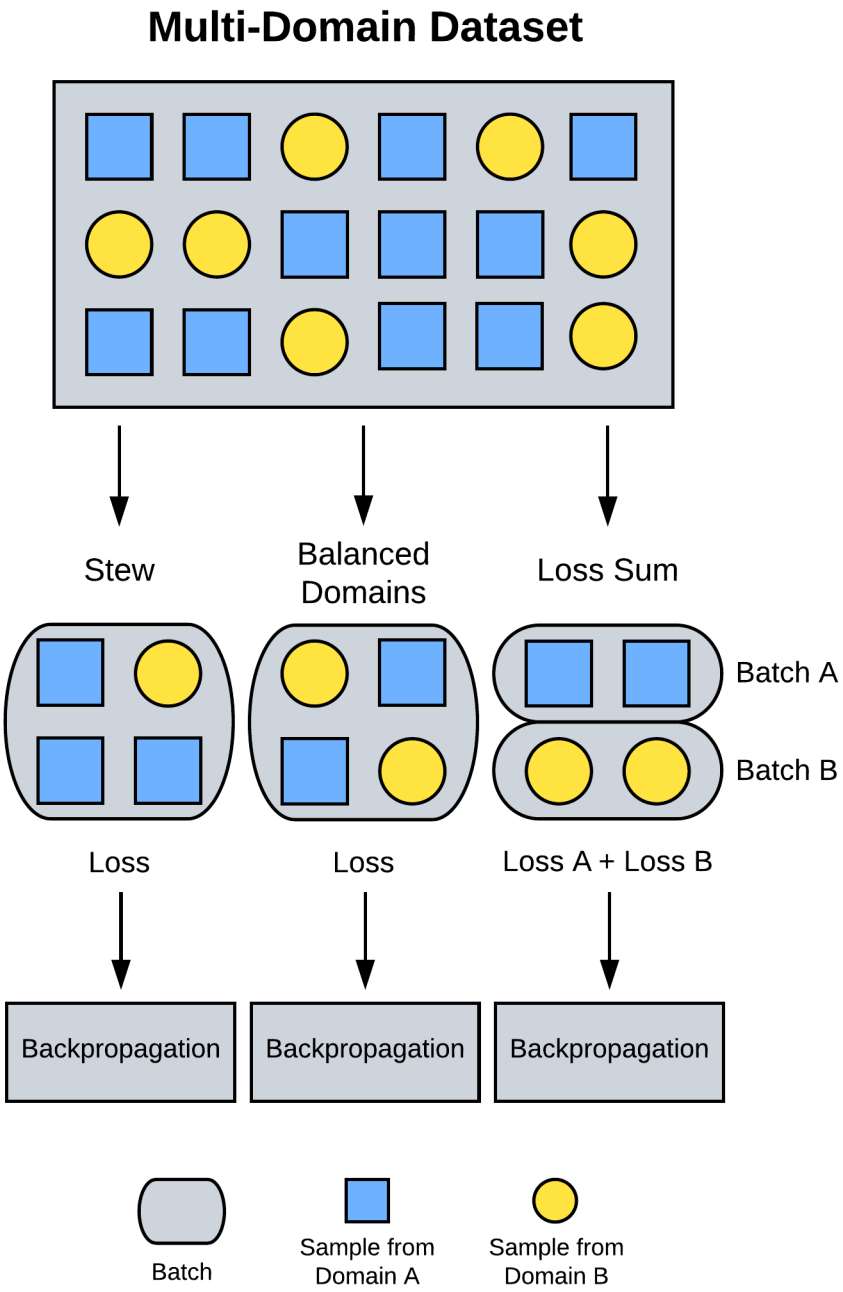

Figure 20 – Evaluated methods illustrated. On the left the Stew method is shown, sampling randomly from the mixed datasets without any kind of balancing, usually having batches where the major domain has more examples. The middle flow describes Balanced Domains, where we also sample from the mixed domains dataset, but strive for similar amounts of each domain. On the right, Loss Sum flow is presented, where we sample from the domains individually and calculate a different loss for each domain batch, then sum them up to achieve the final loss. Notice that Loss Sum has smaller batches for each domain, but the number of samples in all batches amounts to the same batch size in previous methods. Source: Author.

lance. During training, the Stew approach is understood to have no balance of domains whatsoever. Essentially, the batches are expected to have more samples from the majority domains, since batches are randomly sampled from the mixed dataset. This can be visualized in Figure 20. For this reason, there is room to explore regularization to address domain-level selection in training batches.

In classification tasks, the presence of a class imbalance in batches might hinder the model generalization, biasing it towards classes with more examples during the training process. Label balancing addresses this problem and can be especially useful in cases where there are few, highly unbalanced classes.

Similarly, we hypothesize a disproportionate domain presentation on a batch level might also interfere with the model learning. In this case, the model would potentially specialize in the majority domain. Intending to perform well in all presented domains, we propose a variation of the Stew method called Balanced Domains.

To compose the batches seen in model training, instead of sampling from a unique dataset, Balanced Domains samples from each of the available domains separately (as seen in Figure 20). The domain sample size is set to accommodate an equal (or close to equal) composition of each domain while maintaining the original batch size unaffected.

This method is envisioned as a batch-sampling abstraction to enforce each domain to have approximately an equal number of examples in a batch. Thus the training step consists of sampling all obtainable domains, grouping the data into a single batch before presenting it to a model, and backpropagating the calculated loss from the batch.

The number of batches in an epoch is bounded by the largest domain. Whenever the domains differ in their number of samples, the smaller domain entries will be shown multiple times throughout a single epoch. Admittedly, this entails the same implications associated with oversampling and therefore requires the same cautionary practices.

4.2.3 Loss Sum

One way of penalizing the model whenever it underperforms on a training domain is achieved by balancing the domain representation. Another way to implement this is, instead of mixing domain samples into a single balanced batch, to calculate the Loss from each domain separately and sum the Loss across all domains before backpropagating it (as seen in Figure 20). Notably, this sort of approach was previously proposed by (BENDER, 2022) and (TETTEH et al., 2021) for image classification.

The Balanced Domains method regulates domains at a batch level, but this technique does not guarantee that the model will properly learn all domains since it might still prioritize domains with similar features. For instance, in cases where there are several domains, the difference in Loss of a small portion of them might not be enough to pose a difference. The intuition behind the Loss Sum approach is punishing the model

with greater overall Loss values whenever it yields bad results in any domain. In this situation, penalizing the model by calculating the Losses individually and then adding them will lead to a higher Loss value.

It is relevant to note that although each domain is presented separately during model training, the total batch size between the three methods (Stew, Balanced Domains, and Loss Sum) remains unchanged. Loss Sum requires an additional call to the loss function for every domain, therefore it requires more training steps. Smaller batch sizes will also reduce the parallelization achieved by the GPU during training, effectively making it slower.

4.3 Counterfactual and Comparison Methods

This section describes additional methods, mainly counterfactual methods, useful to derive properties when comparing their results to the main methods described in the previous section.

4.3.1 Random Sum

Admittedly, the Loss Sum method operates on a different scale than other regular methods, and this is due to the fact it sums up the loss of multiple domains. Previous studies in image classification have suggested an increase in F1-Score when training neural networks using the Loss Sum approach. However, it is unclear whether this is due to the separate loss calculation and sum operation or due to simply having a higher loss value. For this reason, we devise a counterfactual method that shares the same scale (higher loss) as the Loss Sum method but does not apply the loss function to domains properly separated, but does so in mini-batches containing examples sampled randomly from the entire dataset. Thus, we refer to this method as Random Sum.

The idea behind this counterfactual experimental method is to understand whether the improved performance previously seen while using Loss Sum method is attributed to its higher loss values or some other mechanism. Should a method with a similar loss scale but trained without domain separation (e.g. Random Sum) performs similarly to the Loss Sum method, we may attribute its overall better performance to higher loss values. The reasoning for this potential outcome is related to how a higher loss entails more abrupt weight updates, which may or may not be appropriate for a given task. Alternatively, Random Sum failing to achieve competing performance with the Loss Sum (despite sharing its loss scale) would be evidence supporting the individual domain loss calculation mechanism.

4.3.2 Loss Mean

The Loss Mean method is another counterfactual method, complementing the Random Sum. It is also missing from previous studies with similar batch regularization proposals. In this approach, we perform the same procedure described for the Loss Sum process, but we divide it by the total of domains present in the task before back-propagating the loss. By doing this, we force the loss scale back to being comparable to other regular methods.

Depending on whether this experiment shows results similar to the Loss Sum method, we may collect further evidence for the separated loss calculation improving multi-domain learning tasks. Analogally, the hypothetical scenario where Loss Mean underperforms in comparison to Loss Sum would imply evidence for the higher loss scale being useful for the task (instead of the individual domain loss system).

4.3.3 Sequential And Inverse Sequential

Additionally, we configure two other training methods for comparison purposes. The Sequential training method is defined as training on individual domains in sequence. Conversely, the Inverse Sequential approach does the same, but in the reverse order of domains. Both of these approaches are prone to catastrophic forgetting (where the model forgets previous knowledge, replacing it with new information). Neither of these is expected to show competitive results with the other methods. However, we argue their results offer an interesting perspective on the learning tasks. As such, they enrich the comparison by showing the pitfalls of naive methods — demonstrating catastrophic forgetting in action is not, necessarily, beyond the scope of this work. Another argument to be made is that these methods behaving as one would expect conveys evidence of the correct implementation of the experiments.

4.4 Comparison Metrics

The baseline for our comparison is the Stew training method, as it is commonly used and *de facto* standard in the literature. The performance of models in an experiment for each method (Stew, Balanced Domains, Loss Sum, Random Sum, and Loss Mean) is calculated using the average F1-Score across domains. The F1-Score was chosen because it summarizes the learning objective: learning all domains at the same time while generalizing the classes. It does so by calculating the harmonic average between precision and recall (Equation 12). The recall metric (Equation 14) denotes how many positive samples were identified from to the actual positive example amount (i.e. if we understand our capacity of identifying positive samples as a fishing net, it is the ratio of how many fish we manage to catch from the total fish present in the pond). Whereas the precision metric (Equation 13) depicts how many positive instances were correctly

classified (i.e. using the same previous analogy, it is the ratio of how many of the things caught in our net are actually fish). In the equations below, TP refers to true positive examples, FP refers to false positive examples, TN refers to negative examples, and FN refers to false negative examples. The positive and negative nomenclature alludes to the class value, while the true and false denotes whether the prediction is correct or not. For example, a false negative prediction is an example incorrectly classified as the negative class. In reality, it ought to pertain to the positive class.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{12}$$

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

When calculating F1-Score, we are presented with a choice of whether we use the macro F1-Score, the weighted F1-Score, or the micro F1-Score. The macro version calculates the F1-Score for each individual class, then returns their average value. It is an interesting alternative when the classes themselves are the object of investigation. When the classes are not balanced, that is, some classes have more representation in a given task, the weighted F1-Score provides an alternative by calculating the F1-Score for each class, and returning the weighted average of the result. The weights are typically calculated using the inverse of the example frequency from a given class. Finally, the micro F1-Score sums up the TP, FP, TN, and FN for all classes and calculates a single, unified F1-Score for the model. This approach abstracts the concept of classes and focuses solely on how the model performs.

Because the object of focus for this study is the model itself and not the particular classes, we chose the micro F1-Score calculation. We argue choosing either macro or weighted F1-Scores would, in a way, hide aspects of the model performance behind averages and class weights. This is an unnecessary layer of abstraction that would, in fact, difficult model evaluation. This is particularly noticeable in cases where each domain has a different class distribution. Thus, considering the objective of evaluation in this study, projecting raw scores is the preferable alternative.

4.5 Evaluating Multi-Domain Learning Models

According to Gulrajani; Lopez-paz (2020), there are two major approaches to multi-domain model evaluation that are rarely discerned and seldom discussed.

In the Leave-One-Domain-Out-Cross-Validation approach, one model is trained for every domain: each holding one of the training domains out. The withheld domain

is then used to evaluate its corresponding model. The average of the score metrics across folds is then reported. Expectedly, domain characteristics can greatly impact this evaluation. When domains are similar enough, implicit data leakage can occur. Alternatively, in the scenario where the domains show significant distinct characteristics, covariate domain shift could be an obstacle. For this reason, we refrain from using it to evaluate the datasets.

In the Training-Domain Validation Set approach, each domain is split into training and testing subsets. The resulting partitions are pooled to create multi-domain train and test folds. The model is evaluated using the resulting overall test fold. This strategy assumes a certain similarity between training and testing example distributions. Overall, it is a conservative approach whenever prior knowledge of domain characteristics is limited. Thus, this approach is more appropriate for the current study.

Moreover, we purposely avoid using k-fold cross-validation for similar reasons. When working with multi-domain datasets, there are several relevant distribution characteristics we are interested in controlling cautiously. Take the DAPS dataset, for example: besides guaranteeing a similar distribution between train and test sets for the speaker id (which is the target feature), the speaker gender is also an important feature to control. Furthermore, the book excerpt is another dimension we are interested in controlling. And because we are interested in a more detailed task setup distribution-wise, the randomness from k-fold cross-validation could actually prove harmful for our evaluation. It is useful to remember the scope of this study is to evaluate the training methods and not to solve the classification tasks used for the evaluation.

4.6 Experimental Setup

We perform several experiments using the methods and datasets described previously. The current Section details their configuration and the reasoning behind them. Their results, however, are reserved for a dedicated chapter and can be inspected in Chapter 5.

4.6.1 Differentiating Audio Domains Experiment

We are interested in analyzing the behaviors of the proposed methods in different situations regarding domain and class distributions. Before we do so, one important question to address is: how hard is it to distinguish audio domains? We have reviewed this from the human point of view in Section 2.8. Whereas for images it is trivial for humans to distinguish different domains, for audio data it is unexpectedly hard to do so, either by hearing the audio clips (e.g. differentiating a recording in a conference room from the same recording in an office can be challenging), or by looking at the spectrogram. But more importantly, how hard is this task for computers to accomplish?

If, similarly to humans, they prove ineffective for differentiating domains, then there would be no real benefit in using the training methods described in Section 4.2. In fact, in this scenario the domain similarity would characterize a duplication of examples which could cause data leakage or just harm the learning process overall if not treated properly. Conversely, computers being good at distinguishing audio domains would be further evidence of the potential usefulness of these multi-domain training approaches.

To achieve an estimate of how effective computers are at addressing this task, we perform an experiment in which the task is domain classification in each experiment dataset configuration. For example, DAPS dataset samples will be classified as Ipad_Balcony, IPhone_Confroom, or any other of the domains. The merged bird dataset samples would be classified as either FF1010BIRD or WARBLRB10K. Note the actual class these samples pertain to (presence or absence of bird calls) is irrelevant to this task. In this experiment, we train a machine-learning classification model using a ResNet-34[3] and the configuration described below. Its result can be inspected in Chapter 5.

4.6.2 Dataset Distribution Manipulation Experiments

We artificially partition the datasets to perform 5 major experiments (Table 2). The idea behind these experiments is similar in nature to the idea of ablation studies, a type of experimental analysis performed to understand the importance of specific factors, where researchers systematically modify or remove components to infer their impact on the model performance. The goal is to gain insight into the behavior of each method when used in different situations.

Table 2 – Experiment Description Summary

Experiment Number	Task	Datasets	Domains
Experiment 1	Bird Detection	WARBLRB10K, FF1010BIRD	2
Experiment 2	Bird Detection	WARBLRB10K, FF1010BIRD	2
Experiment 3	Bird Detection	WARBLRB10K, FF1010BIRD	2
Experiment 4	Bird Detection	WARBLRB10K, FF1010BIRD	2
Experiment 5	Speaker Identification	DAPS	14

Experiments 1 to 4 use the bird detection datasets (WARBLRB10K and FF1010BIRD), while experiment 5 uses the speech dataset (DAPS). We choose the bird detection dataset to perform dataset manipulations, as it is rich in terms of class distribution differences between the domains. Additionally, it is easier and cheaper to train timewise.

The experiments use the ResNet-18 architecture[4], pre-trained with imageNet and

[3] https://pytorch.org/vision/main/models/generated/torchvision.models.resnet34.html
[4] https://pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html

fine-tuned using the DAPS or bird datasets (depending on the experiment). The choice of this particular network architecture is because it is relatively simple (the simplicity here is useful when comparing the different training methods), it is well established in the literature, and the fact it is a convolution neural network suitable for the use with audio spectrograms. We use 10 epochs with a batch size of 256 and an 80/20 split for train/test. We maintain configurations across experiments whenever possible. This includes the deep learning neural network and hyperparameters. We do so because our focus is to evaluate multi-domain training methodological approaches and not hyperparameter tuning.

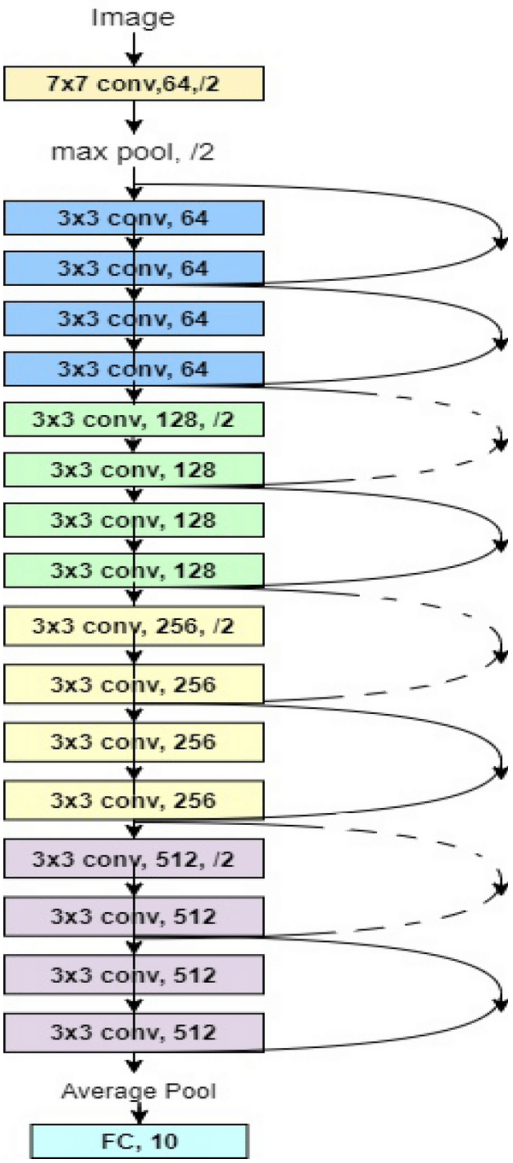

Figure 21 – Architecture Diagram of ResNet-18. Source: He et al. (2016).

Each experiment is repeatedly performed with 30 repetitions with different seeds. The seed values influence the model weight parameter initialization values. In other words, they are responsible for the model configuration starting position in the loss landscape during the optimization process. It is important to guarantee this propri-

ety when comparing the different training methods. Otherwise, some model instances could potentially start at more beneficial locations in the loss landscape, thus complicating the method comparison.

Another relevant propriety to consider is the order examples are shown to the model. The loss landscape traversal is greatly influenced by the order in which a model processes examples, and having it vary across learning methods will hinder a desirable fair evaluation.

- **Experiment 1 — Bird Detection, Original Dataset Size.** In this experiment, we use the proposed methods to train a model on the task of bird detection using the bird dataset (WARBLRB10K and FF1010BIRD) in their entirety. In this scenario, the datasets are similar in terms of example amount. We expect selecting examples stochastically domain-wise would not greatly affect the model, as each dataset would have a similar representation in the training dataset, in terms of example amount. Thus, this experiment evaluates the methods when domains are similar in terms of quantity, although being different in composition.

- **Experiment 2 — Bird Detection, Reduced FF1010BIRD.** We alter the configuration of the domains by artificially reducing one of them, using stratification to maintain the class distribution in each domain. For experiment 2, we randomly remove examples from the FF1010BIRD domain. In practice, this means domain WARBLRB10K is more influential during the training of the model when using a vanilla training approach. Expectedly, potential benefits from balancing domain presence in batches would appear in this scenario.

- **Experiment 3 — Bird Detection, Reduced WARBLRB10K.** This experiment is the natural counterpart of experiment 2. The reduced domain is now WARBLRB10K, while FF1010BIRD remains in its original characteristics. This is useful to review how the model behaves when most of the data comes from the ff distribution. Again, it is expected the regularization of the domains would be notable in this scenario should it improve model learning and generalization. In this experiment, we discard 1/3 of the WARBLRB10K domain, amounting to 5,598 removed audio clips. Only 2,400 examples remain.

- **Experiment 4 — Bird Detection, Reduced Symmetric.** Experiment 4 greatly reduces both domains to 300 samples, also using stratification to maintain the class distribution in each domain. The objective of this experiment is to evaluate the training methods when few examples are available. Despite the small number of examples, similarily to experiment 1, both domains are in equal amounts.

- **Experiment 5 — Speaker Identification, Original Dataset Size.** In this experiment, we use the DAPS dataset, which already has several domains defined.

There are 15 domains in the original dataset. However, we decide to remove the domain clean_raw from this experiment, as it is notably distant from other domains (it is the only domain with breath sounds, lip smacks, and other speech noises). More importantly, it would differ very little from the clean_speech domain and would cause complications by having duplicated examples in the dataset (this is easy to happen when we split the original audio clips into 5-second segments). Despite the domain differences regarding acoustic conditions and recording devices, they are balanced in terms of class distribution.

5 RESULTS

This section presents and discusses the experimental results of the audio classification tasks. We start by estimating how difficult it is for machine learning models to distinguish audio domains, then proceed by presenting the experimental results of dataset manipulations.

5.1 Differentiating Audio Domains Experiment

When using the DAPS dataset for domain classification, we achieve a 0.98 accuracy score in the validation set using a ResNet34. A brief report of the training can be viewed in Table 3. Considering how hard this task is for humans and the significant amount of classes in this task (a total of 15 domains), the first rational response to this result is to suspect the model is overfitting the validation set. But when the model was used to make predictions on the test set, the accuracy score was maintained, achieving 0.99. Figure 22 denotes the confusion matrix for the test set classifications. We can further inspect a batch of model predictions in Figure 23.

Table 3 – DAPS Domain Classification Report

Epoch	Train Loss	Validation Loss	Accuracy
0	0.910	0.534	0.801
1	0.394	0.291	0.895
2	0.219	0.146	0.949
3	0.131	0.099	0.969
4	0.106	0.094	0.971
5	0.071	0.060	0.980
6	0.040	0.054	0.981
7	0.042	0.053	0.983
8	0.026	0.049	0.983
9	0.026	0.049	0.983

When using the bird detection datasets WARBLRB10K and FF1010BIRD for domain classification, we achieve similar results of up to 0.959 validation accuracy score (the report can be seen in Table 4). It is interesting to note the first accuracy score va-

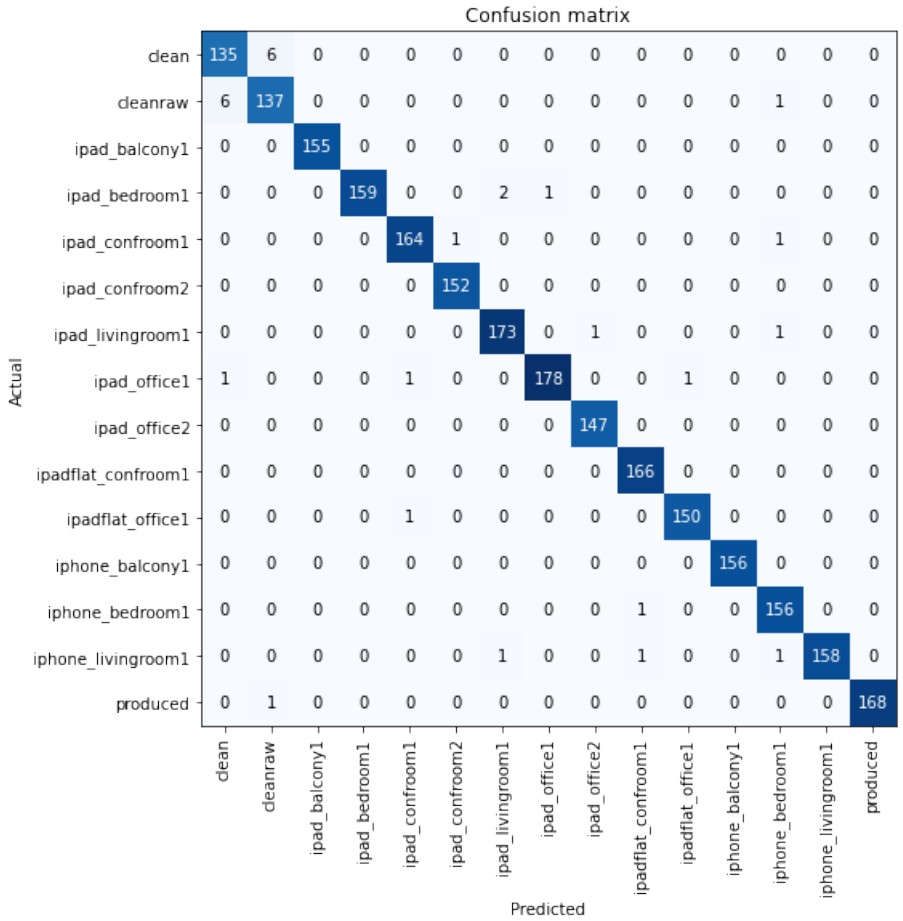

Figure 22 – Confusion matrix of domain classification using the DAPS dataset. Source: Author.

lues from the bird domain classification (binary classification problem using 2 domains) are dramatically worse than the DAPS domain classification (multiclass classification problem with 15 domains). At first sight, this is not an intuitive result, but it can be attributed to the significantly different amount of domain examples forming the bird dataset. WARBLRB10K forms approximately 48% of examples, while FF1010BIRD comprises 52% of them. Hence we can explain this lower initial score by the model initially guessing all examples as WARBLRB10K or as FF1010BIRD during the initial epochs. This behavior is further intensified by the bird dataset having only two classes. The test accuracy results also maintained the values seen before in validation, with the FF1010BIRD domain achieving 94.3%, and the WARBLRB10K domain achieving up to 97.2% accuracy. An example of predictions can be seen in Figure 24.

During the experiment, we still suspected data leakage could be occurring. As such, we also experimented with hiding the filenames from the image data (which actually contained the example domain), but the results remained unchanged. Finally, we performed a SHAP value analysis for image data that yielded no evidence of data leakage. It can be reviewed in Appendix A.

Considering the use of machine learning to perform domain classification, based

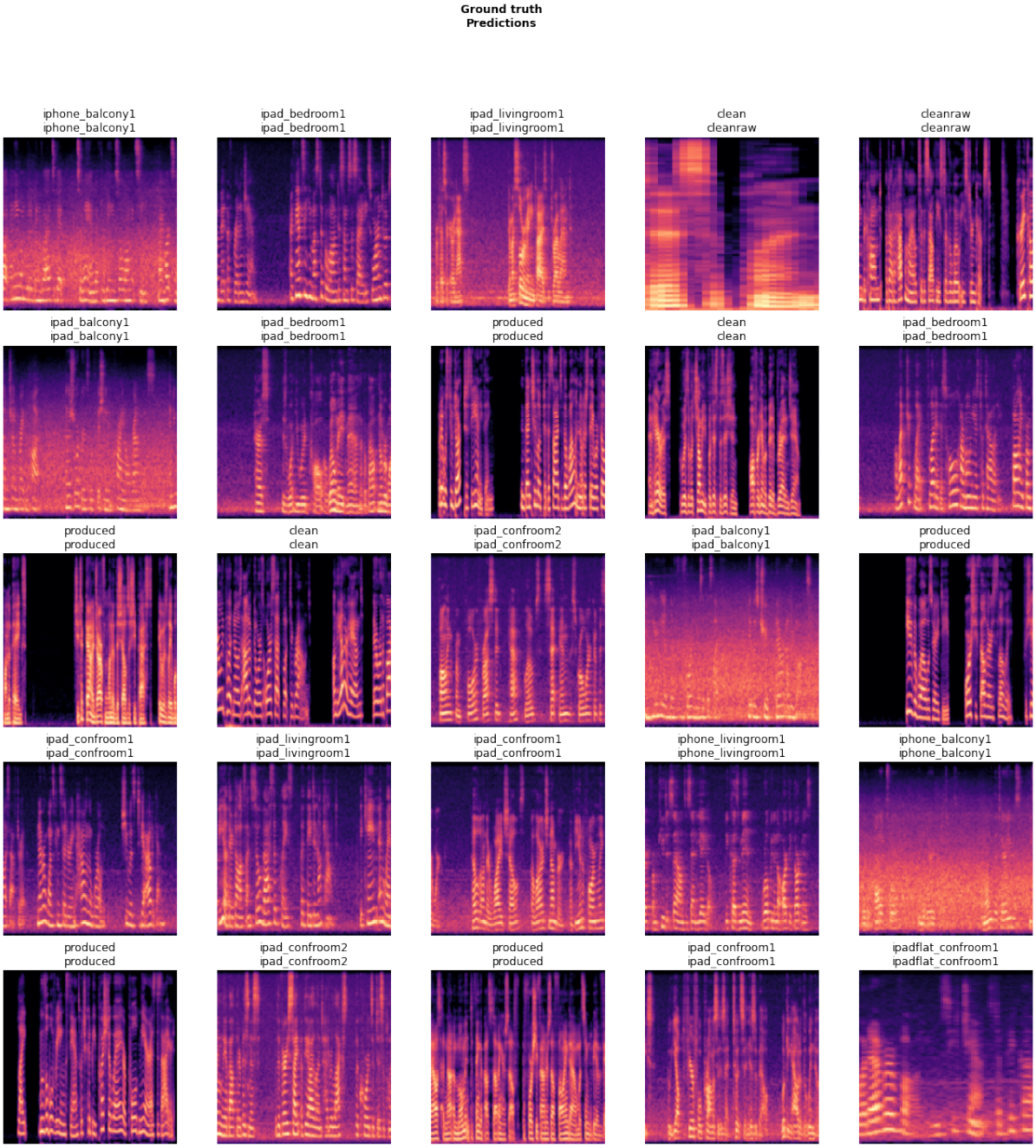

Figure 23 – Illustration depicting model prediction of several examples in the DAPS dataset. Ground truth is the top label and model predictions are shown below it. Source: Author.

on the evidence presented in this section, we theorize this task pertains to the subset of tasks that are quite difficult for humans but unexpectedly trivial for computers. Possibly due to characteristics of individual frequency components either from recording sensors or domain acoustic features (not mutually exclusive). Notably, these results also reflect the potential of multi-domain regularization methods.

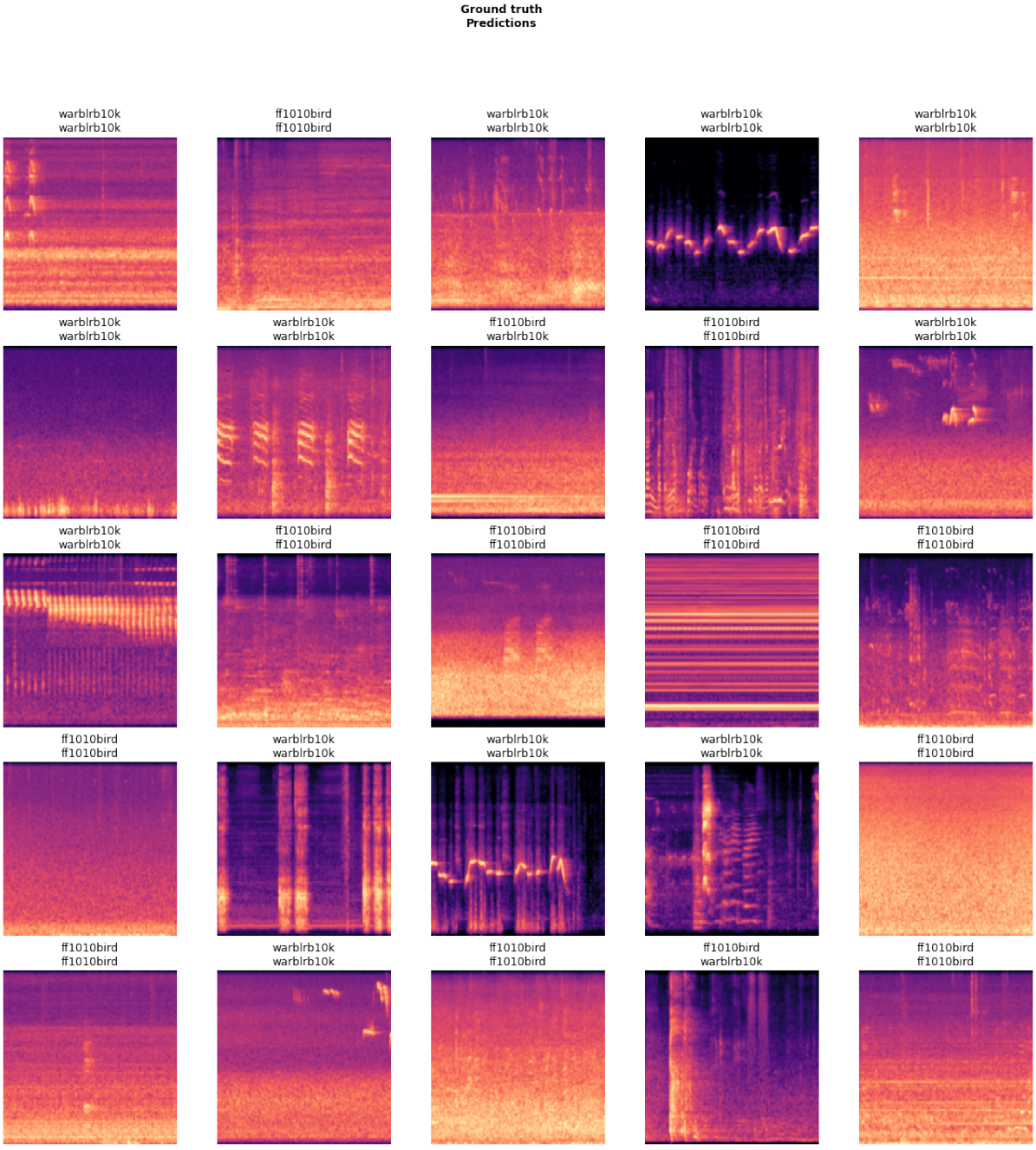

Figure 24 – Illustration depicting model prediction of several examples in the bird dataset. Ground truth is the top label and model predictions are shown below it. Source: Author

Table 4 – Bird Domain Classification Report

Epoch	Train Loss	Validation Loss	Accuracy
0	0.435	3.841	0.484
1	0.315	6.093	0.522
2	0.289	2.140	0.490
3	0.222	0.178	0.940
4	0.175	0.696	0.668
5	0.149	0.175	0.942
6	0.141	0.133	0.953
7	0.124	0.168	0.944
8	0.110	0.116	0.960
9	0.104	0.114	0.960

5.2 Dataset Distribution Manipulation Experiments

This section presents the results of the multi-domain learning experiments in bird classification (Experiments 1 – 4), and speaker identification (Experiment 5).

5.2.1 Experiment 1 — Bird Detection, Original Dataset Size

This experiment uses all of the bird detection datasets (which includes FF1010BIRD and WARBLRB10K as domains). The summarized results can be viewed in Table 5. The summarized results omit the standard deviation and the results from the sequential and inverse sequential methods (which are less relevant to the discussion here). For the expanded table, refer to Appendix B.

Table 5 – Experiment 1 — Bird Detection, Original Dataset Size, Micro F1-Score (Summarized)

Domain	Stew	Balanced	Loss Sum	Random Sum	Loss Mean
WARBLRB10K	0.623	0.694	0.803	0.618	0.799
FF1010BIRD	0.631	0.574	0.779	0.627	0.779
Average	0.627	0.634	0.791	0.622	0.789

When looking at the average score of each method in Table 5, we notice the best result is from using the Loss Sum approach. Additionally, the Random Sum results are far worse than Loss Sum, despite being in the same loss scale. This is evidence against the argument stating that Loss Sum is better because of its higher loss scale. In fact, even the baseline Stew approach performed better than Random Sum. Furthermore, the Loss Mean method performs similarly to Loss Sum, despite not operating in the higher loss scale. This is yet another argument against the higher loss scale being responsible for the Loss Sum improved performance. Notably, the Balanced method also improved the average performance when compared to the Stew baseline. However, there is a tradeoff where the war domain increased in performance at the cost of

lower performance in the FF1010BIRD domain.

We are also interested in how each method evolves regarding its performance during training (some methods may cause a faster generalization during training than others). Figure 25 denotes the test F1-Score evolution during training. We see a significant increase in the F1-Score of the Loss Sum and Loss Mean methods. The Balanced, Stew, and Random Sum methods perform similarly, with the Balanced method being slightly better than the Stew baseline, and the Random Sum performing slightly worse than the baseline. Below we see the Sequential and Inverse Sequential methods, where the moment the training domains change is evidenced by the sudden decline in the 5th epoch.

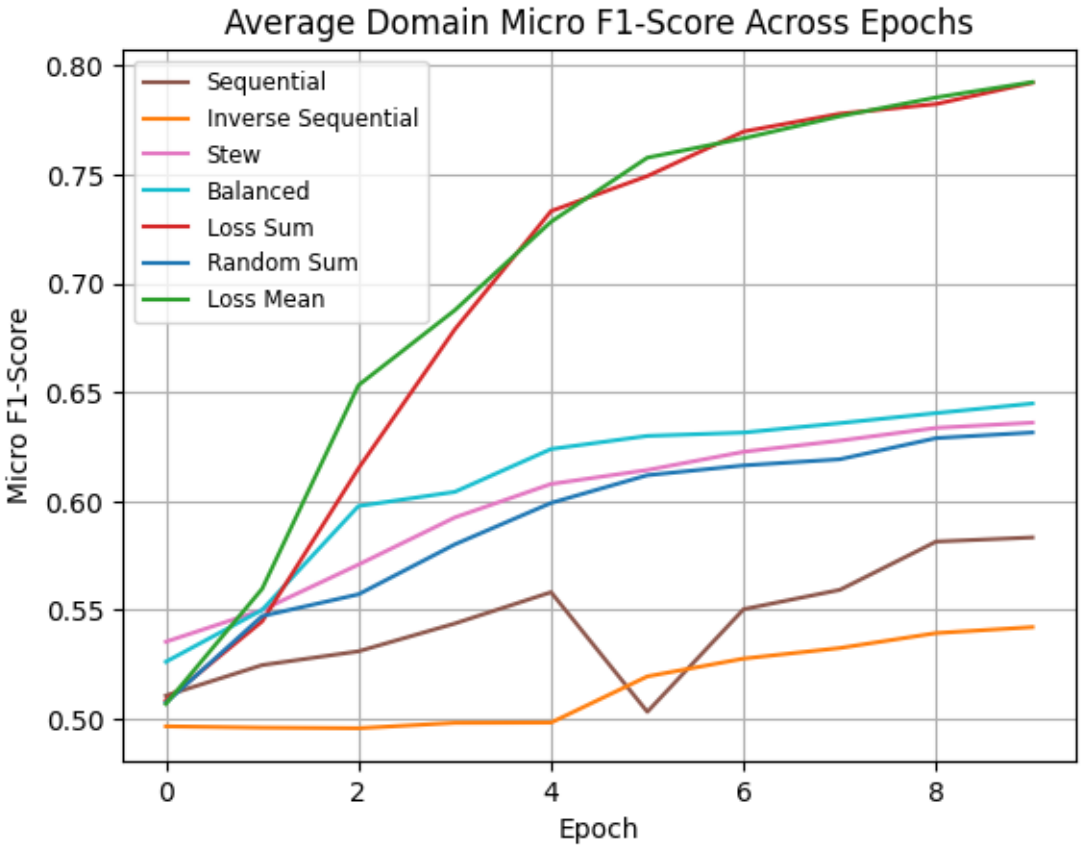

Figure 25 – Average test set domain micro F1-score across epochs for Experiment 1. Source: Author.

Detailed visualizations of loss curve convergence can be seen in Appendices C.

5.2.2 Experiment 2 — Bird Detection, Reduced FF1010BIRD

After reducing the FF1010BIRD dataset, its baseline performance using the Stew method dropped (Table 6 and Figure 26). This was expected, as there are fewer examples to learn from in this domain. Conversely, the performance in the WARBLRB10K domain improved for the same reason: it has more examples, and as a result the model focuses on learning its characteristics and achieves better results on it.

Table 6 – Experiment 2 — Bird Detection, Reduced FF1010BIRD, Micro F1-Score (Summarized)

Domain	Stew	Balanced	Loss Sum	Random Sum	Loss Mean
WARBLRB10K	0.745	0.641	0.797	0.752	0.793
FF1010BIRD	0.475	0.605	0.793	0.512	0.796
Average	0.610	0.623	0.795	0.632	0.795

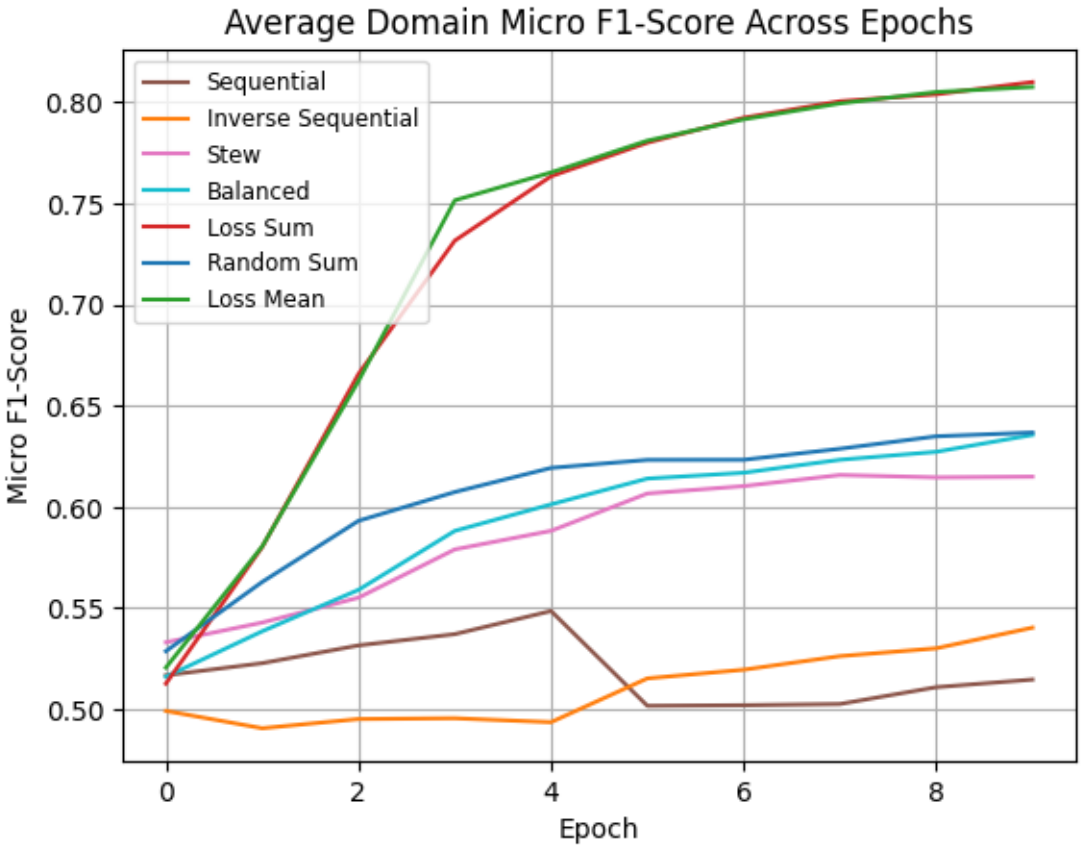

Figure 26 – Average test set domain micro F1-score across epochs for Experiment 2. Source: Author.

The Balanced approach achieved worse results in comparison to using the entire dataset. However, compared to the Stew method, it has maintained performance in the minority dataset (FF1010BIRD), at the cost of some of the performance from the majority dataset. Again, this is aligned with our previous expectations, as the intent behind Balanced Domains is to act as regularization to force the neural network to perform well across domains.

Remarkably, the Loss Sum method achieves slightly better performance when compared to Experiment 1. Despite losing performance in the WARBLRB10K domain, it achieves better performance in FF1010BIRD. This improvement is not entirely expected and might be attributed to its smaller, less reliable test set. Nevertheless, maintaining similar performance to Experiment 1 is interesting evidence of how well Loss Sum

performs when one of the training domains is notably smaller.

The Random Sum method in Experiment 2 performed better than the one in Experiment 1, but when we investigate the domain scores, we see it neglected the reduced domain, similar to the Stew method. The minor improvement is possibly attributed to Random Sum operating on a different loss scale. This is problem-dependent; while this characteristic may help in some problems, in others it will not.

Finally, Loss Mean performs very similarly to Loss Sum. It also potentially suffers from the same optimistic evaluation in the reduced domain.

5.2.3 Experiment 3 — Bird Detection, Reduced WARBLRB10K

Reducing the WARBLRB10K domain yields similar effects to Experiment 2. The Stew method performs marginally better, although it still prioritizes the larger domain (FF1010BIRD).

Table 7 – Experiment 3 — Bird Detection, Reduced WARBLRB10K, Micro F1-Score (Summarized)

Domain	Stew	Balanced	Loss Sum	Random Sum	Loss Mean
WARBLRB10K	0.520	0.649	0.757	0.529	0.768
FF1010BIRD	0.745	0.647	0.771	0.737	0.773
Average	0.632	0.648	0.764	0.633	0.771

In this experiment, we have further evidence depicting how the Balanced method stops the model from focusing solely on the larger domain (Table 6 and Figure 27). Notably, Loss Sum and Loss Mean achieve the best results.

Furthermore, the Random Sum approach does not seem to yield competitive results despite operating on a higher loss scale. This provides evidence against the argument that higher loss values help in this particular task.

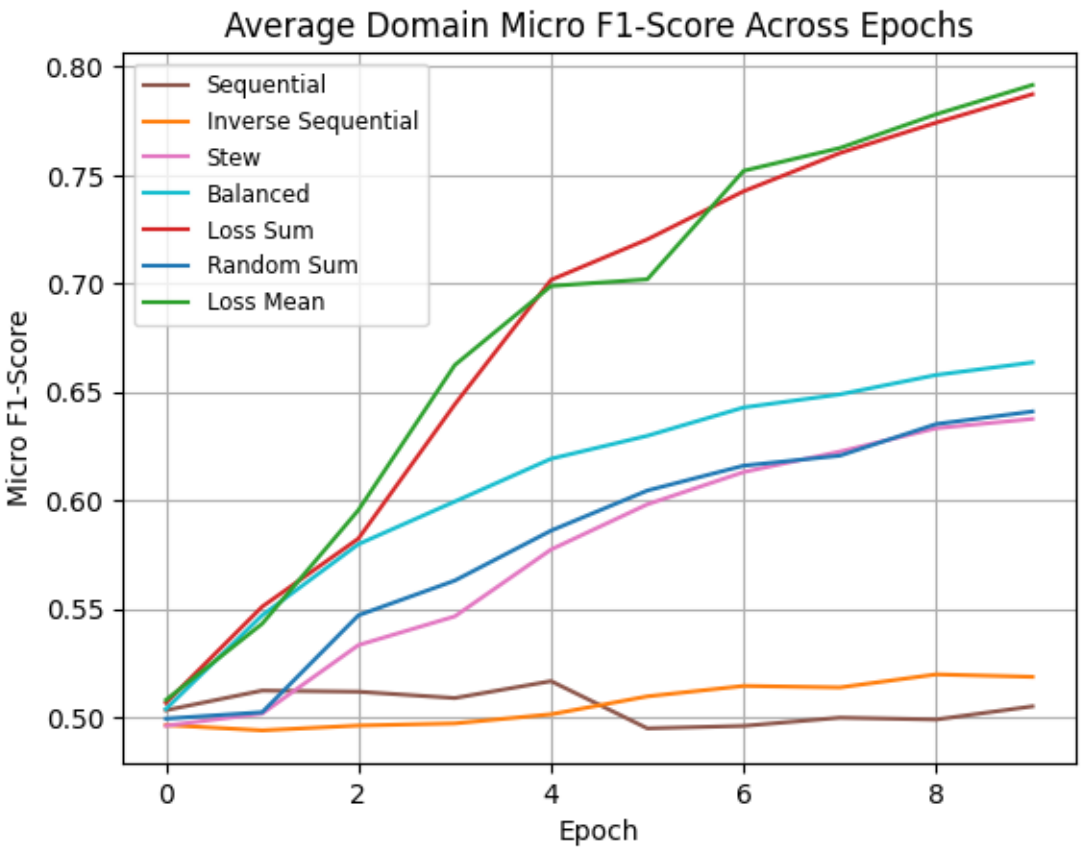

Figure 27 – Average test set domain micro F1-score across epochs for Experiment 3. Source: Author.

5.2.4 Experiment 4 — Bird Detection, Reduced Symmetric, Micro F1-Score

Reducing both domains to a few hundred examples causes model performance to drop dramatically (Table 8). It is relevant to remember this configuration is balanced, as both domains have the same amount of examples.

Table 8 – Experiment 4 — Bird Detection, Reduced Symmetric, Micro F1-Score (Summarized)

Domain	Stew	Balanced	Loss Sum	Random Sum	Loss Mean
WARBLRB10K	0.603	0.538	0.588	0.569	0.656
FF1010BIRD	0.390	0.429	0.424	0.442	0.411
Average	0.497	0.483	0.506	0.505	0.533

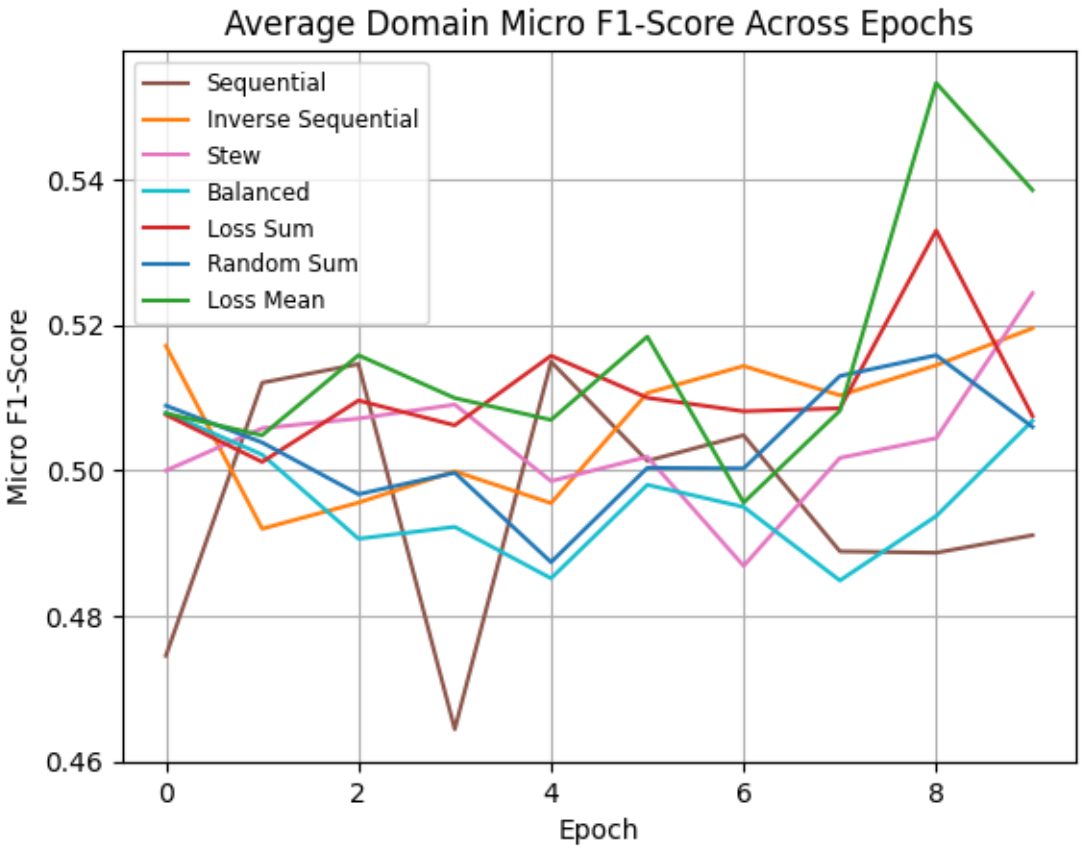

Figure 28 – Average test set domain micro F1-score across epochs for Experiment 4. Source: Author.

In this extreme scenario, the Stew approach still prioritizes one of the domains. Even though the Balanced method performs worse than the Stew one, it mitigates domain favoritism. Again, note there is a significant tradeoff where the favorite domain drops in performance. The Loss Sum and Random Sum perform similarly in this scenario, with the Loss Mean approach yielding the best results.

The scarce amount of examples is insufficient for methods to learn domain-specific distributions. Given the fact the bird domains contain different class distributions, a

shared feature representation representing both domains is not achievable and the F1-Score across epochs is erratic for all training methods (Figure 28).

5.2.5 Experiment 5 — Speaker Identification, Original Dataset Size

This experiment uses DAPS, a larger dataset with several domains. Thus, we evaluate it on a classification task with 20 different classes (Table 9). Expectedly, the results are worse than the previous experiments which used a binary classification task.

Moreover, the results suggest the number of epochs to approach this task could be increased for better results in the target task. The reasoning behind this assessment is the lack of convergence in the F1-Score curves (Figure **??**).

The Stew method achieved competitive results in this experiment. Additionally, the Balanced approach marginally diminished the performance, when compared to the Stew baseline. This is not entirely unexpected, as the domains in this experiment are already balanced. Domain balancing in batches for an already-balanced multi-domain dataset mostly only introduces overhead.

Random Sum and Loss Mean performed below the comparison baseline. But the Loss Sum method achieved similar results to the Stew approach. However, there are some differences in the performance of individual domains. Overall, the training methods behave very differently when used on a high quantity of domains.

Table 9 – Experiment 5 — Speaker Identification, Original Dataset Size, Micro F1-Score (Summarized)

Domain	Stew	Balanced	Loss Sum	Random Sum	Loss Mean
CLEAN	0.132	0.130	0.136	0.129	0.123
IPAD_BALCONY1	0.139	0.139	0.142	0.133	0.136
IPAD_BEDROOM1	0.134	0.130	0.130	0.125	0.127
IPAD_CONFROOM1	0.129	0.129	0.131	0.125	0.125
IPAD_CONFROOM2	0.135	0.132	0.135	0.128	0.121
IPADFLAT_CONFROOM1	0.140	0.137	0.140	0.128	0.133
IPADFLAT_OFFICE1	0.135	0.132	0.132	0.126	0.125
IPAD_LIVINGROOM1	0.135	0.131	0.131	0.128	0.126
IPAD_OFFICE1	0.129	0.127	0.133	0.123	0.121
IPAD_OFFICE2	0.137	0.134	0.134	0.133	0.130
IPHONE_BALCONY1	0.141	0.139	0.146	0.132	0.139
IPHONE_BEDROOM1	0.133	0.130	0.130	0.126	0.126
IPHONE_LIVINGROOM1	0.126	0.125	0.125	0.126	0.123
PRODUCED	0.134	0.130	0.139	0.129	0.128
Average	0.134	0.132	0.134	0.128	0.127

6 CONCLUSION

This work presented an evaluation of multi-domain learning training approaches to regulating domain presence in batches when addressing audio classification tasks. We focused on the following methods: Stew, Balanced Domains, and Loss Sum. Additional counterfactual and comparison methods were investigated, such as Random Sum, Loss Mean, and sequential approaches.

When handling domains with different class distributions, Balanced domains, and Loss Sum seemed to mitigate model domain favoritism. Particularly when some domains are limited in terms of data quantity. Loss Sum consistently presented competitive results in most experiments, improving baseline results in most scenarios.

Despite improving results in several scenarios, Experiment 4 evidenced the necessity of data quantity in domains to better leverage the regulation capacity of the evaluated methods.

The experiments also provide evidence supporting the argument the loss aggregation methods benefit model training because of the individual domain calculation, and not because of the higher loss scale they operate in.

The results suggest that using explicit domain information by presenting them separately in individual batches for each domain potentially benefits the learning when training models in multi-domain tasks. This becomes more evident in experiments with fewer domains with unique class distributions.

It is important to address the limitations of the study. We refrain from hyperparameter tuning to be able to cover multiple dataset partitions, as it is a computationally intensive task. We also lack benchmarks or even similar studies to evaluate this sort of approach for audio classification problems. As a result, validating or comparing these experiments is a difficult task.

Overall, multi-domain learning techniques using individual domain loss calculation, such as Loss Sum, provide an interesting strategy when dealing with multiple domains. Loss Mean performs similarly to Loss Sum likely due to the presence of a similar mechanism. However, according to our experiments, it is not, in fact, due to the higher loss values — as Loss Mean does not operate on a higher loss scale and often achieves

competitive results as well. Upon investigation, we found that the PyTorch implementation might be subject to small floating point precision inaccuracies when representing the sum of loss values. Previous studies have argued for using the quantization of weight parameters in neural networks as a means to curb specialization and memory costs, addressing overfit (YANG et al., 2019; JIN; YANG; LIAO, 2020; ZHANG et al., 2018). However, it is unknown whether the improvements seen in experimental results could be attributed to floating point representation inadvertently causing network weight quantization.

Future studies could also look at how different domain sizes can impact the learning of the model using the proposed methods. Some of the proposed methods may be more robust to domain imbalance than others and further research could analyze this aspect. Additionally, it may be interesting to explore different techniques of combining the loss from different domains, beyond simple arithmetic addition. Using other multi-domain datasets to expand the experiments presented here could also help understand the behavior of the proposed methods. Finally, exploring the interaction of different loss functions with the Loss Sum approach, and replicating the results seen in this study across other tasks beyond classification, such as speech recognition.

REFERENCES

AGARAP, A. F. Deep learning using rectified linear units (relu). **arXiv preprint arXiv:1803.08375**, [S.l.], 2018.

AMINI. MIT 6.S191: Introduction to Deep Learning. In: **Introduction to Deep Learning**. Massachusetts: The MIT Press, 2020.

ARPIT, D.; WANG, H.; ZHOU, Y.; XIONG, C. Ensemble of averages: Improving model selection and boosting performance in domain generalization. **arXiv preprint arXiv:2110.10832**, [S.l.], 2021.

BECKER. Modern Approaches in Natural Language Processing. In: **Modern Approaches in Natural Language Processing**. Munich: The MIT Press, 2020.

BEN-DAVID, S.; BLITZER, J.; CRAMMER, K.; PEREIRA, F. Analysis of representations for domain adaptation. **Advances in neural information processing systems**, Canada, v.19, 2006.

BEN-DAVID, S. et al. A theory of learning from different domains. **Machine Learning**, [S.l.], v.79, p.151–175, 05 2010.

BENDER, I. B. **Evaluating Machine Learning Methodologies for Multi-Domain Learning in Image Classification**. 2022. 51p. Master's Thesis (Computer Science) — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas.

BOEHMKE. **Feedforward Deep Learning Models**. Disponível em: <http://ucr.github.io/feedforward_DNN>. Acesso em: 2023-05-22.

BROWN, T. B. et al. Language Models Are Few-Shot Learners. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 34., 2020, Red Hook, NY, USA. **Proceedings...** Curran Associates Inc., 2020. (NIPS'20).

CHAN, W. et al. Speechstew: Simply mix all available speech recognition data to train one large neural network. **arXiv preprint arXiv:2104.02133**, [S.l.], 2021.

CHEN, Y.; LU, R.; ZOU, Y.; ZHANG, Y. Branch-Activated Multi-Domain Convolutional Neural Network for Visual Tracking. **Journal of Shanghai Jiaotong University (Science)**, Shanghai, v.23, p.360–367, 2018.

CHOJNACKA, R.; PELECANOS, J.; WANG, Q.; MORENO, I. L. Speakerstew: Scaling to many languages with a triaged multilingual text-dependent and text-independent speaker verification system. **arXiv preprint arXiv:2104.02125**, [S.l.], 2021.

CHOWDHERY, A. et al. Palm: Scaling language modeling with pathways. **arXiv preprint arXiv:2204.02311**, [S.l.], 2022.

CREVIER, D. **AI**: the tumultuous history of the search for artificial intelligence. [S.l.]: Basic Books, Inc., 1993.

DALLY, W. J.; KECKLER, S. W.; KIRK, D. B. Evolution of the Graphics Processing Unit (GPU). **IEEE Micro**, Austin, Texas, v.41, n.6, p.42–51, 2021.

DO, H. H.; PRASAD, P. W.; MAAG, A.; ALSADOON, A. Deep learning for aspect-based sentiment analysis: a comparative review. **Expert systems with applications**, Louisiana, v.118, p.272–299, 2019.

DOMINGOS, P. A few useful things to know about machine learning. **Communications of the ACM**, New York, United States, v.55, n.10, p.78–87, 2012.

FRENCH, R. M. Catastrophic forgetting in connectionist networks. **Trends in cognitive sciences**, [S.l.], v.3, n.4, p.128–135, 1999.

GANIN, Y.; LEMPITSKY, V. Unsupervised domain adaptation by backpropagation. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2015, Lille, France. **Anais...** JMLR.org, 2015. p.1180–1189.

GANIN, Y. et al. Domain-adversarial training of neural networks. **The journal of machine learning research**, [S.l.], v.17, n.1, p.2096–2030, 2016.

GARSON, J. Connectionism. In: **Connectionism**. CA, United States: Metaphysics Research Lab, Stanford University, 1997.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. `http://www.deeplearningbook.org`.

GOODFELLOW, I. J. et al. An empirical investigation of catastrophic forgetting in gradient-based neural networks. **2nd International Conference on Learning Representations, ICLR 2014**, Banff, AB, Canada, 2014.

GULRAJANI, I.; LOPEZ-PAZ, D. In search of lost domain generalization. **arXiv preprint arXiv:2007.01434**, [S.l.], 2020.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2016, NV, USA. **Proceedings...** IEEE, 2016. p.770–778.

HINTON, G. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. **IEEE Signal processing magazine**, [S.l.], v.29, n.6, p.82–97, 2012.

HOCHREITER, S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, [S.l.], v.6, p.107–116, 04 1998.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, Massachusetts, v.9, n.8, p.1735–1780, 1997.

JAIN, A. et al. Overview and Importance of Data Quality for Machine Learning Tasks. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY & DATA MINING, 26., 2020. **Proceedings...** Association for Computing Machinery, 2020. p.3561–3562. (KDD '20).

JIN, Q.; YANG, L.; LIAO, Z. AdaBits: Neural Network Quantization With Adaptive Bit-Widths. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2020. **Proceedings...** IEEE, 2020.

JOUPPI, N. P. et al. In-datacenter performance analysis of a tensor processing unit. In: OF THE 44TH ANNUAL INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE, 2017, Toronto, Canada. **Proceedings...** IEEE, 2017. p.1–12.

KANG, G.; JIANG, L.; YANG, Y.; HAUPTMANN, A. G. Contrastive adaptation network for unsupervised domain adaptation. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2019, CA, USA. **Proceedings...** IEEE, 2019. p.4893–4902.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Communications of the ACM**, NY, United States, v.60, n.6, p.84–90, 2017.

KUMAR. **Regarding Linear Separability**. Disponível em: <https://vitalflux.com/how-know-data-linear-non-linear/>. Acesso em: 2023-05-25.

LAPARRA, E.; BETHARD, S.; MILLER, T. A. Rethinking domain adaptation for machine learning over clinical language. **JAMIA open**, Oxford, United Kingdom, v.3, n.2, p.146–150, 2020.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, [S.l.], v.521, n.7553, p.436–444, 2015.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. **Neural computation**, Massachusetts, USA, v.1, n.4, p.541–551, 1989.

LI, D.; YANG, Y.; SONG, Y.-Z.; HOSPEDALES, T. M. Deeper, broader and artier domain generalization. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2017, Venice, Italy. **Proceedings...** IEEE, 2017. p.5542–5550.

LI, H.; PAN, S. J.; WANG, S.; KOT, A. C. Domain generalization with adversarial feature learning. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2018, Salt Lake City, UT, USA. **Proceedings...** IEEE, 2018. p.5400–5409.

LIKHOMANENKO, T. et al. Rethinking evaluation in asr: Are our models robust enough? **arXiv preprint arXiv:2010.11745**, [S.l.], 2020.

LIU, Y. et al. Compact feature learning for multi-domain image classification. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2019, Long Beach, CA, USA. **Proceedings...** IEEE, 2019. p.7193–7201.

MAKHOUL, J.; COSELL, L. LPCW: An LPC vocoder with linear predictive spectral warping. In: ICASSP'76. IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, 1976, Philadelphia, Pennsylvania, USA. **Anais...** IEEE, 1976. v.1, p.466–469.

MINSKY, M.; PAPERT, S. A. **Perceptrons, Reissue of the 1988 Expanded Edition with a new foreword by Léon Bottou**: An Introduction to Computational Geometry. [S.l.]: MIT press, 2017.

MNIH, V. et al. Playing atari with deep reinforcement learning. **arXiv preprint arXiv:1312.5602**, [S.l.], 2013.

MOHAMED, A.-r.; DAHL, G.; HINTON, G. et al. Deep belief networks for phone recognition. In: NIPS WORKSHOP ON DEEP LEARNING FOR SPEECH RECOGNITION AND RELATED APPLICATIONS, 2009, Vancouver, BC, Canada. **Anais...** IEEE, 2009. v.1, n.9, p.39.

MYSORE, G. J. Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech?—a dataset, insights, and challenges. **IEEE Signal Processing Letters**, Florence, Italy, v.22, n.8, p.1006–1010, 2014.

NA, J.; JUNG, H.; CHANG, H. J.; HWANG, W. Fixbi: Bridging domain spaces for unsupervised domain adaptation. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2021, Nashville, TN, USA. **Proceedings...** IEEE, 2021. p.1094–1103.

NAM, H.; HAN, B. Learning multi-domain convolutional neural networks for visual tracking. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2016, Las Vegas, Nevada, USA. **Proceedings...** IEEE, 2016. p.4293–4302.

NARAYANAN, A. et al. Toward domain-invariant speech recognition via large scale training. In: IEEE SPOKEN LANGUAGE TECHNOLOGY WORKSHOP (SLT), 2018., 2018, Athens, Greece. **Anais...** IEEE, 2018. p.441–447.

NEWELL, A.; SHAW, J. C.; SIMON, H. A. Report on a general problem solving program. In: IFIP CONGRESS, 1959, Paris, France. **Anais...** Springer International, 1959. v.256, p.64.

NGUYEN. Mathematical Group Theory. In: **Group Homomorphism**. [S.l.: s.n.], 2008.

NIU, S.; LIU, Y.; WANG, J.; SONG, H. A decade survey of transfer learning (2010–2020). **IEEE Transactions on Artificial Intelligence**, [S.l.], v.1, n.2, p.151–166, 2020.

QUINONERO-CANDELA, J.; SUGIYAMA, M.; SCHWAIGHOFER, A.; LAWRENCE, N. D. **Dataset shift in machine learning**. [S.l.]: Mit Press, 2008.

RADFORD, A. et al. **Robust Speech Recognition via Large-Scale Weak Supervision**. [S.l.]: Technical report, OpenAI, 2022. URL https://cdn. openai. com/papers/whisper. pdf, 2022.

RIBEIRO, J.; MELO, F. S.; DIAS, J. Multi-task learning and catastrophic forgetting in continual reinforcement learning. **arXiv preprint arXiv:1909.10008**, [S.l.], 2019.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, [S.l.], v.65, n.6, p.386, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning internal representations by error propagation**. [S.l.]: California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, [S.l.], v.323, n.6088, p.533–536, 1986.

SAENKO, K.; KULIS, B.; FRITZ, M.; DARRELL, T. Adapting visual category models to new domains. In: EUROPEAN CONFERENCE ON COMPUTER VISION, 2010, Heraklion, Crete. **Anais...** Springer, 2010. p.213–226.

SAMBASIVAN, N. et al. "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In: CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2021., 2021, Okohama, Japan. **Anais...** ACM, 2021. p.1–15.

SCHWEIGHOFER, E. Data-Centric Machine Learning: Improving Model Performance and Understanding Through Dataset Analysis. In: LEGAL KNOWLEDGE AND INFORMATION SYSTEMS: JURIX 2021: THE THIRTY-FOURTH ANNUAL CONFERENCE, VILNIUS, LITHUANIA, 8-10 DECEMBER 2021, 2022. **Anais...** IOS Press, 2022. v.346, p.54.

SICILIA, A. et al. Multi-domain learning by meta-learning: Taking optimal steps in multi-domain loss landscapes by inner-loop learning. In: IEEE 18TH INTERNATIONAL SYMPOSIUM ON BIOMEDICAL IMAGING (ISBI), 2021., 2021, Nice, France. **Anais...** IEEE, 2021. p.650–654.

SINGH, A. et al. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. **ACM SIGCOMM computer communication review**, London, United Kingdom, v.45, n.4, p.183–197, 2015.

STOWELL, D.; PLUMBLEY, M. D. An open dataset for research on audio field recording archives: freefield1010. **arXiv preprint arXiv:1309.5275**, [S.l.], 2013.

SWAPNA. **Convolutional Neural Networks**. Disponível em: <https://developersbreach.com/convolution-neural-network-deep-learning/>. Acesso em: 2023-05-17.

TETTEH, E. et al. Multi-Domain Balanced Sampling Improves Out-of-Distribution Generalization of Chest X-ray Pathology Prediction Models. **Medical Imaging meets NeurIPS**, [S.l.], 2021.

THOPPILAN, R. et al. Lamda: Language models for dialog applications. **arXiv preprint arXiv:2201.08239**, [S.l.], 2022.

VANSCHOREN, J. Meta-learning: A survey. **arXiv preprint arXiv:1810.03548**, [S.l.], 2018.

VASWANI, A. et al. Attention is all you need. **Advances in neural information processing systems**, Long Beach, CA, USA, v.30, 2017.

WEISS, K.; KHOSHGOFTAAR, T. M.; WANG, D. A survey of transfer learning. **Journal of Big data**, [S.l.], v.3, n.1, p.1–40, 2016.

XIE, S.; ZHENG, Z.; CHEN, L.; CHEN, C. Learning semantic representations for unsupervised domain adaptation. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2018. **Anais...** JMLR.org, 2018. p.5423–5432.

XU, T. et al. Cdtrans: Cross-domain transformer for unsupervised domain adaptation. **arXiv preprint arXiv:2109.06165**, [S.l.], 2021.

XU, X. et al. d-sne: Domain adaptation using stochastic neighborhood embedding. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2019, Long Beach, CA, USA. **Proceedings...** IEEE, 2019. p.2497–2506.

YANG, J. et al. Quantization Networks. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2019, Long Beach, CA, USA. **Proceedings...** IEEE, 2019.

ZHANG, D.; YANG, J.; YE, D.; HUA, G. LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks. In: EUROPEAN CONFERENCE ON COMPUTER VISION (ECCV), 2018, Munich, Germany. **Proceedings...** IEEE, 2018.

Appendices

APPENDIX A – Explainability Using SHAP

SHAP is an AI explainability technique (an acronym for SHapley Additive exPlanations). The SHAP values evaluate the impact of features in comparison to the prediction should the feature had some other baseline value. In other words, they allow decomposing prediction into feature importances.

We interpret spectrogram intensity values as SHAP values, and inspect their influence on model predictions. Figures 29, 30, 31, 32, and 33 show SHAP value analysis for a few spectrogram predictions. Our intention is to verify what information the model is using to make its decisions. In all figures, we see consistent SHAP colors across specific spectrogram frequency bands. This is evidence of sound model decisions, as it is using frequency information to perform its predictions, as expected. It is possible the domains present distinct recording sensor frequency characteristics that make it possible for easy domain distinction.

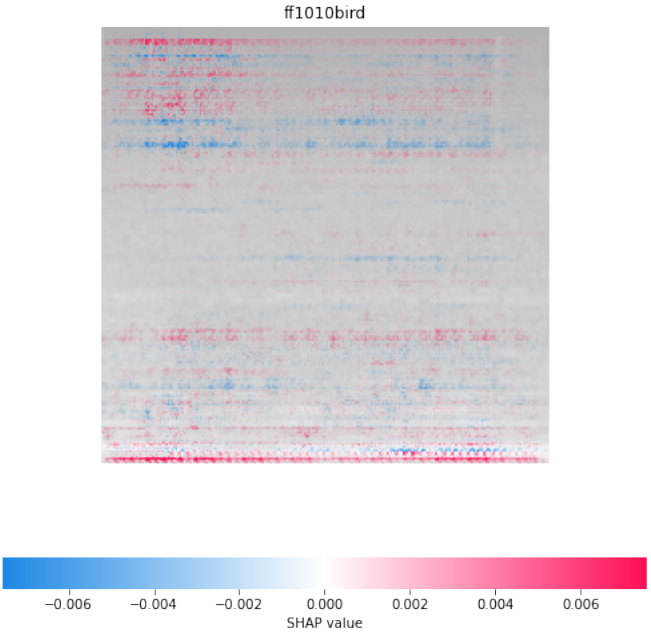

Figure 29 – SHAP analysis of bird detection example 1. Source: Author.

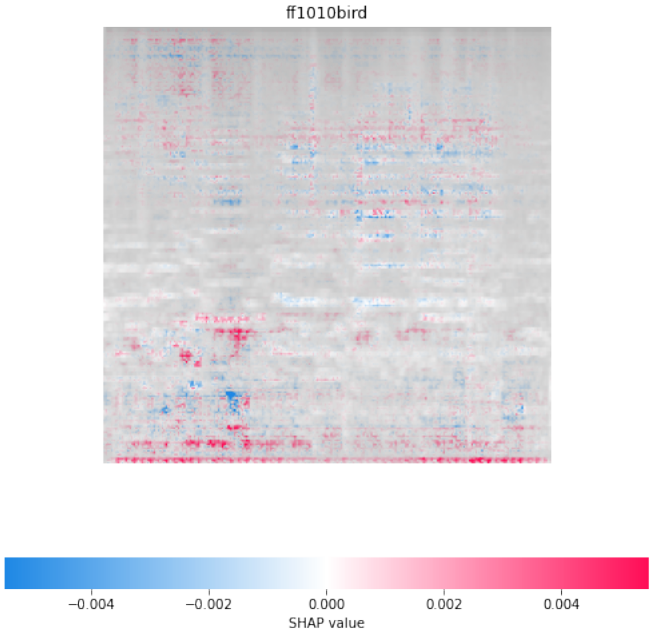

Figure 30 – SHAP analysis of bird detection example 2. Source: Author.

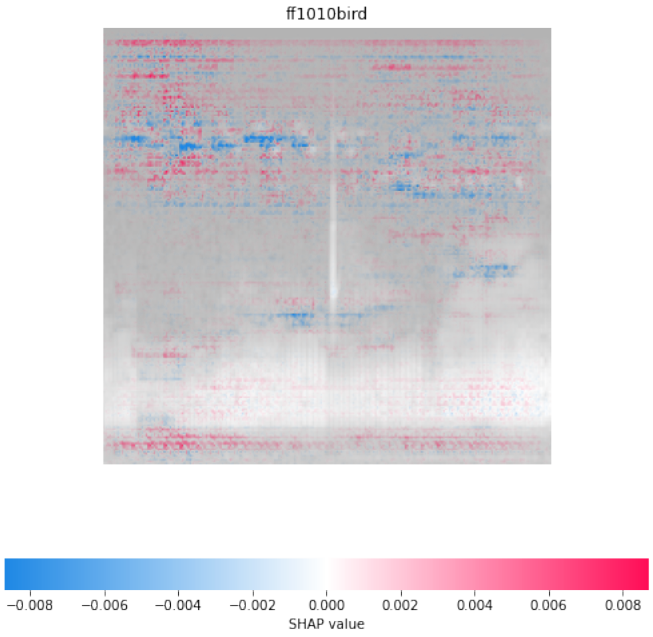

Figure 31 – SHAP analysis of bird detection example 3. Source: Author.

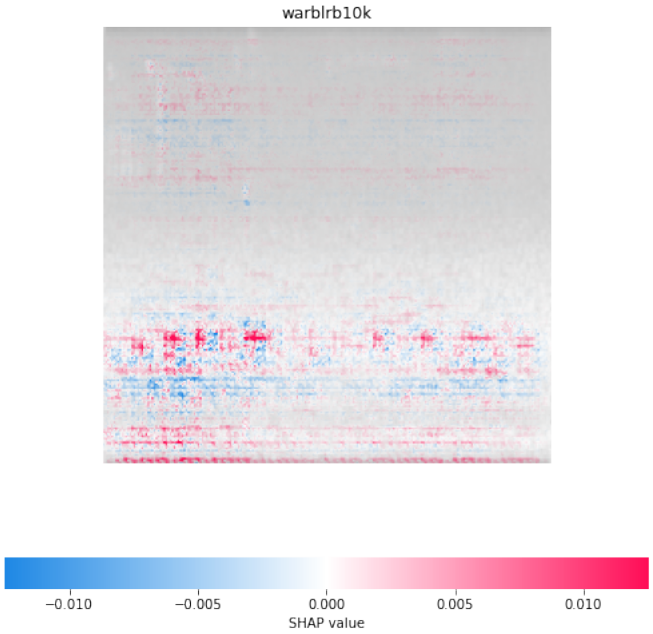

Figure 32 – SHAP analysis of bird detection example 4. Source: Author.

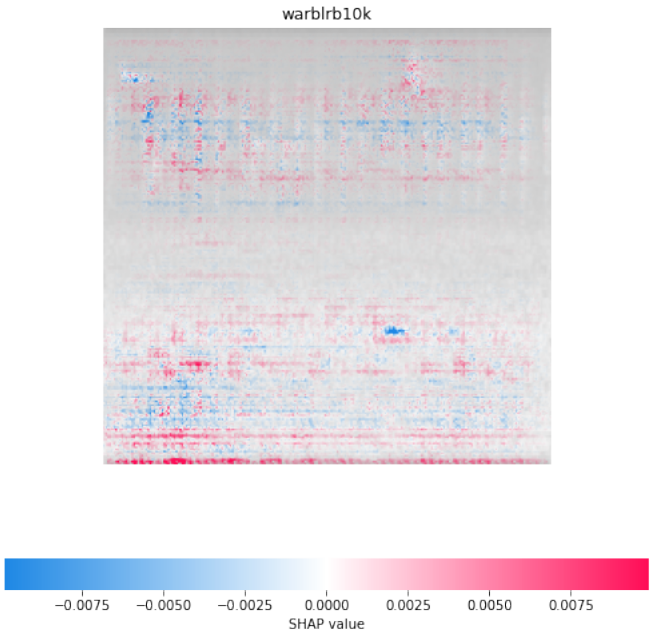

Figure 33 – SHAP analysis of bird detection example 5. Source: Author.

APPENDIX B – Complete Experiment F1-Score Tables

This appendix contains the expanded tables showing the complete experiment results (Tables 10, 11, 12, 13, 14).

Table 10 – Experiment 1 — Bird Detection, Original Dataset Size, Micro F1-Score

Domain	Sequential	Inverse Sequential	Stew	Balanced	Loss Sum	Random Sum	Loss Mean
WARBLRB10K	0.368 ± 0.055	0.786 ± 0.010	0.623 ± 0.026	0.694 ± 0.038	0.803 ± 0.009	0.618 ± 0.033	0.799 ± 0.013
FF1010BIRD	0.791 ± 0.014	0.290 ± 0.030	0.631 ± 0.026	0.574 ± 0.030	0.779 ± 0.029	0.627 ± 0.036	0.779 ± 0.022
Average	0.580 ± 0.033	0.538 ± 0.016	0.627 ± 0.013	0.634 ± 0.015	0.791 ± 0.015	0.622 ± 0.019	0.789 ± 0.015

Table 11 – Experiment 2 — Bird Detection, Reduced FF1010BIRD, Micro F1-Score

Domain	Sequential	Inverse Sequential	Stew	Balanced	Loss Sum	Random Sum	Loss Mean
WARBLRB10K	0.275 ± 0.047	0.777 ± 0.009	0.745 ± 0.037	0.641 ± 0.026	0.797 ± 0.018	0.752 ± 0.028	0.793 ± 0.031
FF1010BIRD	0.754 ± 0.011	0.285 ± 0.029	0.475 ± 0.051	0.605 ± 0.039	0.793 ± 0.015	0.512 ± 0.032	0.796 ± 0.012
Average	0.514 ± 0.028	0.531 ± 0.017	0.610 ± 0.019	0.623 ± 0.018	0.795 ± 0.013	0.632 ± 0.015	0.795 ± 0.019

Table 12 – Experiment 3 — Bird Detection, Reduced WARBLRB10K, Micro F1-Score

Domain	Sequential	Inverse Sequential	Stew	Balanced	Loss Sum	Random Sum	Loss Mean
WARBLRB10K	0.258 ± 0.044	0.766 ± 0.006	0.520 ± 0.061	0.649 ± 0.031	0.757 ± 0.068	0.529 ± 0.047	0.768 ± 0.049
FF1010BIRD	0.751 ± 0.008	0.265 ± 0.007	0.745 ± 0.032	0.647 ± 0.044	0.771 ± 0.023	0.737 ± 0.042	0.773 ± 0.020
Average	0.504 ± 0.025	0.515 ± 0.006	0.632 ± 0.033	0.648 ± 0.024	0.764 ± 0.038	0.633 ± 0.025	0.771 ± 0.029

Table 13 – Experiment 4 — Bird Detection, Reduced Symmetric, Micro F1-Score

Domain	Sequential	Inverse Sequential	Stew	Balanced	Loss Sum	Random Sum	Loss Mean
WARBLRB10K	0.313 ± 0.176	0.761 ± 0.005	0.603 ± 0.221	0.538 ± 0.240	0.588 ± 0.245	0.569 ± 0.234	0.656 ± 0.207
FF1010BIRD	0.698 ± 0.148	0.279 ± 0.090	0.390 ± 0.182	0.429 ± 0.210	0.424 ± 0.231	0.442 ± 0.204	0.411 ± 0.221
Average	0.506 ± 0.080	0.52 ± 0.044	0.497 ± 0.06	0.483 ± 0.073	0.506 ± 0.092	0.505 ± 0.057	0.533 ± 0.095

Table 14 – Experiment 5 — Speaker Identification, Original Dataset Size, Micro F1-Score

Domain	Sequential	Inverse Sequential	Stew	Balanced	Loss Sum	Random Sum	Loss Mean
CLEAN	0.049 ± 0.021	0.049 ± 0.021	0.132 ± 0.018	0.130 ± 0.012	0.136 ± 0.017	0.129 ± 0.019	0.123 ± 0.015
IPAD_BALCONY1	0.048 ± 0.019	0.048 ± 0.019	0.139 ± 0.015	0.139 ± 0.016	0.142 ± 0.018	0.133 ± 0.018	0.136 ± 0.016
IPAD_BEDROOM1	0.051 ± 0.022	0.051 ± 0.022	0.134 ± 0.014	0.130 ± 0.013	0.130 ± 0.015	0.125 ± 0.012	0.127 ± 0.012
IPAD_CONFROOM1	0.052 ± 0.021	0.052 ± 0.021	0.129 ± 0.015	0.129 ± 0.014	0.131 ± 0.013	0.125 ± 0.014	0.125 ± 0.015
IPAD_CONFROOM2	0.048 ± 0.019	0.048 ± 0.019	0.135 ± 0.018	0.132 ± 0.015	0.135 ± 0.017	0.128 ± 0.018	0.121 ± 0.020
IPADFLAT_CONFROOM1	0.050 ± 0.018	0.050 ± 0.018	0.140 ± 0.017	0.137 ± 0.010	0.140 ± 0.016	0.128 ± 0.016	0.133 ± 0.013
IPADFLAT_OFFICE1	0.052 ± 0.021	0.052 ± 0.021	0.135 ± 0.017	0.132 ± 0.012	0.132 ± 0.018	0.126 ± 0.017	0.125 ± 0.012
IPAD_LIVINGROOM1	0.048 ± 0.020	0.048 ± 0.020	0.135 ± 0.016	0.131 ± 0.011	0.131 ± 0.017	0.128 ± 0.014	0.126 ± 0.016
IPAD_OFFICE1	0.051 ± 0.019	0.051 ± 0.019	0.129 ± 0.013	0.127 ± 0.011	0.133 ± 0.014	0.123 ± 0.013	0.121 ± 0.013
IPAD_OFFICE2	0.048 ± 0.018	0.048 ± 0.018	0.137 ± 0.015	0.134 ± 0.011	0.134 ± 0.014	0.133 ± 0.016	0.130 ± 0.013
IPHONE_BALCONY1	0.048 ± 0.016	0.048 ± 0.016	0.141 ± 0.017	0.139 ± 0.017	0.146 ± 0.021	0.132 ± 0.019	0.139 ± 0.020
IPHONE_BEDROOM1	0.051 ± 0.020	0.051 ± 0.020	0.133 ± 0.016	0.130 ± 0.012	0.130 ± 0.016	0.126 ± 0.014	0.126 ± 0.013
IPHONE_LIVINGROOM1	0.046 ± 0.018	0.046 ± 0.018	0.126 ± 0.013	0.125 ± 0.007	0.125 ± 0.012	0.126 ± 0.016	0.123 ± 0.010
PRODUCED	0.048 ± 0.018	0.048 ± 0.018	0.134 ± 0.017	0.130 ± 0.009	0.139 ± 0.018	0.129 ± 0.019	0.128 ± 0.013
Average	0.049 ± 0.018	0.049 ± 0.018	0.134 ± 0.013	0.132 ± 0.010	0.134 ± 0.012	0.128 ± 0.014	0.127 ± 0.011

APPENDIX C – Loss Curve Convergence Visualization

This appendix shows the loss curves during model training on the bird detection dataset, using its original size (Experiment 1).

The real training loss used in backpropagation is depicted in Figure 34. Because some methods operate on higher loss values, the comparison is difficult. We normalize these methods by dividing their loss values by the number of domains (Figure 35). The methods depict similar results towards the end, which may be difficult to visualize. For this reason, we provide Figure showing a zoomed version of the plot including only the last 3 epochs.

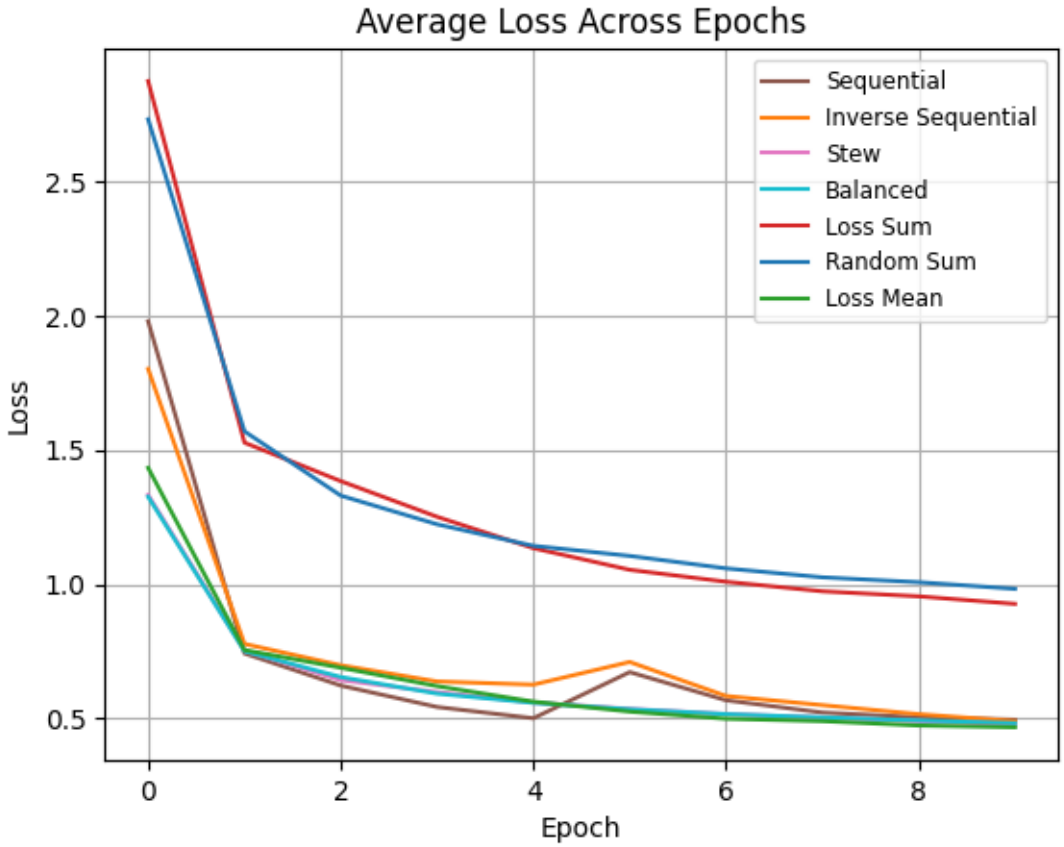

Figure 34 – Average domain loss across epochs. Source: Author.

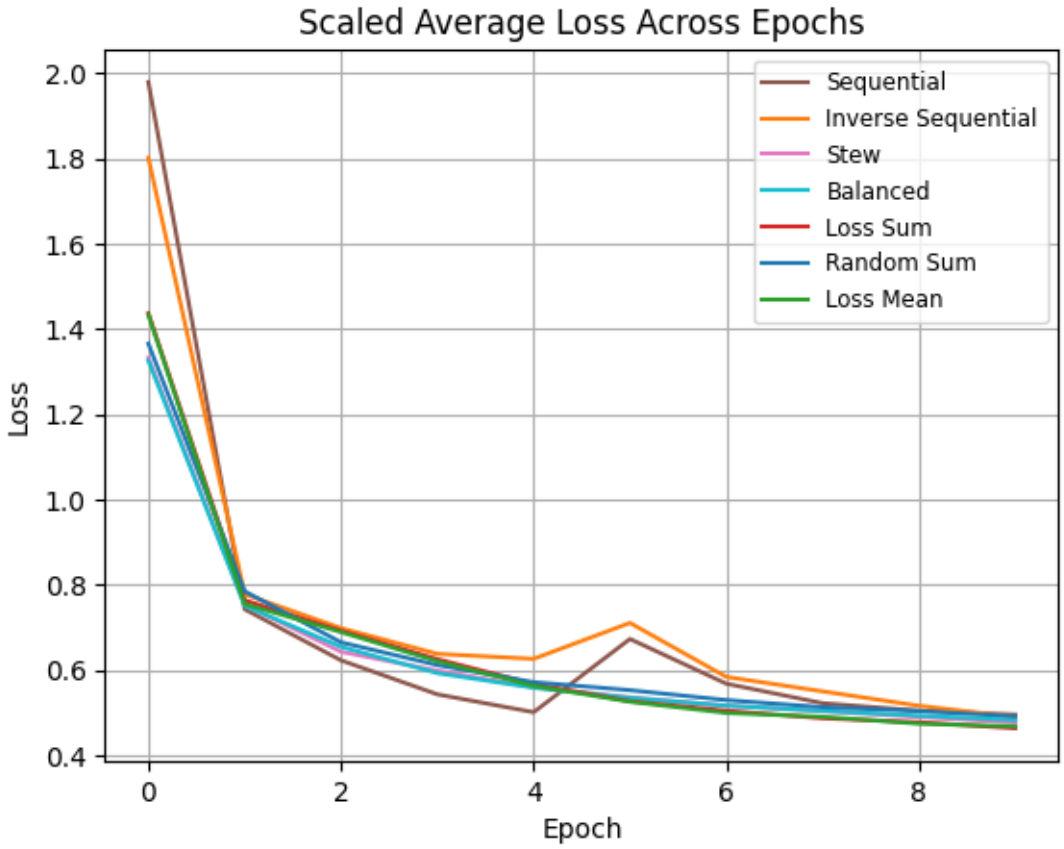

Figure 35 – Average domain loss across epochs. Methods that operate on a higher loss were normalized for comparison purposes (this normalization consists of dividing the loss by the number of domains). Source: Author.

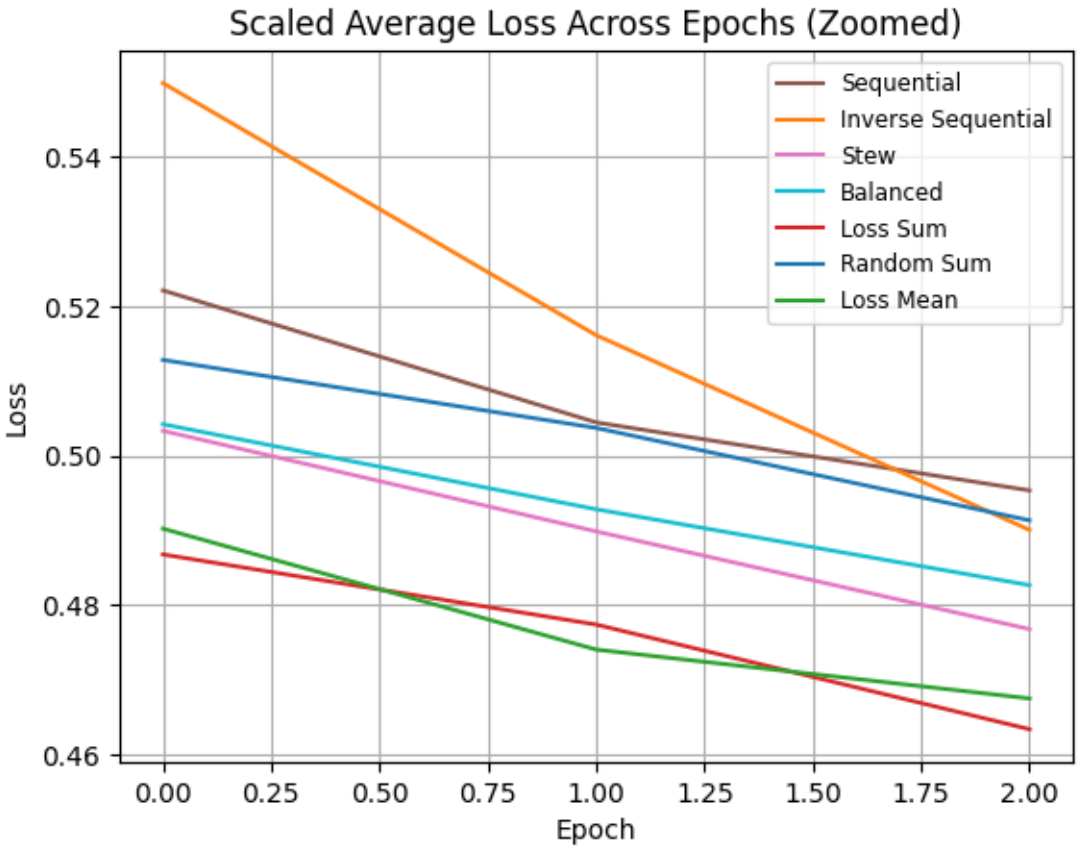

Figure 36 – Average normalized domain loss for the last 3 epochs. Source: Author.