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

Dissertação

Evaluating Machine Learning Methodologies for Multi-Domain Learning in Image Classification

Ihan Belmonte Bender

Pelotas, 2022

Ihan Belmonte Bender

Evaluating Machine Learning Methodologies for Multi-Domain Learning in Image Classification

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 de Araújo

Pelotas, 2022

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

B458e Bender, Ihan Belmonte

Evaluating machine learning methodologies for multi-domain learning in image classification / Ihan Belmonte Bender ; Ricardo Matsumura de Araújo, orientador. — Pelotas, 2022.

53 f.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2022.

1. Aprendizado de máquina. 2. Aprendizado de multiplos domínios. 3. Visão computacional. 4. Inteligência artificial. I. Araújo, Ricardo Matsumura de, orient. II. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Ihan Belmonte Bender

Evaluating Machine Learning Methodologies for Multi-Domain Learning in Image Classification

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: 6 de abril de 2022

AGRADECIMENTOS

Agradeço a todos que de alguma forma fizeram parte da minha caminhada.

O prazer mais nobre é o júbilo de compreender.
— LEONARDO DA VINCI

ABSTRACT

BENDER, Ihan Belmonte. **Evaluating Machine Learning Methodologies for Multi-Domain Learning in Image Classification** . Advisor: Ricardo Matsumura de Araújo. 2022. 53 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2022.

When training machine learning models, it is usually desired that the model learns to execute a specific task. This is commonly achieved by exposing this agent to data related to the task that should be learned. It is also expected that the model is going to be evaluated or used in real world applications receiving as input data samples that are similar to the ones used during training, like images taken from similar devices, therefore having similar features, which we call data domains or data sources. However, there are some cases in which we expect a model to properly perform a task in multiple different domains at the same time, being able to classify images from high definition pictures of objects as well as drawings of the same objects, for example. We propose and evaluate two novel techniques to train a single model to perform well on multiple domains at the same time, for a single task. One of the proposed techniques, we call Loss Sum, was able to achieve good performance when evaluated on different domains, both to domains already seen on training (multi-domain learning) and never seen before domains (domain-generalization).

Keywords: Machine Learning. Multi-domain Learning. Computer Vision. Artificial Intelligence.

RESUMO

BENDER, Ihan Belmonte. **Evaluating Machine Learning Methodologies for Multi-Domain Learning in Image Classification**. Orientador: Ricardo Matsumura de Araújo. 2022. 53 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2022.

Quando se treina um modelo utilizando técnicas de aprendizado de máquina, é comum que se deseje que este modelo aprenda a executar uma tarefa especifica. Normalmente isso é alcançado ao expor o modelo, ou agente, a dados relacionados à tarefa que deveria aprender. Também se espera que o modelo seja avaliado ou utilizado em aplicações recebendo como entrada exemplos de dados que sejam similiares aos dados utilizados durante o treinamento, como imagens obtidas com a utilização de dispositivos similares ou iguais, gerando dados com features semelhantes. A estes dados com características próximas damos o nome de domínio ou fonte. Apesar de normalmente trabalharmos com apenas um domínio no aprendizado de máquina, existem alguns casos onde aprender a realizar a tarefa em mais de um domínio ao mesmo tempo é desejável, como criar um modelo capaz de classificar corretamente imagens tanto em fotos de objetos reais em alta definição quanto em desenhos feitos a mão, por exemplo. Nos propomos e avaliamos dois novos métodos de treinamento de modelos únicos que sejam capazes de ter boa performance em multiplos domínios ao mesmo tempo, para uma mesma tarefa. Uma das técnicas propostas, que chamamos de Soma dos Erros ou *Loss Sum*, foi capaz de alcançar bons resultados quando avaliada em diferentes domínios, tanto os vistos durante o treinamento (aprendizado de múltiplos domínios) quanto os apresentados apenas em etapa de avaliação (generalização de domínios).

Palavras-chave: Aprendizado de Máquina. Aprendizado de Multiplos Domínios. Visão Computacional. Inteligência Artificial.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

MDL	Multi-domain learning
DG	Domain Generalization
GPU	Graphic Processing Unit

CONTENTS

1 INTRODUCTION

In machine learning, one of the most important resources necessary to achieve high performance in learning tasks is data (OVERVIEW AND IMPORTANCE OF DATA QUALITY FOR MACHINE LEARNING TASKS, 2020). It is very difficult to train a model with low quantity or quality of data and still manage to have good results in real world applications. However, images for a training dataset need to be obtained in some way. In images dataset, for instance, data might be collected by taking pictures of objects with a specific camera in a specific environment. Sets of data collected or created under similar conditions are called sources or domains.

A dataset domain can be interpreted as a subset of data in which data samples share a similar set of features. In a labeled dataset of images of dogs and cats, for instance, images might be collected from different sources. Some of them might be obtained by taking high resolution pictures of the animals, while other images might come from sketches drawn by artists. In this situation, we have a single dataset of dogs and cats with two distinct domains or data sources. The first domain is the subset with pictures, while the second is composed by the subset of sketches. Notice that both sources have the same classes (cat and dog) and data type (image), the only difference between them is the data distribution. Figure 1 illustrates the concept.

Most machine learning models are trained using only one source of data, then evaluated in data from the very source. Sometimes, when it is desired to achieve good performance in a domain with few data, models are pre-trained in a different and larger domain and then fine-tuned to the target one. Fine-tuning has been an important tool in the recent years allowing to train machine learning models faster and with more quality in multiple different tasks, specially image classification, while avoiding the overfitting that would probably happen if the model was trained directly in a smaller domain.

However, fine-tuning itself does not solve all of the problems that come from working with datasets with multiple domains. This method is excellent when you have two or more sources of data, but only one domain in which the model is actually going to be tested or used for it is final purpose. When machine learning models that were pre-trained in a source domain are fine-tuned to learn the same task on the target

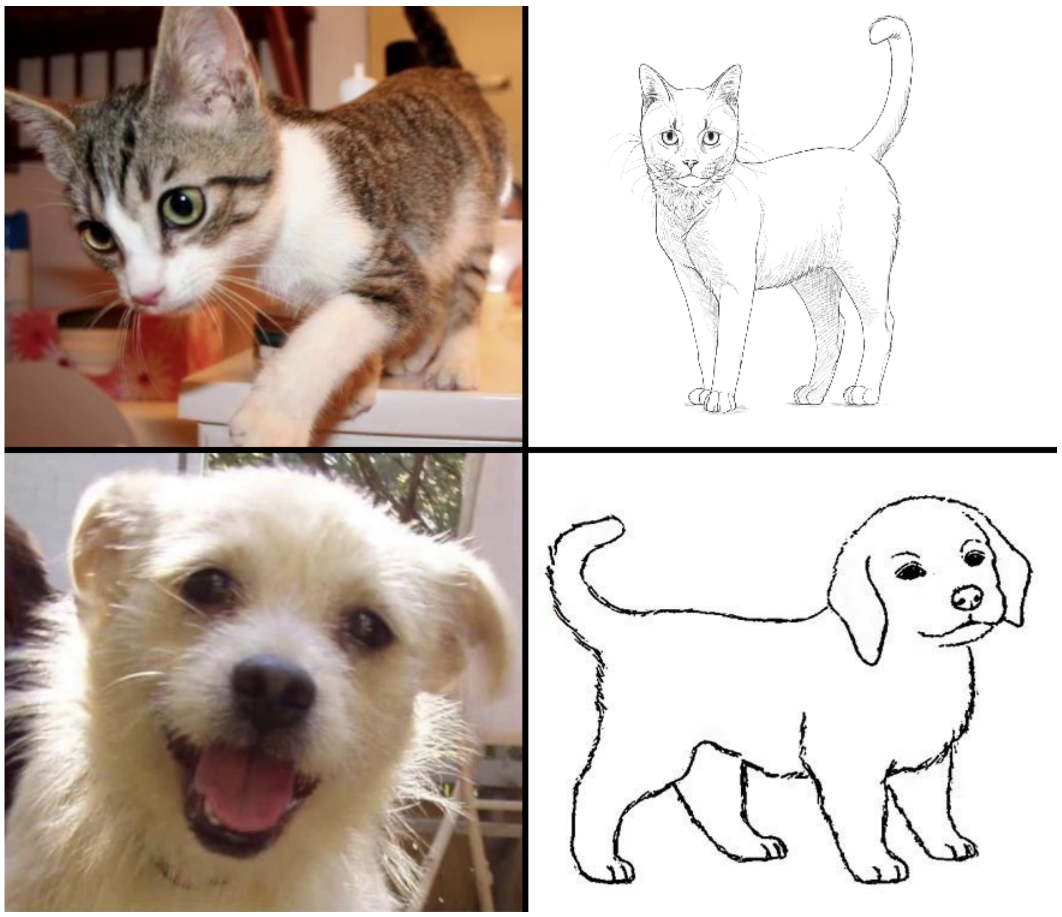
Figure 1 – Example of distinct domains that compose a single dataset. The left side presents images of a cat and a dog taken in a picture, while on the right side images of drawings of the same classes are presented.

dataset, they often suffer from what is called catastrophic forgetting (GOODFELLOW et al., 2015) (KEMKER et al., 2018), where the model's performance in the first domain degrades as it's performance gets better on the new one. Then, what should be done when it is desired to learn all available domains at the same time?

In this work we evaluate different existing and new methodologies to properly train machine learning models using datasets with multiple sources of data on the image classification task. We assume that we can use data from all sources while training, but also test the methodologies behavior when training with some of the domains instead of all of them. Then, we evaluate their performances on both domains seen and not seen during training to evaluate the capacity of both generalizing to new domains and learning multiple known domains at a time. Our study was initially inspired by the desire to understand the best way to learn more than one domain at a time, so that we do not need to keep training different models for different sources of data. In this case, a system that receives images as input but has a model for each different input domain demands not only the image as input, but also the domain it is coming from, adding even more complexity.

We evaluate a naive, but widely used approach that consists of simply mixing all the data from different domains together and training the model normally against two novel proposals. The first one consists of adapting this naive method by also balancing the domains, so that every batch has approximately the same number of samples from every available domain during training. The second proposal consists of a different way to train the network we call Loss Sum, which consists of creating batches balanced by domain, but instead of calculating the loss over the whole batch we calculate the loss for each separate domain, them sum the value for each domain and backpropagate the loss.

We were able to notice that both proposed models present improvements on multi-domain learning (MDL) and domain generalization (DG) tasks when compared to the naive approach. DG nice results were a good surprise, since the focus of the designed methods was to deal with MDL only. While just balancing the domains presented a small but considerable overall improvement on both tasks, Loss Sum was able to outperform the other approaches in the large majority of experiments when training with two or more domains.

2 RELATED WORK

There are multiple sub-fields of research inside machine learning whose objective is to find the best ways to deal with learning multiple domains with the same model. Domain generalization has the objective to make a model perform well on a domain not seen on training data (GULRAJANI; LOPEZ-PAZ, 2020), while domain adaptation has a similar task, the only difference is that domain adaptation methods can deal with unlabeled data on the target domain (GHIFARY et al., 2015).

While the last two presented tasks focus on having good performance on a never-seen domain, other fields try to deal with the problem in a different way. In multi-domain learning (or multi-domain training), it is allowed to use data from all domains to train the model (LIU et al., 2019). Multi-domain techniques can be used to either create a single model that can learn and perform well in multiple domains at a time or even take advantage of multiple sources to create a general purpose model that can be fine-tuned to a target domain later with transfer learning (CHAN et al., 2021).

Even though in recent years we have watched an increase in published works on multiple domain related fields in machine learning, such as (THOMPSON et al., 2019) (LAPARRA; BETHARD; MILLER, 2020) (XIE et al., 2018) in domain adaptation and (GULRAJANI; LOPEZ-PAZ, 2020) (ARPIT et al., 2021) (LAPARRA; BETHARD; MILLER, 2020) (XIE et al., 2018) (LI et al., 2018) (LI et al., 2017) in domain generalization, not much work has been published in MDL. Besides many of the contributions on single-model MDL end up suggesting architectural domain-specific changes (SICILIA et al., 2021) and though these solutions theoretically consist of using a unique model, it is still necessary to create a different architecture for different datasets or if some new domain is added to the original one, making it harder to scale and adding complexity. They usually deal with multiple domains by creating multiple branches within the neural network, one for each domain to be learned, that share the initial part of this network as a feature extractor and domain decider.

One of the main contributions over the last recent years on learning multiple domains with the same model comes from speech recognition. SpeechStew (CHAN et al., 2021) achieves state-of-the-art on speech recognition tasks using different domains to train the same model by simply mixing all of the data together, just like the Naive approach we evaluate in this study. This naive method was not used by the first time with SpeechStew, though. Other previous studies have already trained multi domain models by mixing all the available data (CHOJNACKA et al., 2021) (NARAYANAN et al., 2018) (LIKHOMANENKO et al., 2021), the main difference is that SpeechStew scales to larger models.

3 OBJECTIVES AND METHODOLOGY

This chapter is divided into three sections, where the first is used to present this work objectives. The second section presents the model training methods that are evaluated in our studies and the third section presents the experiments that are used to evaluate those methods, with additional methodological information.

3.1 Objectives

In this master thesis we propose that, when training machine learning models to learn multiple domains at the same time and there is training data on those domains available, the training process takes much advantage from using this data explicitly. Most work so far focus on learning multiple domains using domain generalization methods, where not all domains have training data, leaving a smaller share of the developed work to MDL, the paradigm we approach in our work. In this section we present our general objective and specific objectives.

3.1.1 General Objective

Our general objective in this study is to evaluate different machine learning methods in MDL under the computer vision area. The idea is to understand the best way to use data from multiple domains to train a machine learning model that can properly learn to perform well on evaluation sets for all of those domains, assuming training domains and evaluating domains are the same.

3.1.2 Specific Objectives

1. Propose novel methods to multi-domain learning
2. Compare the proposed methods with the traditional naive method, that does not explicitly considers the domains
3. Evaluate whether explicit domain information usage helps on learning multiple domains.

3.2 Evaluated Methods

We evaluate three distinct methods to train machine learning models to learn more than one domain at the same time on the MDL task. This section presents those three methods, starting by the Naive one, than Balanced Domains and finally Loss Sum.

3.2.1 Naive

The naive method is the most straightforward technique to train one model using data from multiple sources at the same time. The method consists of simply mixing all available data together, without any kind of special processing. This method is already used on the multi-domain training task specially in fields such as speech recognition (CHOJNACKA et al., 2021)(NARAYANAN et al., 2018)(LIKHOMANENKO et al., 2021)(CHAN et al., 2021), but it was not used as frequently on image classification tasks.

We understand that one of the main problems that might come from using this naive approach in image classification is that historically this task has presented some considerable improvements from simple data processing methods, such as dealing with unbalanced labels, for example. In our experiments we decide to make this one change to the naive method and attribute a weight to all dataset entries based on their labels, so that every batch we generate and present to the model has approximately the same amount of data from each class. Another pre-processing we apply that is worth mentioning is image resizing which we use because the ResNet50 architecture, used in all experiments, expects inputs with a specific size.

In this method, every batch consists of 300 images from the dataset that results from mixing all data from all available domains. The size of the batch is related to the hardware we use to train the models and will be approached with more detail later. We shuffle this dataset before we start training as a good practice to avoid one domain being presented to the model at a time. In this specific case, since we select from the dataset randomly with higher weight to minority classes, shuffling should not be really essential, but we keep this step anyway. We run backpropagation after each batch is presented to the model.

Figure 2 presents the flow for all evaluated methods in our study and the naive approach is represented on the left. Notice that, as already mentioned, the batches in this method will usually have more samples from the majoritarian domains, since each batch is sampled randomly from the mixed dataset and all samples have the same chance of being selected. Of course, there is still the label weight, that gives the samples from minority classes a higher probability of being selected, but that does not address domain-level selection.

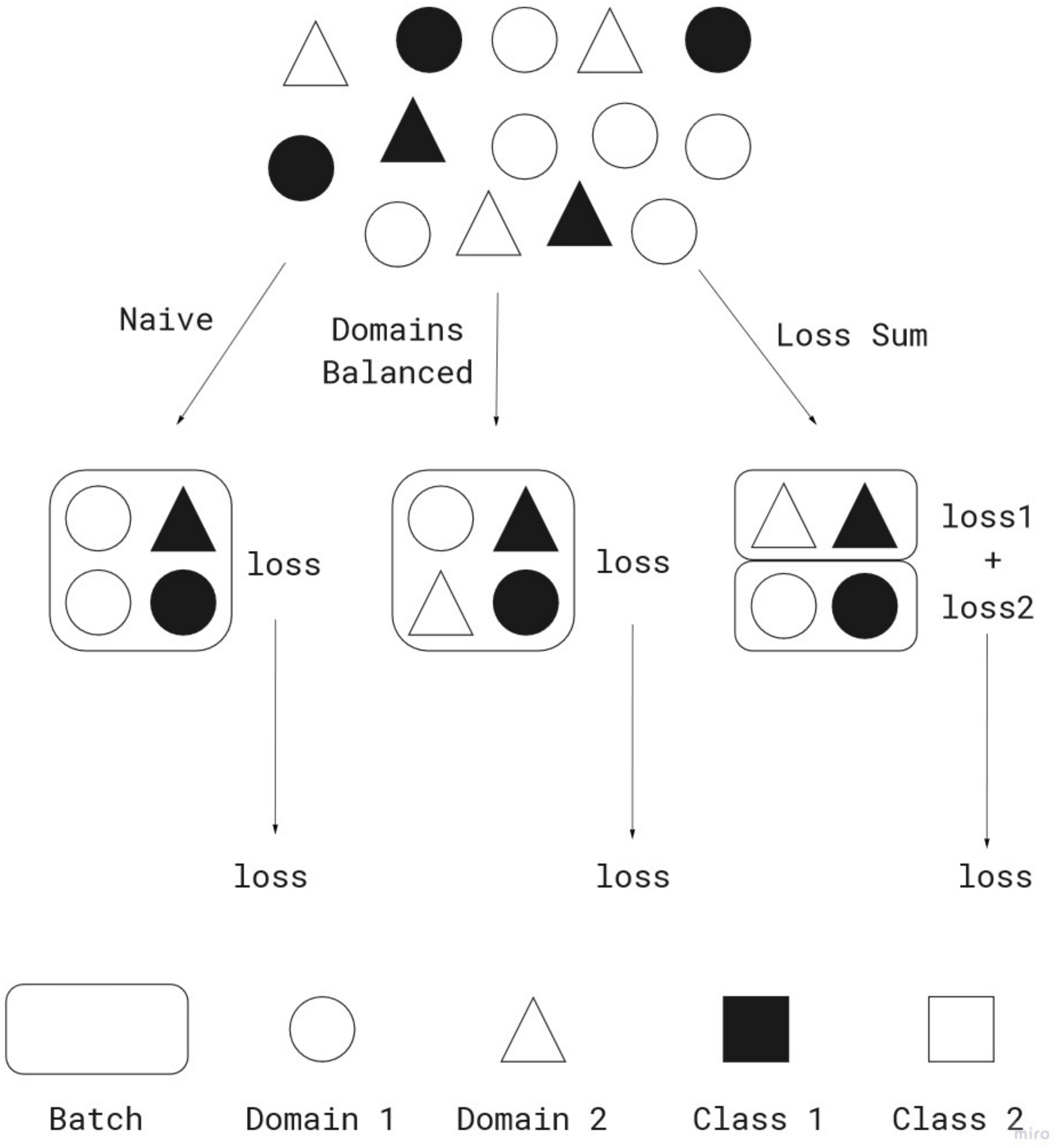

Figure 2 – Evaluated methods illustrated. On the left the naive method is shown, sampling randomly from the mixed datasets without any kind of balancing between domains, usually having batches where the major domain has more examples. The middle flow describes Balanced Domains, where we also sample from the mixed domains dataset, but trying to always have similar amounts from 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 summed is equal to the number of samples in previous methods batches. In all cases, classes keep balanced across the batches, even when domains are not and even though the complete dataset and domains are not naturally balanced.

3.2.2 Balanced Domains

Even though the naive approach is very simple, it is actually a pretty good method to train multi-domain networks in various tasks. For that reason, we decided to keep the naive approach core and add a few changes to create a new method. We understand that, even though the naive approach works pretty well across multiple tasks when training with more than one domain, when it comes to multi-domain training in image classification the simplicity of the naive approach might bring problems.

Unlike speech recognition, where label balancing is not usually an important step in data processing, to properly avoid overfitting in tasks such as image classification it is usually a good idea to deal with this aspect of data, specially when there are few classes and high unbalance in labels between them. We already address this specific matter of label balancing on the naive method, but not for domains. Maybe we should we expect that the trained neural networks would probably overfit to the majority domain just like they do to the majority classes when there is no balancing. We aim to perform well in all of the available and trained domains, and if there are domains with much more data, we believe that the model might get biased towards these domains while losing performance on the ones with smaller share of data. Trying to minimize this problem, we came up with a variation to the naive method we decided to call Balanced Domains.

Balanced domains consists of training the model just like the naive method, but instead of sampling from a unique training dataset, we sample from the available domains separately. We still use batches with the same size as before (300), but now every domain has approximately the same number of samples in each batch, so that the model always deals with all domains equally, or at least with much similar quantities of data. We also keep sampling from the domains in the same way as the naive approach when it comes to labeling, so that the labels are still balanced.

Then, in this case, the training step consists of sampling from all available domains, mixing the sampled data into a single batch and backpropagating the loss from this same batch. The number of batches that are used in each epochs depends on the amount of entries in the largest domain, so in order to keep all Balanced Domains by domain, some of the smaller domains entries will be shown more than once to the model through one epoch.

Again, looking at Figure 2 might help understanding this method. The flow presented in the middle shows how a batch is built in Balanced Domains approach. Notice that, even though the dataset composed from all domains is highly unbalanced towards domain 1, the selected batch ends up having the same number of samples for each domain.

3.2.3 Loss Sum

Even though balancing the domains have the potential of penalizing the model when it does not generalize to all domains seen on training, we go a little further on this penalty. To make the model less likely to undervalue smaller domains, we propose a novel technique called Loss Sum.

In Loss Sum, instead of simply calculating the loss from a batch that has similar amounts of data from each domain, we calculate the losses separately from each source of data and then backpropagate the loss that results from the sum of domains losses. The idea is that, while Balanced Domains approach averages the loss in all domains with equal or almost equal weights, Loss Sum sums them in order to make the overall loss bigger, also penalizing even more the model when having bad results in one domain.

We expect that in some cases, when training models with balanced domains technique, the model might still prioritize one or some domains that have more similar features, for example, since performing badly in only one of the domains might still lead to small losses, specially in cases where there are many domains and the loss of one of them is not enough to make a big difference. Calculating the losses individually, on the other hand, will penalize the model for not improving it's performance in all domains, since having bad performance in one domain leads to a loss at least as high as the worst domain.

Another perk of using loss sum comes from debugging model training. When it is necessary to perform well in multiple domains in real world applications, it might be desired to understand the loss curve of the model to understand how was the learning process. With loss sum, it is not only possible to have the overall loss curve, but it is also easier to analyze each domain's learning curve individually through the epochs with no extra computing necessary.

The way we train models with Loss Sum has many similarities to the way we do on Balanced Domains, from data selection to domain and label sampling. The only difference is that instead of mixing the samples from domains we calculate the losses separately, presenting one domain at a time to the model. Notice that we do not run backpropagation for each domain separately, but instead we gather the different losses and backpropagate their summed value.

It is also worth mentioning that since we present each domain separately and the total batch size stays the same (300), we are actually running more processing steps when using Loss Sum. Consider a dataset that has four available domains, in this case, for each batch we have four smaller batches (one for each domain) with size 75 that are presented sequentially to the model. Since we use GPUs to train our model, that means we are losing on parallelism and therefore taking more time to train the model. We still believe that it is still worth it, but future optimization on the way we calculate the gradients might be welcome in the future. In fact, we could already address this matter by generating batches of size 300 for each domain, taking more advantage of hardware, but we decided that the chosen way was better for a fair comparision between methods. Differences in both data selection and batch size might be observed in Figure 2, at the flow in the right, where it is shown that the domain batches have the same size and the sum of their sizes is 300, just like previous methods.

3.3 Experiments

To evaluate the three proposed methods, we ran several experiments across three different multi-domain datasets. Initially the objective was to evaluate how well those methods can perform MDL, but since we had two available datasets with more than two domains, we also evaluated them on the DG task. We need more than two domains to evaluate the models on this task because it only makes sense to train models with those methods if there are at least two domains available for training, since the training methodology itself only varies from regular training in the way multiple domains are presented to the model across batches. Since we need at least two domains to train the model and DG consists of evaluating on a never seen domain, the smallest number of domains we need is three.

In this study, we consider an experiment to be the training of a model under certain conditions. We first choose one of the three model training methods, then the dataset that is going to be used for training. After that, we define which domains are going to be used for training, while the evaluation is made using all the domains. The reason we evaluate the model in all domains is to have results on both MDL (for the domains seen while training) and DG (for those that are not used for training). The only case where there is no result for DG is when we train with all available domains, generating results only for MDL due to all evaluated domains having already been used to train. We also run experiments training with one domain only and use the results as a baseline to understand how the model deals with multiple domains, specially if performance is reduced when introducing new sources of data.

Performance is calculated using F1-Score. We choose this metric because we believe that it better summarizes the desired result, which is to learn all domains at the same time and being able to learn all classes. Since in many domains of the three datasets the labels are considerably unbalanced, metrics such as accuracy might be misleading when analyzing final results. This might happen if a model always predicts the majority class for a specific dataset, therefore obtaining relatively high accuracy, but not having learned the task as expected. Label unbalancing will be approached in the next chapter: Datasets.

We run experiments for all the possible combinations of method, dataset and training domains across the three datasets and three methods once, using the same architecture and hyperparameter configurations. Besides the already presented configurations of the methods, we also evaluated a few variations of those, like not using label balancing, shuffling each batch before presenting it to the model and we also evaluated a version of Loss Sum without balancing the domains. Since those method variations are independent of each other, we tested all possible combinations of them across methods, generating many more results. In order to summarize the final results we decided to remove those variations because they did not bring any noticeable improvements, and in fact sometimes they made results worse.

The model architecture we use is the ResNet50 (HE et al., 2015) pre-trained on the ImageNet dataset (DENG et al., 2009) and we fine-tune the model in all of our experiments with 50 epochs, with batch sizes of 300, and we use Stochastic Gradient Descent (SGD) optimizer with 0.01 as the learning rate and cross entropy loss. We use the PyTorch (PASZKE et al., 2019) as the library to train and evaluate the models with the Python programming language. Training the model with pytorch includes obtaining the pre-trained ResNet50 and we also use their implementation of optimizer and loss. Also, we use their dataloaders to create the balanced batches on both domains and labels when training with all proposed methods. In total, we run 300 hundred unique experiments, training one model for each experiment during more than 150 hours training with one Nvidia Geforce GTX Titan XP 12Gb.

One of the main reasons for the choice of 300 as the batch size was to take full advantage of the GPU memory and running experiments faster. Also, the low number of epochs we use on the experiments was chosen to make it possible to run as most unique experiments as possible, allowing to search for different configurations. Since we have fewer epochs, we chose to use a learning rate that is much higher than usual so that the model is able to converge fast enough to present interesting results. Prioritizing to test different methodological configurations instead of finding the best hyperparameters for specific configurations is also the reason we do not use any kind of hyperparameter tuning. Instead, we decide to use a fixed set of hyperparameters for all experiments.

4 DATASETS

As mentioned, we run experiment across multiple datasets using different combinations of their domains in the training dataset. In this section we bring more information on each of the three datasets used in our studies.

4.1 Apple Leaves

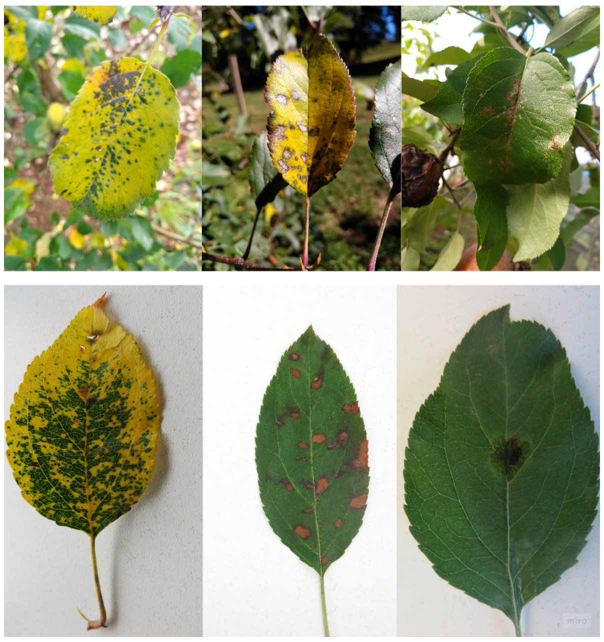

Figure 3 – Image examples from apple leaves dataset containing examples from the three classes: *Desfolha*, *Glomerella* and *Sarna* (left to right). Above, images from the Wild domain, taken outdoors, below images from the Lab domain, which are taken in a controlled environment.

The first dataset is called Apple Leaves and consists of images of diseased apple tree leaves. There are three different classes of diseases: *Glomerella*, *Desfolha* and *Sarna* and two different data domains. The first domain consists of images taken inside a laboratory, with only the diseased leaf and an empty background. These images have controlled light levels and high contrast with the background. This domain is called "Lab" in this paper, for short.

The other domain consists of images from the same kinds of leaves, with the same diseases, but instead of taken under controlled environmental variables inside a laboratory, the pictures are taken directly on the fields and in most images the leaf is still attached to the tree. For that reason, the background of the domain's images usually have colors that are pretty similar to the leaf itself. The pictures have a slightly blurred background, which highlights the object of interest, but there is still a notable difference between domains and the second one might be much more difficult to deal with. We call this domain "Wild" and the difference between the domains can be observed in Figure 3.

This dataset was not only chosen to be part of our studies, but also inspired this study from the start. It consists of a small part of another dataset which contains many more classes of leaf diseases and also different apple fruit diseases. In (GARCIA NACHTIGALL; ARAUJO; NACHTIGALL, 2017) and (BALLESTER et al., 2017) the full dataset was already used.

While working on image classification with the full dataset, it was noticed that regularly the less represented domain would have worse performance when training with all the data together, while the largest one kept having good results. This behavior is acceptable if the model is performing well on the domain from which images will be extracted when the model is actually being used, but if it is desired to perform well on the other domain, it is not a good result. Of course, one can always fine-tune the model that performs well on the larger domain to transfer it's learning to the smaller one, but that only transfers the problem from one side to the other.

4.2 Office-31

The second dataset is Office-31 (or simply Office) (SAENKO et al., 2010), which consists of pictures of office related objects. There are three domains with 31 classes, being the domains Amazon (2,817 images), DSLR (498 images) and Webcam (795 images). The office dataset was chosen in our experiments because it is widely used in studies that deal with training machine learning models in multi-domain datasets, specially in domain adaptation (XU et al., 2021) (NA et al., 2021) (KANG et al., 2019) (XU et al., 2019). Even though domain adaption is a considerably different task from multi-domain learning, the dataset is good in both cases, since it has images from different sources but with the same classes.

Office dataset presents some challenges when trying to perform well in all domains using a single model. The first problem addresses to the sizes of the domains, since there is a considerable difference between the number of examples across the three available domains. The largest domain, Amazon, has more examples that the other two domains together. In fact, the Amazon domain has more than twice the examples of other domains together.

This huge unbalance in the data sources might lead to bias towards the majority domain, specially if we decide to train the model with a naive approach, where the model is presented to more examples of one of the domains. In order to achieve a lower loss, the model might start prioritizing learning specific features to the Amazon domain, for example. On Figure 4 it is possible to observe some examples of images from the same classes on the three datasets. The difference between domains is not huge, but is still considerable. Notice that the amazon dataset images (top row) have white background, while other images do not.

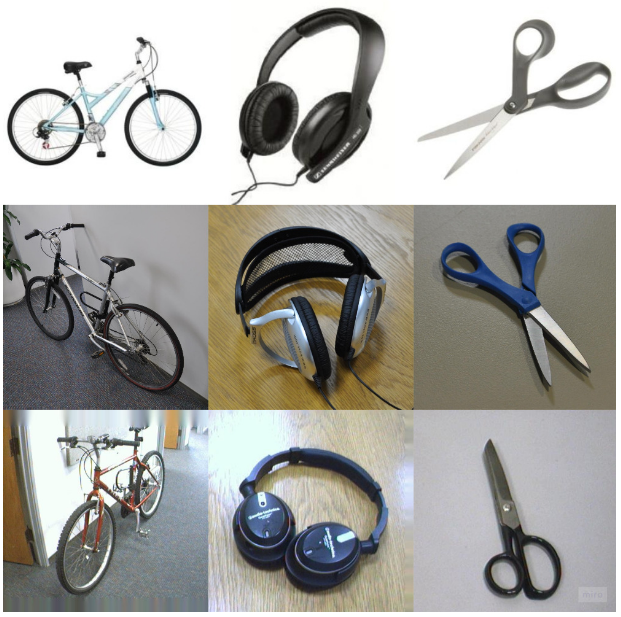

Figure 4 – Examples from the Office-31 dataset. Each line presents examples of the classes bike, headphone and scissors for a domain. The domains are Amazon, DSLR and Webcam, from the top to the bottom.

Another problem is the quality of images. Although Webcam dataset has more images than DSLR, the quality of the images are really different. While Webcam's images have low resolution (640×480), DSRL dataset contains low-noise high resolution images (4288×2848), which might make it easier to the model to distinguish between classes, therefore treating the domains differently and learning more from one than the other.

Also, we can notice a large difference in samples from the different labels. Office-31 has 31 different classes, which is a considerable number of classes to learn simultaneously, and the problem is even bigger due to the difference in the number of examples on the classes. On the amazon dataset, for example, the "bottle" class has 19 training images after our data splitting, while the "mobile phone" class has 64, more than three times the amount on the previous one. This difference reinforces the need of label balancing treatment we use when evaluating all the methods.

The last issue we found when analyzing the dataset was that there were many repeated images in all domains and also multiple pictures of the same object on the minority domains (DLSR and Webcam). We manually removed the repeated ones and created groups to identify which images consisted of the same objects. With that new information we were able to properly split the dataset into train, validation and test without worrying about data leak.

4.3 VLCS

The third used dataset is called VLCS, that consists of 10,729 images for 5 classes (bird, car, chair, dog, person) distributed across 4 domains (VOC2007, LabelMe, Caltech, SUN09). VLCS dataset is one of the main framework datasets for the domain generalization task and is used in works such as (ARPIT et al., 2021) (THOMAS et al., 2021) (CHA et al., 2021). Besides being widely used in a task that is similar to multi domain learning, another reason of the choice to use VLCS is the large number of domains with a considerable quantity of data, where all domains have samples from all existing classes.

In this dataset none of the domains present much more data than the others and for that reason learning all of them with similar priority should not be a big problem. However, when it comes to label unbalancing, all domains have serious unbalancing.

Table 1 – Number of examples for each class in VLCS domains.

	Caltech	LabelMe	VOC 2007	SUN09
Bird	166	56	231	14
Car	86	846	489	652
Chair	83	62	300	725
Dog	47	29	294	21
Person	609	866	1049	885

Table 1 shows the number of samples available in each VLCS domain and it is clear that in all domains there are classes that have a much bigger share when compared to others. The "person" label, for example, has the most samples across all domains, while "dog" is the least represented class in two domain and the second least represented on the other domains, ahead only of the "bird" class that is also very underrepresented. Similarly to the Office dataset, label unbalancing should be treated to avoid label bias, reinforcing the need of our label balancing strategy.

As for the difference between domains, VLCS seems to have an even larger distance between images of the same class in different domains, as seen on Figure 5. While the Caltech domain usually presents images where the object of interest (the object that assigns the image to that class, like a bird or chair) is centralized and focused, other domains have many images where the object of interest is a smaller part

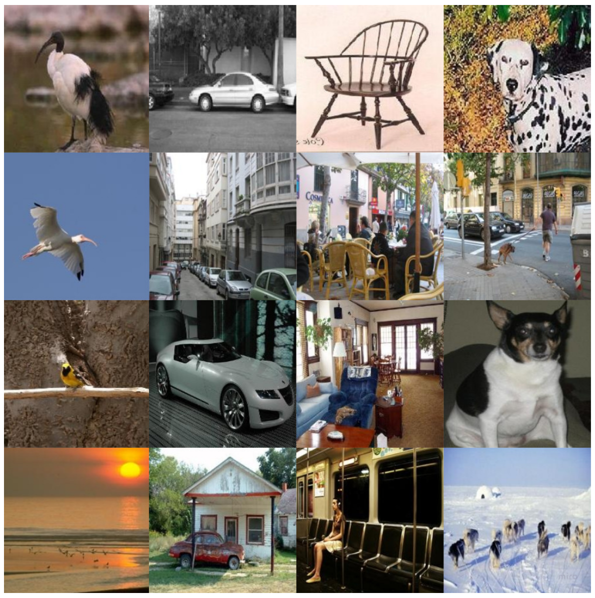

Figure 5 – VLCS example from four classes: Bird, Car, Chair and Dog (columns). Domains are represented by lines, from the top to the bottom the domains are Caltech, LABELME, VOC and SUN

of the image or is even almost not visible, specially on LabelMe and Sun09. The VOC domain is a middle ground between Caltech and the other domains in this matter. Even though most images have a centralized and highlighted image, there are still many images that have several other objects that are even bigger than the main one in some cases.

5 RESULTS

After running experiments across all domain combinations in all three datasets evaluating the three proposed methods, we obtain several results on their performance in both multi-domain learning and domain generalization tasks. In this chapter we present those results, first individually in each dataset and then in a summarized way at the end. We start evaluating performance in both tasks on Office-31 and then VLCS. The Apple Leaves dataset is the last individual presented and with only the multi-domain learning task, due to lack of enough available domains for domain generalization with our methods. It is worth mentioning that many results were removed from this section, as already briefly explained on the methodologies chapter, experiments session.

5.1 Office-31

This dataset section is split into four parts. On the first three, we start by presenting results on the baselines, obtained from training the model in one domain at a time than evaluating in all domains. Then, we present the results obtained on the MDL task from both training in two domains and training in all three. When evaluating MDL on two domains, we exclude non-seen domains from the evaluation. Finally, we present results obtained for the DG task, which consists of performance on not seen domains when training on two domains only (therefore, evaluating using the remaining one). The fourth part consists of a quick analysis on general results obtained from this dataset composed from the previous ones.

An important disclaimer for the Office-31 dataset addresses the learning of all classes on this dataset. As mentioned in the methodology chapter, under experiments, we use F1-Score as the performance metric to properly evaluate our models on an unbalanced classes environment, caused by many of the domains across datasets. On the Office-31 dataset, generalizing to all classes is pretty difficult, at least when compared to the other used datasets due to the number of classes. While VLCS has five and Apple Leaves three, Office-31 has thirty-one different classes.

In all of our experiments on Office-31, none of the trained model was able to have more than zero percent accuracy in all classes when evaluating. That means that in all cases, at least one of the classes was not learned at all. This might happen due to the fewer epochs associated with lower learning rate, as well as having few examples of those classes, but the fact that we use F1-Score as evaluation metric already addresses this problem and comparing the models on our experiments is still possible and fair. The model with the best F1-Score, for example, was also the one with less non-learned classes, having zero percent accuracy in three of the thirty-one classes. This model is presented under the MDL results session and we could see that this problem was reduced as we added more domains and therefore more examples to training.

5.1.1 Baselines

Table 2 presents the F1-Score for when evaluating on each domain, for each training domain. Notice that in most cases the best result is obtained when evaluated on the same domain the model was trained on, as expected, but when training on the Webcam domain the best result was actually on the DLSR domain by little. This result might occur due to similarity between domains, which might be observed on the results of models trained only on DLSR. In this case, the model performs better on Webcam than the Amazon dataset. Also, when analyzing the results of the model trained on the Amazon dataset, it's the performance on both DSLR and Webcam are really similar.

Even though training on Webcam leads to better performance on DLSR, the opposite is not true. We believe that this result happens due to the Webcam domain having better quality data, which would lead to better generalization overall. This thought is reinforced when we look at the Macro-F1 column, which stands for the mean of each line. In this column the Amazon domain has the best results overall, followed by Webcam and then DLSR. As mentioned on the Datasets chapter, Amazon is the domain with larger quantity and variety of data. It is also interesting to notice that the models trained on the Amazon domain perform much better than the others in their own training domains, while Webcam performs better than DSLR in the same way. This shows once again how important the quality and quantity of data is important in those cases.

Table 2 – F1 Score on models trained in each domain. Scores in bold represent those obtained from training and evaluating the model on the same domain. Macro F1 column has the mean between scores obtained from training in each domain.

Train Domain	Evaluation Domain			Macro F1
	Amazon	DSLR	Webcam	
Amazon	**0.120**	0.112	0.111	0.114
DSLR	0.021	**0.033**	0.027	0.027
Webcam	0.034	0.054	**0.053**	0.047

5.1.2 Multi-domain Learning

In this dataset, MDL results are obtained from four experiments for each of the three methods. Three of them consist of training in two domains (and evaluating on the same two domains) while the last experiment comes from training in all three domains and also evaluating the trained model with all domains. For each method we present a different table with MDL results, to make comprehension of individual results easier.

Table 3 – F1-Score for the **Naive** method when training with two or three domains at the same time for the multi-domain learning task. Cells where results for domains not seen on training are blank due to the scope of the evaluated task. Those results will be presented in the next session for domain generalization.

Training Domains	Domain Evaluated		
	Amazon	DSLR	Webcam
DSLR, Webcam	-	0.062	0.077
Amazon, Webcam	0.097	-	0.092
Amazon, DSLR	0.124	0.152	-
Amazon, DSLR, Webcam	0.132	0.177	0.170

Table 4 – F1-Score for the **Balanced Domains** method when training with two or three domains at the same time for the multi-domain learning task. It is possible to notice that the difference between using or not Amazon domain grows in this case.

Training Domains	Domain Evaluated		
	Amazon	DSLR	Webcam
DSLR, Webcam	-	0.045	0.019
Amazon, Webcam	0.279	-	0.297
Amazon, DSLR	0.233	0.233	-
Amazon, DSLR, Webcam	0.353	0.452	0.426

Table 5 – F1-Score for the **Loss Sum** method when training with two or three domains at the same time for the multi-domain learning task. Using Loss Sum brings much better results in all cases when compared to previous methods

Training Domains	Domain Evaluated		
	Amazon	DSLR	Webcam
DSLR, Webcam	-	0.156	0.173
Amazon, Webcam	0.422	-	0.424
Amazon, DSLR	0.379	0.406	-
Amazon, DSLR, Webcam	0.735	0.830	0.764

Tables 3, 4 and 5 present the F1-Scores for each method on theMDL task. Notice that, since this task consists of evaluating only on domains seen on training, results for never seen domains when training with only two of them are removed from the analysis.

It is possible to notice that both Naive and Balanced Domains methods have some difficulty, specially when it comes to training without the amazon domain, the larger and most organized of this dataset. In fact, performance on the Webcam domain when using Balanced domains is even worse than the baseline, indicating that DSLR might have introduced noise to the model training deteriorating performance when compared to training with only one domain. Still, in all other cases, introducing new domains to the model training did not reduce performance and actually led to a considerable increase in F1-Score.

When it comes to domain quantity, apparently introducing new domains in this dataset shows good results, specially when amazon is one of them. Training with all domains at the same time has the best results for all domains when compared to any other combination in all training methods as well, which shows how they are adapted to the MDL task, at least with those domains.

Another interesting result is that, when training with all domains, all methods have the best results on DSLR, followed by Webcam and then Amazon. This is curious because DSLR is the domain that less contributes to raising performance when used as training domain, while Amazon is the one that makes the largest difference. So, even though some domains are "easier" to perform, they are not that good to be used on training. It is an interesting idea to check for this behaviour on further results. Still, when looking at Table 6, Amazon has the best average result, but this score comes from the mean between all F1-Scores obtained from evaluating at that domain and since the worse results come from training without amazon, therefore evaluating only on DSLR and Webcam, these two domains end up having worse final scores.

Table 6 – Averaged F1-Score for each method on the Office-31 domains. The average consists of the mean between all multi-domain learning results obtained by each method in each domain.

Method	Domain Evaluated		
	Amazon	DSLR	Webcam
Loss Sum	0.512	0.464	0.454
Balanced Domains	0.288	0.243	0.247
Naive	0.118	0.130	0.113

Comparing the methods between them, Loss Sum presents by far the best results in all cases. Even when training without the Amazon domain, performance is much better than the obtained with other methods and the tendency follows for all other combinations. In fact, when comparing the three methods on training with all available domains leads to the largest difference in F1-Score. While the Naive method reaches 0.177 F1-Score on the DSLR domain as the highest result and Balanced Domains reaches 0.452, in the same domain, the lowest F1-Score for the Loss Sum method is 0.735 on the amazon domain. The best result is also on DSLR, with 0.830. Also, Table

6 which contains a compiled version of the results for each method on the domains helps to directly compare the methods and also makes it easier to see the difference on F1-Score achieved by the methods.

5.1.3 Domain Generalization

The three methods performed relatively well on the MDL task for the Office-31 dataset when compared to the baseline. Also, they were able to take advantage of the insertion of new domains to the training set, which is an important behaviour since the object is to learn as many domains needed at the same time. Now we present results on the domain generalization task. We present once again results for each method and then a table unifying those results to make comparison easier. The format of the tables is really similar to those shown before and we even keep MDL results on the table to help understanding how well models generalize by comparing generalization score with MDL score. The only removed results were of the experiments where all domains were used for training, since they do not generate results for DG.

Table 7 – F1-Score for the **Naive** method when training with two domains. Bold cells represent results on the domain generalization task, while the other results represent multi-domain learning results. Results show that when using amazon domain dataset, domain generalization is much better then when not using it. In fact, domain generalization has better results than learning amazon for multi-domain learning.

Training Domains	Domain Evaluated		
	Amazon	DSLR	Webcam
DSLR, Webcam	**0.050**	0.062	0.077
Amazon, Webcam	0.097	**0.098**	0.092
Amazon, DSLR	0.124	0.152	**0.126**

Table 8 – F1-Score for **Balanced Domains** method. Bold cells represent domain generalization results, the remaining are the same from the multi-domain learning session. Notice how this method actually performed better on never seen domains when using amazon as one of the domains.

Training Domains	Domain Evaluated		
	Amazon	DSLR	Webcam
DSLR, Webcam	**0.025**	0.045	0.019
Amazon, Webcam	0.279	**0.304**	0.297
Amazon, DSLR	0.233	0.233	**0.239**

Tables 7, 8 and 9 present the results for Naive, Balanced Domains and Loss Sum methods, respectively. Results are really similar to the ones observed on the MDL results, being sometimes even better than them. In all methods, training with amazon and webcam domains and evaluating on DLSR resulted on best F1-Score for DG than

Table 9 – F1-Score for **Loss Sum** method. Results in bold represent domain generalization task results. Notice that once again results are best for domain generalization than multi-domain learning when using amazon and webcam as training domains. On the other cases performance on DG and MDL was pretty similar.

Training Domains	Domain Evaluated		
	Amazon	DSLR	Webcam
DSLR, Webcam	**0.125**	0.156	0.173
Amazon, Webcam	0.422	**0.439**	0.424
Amazon, DSLR	0.379	0.406	**0.400**

for MDL. Also, when using Domains Balanced method, training with Amazon and DSLR led to the best result overall being for Webcam, the never-seen domain.

5.1.4 General Results

Even though the methods are designed to deal with the MDL task and have no special treatment for generalizing to other domains as seen on previous work, all of them performed well on DG when compared to MDL results. Of course, it is important to understand if this follows through the other used datasets, since this result might just be a coincidence or even show us that the domains are so similar that not using them for training does not affect performance that much, and what really affects performance is using or not amazon domain, not necessarily because of the domains image quality but because of the quantity of samples.

It is valid to mention as well that once again the best results are obtained by using Loss Sum. Table 10 compares directly results across the training methods. Similarly to MDL results, Loss Sum has the best scores for all classes, followed by Balanced Domains and then Naive method. Even though the naive method outperforms Balanced Domains on amazon domain, it is surpassed by far on the other two domains.

Table 10 – F1-Score for the Office-31 dataset using each training method. Results obtained from training on the remaining domains only.

Method	Domain Evaluated		
	Amazon	DSLR	Webcam
Loss Sum	**0.125**	**0.439**	**0.400**
Balanced Domains	0.025	0.304	0.239
Naive	0.050	0.098	0.126

5.2 VLCS

The section presents results for the VLCS dataset in a similar way to he previous section for Office-31. We also start by presenting the baseline results, then the MDL and finally DG. Unlike Office, which has three domains, VLCS has four, which allows us to generate more results on both MDL and DG. In each section we explain the showed results and how we combine them.

5.2.1 Baselines

Table 11 has the F1-Scores results from the mean of three experiments. In each a model was trained in one domain only but evaluated in all of them, the same way as done with the Office-31 dataset. As expected, the best results are obtained mainly when evaluating on the same domain as trained, except for the case where the model is trained with VOC and evaluated on Caltech. In fact, training with the VOC domain leads to pretty good results on never seen domains when compared to the other domains in VLCS, since when evaluating on both LabelMe and SUN09, training with VOC brings the best F1-Score after training with the same domain.

It is also possible to notice that Caltech seems to be the easier domain to perform well on both cases, while LabelMe presents the worse results overall, which might indicate that this domain is a little harder than the others. The macro-f1 column gives us an indication of the best domains for training models, where we can notice that while VOC has the best results, specially due to it's peformance on the Caltech domain, SUN09 has the worst macro F1-Score as a training domain. So, even though LabelMe has the worst performance when training and evaluating on itself, when generalizing to others it's performance increases, being the second best training domain only after VOC itself.

Table 11 – F1-Score of training only with one domain and evaluating in others for the **VLCS** dataset. Results presented on the table are the mean between the results of three trained models. Macro F1 represents the mean between the results for each training domain.

Train Domain	Evaluation Domain				Macro F1
	VOC	LabelMe	SUN09	Caltech	
VOC	**0.890**	0.478	0.521	0.915	0.701
LabelMe	0.574	**0.803**	0.474	0.671	0.630
SUN09	0.487	0.380	**0.873**	0.553	0.574
Caltech	0.634	0.378	0.430	**0.974**	0.604

5.2.2 Multi-Domain Learning

VLCS dataset contains four domains, which generates twelve different domain combinations with size bigger than one. If we presented the results in tables such as those on the Office-31 MDL results, each of the three tables would have multiple lines, which would make it really difficult to interpret the results. For that reason, we decide to create three different tables, one for each number of training domains. The first one, Table 12 presents results for training with two domains, Table 13 for three and Table 14 four.

For the entirety of results the reader should consult the tables under the Appendix 1: Detailed VLCS Results.

We combine those results by calculating the mean of the F1-Scores obtained by training a model with the given method (first column, left to right) having the given domain (second row, top to bottom) as one of the domains used in training. Table 14 consists of the results from training with all domains, hence not being result of any kind of averaging, but the original results.

Table 12 – Mean of F1-Scores from training with **two domains** and evaluating on the same domains (MDL task).

Training Methods	Domain Evaluated				Macro F1
	VOC	LabelMe	SUN09	Caltech	
Naive	0.899	0.693	0.780	0.971	0.836
Balanced Domains	0.888	0.796	0.845	0.975	0.876
Loss Sum	0.941	0.883	0.912	0.985	0.930

Table 13 – Mean of F1-Scores from training with **three domains** and evaluating on the same domains (MDL task).

Training Methods	Domain Evaluated				Macro F1
	VOC	LabelMe	SUN09	Caltech	
Naive	0.898	0.624	0.736	0.973	0.808
Balanced Domains	0.889	0.784	0.830	0.973	0.869
Loss Sum	0.954	0.924	0.932	0.982	0.948

Table 14 – F1-Score from training and evaluating in **all four domains** for each method.

Training Methods	Domain Evaluated				Macro F1
	VOC	LabelMe	SUN09	Caltech	
Naive	0.899	0.580	0.733	0.980	0.798
Balanced Domains	0.884	0.781	0.814	0.970	0.862
Loss Sum	0.950	0.934	0.951	0.981	0.954

In any quantity of training domains, Loss Sum presents the best results on the macro F1-Score, which consists of the mean between the F1-Score in each separate domain. Balanced domains has the second best score, while the Naive approach has the worst results. Some behaviors are pretty similar to the baselines, like all methods presenting best results for the Caltech domain and the worst for LabelMe but unlike Office, where using more data to train models led to improvement on performance comparing to the baseline, now only Loss Sum method is able to achieve such results.

Naive and Balanced Domains methods results were worse than the baseline, and they also get worse as we add more domains to the training set. Apparently, on this dataset, the noise introduced by the addition of new domains is enough to deteriorate the model's capacity to learn and generalize to all of them. Of course, training with all domains is still generally better than training in only one on the macro F1-Score, but in this case training with multiple domains costs individual domain performance in all methods except Loss Sum.

5.2.3 Domain Generalization

Like in Office-31, we also evaluate the three methods in the Domain Generalization task. Tables 15 and 16 present results for DG with two and three training domains, respectively. While Table 15 results consists of the mean from all results on each domain for DG, Table 16 presents the original results.

Table 15 – F1-Scores for the Domain Generalization task training with **two domains**. Results are obtained by calculating the mean between all possible combinations of training domains that do not use the target domain.

Training Methods	Domain Evaluated				Macro F1
	VOC	LabelMe	SUN09	Caltech	
Naive	0.707	0.469	0.571	0.863	0.653
Balanced Domains	0.681	0.481	0.565	0.876	0.651
Loss Sum	0.708	0.500	0.576	0.879	0.666

Table 16 – F1-Score for Domain Generalization using **three domains** for training.

Training Methods	Domain Evaluated				Macro F1
	VOC	LabelMe	SUN09	Caltech	
Naive	0.730	0.489	0.626	0.870	0.679
Balanced Domains	0.735	0.525	0.608	0.894	0.691
Loss Sum	0.742	0.535	0.657	0.901	0.709

Unlike Office-31, where DG results were really similar to MDL, results on DG for the VLCS dataset are considerably worse than MDL training with two and three domains. Once again, it seems that Loss Sum has a little better performance than the other methods, but when comparing to MDL results it is easy to notice how performance deteriorates by not having previous contact with the target domain. Using three domains seems to lead to better results in all cases, differently from MDL where only Loss Sum took advantage of domain insertion in all levels. It is possible that Office-31 has much closer domains than VLCS and for that reason the methods could perform well on DG then, but not now.

Performance in individual domains was already worse than the baseline in MDL for Naive and Balanced Domains approach, but it is even worse now. Even Loss Sum, that performed pretty well on MDL was not able to transfer the good results to DG.

From this results we understand that domain generalization capacities of our methods is worse than training a new model directly into the new domain, at least for the VLCS dataset.

5.2.4 General Results

Results seem to indicate that VLCS in a much more difficult dataset to transfer knowledge between domains when compared to Office-31. Even though the obtained F1-Score was much better on VLCS than it was in Office for MDL and DG, the correct comparison to understand the methods performance on such tasks should be compared to the baselines and in VLCS domains training directly with the domain only already leads to high scores. This happens probably because there is more data and there are not as many classes in VLCS as there are in Office-31.

The only method that was able to improve performance in all domains by adding new domains to the training set was Loss Sum, and this better performance was obtained only on the MDL task, not on DG. This result should not be surprising, since the method was designed to deal with the first task and not the second, but since results in the Office dataset were really promising, it was expected that the method could perform pretty well on DG too.

5.3 Apple Leaves

The final dataset results we present is for the Apple Leaves dataset. There are much less results generated for this one, since there are only two domains. Also for that reason, we only present results for the baselines and MDL, since our methods need at least three domains to perform DG. We start by presenting baseline results, then MDL.

5.3.1 Baselines

We average three experiments for each domain used as only source in training and evaluate on both of then to understand how well the model performs on these circumstances. Table 17 shows how performance varies when evaluating on the training domain and on the remaining one. While the results presented in bold have F1-scores higher than 0.7, evaluating on a never seen domain presents results lower than 0.6 in both cases. In fact, training and evaluating on the lab domain, whose images are taken under a much more controlled environment, gives us 0.911 score, a really high value. However, training on the same domain and evaluating on Wild leads to the worst result on the table, 0.396. Training on the wild domain leads to a little more balanced set of results, where the difference on evaluation domains is smaller, but still results for the training domain are considerably better.

Those results might indicate that there is a great difference between domains and generalizing for both might be a challenge, since finding shared features might not be easy to the model. Also, generalizing from Wild to Lab domain looks easier, maybe because Lab domain is actually an easier domain since it's images come from controlled environment, being easier to perform well on but harder to transfer knowledge to harder domains. We present results for the three methods on the MDL task on the next session.

Table 17 – F1-Score for training with one domain at a time. Bold results represent results on training and evaluating on the same domain.

	Wild	Lab
Wild	**0.778**	0.585
Lab	0.396	**0.911**

5.3.2 Multi-Domain Learning

In this dataset, MDL task is evaluated on a single table composed by training with both domains and evaluating on them separately. Table 18 presents the obtained results for the three proposed methods. It is possible to see that all methods performed better than the baseline on the Lab domain, which is an interesting result and shows that all of them were able to use Wild to improve performance on that domain. However, similarly to what is seen on VLCS dataset, Naive and Balanced Domains methods end up having worse performance on Wild than the baseline. Of course, Balanced Domains result is really close to the one obtained on the baseline and generalization to the lab dataset was really good, so overall this method performed pretty well but the naive method on the other hand had some considerable performance reduction on the Wild domain.

Table 18 – F1-Score for training with both domains with the three methods. Evaluation is done with each domain separately.

Methods	Domain Evaluated	
	Wild	Lab
Naive	0.615	0.927
Balanced Domains	0.777	0.927
Loss Sum	0.790	0.935

Loss Sum method was able to outperform the other two methods and also the baseline not by much, but was still able to make a single model perform better than two separate models trained only for one domain each. This result is really good, specially when the baselines show us that there is a considerable difference between the domains and it was expected that learning both could be a problem. Unfortunately, we do not have enough domains to evaluate the methods on Domain Generalization, which would probably be a challenge like it was on VLCS due to domain difference.

6 CONCLUSION

In this chapter we bring some discussion on the implication of our results on the methods we propose on MDL and DG. We also discuss about our experiments and the limitations of this work also bring some possible future work to address the limitations and build more knowledge on the proposed methods and on the field in general, specially for MDL.

6.1 Discussion

From the obtained results, we can notice that Loss Sum performed well in all experiments we have proposed for MDL. Of course, in some cases, the difference between the scores from Loss Sum to other methods is not big, but it still performs better in all tasks across all datasets in our study, which leads us to believe it is a considerably better method.

An interesting result we observed was that the naive and balanced domains approaches perform better in some domains under really specific experiments. It is hard to know if this result consists of noise, since we run each experiment only once, but all of the results coming from experiments on the VLCS dataset when training with two of the four domains might be a clue of a deeper understanding of these techniques we can investigate in the future. We do not show those results on the results chapter, since they happen in few of the many results we average, but it is important to mention that they happen and even though Loss Sum performs better overall and in most individual experiments, that does not happen in all of them. In the Appendix 1, Tables 26, 27, 28, 29 show some results where Domains Balanced and Naive approach perform better for Domain Generalization in some domains, while Table 25 shows the only case in which Domains Balanced surpassed Loss Sum performance in multi-domain learning in one of the domains.

We also could notice interesting results on the Domain Generalization task, even though all the proposed methods are designed to deal with the Multi-domain learning task. Of course, on datasets where domains have more differences, like the VLCS dataset, neither of our methods performed as well as the Office, where domains are more similar, but results are still promising and some adaptations to our approaches might lead to new ways of dealing with the problem.

In any case, our results indicate that it is significantly better to use Loss Sum to train models on multiple domains to learn them all at the same time, at least for image classification with the evaluated quantities of domains. We expect that Loss Sum might be able to repeat those good results in many other tasks, both using images or other kind of data and also training with even more domains at the same time. This method might be a really big improvement for many real world applications were there are few sources of data from a single source, but at the same time many sources of data exist and it is desired to learn them all.

Of course, even if we present really good results, our study has limitations regarding number of experiments, datasets and domains, and those performance increases might be obtained from reasons attached to some of the choices in our experiments. In the next session we discuss some limitations of this study and how they might affect results. Later, we propose new experiments for future work that address those points and should help us understand how good are the proposed methods.

6.2 Study Limitations and Future Work

6.2.1 Number of Experiments

Our experiments consist of running one combination of domains and methods only once for few epochs without any kind of hyperparameter tuning. We chose to run experiments under those conditions to be able to cover a large number of variations in our methods. We generated results for twelve unique methods and only three of them presented the most interesting results, and those are the ones that are presented on this study. Even though we were able to test different combinations in our approaches, we were not able to run multiple experiments for the same settings, which reduces our control over noise, for example. Some of the presented results might be generated by chance, and this is one of the main limitations of this study.

Now that we selected the most promising configurations of the methods, it would be important to generate even more results using the same experiment settings, but with randomized starting values like presented batches. It would be interesting to understand if the model achieves good performance frequently when using our methods across the datasets. A simple future work could consist of running experiments in the same conditions we did, but multiple times, and understand how performance varies and properly compare different methods with statistical analysis.

6.2.2 Hyperparameters

Another possible source of misleading results is the number of epochs associated with the high learning rate. In order to train the largest number of unique models across different datasets and methods, we had to use a pretty small number of epochs (50) when compared to other image classification training processes. To make models converge under such a few number of epochs, we decided to use a learning rate of 0.01, which is much larger than most contemporaneous experiments use. We understand that Loss Sum having the best results in most cases might be strongly attached to those choices. Since this methods idea is to make loss even bigger than other methods, training with fewer epochs might favor this method a lot, and it's better performance might only happen because it converged faster than other methods, which did not have the time to achieve it's peak performance. In any case, even if this is the case, Loss Sum might not achieve the best final result, but being able to converge faster is also an interesting feature.

It is important to create new conditions to train those models. Preferentially, training the model with a much larger number of epochs and smaller learning rate. It would also be interesting to perform hyperparameter search to properly understand which hyperparameters are most suited for each method. Also, it is an interesting idea to observe the converging curve for each trained model and check whether Loss Sum really converges faster than other methods and which setting of hyperparameters best takes advantage of this method.

6.2.3 Number of Domains

Also, we run experiments across three datasets, with four, three and two domains. We used those datasets trying to cover many variables on multi-domain datasets, like number of domains and difference between domains. However, we were not able to test the methods using more domains than that. SpeechStew (CHAN et al., 2021), for instance, uses a dataset with seven distinct domains. Loss Sum tries to increase loss by penalizing the model when it performs badly in a domain by summing losses instead of averaging, but it is possible that on larger number of domains this method is not enough to make few domains have considerable impact on loss, so it might not be suited for these kinds of datasets.

Still on domains, we only run tests on the image classification task, therefore using only images as data. Multi-domain learning is an important task on image classification, but it would be important to understand if Loss Sum is really suited to other types of data, like speech, both with audio and written, or general categorical data, for example. In the next session, we address those limitations by proposing future work that might help us learning more about the proposed methods and validating it's quality under other variables.

To deal with those limitations, it is important to expand the usage of this method on other image classification datasets that have more domains and domains with more variety and difference between them. Also, it would be interesting to expand to other fields such as speech recognition and compare our proposed methods with the current used ones.

REFERENCES

ARPIT, D.; WANG, H.; ZHOU, Y.; XIONG, C. Ensemble of Averages: Improving Model Selection and Boosting Performance in Domain Generalization. **CoRR**, [S.l.], v.abs/2110.10832, 2021.

BALLESTER, P. L.; CORRÊA, U. B.; BIRCK, M. A. F.; ARAÚJO, R. M. Assessing the Performance of Convolutional Neural Networks on Classifying Disorders in Apple Tree Leaves. In: LATIN AMERICAN WORKSHOP ON COMPUTATIONAL NEUROSCIENCE, 2017. **Anais. . .** [S.l.: s.n.], 2017.

CHA, J. et al. **SWAD**: Domain Generalization by Seeking Flat Minima.

CHAN, W. et al. **SpeechStew**: Simply Mix All Available Speech Recognition Data to Train One Large Neural Network.

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.

DENG, J. et al. ImageNet: A large-scale hierarchical image database. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2009., 2009. **Anais. . .** [S.l.: s.n.], 2009. p.248–255.

GARCIA NACHTIGALL, L.; ARAUJO, R.; NACHTIGALL, G. Use of Images of Leaves and Fruits of Apple Trees for Automatic Identification of Symptoms of Diseases and Nutritional Disorders. **International Journal of Monitoring and Surveillance Technologies Research**, [S.l.], v.5, p.1–14, 04 2017.

GHIFARY, M.; BALDUZZI, D.; KLEIJN, W. B.; ZHANG, M. Scatter Component Analysis: A Unified Framework for Domain Adaptation and Domain Generalization. **CoRR**, [S.l.], v.abs/1510.04373, 2015.

GOODFELLOW, I. J. et al. **An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks**.

GULRAJANI, I.; LOPEZ-PAZ, D. In Search of Lost Domain Generalization. **CoRR**, [S.l.], v.abs/2007.01434, 2020.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep Residual Learning for Image Recognition. **CoRR**, [S.l.], v.abs/1512.03385, 2015.

KANG, G.; JIANG, L.; YANG, Y.; HAUPTMANN, A. G. **Contrastive Adaptation Network for Unsupervised Domain Adaptation**.

KEMKER, R. et al. Measuring Catastrophic Forgetting in Neural Networks. **Proceedings of the AAAI Conference on Artificial Intelligence**, [S.l.], v.32, n.1, Apr. 2018.

LAPARRA, E.; BETHARD, S.; MILLER, T. A. Rethinking domain adaptation for machine learning over clinical language. **JAMIA Open**, [S.l.], v.3, n.2, p.146–150, 04 2020.

LI, D.; YANG, Y.; SONG, Y.-Z.; HOSPEDALES, T. M. Deeper, Broader and Artier Domain Generalization. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 2017. **Proceedings. . .** [S.l.: s.n.], 2017.

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 (CVPR), 2018. **Proceedings. . .** [S.l.: s.n.], 2018.

LIKHOMANENKO, T. et al. **Rethinking Evaluation in ASR**: Are Our Models Robust Enough?

LIU, Y. et al. Compact Feature Learning for Multi-Domain Image Classification. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2019., 2019. **Anais. . .** [S.l.: s.n.], 2019. p.7186–7194.

NA, J.; JUNG, H.; CHANG, H. J.; HWANG, W. **FixBi**: Bridging Domain Spaces for Unsupervised Domain Adaptation.

NARAYANAN, A. et al. **Toward domain-invariant speech recognition via large scale training**.

JAIN, A. et al. **Overview and Importance of Data Quality for Machine Learning Tasks**. New York, NY, USA: Association for Computing Machinery, 2020. p.3561–3562.

PASZKE, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: WALLACH, H. et al. (Ed.). **Advances in Neural Information Processing Systems 32**. [S.l.]: Curran Associates, Inc., 2019. p.8024–8035.

SAENKO, K.; KULIS, B.; FRITZ, M.; DARRELL, T. Adapting Visual Category Models to New Domains. In: COMPUTER VISION – ECCV 2010, 2010, Berlin, Heidelberg. **Anais. . .** Springer Berlin Heidelberg, 2010. p.213–226.

SICILIA, A. et al. Multi-Domain Learning by Meta-Learning: Taking Optimal Steps in Multi-Domain Loss Landscapes by Inner-Loop Learning. **CoRR**, [S.l.], v.abs/2102.13147, 2021.

THOMAS, X.; MAHAJAN, D.; PENTLAND, A.; DUBEY, A. **Adaptive Methods for Aggregated Domain Generalization**.

THOMPSON, B. et al. Overcoming Catastrophic Forgetting During Domain Adaptation of Neural Machine Translation. In: CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, VOLUME 1 (LONG AND SHORT PAPERS), 2019., 2019, Minneapolis, Minnesota. **Proceedings. . .** Association for Computational Linguistics, 2019. p.2062–2068.

XIE, S.; ZHENG, Z.; CHEN, L.; CHEN, C. Learning Semantic Representations for Unsupervised Domain Adaptation. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 35., 2018. **Proceedings. . .** PMLR, 2018. p.5423–5432. (Proceedings of Machine Learning Research, v.80).

XU, T. et al. **CDTrans**: Cross-domain Transformer for Unsupervised Domain Adaptation.

XU, X. et al. d**-SNE**: Domain Adaptation using Stochastic Neighborhood Embedding.

7 APPENDIX 1: DETAILED VLCS RESULTS

Table 19 – F1-Score using **VOC2007, LabelMe, SUN09 and Caltech** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.899	0.580	0.733	0.980
Balanced Domains	0.884	0.781	0.814	0.970
Loss Sum	0.950	0.934	0.951	0.981

Table 20 – F1-Score using **VOC2007, LabelMe, Caltech** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.906	0.604	0.626	0.975
Balanced Domains	0.886	0.778	0.608	0.963
Loss Sum	0.955	0.915	0.657	0.986

Table 21 – F1-Score using **VOC2007, LabelMe, SUN09** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.898	0.582	0.674	0.870
Balanced Domains	0.885	0.785	0.838	0.894
Loss Sum	0.960	0.924	0.939	0.901

Table 22 – F1-Score using **VOC2007, SUN09, Caltech** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.889	0.489	0.697	0.980
Balanced Domains	0.896	0.525	0.816	0.977
Loss Sum	0.947	0.535	0.921	0.979

Table 23 – F1-Score using **LabelMe, SUN09, Caltech** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.730	0.687	0.837	0.963
Balanced Domains	0.735	0.790	0.837	0.979
Loss Sum	0.742	0.921	0.936	0.980

Table 24 – F1-Score using **VOC2007, LabelMe** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.913	0.632	0.579	0.896
Balanced Domains	0.890	0.777	0.620	0.913
Loss Sum	0.943	0.872	0.655	0.963

Table 25 – F1-Score using **SUN09, Caltech** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.691	0.438	0.805	0.973
Balanced Domains	0.671	0.466	0.840	0.992
Loss Sum	0.695	0.499	0.921	0.984

Table 26 – F1-Score using **LabelMe, Caltech** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.704	0.754	0.549	0.970
Balanced Domains	0.660	0.786	0.529	0.975
Loss Sum	0.699	0.881	0.534	0.986

Table 27 – F1-Score using **LabelMe, SUN09** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.725	0.692	0.839	0.771
Balanced Domains	0.711	0.826	0.860	0.806
Loss Sum	0.729	0.895	0.925	0.760

Table 28 – F1-Score using **VOC2007, SUN09** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.896	0.491	0.695	0.922
Balanced Domains	0.892	0.507	0.836	0.908
Loss Sum	0.943	0.511	0.890	0.915

Table 29 – F1-Score using **VOC2007, Caltech** for training.

Method	Evaluation Domain			
	VOC F1	LabelMe	SUN09	Caltech
Naive	0.887	0.478	0.586	0.971
Balanced Domains	0.882	0.470	0.546	0.959
Loss Sum	0.937	0.490	0.538	0.984