ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

PEDRO LEMOS BALLESTER

SEMI-SUPERVISED LEARNING METHODS FOR UNSUPERVISED DOMAIN ADAPTATION IN MEDICAL IMAGING SEGMENTATION

Porto Alegre
2019

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

SEMI-SUPERVISED LEARNING METHODS FOR UNSUPERVISED DOMAIN ADAPTATION IN MEDICAL IMAGING SEGMENTATION

PEDRO LEMOS BALLESTER

Dissertation submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fullfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Dr. Rodrigo Coelho Barros

Porto Alegre
2019

Ficha Catalográfica

B191s Ballester, Pedro Lemos

Semi-supervised Learning Methods for Unsupervised Domain Adaptation in Medical Imaging Segmentation / Pedro Lemos Ballester . – 2019.

79 p.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rodrigo Coelho Barros.

1. Deep learning. 2. Domain adaptation. 3. Self-ensembling. 4. Semi-supervised learning. I. Barros, Rodrigo Coelho. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS com os dados fornecidos pelo(a) autor(a).

Bibliotecária responsável: Salete Maria Sartori CRB-10/1363

Pedro Lemos Ballester

Semi-supervised Learning Methods for Unsupervised Domain Adaptation in Medical Imaging Segmentation

This Dissertation has been submitted in partial fulfillment of the requirements for the degree of Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on August 2, 2019.

COMMITTEE MEMBERS:

Prof. Dr. Luís da Cunha Lamb (UFRGS)

Prof. Dr. Felipe Rech Meneguzzi (PPGCC/PUCRS)

Prof. Dr. Rodrigo Coelho Barros (PPGCC/PUCRS - Advisor)

I dedicate this work to Davi Lemos Ballester, my friend, my mentor, and, by sheer luck, my brother.

"There is nothing like first-hand evidence."
(Sherlock Holmes)

ACKNOWLEDGMENTS

Minha história na área de Inteligência Artificial começa muito antes de qualquer contato com um computador (ou com um livro, que seja). Inicialmente, agradeço as famílias emprestadas que tive durante os anos. Iria Gaspar e Oracio Arejano; Ilka Simone Pereira e Roberto Pereira; Liliane Braunstein e Osvaldo Cunha; Luciana Peixoto e Roberto Pinelli. No âmbito profissional, não poderia deixar de citar as pessoas que me acolheram em 2013, sem nenhum conhecimento além de fritar um ovo (pensando bem, acho que nem isso), e acreditaram na minha vontade de ser um pesquisador. Felipe Codevilla, Luan Silveira e Silvia Botelho, obrigado pela oportunidade de encontrar a melhor profissão que alguém pode ter. Na UFPel, pude continuar meu trabalho com a orientação de um dos melhores mentores que já encontrei, Ricardo Araujo. Também em Pelotas tive a coorientação de um grande docente, Ulisses Correa; um mentor excepcional que hoje tenho o orgulho de chamar de amigo. Já em Porto Alegre, pude ver de perto o trabalho dos profissionais do Hospital de Clínicas de Porto Alegre, graças ao professor Ives Passos. Também agradeço meu atual orientador, Rodrigo Barros, pela parceria durante todo o mestrado. Aos meus colegas de programa de pós-graduação, Eduardo Pooch e Olimar Borges, sou grato pela ajuda de vocês para a conclusão do meu volume. Aos meus amigos Gabriel Simões, Mauricio Armani e Wanderson Valente, a parceria de vocês foi, e é, essencial. Finalizo agradecendo a minha família. Obrigado Pai e Mãe por terem me proporcionado tudo que precisei e por nunca terem cogitado parar a minha educação, realidade ainda de muitos do nosso país. Vejo isso como um privilégio meu e mérito de vocês em meio a todos os obstáculos que a vida os proporcionou. Obrigado Mana e Cunhado pelo apoio, compreensão e por terem me dado a alegria de ter uma sobrinha maravilhosa, Catarina. Obrigado Mano e Cunhada, vocês foram os que mais tiveram que suportar minhas peripécias e, por motivos de serem pessoas incríveis, se mantém do meu lado até hoje dando todo o apoio. Obrigado aos meus irmãos Caio Peixoto, Fabrício Pereira e Lucas Braunstein. Tanto tempo de amizade que não lembro como era a vida antes de vocês. Fico feliz de compartilhar esta, todas as anteriores, e todas as futuras conquistas com vocês.

MÉTODOS DE APRENDIZADO SEMI-SUPERVISIONADO PARA ADAPTAÇÃO DE DOMÍNIO NÃO-SUPERVISIONADA EM SEGMENTAÇÃO DE IMAGENS MÉDICAS

RESUMO

Aplicações com aprendizado de máquina possuem diversas suposições sobre o cenário em que são colocadas. Uma suposição comum é a de que o ambiente de teste segue a mesma distribuição dos dados de treino. Essa suposição é sistematicamente quebrada em cénarios do mundo real; a diferença entre essas distribuições é conhecida como *domain shift*. Adaptação de domínio não-supervisionada visa mitigar esse problema impulsionando o conhecimento dos modelos com dados do ambiente de teste. Uma das áreas mais sensíveis a *domain shift* é a de imagens médicas. Devido a heterogeneidade das distribuições de dados das máquinas de aquisição de imagens, os modelos tendem a variar sua performance preditiva quando lidam com imagens vindas de máquinas sem nenhum exemplo no conjunto de treino. Este trabalho propõe duas contribuições. Primeiramente, o uso de *self-ensembling* em adaptação de domínio para segmentação de imagens médicas para segmentação de substância cinzenta na medula espinhal é introduzido. Em seguida, baseado no sucesso do *self-ensembling*, outros trabalhos recentes da literatura de aprendizado semi-supervisionado são adaptados para o contexto apresentado, nominalmente, unsupervised data augmentation e MixMatch. Foram conduzidos estudos de ablação e experimentos para compreensão do comportamento dos métodos e comparação dos seus melhores resultados. Os resultados indicam uma melhoria em relação a treinamento puramente supervisionado, além de demonstrar que os métodos recentes de aprendizado semi-supervisionado são promissores para o campo de adaptação de domínio em segmentação de imagens médicas.

Palavras-Chave: Aprendizado profundo, Adaptação de domínio, Self-ensembling, Aprendizado semi-supervisionado.

SEMI-SUPERVISED LEARNING METHODS FOR UNSUPERVISED DOMAIN ADAPTATION IN MEDICAL IMAGING SEGMENTATION

ABSTRACT

Machine learning applications make several assumptions regarding the scenario where they are employed. One common assumption is that data distribution in the test environment follows the same distribution of the training set. This assumption is systematically broken in most real-world scenarios; the difference between these distributions is commonly known as domain shift. Unsupervised domain adaptation aims at suppressing this problem by leveraging knowledge with unlabeled data from the test environment. One of the most sensitive fields for domain shift is medical imaging. Due to the heterogeneity in data distributions from scanners, models tend to vary in predictive performance when dealing with images from scanners with no examples in the training set. We propose two contributions in this work. First, we introduce the use of self-ensembling domain adaptation in the field of medical imaging segmentation in a spinal cord grey matter segmentation task. Next, based on the success of self-ensembling, we adapt two other recent work from the semi-supervised learning literature to the same task, namely, unsupervised data augmentation and MixMatch. We conduct ablation studies and other experiments in order to understand the behavior of each method and compare their best results. The results show improvements over training models in a supervised learning fashion and demonstrate that recent semi-supervised learning methods are promising for domain adaptation in medical imaging segmentation.

Keywords: Deep learning, Domain adaptation, Self-ensembling, Semi-supervised learning.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

CNN – Convolutional Neural Network

CT – Computed Tomography

EMA – Exponential Moving Average

FCN – Fully Convolutional Network

GAN – Generative Adversarial Network

MIOU – Mean Intersection over Union

MSE – Mean-squared Error

MRI – Magnetic Resonance Imaging

PET – Positron Emission Tomography

SCGM – Spinal Cord Grey Matter

CONTENTS

1. INTRODUCTION

Deep learning has advanced the field of artificial intelligence and gained popularity due to its ability to handle unstructured data, such as image and text. Whereas in traditional machine learning the models rely on manual feature engineering for dimensionality reduction, deep learning methods learn both the task and the features that are important for solving the problem simultaneously. This is an important paradigm shift, since handcrafted feature extraction algorithms are cumbersome to design and are usually only useful for a specific domain of application [4]. Handcrafted features are also limited by human knowledge, requiring domain specialists to design them when dealing with highly-complex tasks.

Deep neural network architectures comprise several building blocks — or layers — that are combined sequentially to extract and process information. In deep learning, convolutional layers are arguably the most important feature of an artificial neural network, being the most vital components of Convolutional Neural Networks (CNNs) [40]. Though proposed around 1990 [39], the at the time prohibitive computational cost made CNNs become popular much later [37], when graphics processing units were more powerful and widespread and data became abundant. To this date, convolutional neural networks are the state-of-the-art in a variety of tasks, such as object detection [42, 64] and image classification [87, 29].

Although deep learning is a solid approach to handle unstructured data, there are several drawbacks in letting machines operate in an end-to-end manner. These issues can sometimes be detected through a sound validation protocol, but in other situations remain undetected and present a high risk of undesirable behavior. The process of learning how to extract information from raw data is nontrivial. Ultimately, we must feed the model with enough data to find patterns that can properly solve a task. This means that for very complex scenarios, the amount of necessary data can grow substantially, becoming a limitation when data acquisition is expensive or logistically prohibitive [23].

Alternatively, learning from just a few data points increases the chance of one of the most dangerous issues of machine learning and deep learning, *overfitting*. Overfitting consists of overly adjusting a model for a given dataset. When a curve is more adjusted than it should to the data, it will possibly poorly interpolate points and thus fail to accurately predict the outcome for instances that are unseen during training. This type of issue is easily spotted by creating validation sets or performing cross-validation experiments [2].

However, there is a misconception that validating a model on an unseen set properly describes its effectiveness when dealing with new environments. Collected data can suffer from a number of biases, and models may not be prepared to handle situations that the engineering process did not take into account [73]. Most of the time, validation and training sets are sampled from the same context and follow somewhat the same distribution. Nonetheless, a new scenario could slightly shift data distribution due to the lack of control of confounding factors. The difference between the originally trained distribution to a new, unseen set is called *domain shift*,

and it is the object of study of *domain adaptation*. Domain adaptation has as its main goal to prepare a model to deal with data from a new, previously unseen domain.

More formally, in domain adaptation the task is to treat the discrepancy of how a trained model handles samples following the source training distribution $x_s \sim p(X_s)$ and samples from a target distribution $x_t \sim p(X_t)$. Frequently, samples from $p(X_s)$ are annotated data, while samples from $p(X_t)$ are not. In that case, the task is called *unsupervised domain adaptation*.

One of the many fields of application for domain adaptation is medical imaging. Due to the lack of standardization of scanners among companies and the heterogeneity of acquisition parameters among studies, the variability can have implications in the performance of deep learning models [76, 12]. Here we opt to call medical imaging any modality of image acquisition that is frequently used for health care purposes. Frequently used modalities are magnetic resonance imaging (MRI), computed tomography (CT), positron emission tomography (PET), and several others, each one with their unique properties.

Within the overlapping field of deep learning and medical imaging, MRI is the most frequent among scans [43], arguably because they are widespread, have dataset available for research [51], and do not present any form of ionizing radiation. However, it also comes with several challenges. MRI scans are highly susceptible to changes in the image due to intrinsic characteristics of the scanner. For example, the intensity of the magnetic field can drastically impact the resolution of the generated image. There are also extrinsic aspects, such as the machine parametrization, which can affect the final result [12].

Prior to exams, subjects can also have contrast agents injected intravenously, such as Gadolinium. Contrast agents change the stabilization time and have the benefit of highlighting tumors and other tissues of interest [84]. For deep learning, however, it means yet another data distribution the model should be able to cope with. We also have to deal with preprocessed images, which probably are the biggest source of domain shift in this scenario. There is a wide literature for preprocessing MRI images, each one leading to a slightly different distribution [77].

As our object of study, we picked the most common task in deep learning for medical imaging, segmentation [43]. From the multiple subtasks we could choose, we picked spinal cord grey matter segmentation due to the lack of similar work in the field and the recent release of a multi-center dataset [60]. Spinal cord segmentation is an important task that can help identify several non-traumatic neurological disorders [1], also improving the visualization of spinal cord damages.

We have found some methods that are concerned with domain adaptation for segmentation in medical imaging with promising results. Most of them use adversarial training-based frameworks to overcome domain shift [11, 47, 48]. However, throughout our research, we have not found methods that originated from more recent semi-supervised learning literature being applied for domain adaptation in medical imaging. Since we found no positive or negative evidence on the effectiveness of self-ensembling in this context [18], and due to its success in

other fields, we intended to investigate whether this framework is a suitable alternative for researchers when applying their models in new domains.

After promising results from self-ensembling, we decided to adapt other state-of-the-art methods from the semi-supervised learning literature for domain adaptation in medical imaging segmentation. We opt to investigate recent methods that surpassed self-ensembling results in standard semi-supervised learning benchmarks that have the same underlying assumptions and strategies, namely, *unsupervised data augmentation* [80] and *MixMatch* [5].

We evaluate two hypotheses regarding the performance of semi-supervised learning methods on our context. The results showed that semi-supervised learning methods can improve purely supervised learning for domain adaptation when models handle data from an unseen domains and even improve results in the original training domain.

The rest of this dissertation is organized as follows: In Chapter 2 we introduce the nonstandard learning paradigms, such as transfer learning and multitask learning and how they relate to the field of domain adaptation. We also detail the task of medical imaging segmentation. In Chapter 3 we present current methods of domain adaptation and detail the methods that create the groundwork for ours; when also detail some of our decisions and present our hypotheses. Chapters 4 and 5 we discuss our work and present our experiments.

2. BACKGROUND AND DEFINITIONS

Unsupervised domain adaptation is a field that can be integrated at virtually any machine learning paradigm and task. To further discuss domain adaptation we must be able to properly define its relationship with other fields and, most importantly, how it differs from other research areas such as semi-supervised learning and transfer learning.

2.1 Learning Paradigms

Machine learning is a composition of multiple learning paradigms. The most common are usually classified as supervised learning, unsupervised learning, and reinforcement learning [6]. In this study, we will focus on supervised and unsupervised learning as they represent a large portion of current unsupervised domain adaptation applications. Reinforcement learning has seen a growth in the number of domain adaptation work [27, 8], but the field has grown on its own, and for that reason usually depends on developing dedicated approaches.

In supervised learning, the model receives a set of tuples $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ corresponding to n training instances x_i with their corresponding annotation y_i. We can further define an annotated dataset as X with $p(X)$ representing the distribution of its training instances and $p(Y|X)$ representing the label distribution with respect to the input. Fitting a model means that it learns a mapping $M : X \rightarrow Y$ from the input space X to label space Y. Traditional supervised learning tasks are classification, regression, segmentation, among others.

In unsupervised learning, the model receives a set of instances $D = \{x_1, ..., x_n\}$. In this case, there are no labels and the model should be able to learn relevant patterns on data alone. The distribution $p(X)$ is present, but unlike in supervised learning, the values from the distribution $p(Y|X)$ for training samples are unknown. Unsupervised tasks are mostly concerned with clustering and dimensionality reduction. In contrast to supervised tasks, unsupervised learning usually does not have a clear objective in terms of a task, since the result is frequently open to interpretation (although in most cases has a clear optimization objective).

Following the distinction of learning paradigms, we can also define the difference between datasets. Generally speaking, we can group data into two groups, labeled (annotated) and unlabeled, when labels from label space distribution $p(Y)$ are present and when they are not, respectively. While annotated data represents important information for encoding a specific objective that the model should learn, most data available belong to the latter group. Labeling data is a cumbersome and expensive process. Annotation usually depends on domain experts and in some cases it is not as straightforward as describing a picture. In brain tumor segmentation tasks in magnetic resonance imaging, for instance, training a model in a standard supervised learning fashion would take hundreds of images annotated at pixel-level by neurologists, radiologists, or other qualified professionals. Such professionals frequently do not

agree among each other in their annotations, leading to inter-rater inconsistency [3]. Dealing with inconsistency has been the object of study of Bayesian approaches towards uncertainty modeling, and it is not within the scope of this work [36].

Collecting annotated datasets is a consuming and error prone process and thus we should try to find approaches that can deal with both labeled and unlabeled instances. The semi-supervised learning paradigm offers a solution for such a scenario. This paradigm takes two datasets, the first composed of training instances $p(X_l)$ with available labels from $p(Y|X_l)$ and another set with $p(X_u)$ instances and unavailable labels from $p(Y|X_u)$. The aim is to leverage models knowledge on its $M : X_{\{l,u\}} \rightarrow Y_{\{l,u\}}$ mapping with both $p(X_l)$ and $p(X_u)$ samples. There are numerous ways of introducing $p(X_u)$ to the traditional training protocol, and we further discuss those methods in Chapter 3. This type of paradigm can greatly improve results when little annotated data is available but sometimes fail to introduce improvements when labels are not scarce [54]. One important assumption of semi-supervised learning is that $p(X_u) \simeq p(X_l)$, which means the input spaces follow the same distribution, and that, consequently, the conditional distribution of labels with respect to the input is somewhat the same, or $p(Y|X_l) \simeq p(Y|X_u)$.

The former assumption of data distribution is of extreme importance for semi-supervised learning to work properly since most methods use this to improve their training data. Exemplifying, label propagation approaches [86] label data from $p(X_u)$ using the closest similar instance, generally in the input or feature space. To ensure that this assumption does not break, we must guarantee that the environment where data is collected is controlled by several factors. Not controlling said factors might cause the input data to shift too much from the original training mean and lead to undesired model behavior. In more extreme cases, the new dataset distribution may follow a completely different manifold (a generalization of a mathematical surface for multidimensional spaces). The premise of $p(X_l) \simeq p(X_u)$ is broken in many scenarios. In practical tasks, such as autonomous driving, where each city has a unique architecture, or medical imaging, where each center gathers data following different protocols, semi-supervised learning could fail to improve results. For those reasons, we must be able to learn from data that differ in distribution. In such situations, domain adaptation, transfer learning, and multitask learning should be more suitable alternatives for the challenge.

2.2 Domain adaptation, transfer learning, and multitask learning

In this section we discuss the multiple paradigms that have been proposed to learn in these scenarios of multiple and diverging distributions in the input space. We consider this to be one of many challenges in deep learning research. Effectively learning when facing this situation would allow us to spread deep learning models for many places that lack annotated data of their datasets, alongside building more generic and practical models that can face realistic situations.

A frequent approach to learning when the distributions shift is to use transfer learning. This is probably the most widespread paradigm of the ones discussed, but it is somewhat limited in terms of benefits. The first step in a standard transfer learning protocol is to train a model in a dataset, called *source dataset*, in its task, called *source task*. To properly benefit from transfer, researchers train models in a large dataset such as ImageNet [66]. This whole process is frequently mentioned as the pretraining of the network. The second step is to fine-tune the model in the *target dataset* in its task (not the same as the source), which is called *target task*.

There are mainly two reasons as to why transfer learning works. First, transfer learning can be perceived as a form of initializing the network. Much like other initialization methods [22, 26], changing weight distribution from a gaussian distribution to something more similar to the final weight distribution can improve convergence. We refer the reader to [22] for a more thorough discussion on the poor performance of random initialization. A second perspective on why transfer learning works is the task-related knowledge inheritance. The kernels from a trained network were designed specifically for extracting features from an input. When the target task is somewhat related to the source, the features can usually be quite useful and speed-up training.

Due to the aforementioned reasons, the model is able to adapt its knowledge from a source task to a target task without much effort. Transfer learning has arguably gained popularity due to other properties as well. A popular advantage of transfer learning is that it can potentially improve results against training purely on the target task, even with a large target dataset [82]. This idea of pretraining improving the final results is currently being challenged in [25]. The work shows experiments that do not present statistically significant accuracy changes with respect to training without pretraining. The convergence speed for transferring models is shown in both work to decrease significantly. Transfer learning does not require the distributions to be similar, nor the tasks from the source and target domain to be related. However, it needs some annotated data from the target task. Depending on how much the tasks and distributions differ, the number of images that are necessary for the transfer may increase significantly.

Another way to take advantage of training with additional data and also optimize the desired task is to train in a multitask fashion [9]. During training, the model optimizes multiple objectives. By sharing parameter spaces between tasks, the chance of overfitting theoretically decreases and could potentially benefit both tasks by learning a more generalized feature space. In that sense, one could leverage the knowledge from a model by introducing the optimization of a related task with additional data. Multitask learning also provides a way to create a single model that can solve multiple tasks alone, particularly useful when there are memory and processing restrictions in a system. Furthermore, a key difference between transfer learning and multitask learning is the phenomenon called catastrophic forgetting [19, 50, 49, 62]. Machine learning models fail to persist with encoded information, highly favoring samples seen in later stages of the training. Researchers are conducting studies to overcome catastrophic

forgetting [35], but, to this date, transfer learning still usually degrades model accuracy rapidly in its source task, specially when the target task diverges much from the original. Both transfer learning and multitask learning demand annotated data from the target domain.

Despite the existence of multiple datasets that can be used to introduce both transfer and multitask learning during training, there are some situations where it fails to generate generalized models. For instance, consider again the task of end-to-end autonomous driving. Training with multiple objectives such as both driving directives and passenger detection (multitask) by fine-tuning a model originally trained for semantic segmentation in streets (transfer) could possibly improve results. However, there are no guarantees that generalization improved sufficiently to maintain accuracy at new cities [81].

This is a case where a model must be prepared for a specific environment (application in a new city, in that case). This is where domain adaptation provides the most benefit. By using data from a source domain X_s for a task T and data from a target domain X_t for the same task T, adapt the knowledge of the model mainly to handle distribution $p(X_t)$. Intuitively, using data from a source domain to improve the target domain on a specific case. In our previously described situation, one could use data from the city of London and Melbourne to better learn the city of Toronto. Ultimately, the model can even improve learning with annotated data from simulations [33]. This differs from transfer learning because we assume that the task stays the same, which comes with interesting properties to explore. Finally, when labeled data is only available for $p(X_s)$ (or London and Melbourne), in this scenario we face the task of unsupervised domain adaptation, which we refer to, from now, simply as domain adaptation.

Like semi-supervised learning, domain adaptation consists of two distributions for the same task, one with annotated data and the other without. We use a different notation for the available sets. In semi-supervised learning, as discussed, we have a labeled dataset X_l and an unlabeled one X_u. In domain adaptation, we also have a labeled and an unlabeled dataset, but we call it source (X_s) and target (X_t) as a reference to the domain from where the dataset was collected. Following the same logic, while we call the label distribution with respect to the input for unlabeled samples $p(Y|X_u)$, in domain adaptation we call it $p(Y|X_t)$.

More formally, both semi-supervised learning and domain adaptation assume that samples from $p(X_u)$ and $p(X_t)$ from the training set have no annotated Y. By contrast, domain adaptation does not assume that $p(X_s) \simeq p(X_t)$ much like semi-supervised learning assumes $p(X_l) \simeq p(X_u)$. Finally, the key difference is that relying on that assumption, semi-supervised learning tries to improve the mapping M for both $p(X_l)$ and $p(X_u)$, while in domain adaptation, the focus is to improve the mapping M for $p(X_t)$ and eventually disregard the importance of $p(X_s)$ in the final model, which means any accuracy drop in the source task is irrelevant. We show an example of this difference in Figure 2.1. Points in the image are instances and the colors are their labels. We see in the semi-supervised learning part that instances from the same class ($X_l = 1$ and $X_u = 1$, and $X_l = 0$ and $X_u = 0$) belong to somewhat the same data manifold

(expected due to the assumption $p(X_l) \simeq p(X_u)$. In domain adaptation, however, we see a shift from instances of the same class, which characterizes the problem.

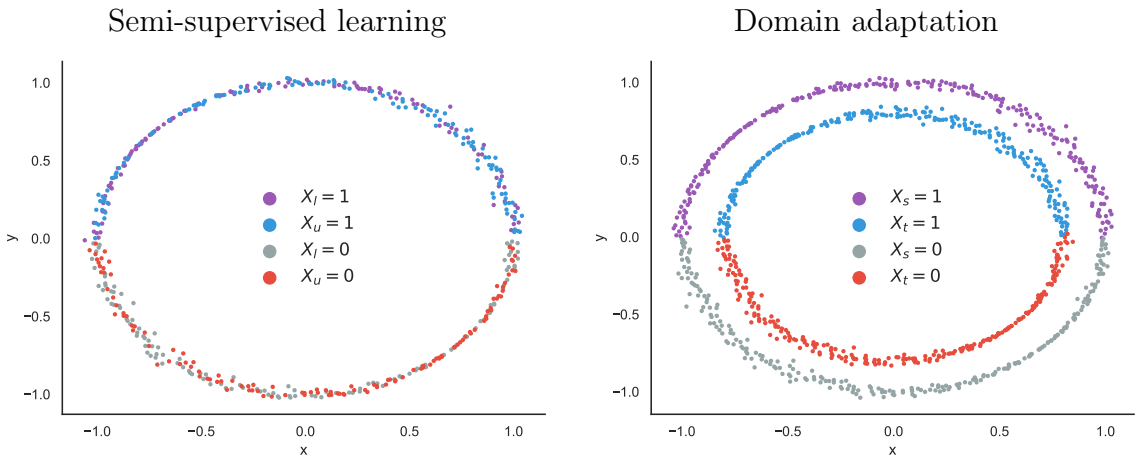


Figure 2.1: Difference between semi-supervised learning and domain adaptation scenarios.

2.3 Semantic segmentation and medical imaging segmentation

Semantic segmentation extends the notion of other tasks such as classification and detection for the pixel level. Whereas in classification, for a given input image, the model predicts a class based on a set of available classes, in segmentation the model predicts one of these classes for each pixel within the image. Segmentation thus produces an output with proportional (or same) size as the input (e.g. an image mask). We see an example of a semantic segmentation task in Figure 2.2.

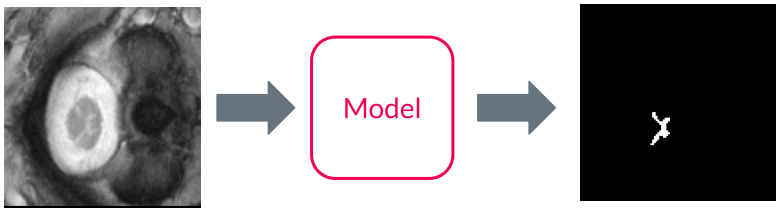


Figure 2.2: Depiction of a semantic segmentation pipeline. In this example, we show the segmentation of the grey matter of a spinal cord.

Within the field of deep learning for medical imaging, we see segmentation as the most frequent task [43]. Applications vary, ranging from prostate segmentation [52] to brain tumor segmentation [56]. Initial work on the field focused on 2-dimensional patch-based methods, where a model received part of a slice from a CT or MRI and produced the result for this part only. Patch-based methods can only evaluate a whole slice by passing a sliding window passes through the input, evaluating the model to predict fixed-size segmentation patches. Results from different portions of the slice are concatenated in order to produce the final segmentation mask. There are studies that aim to mitigate the core problems of patch-based methods to this

date. Some work involve introducing other simple deep learning concepts, such as layer-wise pretraining as autoencoders [79] in the attempt to create more robust data representations. Others use cascaded convolutions, processing patches at different resolutions [30].

Despite recent efforts on patch-based approaches, the most common medical imaging segmentation paradigm shifted with the release of fully convolutional networks (FCNs) [45]. This type of network has no fully connected layers and their output comes straight from convolutional layers, producing an output with a shape proportional to the input shape. Fully convolutional networks effectively popularized in medical imaging with the U-Net [65]. The U-Net is a fully convolutional network designed as an encoder-decoder to maximize the global information capture by the model. The authors also introduce the use of skip connections in medical imaging segmentation, connecting layers from the encoder to layers of the decoder, thus speeding up optimization and mitigating the vanishing gradients problem.

3. METHODOLOGY

In this chapter, we review current methods for unsupervised domain adaptation and their similarities. We then show the most important work on domain adaptation designed specifically for medical imaging segmentation. We also thoroughly discuss self-ensembling, which we employ in our task. We move on to explain unsupervised data augmentation [80] and MixMatch [5], two recent methods of semi-supervised learning that have yet to be tested in domain adaptation for medical imaging segmentation. Finally, we present the dataset used in our experiments, our baseline, and introduce our research hypotheses.

3.1 Related Work

While advances in unsupervised domain adaptation have been significant, much work is still left undone [55, 13, 76]. In this section, we discuss traditional approaches of domain adaptation. Traditional methods mainly rely on computing some sort of statistical divergence on the model for source and target domain inputs. We then move on to adversarial training, which comprises more robust methods of domain adaptation and includes several state-of-the-art methods. Adversarial-based methods excel at visual domain adaptation, specially when the domains differ mostly in texture. Understanding these concepts, we are capable of discussing the specific methods of domain adaptation for medical imaging segmentation. We then discuss self-ensembling and its background and why it is relevant for the field. Finally, we show two recently released approaches for semi-supervised learning, namely, unsupervised data augmentation [80] and MixMatch [5], which we adapt for our context.

3.1.1 Traditional Approaches on Unsupervised Domain Adaptation

We define traditional approaches as methods that directly minimize the discrepancy between predictions on source and target domain through distance metrics or higher-order statistics. This assumes that minimizing discrepancies between predictions improves the prediction space, which means the model more effectively creates inner representations that handle both domains.

Seminal work in the field involved creating a bottleneck layer followed by a loss to compare domains at feature space [75]. The network was optimized to minimize what was called domain discrepancy, or in other words, maximize *domain confusion*. This was essentially a mean-squared error loss of predictions from different domains. CORAL [71] extends this notion to minimize the covariance of such predictions instead of the squared error. A more complex approach introduced minimizing the Jacobian between domains [70]. The work shows that the way prediction space is interpolated can vary drastically among models and yield poor results.

Minimizing the Jacobian would lead to more smooth interpolations among predictions which would improve the knowledge transfer among domains.

Other approaches directly change computed statistics inside the network. A common layer that encodes such information is the batch normalization [31]. It involves storing channel-wise mean and standard deviation of activations during training. This notion was extended for domain adaptation [41]. The adaptation extension computes different statistics for source and target domains, improving how the network treats information from both domains, creating more domain-invariant features. During inference time, the model must know from which domain the data is coming to use the proper statistics for normalization.

3.1.2 Adversarial training and image-to-image translation

Popularized in 2014, adversarial training has become the state-of-the-art for multiple tasks in deep learning that require generative models [24]. The original framework, called Generative Adversarial Networks (GANs), comprises two models, known as discriminator and generator. During training, both models participate in a zero-sum game, where the generator aims at generating realistic content while the discriminator identifies whether samples originated from the generator or from a real (training set) distribution.

The training consists of two basic steps. First, the generator is frozen (not updated during backpropagation) and the discriminator is trained to predict 1 when samples come from a real data dataset and 0 when samples come from the generator. In order to do that, the generator creates b samples where b is the batch size for real samples. These are batch-wise concatenated with samples from the training set and the error is calculated using a binary cross-entropy loss. Then, the discriminator is frozen. The generator generates more samples and evaluate them against the discriminator, being updated according to whether the samples tricked or not the discriminator into predicting they originated from the training set distribution. The gradients flow through the discriminator and update the generator.

Both generator and discriminator become increasingly better as the training progresses, culminating into a generator capable of generating realistic samples. Another facet of generation is being able to generate multiple and diverse samples. The way GANs solve this problem is by learning to map a known distribution $p(Z)$ into the unknown real data distribution $p(X)$. The known distribution $p(Z)$ can follow any known probability density function, such as gaussian or uniform. We can define this mapping as $M : Z \to X$. Concurrently, the discriminator learns a mapping $M : X \to \{0, 1\}$. Theoretically, the zero-sum game between the generator and discriminator converges to a scenario where the generator generates samples that follow the exact same distribution as the real data, and the discriminator reaches an accuracy of 0.5 (equivalent to random guess). In practice, both models are limited by the number of parameters, thus usually not reaching a perfect equilibrium nor achieving perfect generation. The models optimize the following objective, one to minimize it and another to maximize it:

$$\min_{m_g} \max_{m_d} \mathcal{L}_{gan}(m_d, m_g) = \mathbb{E}_{\mathbf{x}\sim p(X)}[\log m_d(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(Z)}[\log(1 - m_d(m_g(\mathbf{z})))] \tag{3.1}$$

where m_g and m_d are the generator and discriminator models, and x and z are sampled values from the data distribution $p(X)$ and a noise distribution $p(Z)$ respectively. We also depict the same generative adversarial network framework in Figure 3.1.

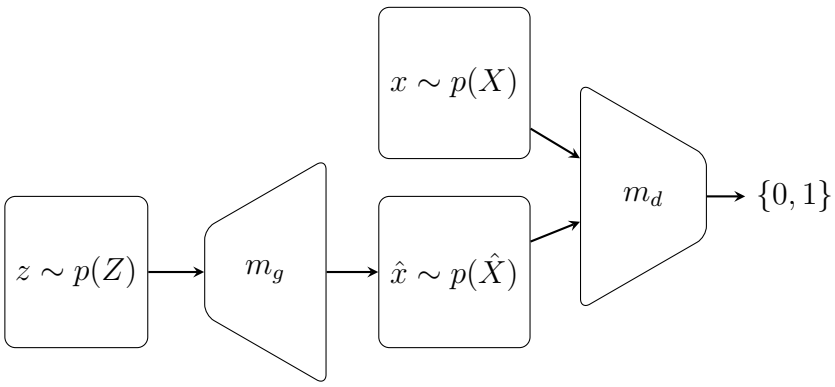


Figure 3.1: Generative Adversarial Network training framework. We use x to represent a sample from the real distribution X and $\hat{x}$ to represent a sample from the fake distribution $\hat{X}$.

The framework has been incrementally developed to improve quality and control over generated samples. Regarding quality of generation, the extension of the framework for deep convolutional networks [61] heavily influenced subsequent GANs work. Current state-of-the-art for the task, known as BigGAN [7], uses deep convolutional GANs alongside other advances in training procedures to generate images with 512×512 resolution. However, BigGAN still suffers from some limitations, specially mode collapse, where regions of the latent space collapse to generate the same content. Current work in variational autoencoders are challenging GANs in the task by posing a solution to said limitations [63], but are out of the scope of this work.

On the control of the generated image, the original framework could not control the results in terms of specific characteristics. To generate a specific content, the user needed to sample different vectors from Z and observe the changes. To fix this issue and increase the applicability of GAN models, a conditional vector is inserted during training that consists of characteristics that are intended to be controlled during inference. This kind of model is known as CGAN [53].

In the context of domain adaptation, adversarial training appears in many shapes. In CoGANs [44], both generator and discriminator models are used for image-to-image translation. In image-to-image translation we attempt to, given an image from domain A, generate its counterpart in domain B. When trained with unpaired examples from both domains, this task has an intrinsic relationship with unsupervised domain adaptation. For that reason, the authors evaluated their framework in a digit recognition domain adaptation task, achieving the, at the time, state-of-the-art. The good results from the framework come from the ability of accurately translating the image from one domain to another. Building upon this concept, subsequent work

based on CycleGANs [85] (an image-to-image translation method), CyCADA aims at solving the task of unsupervised domain adaptation using cycle consistency [28].

Other methods rely solely on the use of discriminators alongside the task loss. For instance, the most common approaches branch the network at intermediate layers, sharing the initial layers and creating two different paths, one for each domain [20, 74, 10]. This type of framework trains the source branch to minimize the task loss and the target branch to minimize an adversarial loss, based on the difference of the distributions of the branches when presented with their respective samples. This assumes that in order to predict properly for the target domain, high-level feature space should be similar when predicting on source and target images. This is a common premise, frequently seen in seminal papers of deep domain adaptation [75].

3.1.3 Domain adaptation in medical imaging segmentation

There are not many popular methods designed specifically for the purpose of domain adaptation in medical imaging. Existing work focus mostly on using adversarial-based methods in a similar fashion as they were originally proposed. We discuss some of the papers and present their current efforts to solve the task.

Chen et al. [11] developed a domain adaptation method based on CycleGANs [85]. Their work resembles CyCADA [28], with the difference of also introducing a discriminator for the network output, creating what they called semantic-aware generative adversarial networks. Javanmardi and Tasdizen [32] use a framework very similar to domain-adversarial neural networks [20] that use a domain classification network with a gradient reversal layer (similar effect of a discriminator) to model a joint feature space. Some work focus on domain adaptation for cross-modality segmentation. Dou et al. [17] designed an adversarial-based method that learns both domain-specific feature extraction layers and a joint high-level representation space that can segment data from both MRI or CT data.

Gholami et al. [21] proposes a biophysics-based data augmentation method to produce synthetic tumor-bearing images for the training set. The authors argue that this augmentation procedure improves model generalization. Mahmood et al. [48] presents a generative model that translates images from a simulated endoscopic images domain to a realistic-looking domain as data augmentation. The authors also introduce a l1-regularization loss between the translated and the original image to minimize distortions. Madani et al. [47] introduced a generative adversarial network with a discriminator capable of detecting cardiac disease alongside the adversarial objective. They employ their network in a semi-supervised scenario and also evaluate their model for domain adaptation with promising results.

We include these studies in the related work for completeness but do not compare them in our experiment section with the methods we employ. Some methods from this section are very domain-specific and demand biology and radiology specialists to translate them for different

machines and body regions, which falls out of our scope of more easily generalizable approaches. We focus only on evaluating recent work from the semi-supervised learning literature, since they were shown to produce some of the best results in domain adaptation while not relying on domain-specific knowledge nor on, what tend to be, unstable procedures, such as adversarial learning.

3.1.4 Temporal ensembling and self-ensembling

Temporal ensembling and self-ensembling are a different way of posing the task of domain adaptation. In this case, the idea originates from the related field of semi-supervised learning. The main concept behind it dates back to Polyak-Ruppert averaging [59] or Polyak averaging for short (sometimes also called stochastic approximation). Generally speaking, Polyak averaging is a method that improves optimization following an Exponential Moving Average (EMA). We see in Figure 3.2 an example of the EMA behavior, where, as originally proposed, an exponential moving average over the weights at every training iteration leads to smoother loss curves, controlled by a hyperparameter α. In this section we describe EMA in the context of semi-supervised learning, the two most important work in the field, and how it has been modified for domain adaptation.

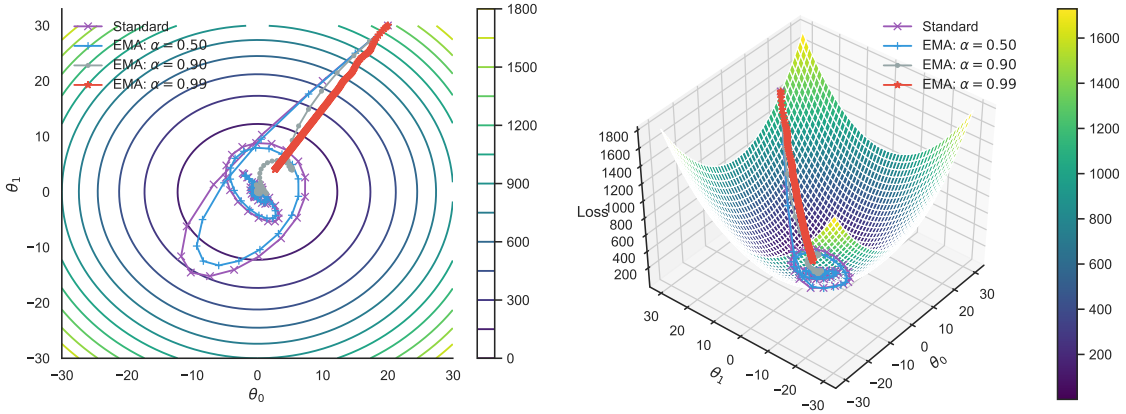


Figure 3.2: Optimization with Adam optimizer [34] of a model with two weights (θ_0 and θ_1) in a convex loss surface with exponential moving average for multiple α. We start at $\theta = [20.0, 30.0]$ and set the learning rate to 10. The loss function is defined as $Loss = {\theta_0}^2 + {\theta_1}^2$.

Temporal ensembling

The first method to propose EMA during training for semi-supervised learning called it temporal ensembling [38]. The paper describes that combining predictions over time generates more reliable estimates that can be used as pseudo-labels. The framework keeps a running exponential moving average of predictions for every sample, updated once every epoch. These pseudo-labels are then used as ground truth for unlabeled examples $p(X_u)$ and trained jointly

with the labeled set $p(X_l)$. The following equation defines the generation of pseudo-labels:

$$\hat{y}_t = \alpha\hat{y}_{t-1}(1-\alpha)m(x) \tag{3.2}$$

where α is a factor that modulates the influence of previous weights for the pseudo-label and m is the model. The temporal ensemble $\hat{y}_t$ is the pseudo-label for sample x at epoch t. Temporal ensembling has two independent losses that we see in posterior frameworks. The first loss is called task loss, or $\mathcal{L}_{task}$, computed for labeled instances using the ground-truth y. The second loss is called consistency loss, or $\mathcal{L}_{cons}$, and it is a distance measurement (e.g. mean-squared error) of the ensembled predictions $\hat{y}$ and the current model prediction. We also depict the temporal ensembling framework in Figure 3.3.

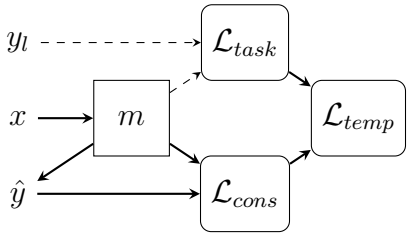


Figure 3.3: Temporal ensembling framework. Straight lines represent flow that happens for every sample in a batch. Dashed lines are pathways that flow only when evaluating labeled instances. The input x is a sample drawn from either the labeled or unlabeled dataset. When x is sampled from the labeled set, y_l becomes its label. In both cases, $\hat{y}$ is the exponential moving average of the model m predictions. $\mathcal{L}_{task}$, $\mathcal{L}_{cons}$, and $\mathcal{L}_{temp}$ are the task loss, consistency loss between predictions and moving average expected values, and the total final loss.

Although useful and with promising results, the temporal ensembling framework demands that an exponential moving average representing the expected output for each training instance must be stored. This is very limiting since datasets are usually composed of thousands, if not millions, of instances. Furthermore, the benefits of the EMA can only be seen after several epochs, since the effect of updating $\hat{y}$ happens only once per epoch for each instance. There are also a few other tunings in the method. For example, there is a clear tradeoff betweeen the amount of images sampled from labeled and unlabeled datasets to compose the batch, which is controlled by a hyperparameter.

Self-ensembling

A substantial incremental change was proposed following the seminal work of temporal ensembling in what is called the mean-teacher framework [72]. The mean-teacher proposes to change the EMA from the prediction space to the parameter space. The framework comprises two different models, which are called *student* and *teacher*. The former is trained with backpropagation, whereas the latter is an EMA of the student weights over time, following the equation:

$$\hat{\Theta}_t = \alpha\hat{\Theta}_{t-1} + (1-\alpha)\Theta_t \tag{3.3}$$

where Θ are parameters of the student model and $\hat{\Theta}$ are parameters of the teacher model.

The parameters of the teacher model are updated once every iteration. First, an adaptive α hyperparameter that is increased as a function of the number of epochs. Intuitively, model predictions are less accurate during early steps of training and can be forgotten faster. As the training progresses, teacher weights benefit from the use of a longer memory. The authors also propose a learning rate schedule with an initial rampup (preventing unstable training) followed by a rampdown (exploits local minima) until achieved the total number of epochs.

The change from predictions to parameters brings some advantages over the previous framework. First of all, in temporal ensembling the effect of the EMA was perceived only after several epochs and for large datasets there is a big delay between every update for a single sample. However, the pseudo-labels in the mean-teacher are computed every time a mini-batch is sampled. Basically, instead of storing the exponential moving average of instances, the batch is just forwarded through the teacher network and the output compared against the student's. The loss function comprises a distance between predictions, called $\mathcal{L}_{cons}$, from the student and teacher models for both labeled and unlabeled samples and the task loss $\mathcal{L}_{task}$ for labeled samples. The teacher model is updated once every mini-batch after the student update:

$$\mathcal{L}_{stu}(m_s, m_t, x_l, x_u, y_l) = \mathcal{L}_{task}(m_s, x_l, y_l) + \gamma \mathcal{L}_{cons}(m_s, m_t, [x_l, x_u]) \tag{3.4}$$

where $\mathcal{L}_{stu}$ is the total loss, $\mathcal{L}_{task}$ is the loss for the task that is being targeted during training, and $\mathcal{L}_{cons}$ is the consistency loss between the teacher and student predictions for an instance. Finally, $[x_l, x_u]$ is just a set of both independent labeled and unlabeled examples.

We describe one possible instantiation of the framework to explain it. In a standard classification semi-supervised learning task, the model receives gradients from both labeled and unlabeled instances. For labeled instances, two losses are computed, a categorical cross-entropy between x_l and y_l and a mean-squared error for student and teacher predictions. For unlabeled instances, only a mean-squared error is computed. Every batch comprises both labeled and unlabeled samples and the batch average loss is backpropagated to the student model. The teacher is then updated following the exponential moving average equation with a chosen α using the new parameters from the student. One way to think intuitively about why this works is that we assume the teacher model to always be one step ahead of the student (due to Polyak averaging). For that reason, when one iteration of the backpropagation happens in the student, it is simultaneously trying to approximate the teacher model and getting closer to the task objective.

Unsupervised domain adaptation and semi-supervised learning share several key characteristics. For that reason, it is possible to adapt the mean-teacher framework for domain adaptation tasks. Such an adaptation of the original framework has already been proposed [18]. The main contributions are among the mini-batch composition of labeled and unlabeled samples and separate pipelines for data from the source and target domains. Data from the source domain have the loss computed only on the task, whereas samples from the target domain are

only evaluated by the discrepancy between the teacher and student predictions. We show how information flows through the mean-teacher for domain adaptation framework in Figure 3.4.

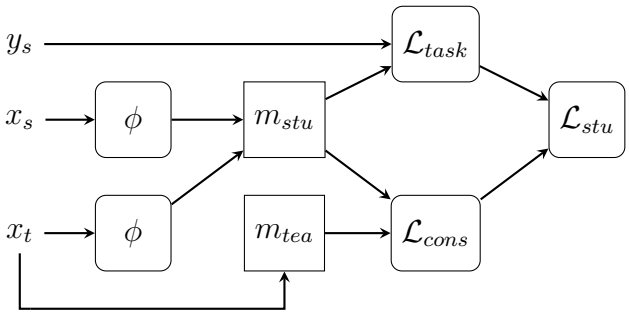


Figure 3.4: The mean-teacher framework for domain adaptation. The $\mathcal{L}_{task}$, $\mathcal{L}_{cons}$, and $\mathcal{L}_{stu}$ are the task loss, usually a cross entropy between predictions and expected values, consistency loss between models, and the total student model loss. We call the student model m_{stu} and the teacher model m_{tea}. The inputs x_s and x_t are data from the source and target domain, while y_s is the expected value for the source input. We depict an augmentation policy as ϕ.

One last important characteristic of all self-ensembling-based frameworks is data augmentation. Data augmentation means introducing noise, such as random and gaussian, and transformations, such as rotation and translation, in the input data to improve regularization and, consequently, boost generalization. Augmentation plays a key role in all these frameworks because we use different augmentations when calculating the consistency loss. Since the student and teacher receive the same input, but with slightly different modifications, the consistency loss also helps the model to be invariant to the augmentation included in the training.

3.1.5 Recent semi-supervised learning methods

Recent work on semi-supervised learning has surpassed the state-of-the-art set by the mean-teacher algorithm. Both unsupervised data augmentation [80] and MixMatch [5] reached outstanding results in object classification tasks with missing labels. We detail both methods and show several key points that are reviewed in Chapter 4 to prepare them for domain adaptation in medical imaging segmentation.

Unsupervised data augmentation

Unsupervised data augmentation [80] builds upon the concept introduced in AutoAugment [14]. AutoAugment is an algorithm that finds data augmentation policies straight from data. The key motivation lies in the different missing variability contained at each dataset (e.g. one can be lacking rotation variability in training data while the other might be missing translation). The method selects a set of augmentation procedures (e.g. translation, rotation, Cutout [15], and others) and uses reinforcement learning to create an augmentation policy based on a subset of them. It also automatically identifies the magnitude for each of the procedures.

The training is composed of a controller, responsible for predicting augmentation strategies and a child network, trained to validate the strategy. The process is conducted as follows:

1. The controller predicts a strategy S composed of 5 sub-policies, each one containing an operation type, a probability of being applied, and a magnitude,
2. A child network is initialized and trained using samples augmented by S. It is important to note that each example is augmented using only one, randomly chosen, sub-policy from S and validation samples are not augmented,
3. The controller updates its weights by using a policy gradients method with the validation accuracy from the child network as a reward.

Unsupervised data augmentation, at its core, compares model predictions between augmented and non-augmented samples to build a consistency. In the mean-teacher, the authors build a consistency loss between predictions from the student and teacher models, but in unsupervised data augmentation, due to the benefits of AutoAugment, the divergence of predictions is calculated over the same model, as follows:

$$\mathcal{L}_{cons} = \mathbb{E}_{x \sim p(X_u)} \mathbb{E}_{\hat{x} \sim \phi(x)} [KL(m(\hat{x}), \hat{m}(x))] \tag{3.5}$$

where ϕ is an augmentation policy from AutoAugment and KL is the Kullback-Leibler divergence of predictions from the model. Models m and $\hat{m}$ are exact copies, but gradients do not flow through the latter. The Kullback-Leibler divergence is defined as follows:

$$KL(p, q) = -\sum p * \log \frac{p}{q} \tag{3.6}$$

where p and q are discrete probability distributions. We can also understand it as the difference between the cross-entropy of p and q and the entropy of p, as:

$$KL(p, q) = CE(p, q) - CE(p, p) \tag{3.7}$$

where CE is the cross-entropy, defined as:

$$CE(p, q) = -\sum p * \log q \tag{3.8}$$

Unsupervised data augmentation has a second objective that is optimized simultaneously. Since deep learning models tend to easily overfit in the labeled training data, the authors propose a way to limit the amount of gradient information coming from labeled samples. They introduce the Training Sinal Annealing (TSA), which is a strategy to remove samples from backpropagation that the model is already confident about. The allowed confidence for backpropagating samples grows as the training procedure progresses. Intuitively, the model should not produce highly-confident predictions for labeled samples in early stages of the training

to allow for more gradient information to come from unlabeled and uncertain samples. The supervised objective is a simple change from the cross-entropy objective, designed as follows:

$$\mathcal{L}_{task} = -\frac{1}{Z}\sum y * \log(m(x)) * mask \tag{3.9}$$

where mask is a vector with the same length as the number of classes, designed as follows:

$$mask = \begin{cases} 1, & \text{where } \max[y * m(x)] < \eta_t \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

and, since y is a one-hot vector indicating the correct class, max will yield the predicted probability for the expected class. Note that this means high-confidence predictions for incorrect labels are still backpropagated. The confidence threshold is defined as follows:

$$\eta_t = \frac{1}{k} + \lambda_t * (1 - \frac{1}{k}) \tag{3.11}$$

where k is the number of classes and λ_t is a parameter that increases with the training progression, going from 0 when the training starts to 1 when the training ends. In practice, η_t grows from $\frac{1}{k}$ to 1, which means that earlier in training, the propagation only happens in labeled samples with predictions lower than random confidence. In later training stages, the propagation happens for every sample. The authors show that λ_t can grow in a logarithmic, exponetial, or linear fashion, which is a choice for the user to make depending on the amount of labels available. When few labels are available, λ_t should grow slower to more severely prevent overfitting. The η_t value over the training epochs for each schedule is shown in Figure 3.5.

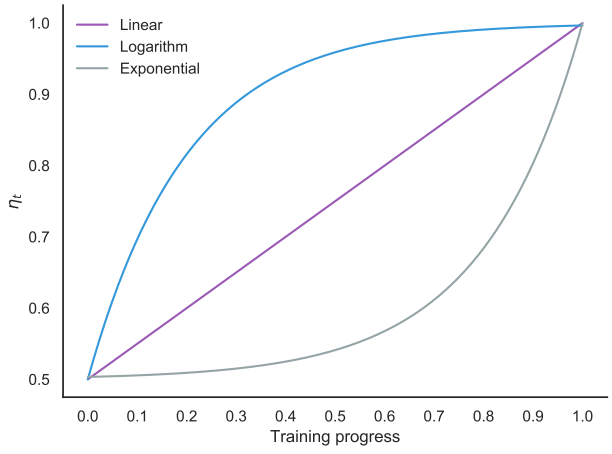


Figure 3.5: η_t over time for each λ_t training schedule.

The composite loss for unsupervised data augmentation is defined as $\mathcal{L}_{comb} = \mathcal{L}_{task} + \mathcal{L}_{cons}$. The architecture for this framework can be seen in Figure 3.6.

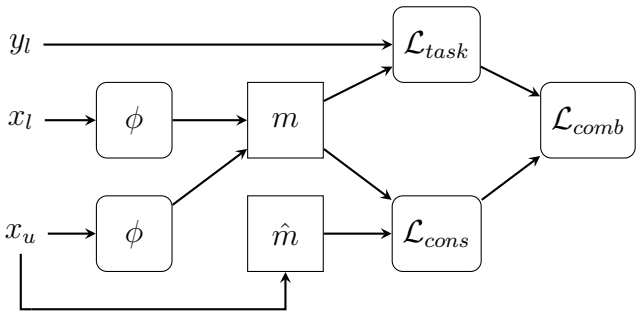


Figure 3.6: Depiction of the unsupervised data augmentation framework. The $\mathcal{L}_{task}$, $\mathcal{L}_{cons}$, and $\mathcal{L}_{comp}$ are the task loss, usually a cross entropy between predictions and expected values, consistency loss between models, and the total model loss. We call the model m and data from the source and target domain x_s and x_t with source labels y_s.

MixMatch

The second algorithm, MixMatch [5], builds upon a regularization technique called MixUp [83]. In MixUp, samples from the training set have their values and respective labels mixed in order to smooth the input space manifold. Basically, for two randomly chosen input and target pairs from the dataset, MixUp computes a new input and target pair:

$$\begin{aligned} \tilde{x} &= \lambda x_1 + (1-\lambda)x_2 \\ \tilde{y} &= \lambda y_1 + (1-\lambda)y_2 \end{aligned} \tag{3.12}$$

where x_1 and x_2 are samples from the training set and λ is a random value sampled from a beta distribution $\beta(\alpha, \alpha)$, with α as a hyperparameter. The new input and target pair $\tilde{x}$ and $\tilde{y}$ are the only values to be used during backpropagation, discarding x_1 and x_2. With this setup and the recommended values from the paper, the beta distribution has a high density for values close to 0 and close to 1, keeping the distortion of the samples fairly close to the original points (values near 0 lead to a sample close to x_2 and values near 1 lead to a sample close to x_1). We show the β distribution for multiple choices of α in Figure 3.7.

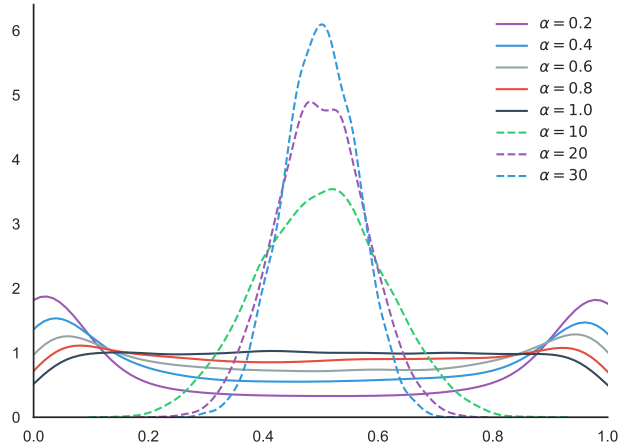


Figure 3.7: $\beta(\alpha, \alpha)$ distribution for multiple α.

In practice, when we increase α, the points from the dataset become more continuous, since the distribution tends to a 0.5-centered gaussian distribution. The tradeoff is clear, we have overfitting as $\alpha \to 0$ (training samples stay the same as we only sample 0 or 1) and underfitting as $\alpha \to \infty$ (samples become too far from their original points). We show the effect in training data for multiple choices of *alpha* in Figure 3.8. We can see how the input space becomes smoother as we increase α, with the drawback of possibly becoming noisier for higher values.

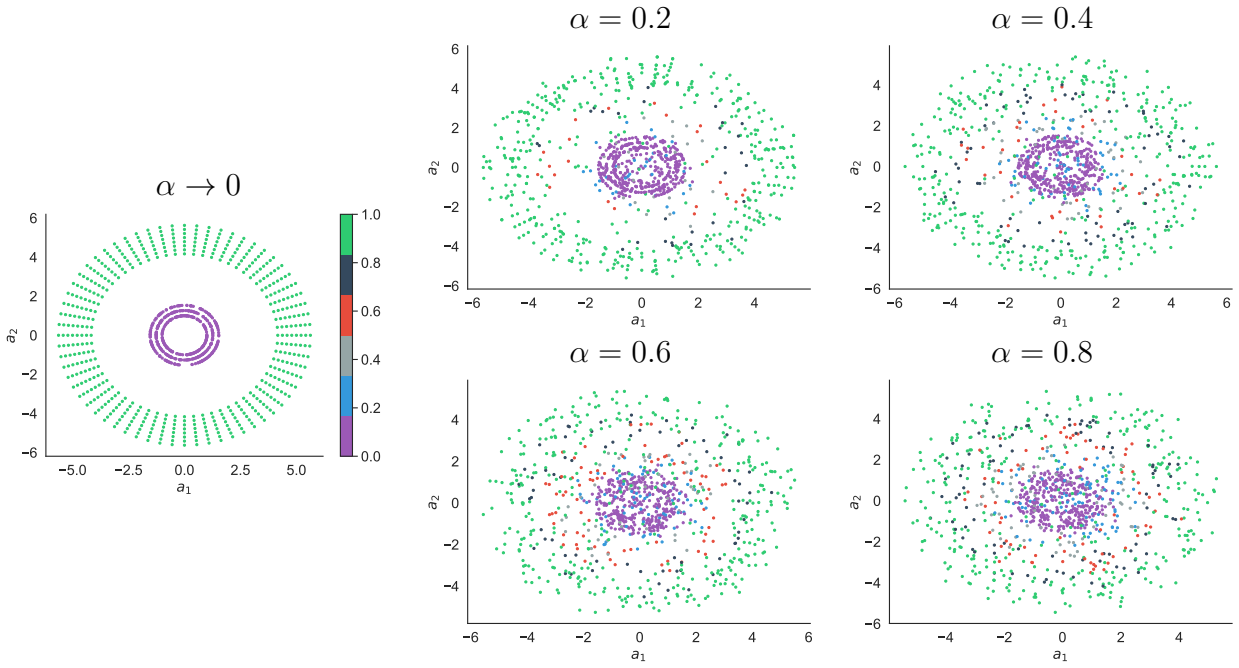


Figure 3.8: MixUp effect for multiple α. This depicts a toy dataset with two attributes called a_1 (x axis) and a_2 (y axis). Colors represent labels.

MixMatch also uses augmentation to enforce consistency among predictions with similar data points. MixMatch receives a labeled and an unlabeled batch as input and returns two batches with pseudo-labels, the ones effectively used for training. We define the algorithm as a sequence of steps that we later depict in a diagram in Figure 3.9:

1. Sample one batch from the labeled set X_l and one from the unlabeled set X_u. The labeled batch is defined as $B_l = \{(x_l^1, y_l^1), ..., (x_l^b, y_l^b)\}$ and the unlabeled batch is defined as $B_u = \{x_u^1, ..., x_u^b\}$ where b is the batch size,

2. Augment each sample x_l from the labeled batch B_l, generating $\hat{B}_l$,

3. Create K augmented samples for each instance x_u from the unlabeled batch B_u, generating $\hat{x}_u$ instances belonging to $\hat{B}_u$. Note that $|\hat{B}_u|$ is equal to $K \times |B_u|$,

4. Compute predictions $\hat{y}_u$ from the model for every augmented unlabeled instance $\hat{x}_u$,

5. Average predictions $\hat{y}_u$ for each unalabeled sample originated from the x_u,

6. Sharpen predictions $\hat{y}_u$ with a temperature-based softmax-like function and update $\hat{B}_u$ to include them as pseudo-labels for their respective samples,

7. Create a set B_c by concatenating $\hat{B}_l$ and $\hat{B}_u$ and shuffle,
8. Create $\tilde{B}_l$ by using the MixUp regularizer between $\hat{B}_l$ with part of B_c,
9. Create $\tilde{B}_u$ by using the MixUp regularizer between $\hat{B}_l$ with the remainder of B_c,
10. Use $\tilde{B}_l$ and $\tilde{B}_u$ as input for the training step.

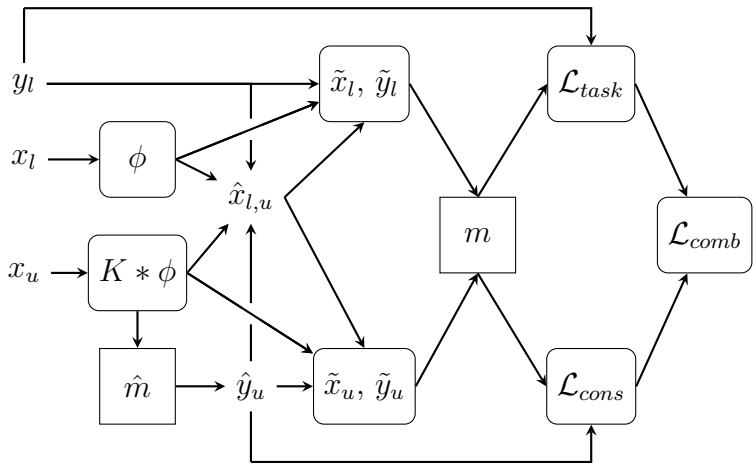


Figure 3.9: MixMatch framework. We call the augmentation protocol ϕ (stages 2 and 3). The model m computes pseudo labels $\hat{y}_u$ (4). Instances are combined to create $x_{l,u}$ (7). We create two independent subsets by applying MixUp ($\tilde{x}_l$ for stage 8 and $\tilde{x}_u$ for stage 9). Finally, the model evaluates the labeled instance and the K unlabeled instances, combining to a final loss $\mathcal{L}_{comb}$. The model $\hat{m}$ is an exact copy of m, but it does not receive gradients to update m.

The final loss function for the MixMatch method is a combination of a labeled and an unlabeled component, which we call here $\mathcal{L}_{task}$ and $\mathcal{L}_{cons}$ to create a parallel with the other two methods, the mean-teacher and the unsupervised data augmentation. We define the loss as:

$$\mathcal{L}_{comb} = \mathcal{L}_{task}(m, \tilde{x}_l, \tilde{y}_l) + \gamma \mathcal{L}_{cons}(m, \tilde{x}_u, \tilde{y}_u) \tag{3.13}$$

where the task loss is an entropy-based function and the consistency is a mean-squared error.

The sharpen function described in step 6 is defined as a softmax-like function of the predictions with a temperature to adjust the *softness* of the result as follows:

$$sharpen(p_j, \tau) = \frac{p_j^{\frac{1}{\tau}}}{\sum_{i=1}^{C} p_i^{\frac{1}{\tau}}} \tag{3.14}$$

where p_j are the average predictions from the model and p_i is the average prediction for class i. C is the number of classes. The temperature τ hyperparameter defines the amount of *sharpening* for the averaged predictions. When $\tau \to \infty$, the distribution will tend to a uniform distribution, while when $\tau \to 0$, the distribution becomes a one-hot. We show this pattern in Figure 3.10.

MixMatch slightly changes the MixUp formulation. Instead of using λ straight from the β distribution, the authors propose to use $\max(\lambda, 1 - \lambda)$. The idea behind it is ensuring

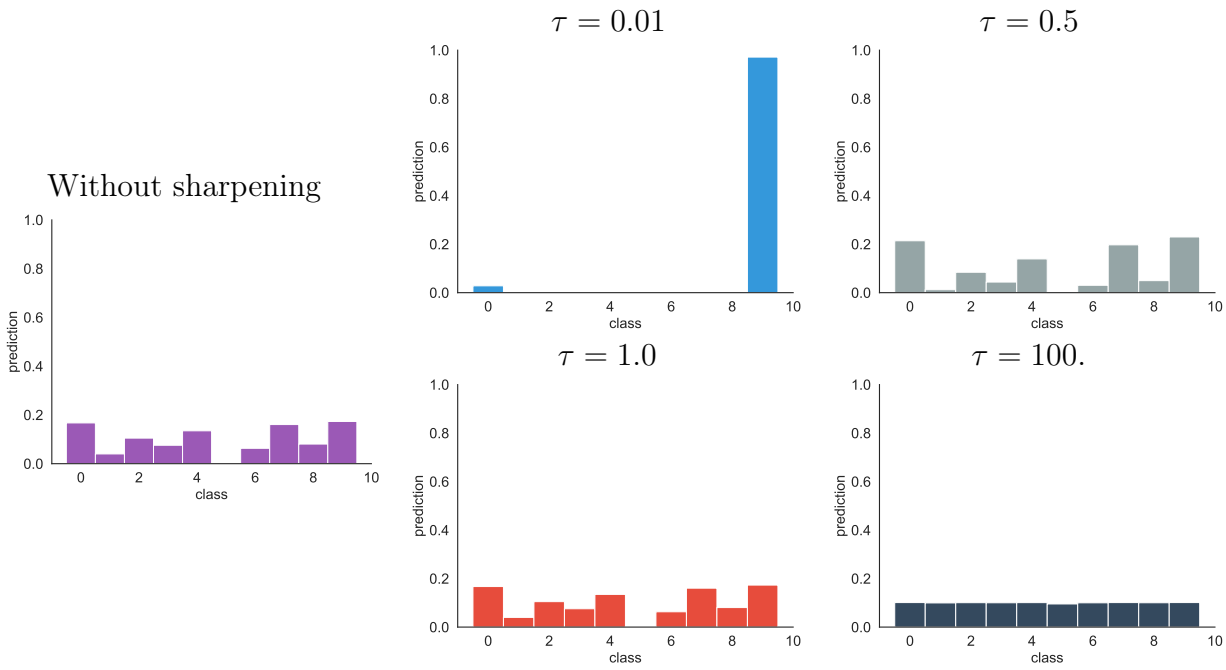


Figure 3.10: Sharpen effect for multiple τ. X axis represents classes (in a hyphotetical 10-class classification task) and Y axis the model prediction for each class.

that when mixing the labeled set with the combined set, λ always enforces more presence of the labeled instance. The same happens when mixing the unlabeled set with the combined set.

3.2 Metrics

The metrics presented here are either frequent for the task of semantic segmentation or for the medical imaging literature. Table 3.1 shows a 2×2 confusion matrix, which will be the basis for building the other metrics. The confusion matrix summarizes the predictions of the model in comparison with the expected values. We use a 2×2 matrix for simplicity and this is what we in fact use in our work, but the concept generalizes for any number of classes.

Table 3.1: Confusion matrix. Rows represent the predicted value from the model while columns are the expected values for each instance. TP, FP, TN, and FN are the true positive, false positive, true negative, and false negative instances.

		Expected	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

3.2.1 Sørensen-Dice coefficient

The Sørensen-Dice coefficient, or just Dice coefficient for short, is used to measure the similarity of a sample and its ground-truth. It has been frequently used in seminal papers of medical imaging segmentation [52]. We can define it in function of its confusion matrix values:

$$Dice = \frac{2TP}{2TP + FP + FN} \tag{3.15}$$

Technically speaking, the dice coefficient is not a metric as it does not follow the triangle inequality. Dice has been empirically shown in ecology literature as an useful measurement [16], working properly when the class distributions are heterogenous. The Dice coefficient is also known as F-score, and is the harmonic average of the precision and recall.

3.2.2 Mean intersection over union

Mean Intersection over Union (mIoU), or the Jaccard index, is also a frequently used metric for semantic segmentation. The intersection over union is frequently seen in object detection papers as a way of determining whether a bounding box was detected or not. We define it in terms of its confusion matrix as follows:

$$mIoU = \frac{TP}{TP + FP + FN}. \tag{3.16}$$

The Jaccard index is very similar to Dice, however, it respects the triangle inequality.

3.2.3 Metrics intuition

Each metric measures a different aspect of how a model treats samples. Alongside the Dice and mIoU, we also employ recall, precision, and specificity. We show how each of them behaves in a 5×5 segmentation example in Figure 3.11. We show two degenerated prediction distributions, one which the model predicts all pixels as positive and another which it predicts all pixels as negative. These examples showcase how either recall (Figure 3.11.a) or specificity (Figure 3.11.b) alone can be maximized with a degenerated model.

Furthermore, we present two other examples, one which the model does not output any false negatives and another which the model does not output any false positives. Indicating which prediction is the best is not trivial. Generally speaking, deciding the best model depends on the application scenario, since the final application might demand a more conservative or a more liberal model. In our case, due to class-imbalance, the Dice loss favors models with less false negatives than false positives, as shown in the Figure 3.11.c and Figure 3.11.d.

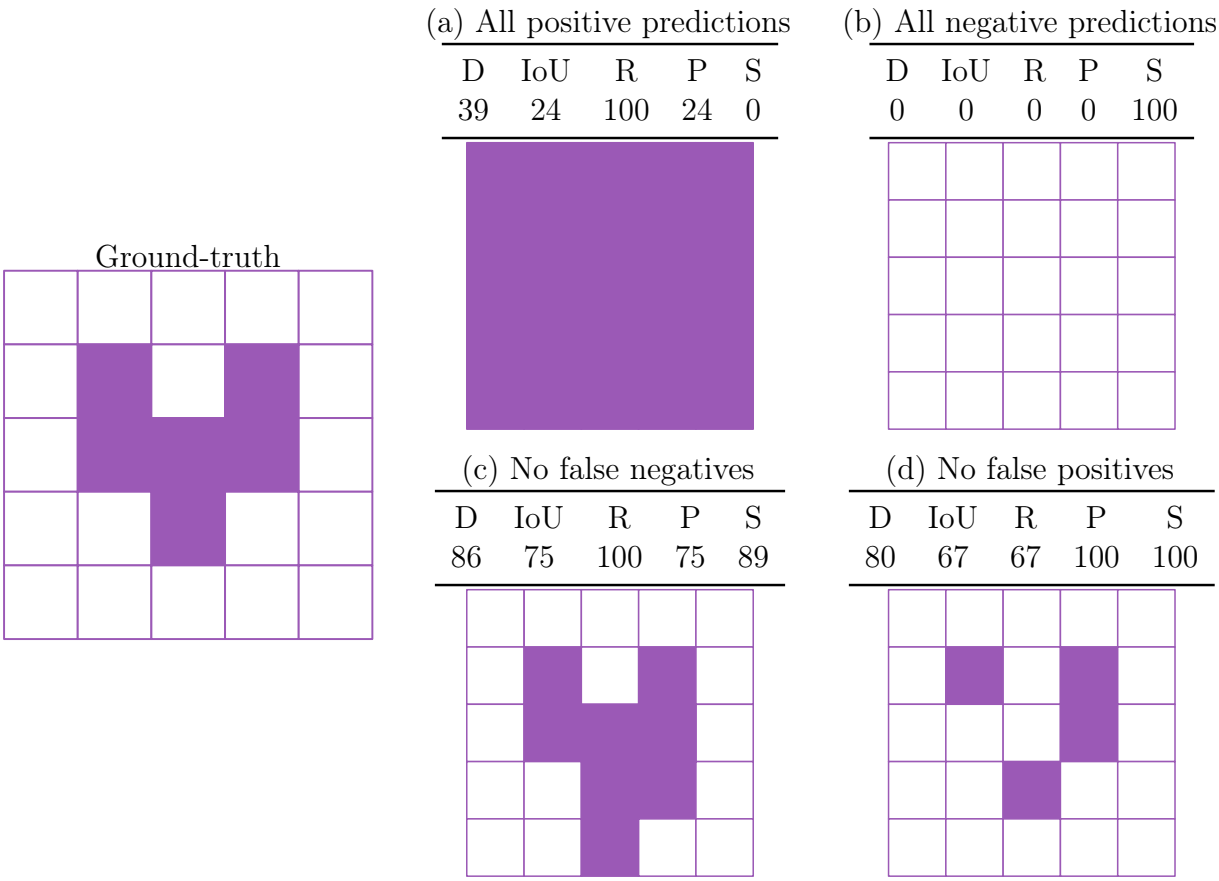


Figure 3.11: Depiction of how each metric behaves. D is the Dice coefficient, IoU is the mean Intersection over Union, R is recall, P is precision, and S is specificity. Colored squares mean positive values.

3.3 Spinal Cord Grey Matter segmentation challenge dataset

The Spinal Cord Grey Matter Segmentation Challenge (SCGM) dataset is a collection of MRI data from four different centers [60]. The set is composed of MRI data from 80 healthy subjects, 20 from each center. Half of the subjects from each center are labeled, totaling 40 spinal cord segmentation pixel-level labels created by 4 independent expert raters. The voxel size resolution ranges from 0.25×0.25×2.5mm to 0.5×0.5×5.0mm.

We had to do some minor changes in the split. The challenge has a server to evaluate center 4 test set, which means that labels are not available for download. We ended up using center 4 test set as the unlabeled examples for domain adaptation and its training set, which we have the labels available, as the test. The final split comprises centers 1 and 2 training set for training, center 3 test set for validation and center 4 training set as test.

Figure 3.12 shows an example from each center from the dataset. We can observe the resolution variability when comparing the samples from each center. Figure 3.13 presents the unnormalized distribution of all centers. We also see a somewhat bimodal distribution at every center, which represent the white and grey matter intensities. The domain shift between each center becomes evident when observing how the intensities range can change and also

how different are the grey and white matter intensities. We see in center 3, for example, a dramatically different presence of low-level intensities with respect to higher ones, while in center 1 they appear in a more uniform manner.

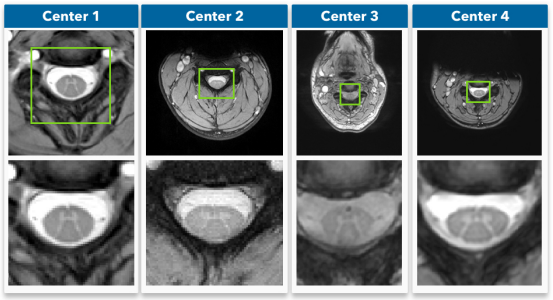


Figure 3.12: Samples from each center from the Spinal Cord Grey Matter Challenge.

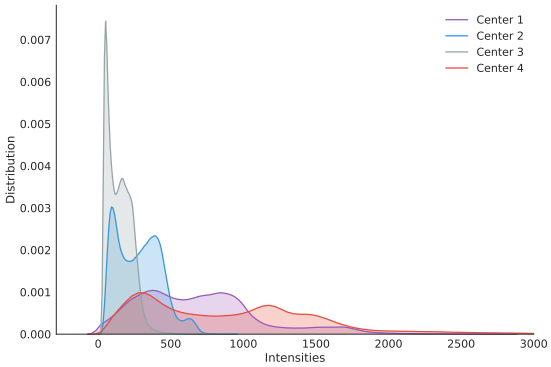


Figure 3.13: Unnormalized pixel intensity distribution for every center in the Spinal Cord Grey Matter segmentation dataset. Colors represent centers.

We conducted all of our experiments in the SCGM dataset. We compare the results for each approach using the metrics detailed in Section 3.2. When conducting adaptation experiments in the ablation, we used the training set from center 3 as adaptation and the validation set as validation. In our final experiments in Section 5.4, we also employ center 4 validation set as adaptation (due to its missing labels) and its training set as validation.

3.4 Baseline

To perform our experiments we needed to find a validated network for the segmentation task. There is a good range of architectures for segmentation, but we decided on one that has been widely applied in medical imaging, U-Net [65]. Using a standard network narrows the interpretation of our results, reducing the number of ablation studies we must conduct.

Since we define our problem as a 2-class segmentation task, our U-Net has as its final layer a sigmoid activation. All slices are center cropped to a $(b, 1, 200, 200)$ shape, and the output is, consequently $(b, 1, 200, 200)$, where b is the batch size. Due to our sigmoid single layered output, we depend on a threshold for deciding between positive (grey matter) and negative (not grey matter) classes. We set this threshold to 0.9 based on our early experiments and never change it throughout the others. We also set the batch size to $b = 12$ for all experiments.

We show the full architecture of our network in Figure 3.14. The only architectural changes during experiments were in the normalization, where we conduct runs to determine which normalization yields the best model, discussed in Section 4.1. Our instantiation of the network always presents two subsequent blocks of convolution with 3×3 kernels with a padding of 1 pixel to maintain the same proportions of the image, normalization, and ReLU layers. We

then do either a downsampling with a 2×2 max pooling (in the first half of the network), a bilinear upsampling with a scale factor of 2 (in the second half of the network), or a 3×3 convolution (prior to the sigmoid activation).

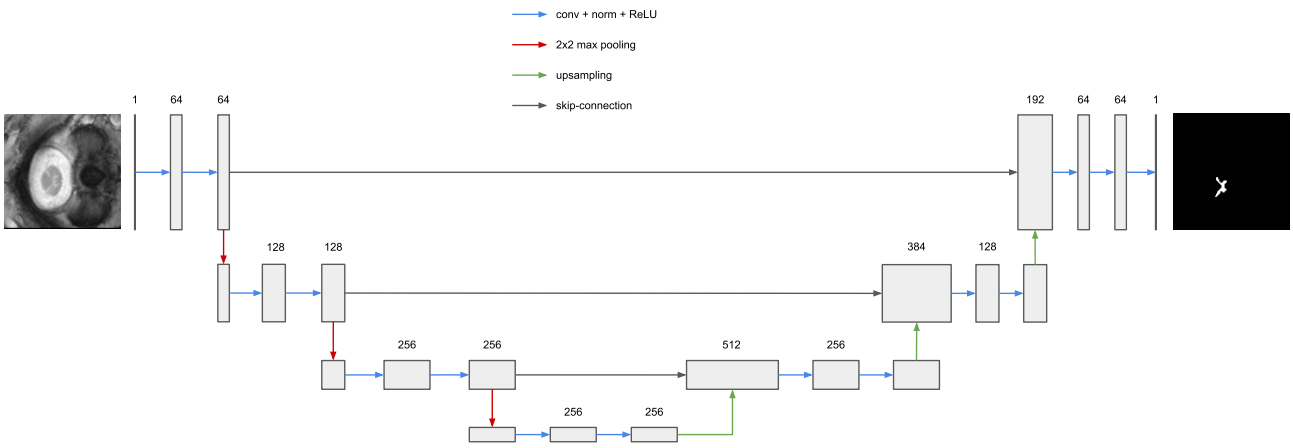


Figure 3.14: The architecture used in all experiments. Boxes represent the volume at a specific part of the network during a forward pass. Numbers above boxes are the number of feature maps at its respective part. Blue arrows represent convolutional operations with 3x3 kernel size. Red arrows are 2x2 max pooling for dimensionality reduction. Green arrows are bilinear upsampling operations. Black arrows represent skip connections (concatenation of feature maps).

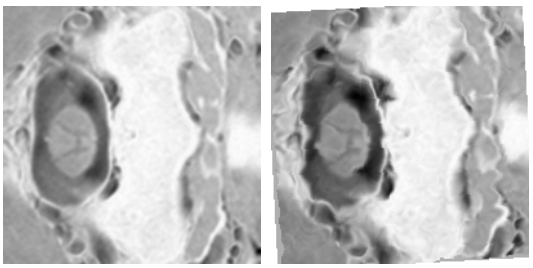
We alter the original loss function for U-Net to a more recent one. We use the dice loss, a loss function based on the dice coefficient, shown in Section 3.2, defined as:

$$Dice = \mathcal{L}_{task}(m, x, y) = -\frac{2 * \sum[m(x) * y]}{\sum m(x) + \sum y} \tag{3.17}$$

where m is a model and y are the expected values for each voxel from each instance x. This loss function was proposed in V-Net [52] for prostate segmentation in MRI scans.

We carried our experiments by implementing the semi-supervised learning methods with the U-Net as the core network. All baseline values we show in the experiments section are a single U-Net trained in a supervised learning fashion on centers 1 and 2 from the SCGM dataset. In order to train the network we use Adam Optimizer [34] with weight decay ($\epsilon = 6e-4$) for regularization alongside a dropout rate of 0.5 at every layer. To further improve regularization, we conduct data augmentation at every batch. We pass every training sample through elastic transformation ($28. \leq \alpha_{et} \leq 30.$, $3.5 \leq \sigma_{et} \leq 4.0$ with a probability of 0.3 of being applied) [69], rescale (factor between 0.98 and 1.2), translation (± 0.03 pixels), rotation (± 4.6 degrees), and uniform noise (between -0.1 and 0.1). We show an input sample and one possible augmented counterpart in Figure 3.15.

We train the network for 350 epochs. We also employ a learning rate schedule to improve the search for the model. We use a sigmoid rampup for 50 epochs until it reaches the maximum value ($\eta = 6e-4$), followed by a cosine rampdown for the remaining epochs. The



Original Augmented

Figure 3.15: Sample and a possible augmented counterpart. The image is inverted for improved visualization.

full learning rate schedule follows the equation:

$$lr(e) = \begin{cases} \eta * rampup(e), & \text{when } e \leq e_{up} \\ \eta * rampdown(e), & \text{otherwise} \end{cases} \tag{3.18}$$

where e is the current epoch and e_{up} is a hyperparameter defining the number of epochs for the rampup to reach the maximum learning rate value η. The rampup is defined as follows:

$$rampup(e) = \exp\left[-5. * (1 - \frac{e}{e_{up}})^2 \right] \tag{3.19}$$

we can also define the rampdown function:

$$rampdown(e) = .5 * \left[cos\left(\pi * \frac{e - e_{up}}{350. - e_{up}} \right) + 1. \right]. \tag{3.20}$$

We can see the behavior of the learning rate schedule in Figure 3.16.

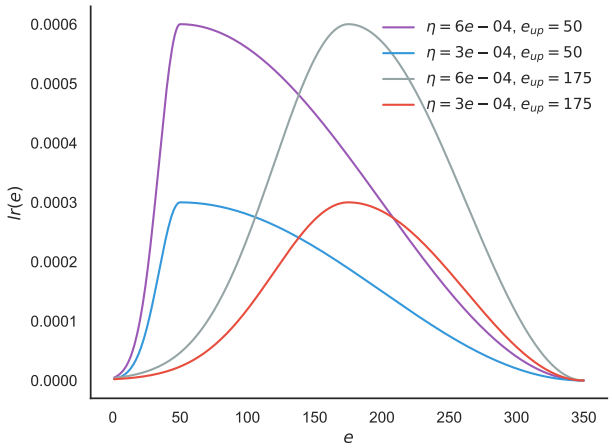


Figure 3.16: Learning rate schedule for different η and e_{up}

In order to observe the magnitude of the domain shift in inner representations of the network, we decided to visualize its predictions. We used t-SNE [46], a non-linear dimensionality reduction technique to reduce our output to two dimensions. We can see the results in Figure 3.17.

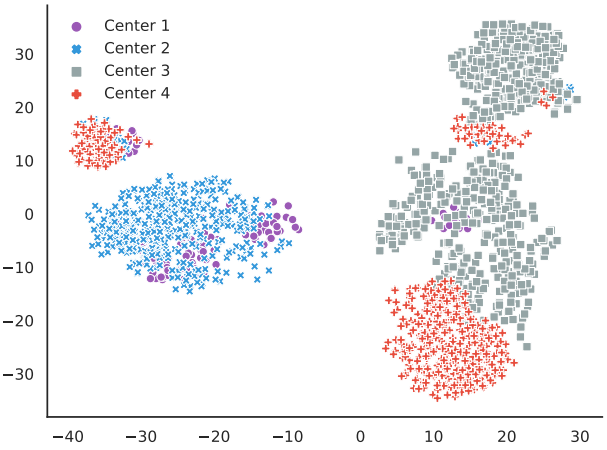


Figure 3.17: t-SNE plot of predictions from the trained baseline model. Colors represent centers.

Each point in the plot represents the model predictions prior to the sigmoid activation for each image unseen during training, where the colors represent the center from where they are sampled. We can clearly observe how clusters formed based on the center from where images are sampled. This indicates that model predictions are highly biased by the source of input, which can implicate in different accuracies over different centers. One can argue as well that points from centers seen during training (1 and 2) have, generally speaking, their points closer, shown by the blue crosses and purple dots cluster. This behavior of a cluster of points for centers seen during training is expected, as these centers represent the distributions the model already included in its learning process.

3.5 Hypotheses

We propose to evaluate two hypotheses: a) *Self-ensembling-based methods are suitable for unsupervised domain adaptation in medical imaging segmentation*; b) *Other state-of-the-art semi-supervised learning algorithms can be applied to our task and surpass the self-ensembling performance.*

Among our objectives, we intend to introduce and help to understand the behavior of semi-supervised learning methods applied to the field of domain adaptation in medical imaging segmentation. We propose several modifications for state-of-the-art methods and validate them in our context, comparing the results of the multiple approaches as fairly as possible.

4. SEMI-SUPERVISED LEARNING ALGORITHMS FOR DOMAIN ADAPTATION IN MEDICAL IMAGING SEGMENTATION

We discussed in Chapter 1 the importance of domain adaptation for medical imaging due to the variability presented in its modalities, in special MRI. We also discussed the possibility of introducing the research on self-ensembling as a suitable domain adaptation for such sensitive scenario. Due to the success of the mean-teacher algorithm for semi-supervised learning [72] and subsequently for domain adaptation [18], we decided to evaluate its predictive performance in our context. Our experiments showed benefits with respect to our baseline on using the domain adaptation version of the mean-teacher, so we modified two recent methods for semi-supervised learning that surpassed the mean-teacher in standard benchmarks to see whether they were able to improve our results. In this chapter we show the changes we did in the network and in each algorithm to prepare them for domain adaptation in a medical imaging segmentation task.

4.1 Network changes in normalization layers

Batch normalization is a normalization strategy originally designed to remove the internal covariate shift — the shift of the distribution that happens inside the network, similarly to the domain shift in the input — to improve convergence and generalization [31]. Although it has been shown that this is not what happens when using batch normalization, and instead the loss function is smoothed [68], consequently introducing more gradient stability, it is still the go-to normalization strategy and it is very useful for many tasks.

In the context of domain adaptation in medical imaging segmentation, batch normalization presents some issues. First of all, during training, each channel is normalized by the average and standard deviation in the batch. With small batches, a common scenario in medical imaging due to the dimensions of the input and output, batch statistics misrepresent the dataset statistics, leading to poor normalization [78]. Second, batch normalization layers keep running averages and standard deviations statistics for each feature map. When done training, inner representations of the network are normalized with the global averages and standard deviations (collected during training). Intuitively, this becomes a problem with data from multiple domains, as the shift over domains would lead to single mean and standard deviation statistics that represent multiple distributions. This has already been somewhat fixed in adaptive batch normalization [41], where multiple statistics are kept in the normalization layers, representing the mean and standard deviation for each domain. This, however, requires the network to know at inference time from which domain the input was sampled.

Another alternative to batch normalization limitations is to use weight normalization [67] instead. In weight normalization, the weight vectors have their norm controlled in order

to improve convergence. Though not depending on minibatch statistics — our main concern with batch normalization — research showed that it fails to yield competitive results in many tasks [78]. We chose group normalization [78] as the normalization method for both our baseline and adaptation experiments. It works better than batch normalization for small batches and also does not require running statistics, which, as discussed in adaptive batch normalization [41], could harm adaptation. Group normalization works independent of batch, and instead divides channels at the layer in groups, computing the mean and variance for group-wise normalization.

4.2 Changes in self-ensembling

We opt to use only the mean-teacher framework as the ensembling-based method due to its improvements over temporal ensembling. We use the terms self-ensembling and mean-teacher interchangeably from this point forward in the text.

To overcome the technical challenges of this task we partnered with an important medical imaging group, the NeuroImaging Research Laboratory at Polytechnique (NeuroPoly) from Polytechnique Montréal. NeuroPoly hosts several important researchers of the field and their expertise on the subject has been of great importance for this part of our work. Our partnership culminated in an approved paper at NeurIPS Medical Imaging Workshop: Medical Imaging Meets NeurIPS (Med-NeurIPS 2018) and in a pre-print paper on ArXiv [58], later published in the Neuroimage journal [57].

4.2.1 Hyperparameters

We use the same hyperparameters from the baseline for all of our experiments with the exception of those specific for the mean-teacher framework. We set the consistency weight (γ) to 1.5 and do a initial rampup for 100 epochs following $\gamma_t(e) = \gamma * rampup(e)$ with e being the current epoch and *rampup* being the sigmoid rampup defined in Equation 3.3. As discussed in the mean-teacher 3.1.4, later stages of training benefit from a higher α (to increase the memory from the exponential moving average). We define $\alpha = 0.99$ until the $50th$ epoch and then set it to 0.999. The batch size for the unlabeled examples is set as the same as the labeled ones (12).

4.2.2 Augmentation

In segmentation tasks, translation and rotation in the input space make the labels mismatch. Furthermore, the framework also assumes that the inputs for student and teacher models have different augmentations. For that reason, we do a small change in how the augmentation is conducted in the framework. We first apply the augmentation procedure in

the input for the student, leaving the input for the teacher untouched. After both inputs have been processed and before both models outputs are compared, we augment the output of the teacher with the same parameters used before in the student input. This leaves us with spatially-matching outputs and also with the variability desired in both models. We formulate this by the equation:

$$\mathcal{L}_{cons}(m_s, m_t, x, \theta) = f(m_s(\phi(x, \theta)), \phi(m_t(x), \theta)) \tag{4.1}$$

where m_s and m_t are the student and teacher models, ϕ is the augmentation with parameters θ, and f is a consistency function, such as mean-squared error or cross-entropy. We later show that the best results we achieve are using the mean-squared error presenting also a more stable training. We also show how the changes affect the loss computation in Figure 4.1.

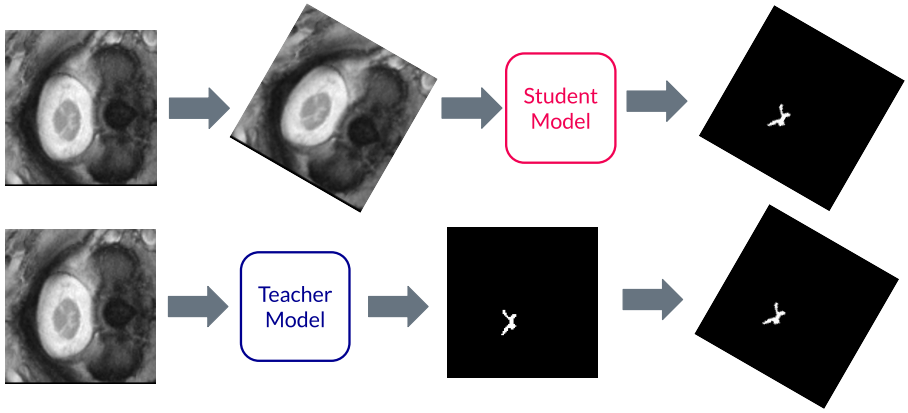


Figure 4.1: Behavior of Equation 4.1

4.3 Changes in unsupervised data augmentation

We have used the same intuition behind the changes for self-ensembling for this method. In order to evaluate the similarity between predictions, we decided to change from the KL-Divergence to the mean-squared error. The KL-Divergence is a distribution similarity function very similar to the cross-entropy, and we assumed that, since the cross-entropy, as we show later, failed to yield a stable training when used as consistency loss in the mean-teacher, we would experience similar behavior in this method. We also transform the output from the model for the non-augmented sample using the same transformation applied to the augmented sample. This makes both outputs directly-comparable, providing more reliable gradients.

Similarly to the mean-teacher consistency equation presented in Equation 4.1, we can define the unsupervised data augmentation consistency as:

$$\mathcal{L}_{cons}(m, x, \theta) = f(m(\phi(x, \theta)), \phi(\hat{m}(x), \theta)) \tag{4.2}$$

where m is the model and $\hat{m}$ is an exact copy of it with no gradients flowing, which means it is not updated during backpropagation. ϕ is the augmentation with parameters θ, and f is a consistency function.

Much like the mean-teacher, we change our task loss $\mathcal{L}_{task}$ to the dice loss. However, the training signal annealing becomes a concern for our task. The original equation was designed to remove highly-confident correct predictions from the gradients computation of the task loss. We use the standard dice loss when not using training signal annealing. In our first attempt of introducing training signal annealing for the task, we adopted the following task loss:

$$\mathcal{L}_{task}(m, x, y) = -\frac{2 * \sum[m(x) * y * mask]}{\sum[m(x) * mask] + \sum[y * mask]} \tag{4.3}$$

where m is a model and y are the expected values for each voxel from each instance x. The arguments for the mask function are ommited. The mask is defined as follows:

$$mask(m, x, y, \eta_t) = \begin{cases} 1, & \text{where } (1 - |m(x) - y|) < \eta_t \\ 0, & \text{otherwise .} \end{cases} \tag{4.4}$$

Intuitively, the mask evaluates the absolute error between the prediction and the expected value and inverts it to represent the confidence. The mask thus becomes 1 when the confidence is below the threshold and 0 otherwise. We preserve the threshold scheduling as presented in Equation 3.11. We introduce three other terms to the loss, one in the numerator and two in the denominator. When training with our dataset, we perceived that the model converged to predict positive in a wide region in the center of the image instead of trying to predict the proper region. We show this pattern in Figure 4.2. We see that the prediction blob fully includes the ground-truth, which means that $y * mask$ is 0 at all (or almost all) voxels. The training thus stagnates and finish as this prediction blob for any annealing schedule.

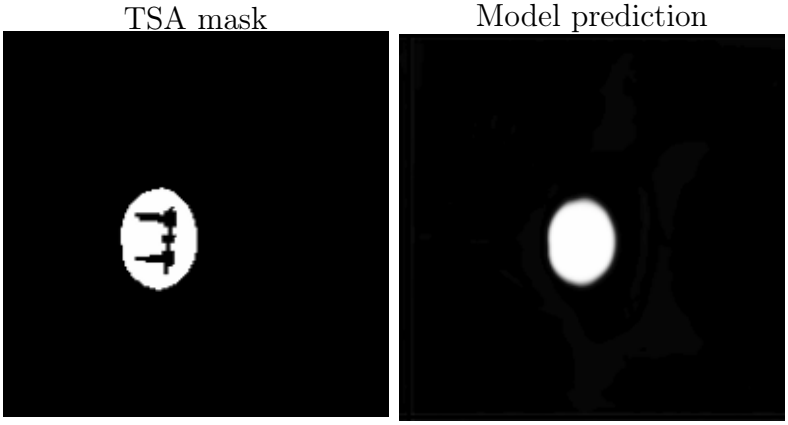


Figure 4.2: Training signal annealing behavior. In the left, the training signal annealing mask for an instance. In the right, model prediction for the same instance.

To change this behavior, we feed the network with additional terms that only focus on the positive (spinal cord grey matter) regions of the image:

$$\mathcal{L}_{task}(m, x, y) = -\frac{2 * \left(\sum[m(x) * y * mask] + \sum[m(x) * y] \right)}{\sum[m(x) * mask] + \sum[y * mask] + \sum[m(x) * y] + \sum y}. \tag{4.5}$$

We can see the change in Figure 4.3. The signal annealing now focuses on border regions (those that are more likely to yield false positives) and the model predictions are somewhat similar to a real ground-truth segmentation mask. We also experimented with other functions, such as the binary cross-entropy to enforce the signal annealing, but all yielded poorer results.

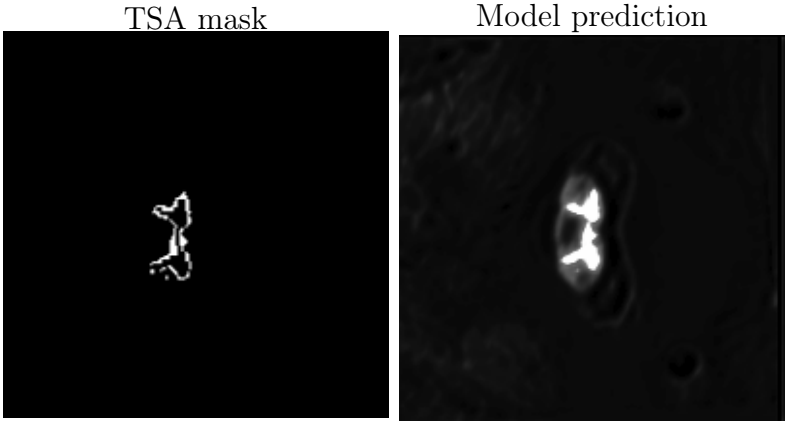


Figure 4.3: Training signal annealing behavior. In the left, the training signal annealing mask for an instance. In the right, model prediction for the same instance.

Finally, the authors find an augmentation policy in a data-driven way using AutoAugment [14]. We, instead, keep the same augmentation policy used in the mean-teacher to preserve a fair comparison. Ultimately, all methods (unsupervised data augmentation, mean-teacher, and MixMatch) should be compared with both the augmentation we empirically found and with an AutoAugment policy, but we leave this for future work.

4.4 Changes in MixMatch

MixMatch poses several challenges when adapting it to our context. First of all, it involves creating K different augmentations and averaging model predictions on them to create pseudo-labels. The first challenge is a consequence of the dense predictions of segmentation. Average predictions from multiple augmented samples leads to a non-ideal scenario that averages are computed over unmatching pixels. To fix this, we are adapting the same idea behind the unsupervised data augmentation and mean-teacher for this problem. We believe that by conducting 1 augmentation for every sample and then aligning all outputs by conducting $K - 1$ augmentations at every sample (one for each of the other K samples) will overcome this limitation. This could become a problem for high values of K, as the distortions would become too large, but the original paper suggests that low values yield the best results (e.g. $K = 2$).

For reference, this should fix the problems that arise from steps 2 through 5 in our definition of MixMatch in Section 3.1.5. We show our intuition in a pseudocode presented in Algorithm 4.1.

Algorithm 4.1: Average Predictions

Input Receives an unlabeled sample x, augmentation procedure ϕ, and a list of independent augmentation parameters θ of size K. The model is called m.
for $i = 1$ **to** K **do**
    $y_k = \phi(x, \theta_i)$
    for $j = 1$ **to** K **do**
        if $i \neq j$ **then**
            $y_k = \phi(y_k, \theta_j)$
$y = \frac{1}{K} \sum_{k=1}^{K} y_k$
return y

The sharpening introduced in MixMatch is a softmax-based function. The output of our network has two classes and is represented with a single value (a threshold divides the two classes). For that reason, we change the sharpening formulation to a sigmoid-based function that should encompass the same characteristics. We formulate in a manner that when $\tau \to 0$ it unsharpens predictions into their maximum uncertainty (0.5) and when $\tau \to \infty$ it becomes a .5-centered step-function (predictions become either 0 or 1). We formulate this as:

$$sharpen(p, \tau) = sigmoid([p - 0.5] * \tau) \tag{4.6}$$

where p are the predictions from the model and τ is the hyperparameter. Sigmoid is a standard sigmoid function defined as $sigmoid(p) = \frac{1.}{1.+exp(-p)}$. We show the behavior of our sharpening function for different values of temperature τ in Figure 4.4.

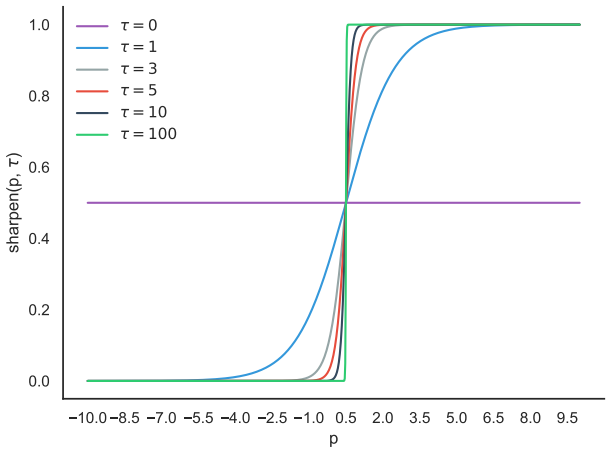


Figure 4.4: Sharpening for multiple τ.

One big issue as well is a dice loss complication. When doing MixUp — steps (8) and (9) of our MixMatch definition — the target becomes continuous, which means that the loss should be computed over two continuous distributions. The dice loss, however, assumes a binary

target distribution. Take, for instance, two single-valued vectors, one to emulate the prediction and another to emulate the target, both set to [0.5]. It is expected for a loss function, when both the prediction and ground-truth have the same value, to present the minimal possible value. Following the formulation of the dice loss, the value for that scenario becomes $\frac{-2*(0.5*0.5)}{.5+.5}$, or -0.5. However, if we take a scenario where the model prediction was 1, we end up with $\frac{-2*1*0.5}{1+0.5}$, or -0.67, a lower number than the former. To deal with this limitation we slightly change the MixMatch loss function so it keeps its regularization properties while including our task loss:

$$\mathcal{L}(m, x, y, \tilde{x}_l, \tilde{y}_l, \tilde{x}_u, \tilde{y}_u) = \mathcal{L}_{task}(m, x, y) + \gamma \mathcal{L}_{cons}(m, \tilde{x}_l, \tilde{y}_l, \tilde{x}_u, \tilde{y}_u) \tag{4.7}$$

where $\mathcal{L}_{task}$ is the dice loss, $\tilde{x}_l$ and $\tilde{y}_l$ are samples from $\tilde{B}_l$, and $\tilde{x}_u$ and $\tilde{y}_u$ are samples from $\tilde{B}_u$ (steps 8 and 9). $\mathcal{L}_{cons}$ is defined as:

$$\mathcal{L}_{cons}(m, \tilde{x}_l, \tilde{y}_l, \tilde{x}_u, \tilde{y}_u) = \frac{1}{|\tilde{y}_l|}||m(\tilde{x}_l) - \tilde{y}_l||^2 + \frac{1}{|\tilde{y}_u|}||m(\tilde{x}_u) - \tilde{y}_u||^2 \tag{4.8}$$

we can see this as the mean-squared error of predictions and pseudo-labels created with MixUp.

With this formulation we can use the dice loss without worrying about non-binary targets, and, more importantly, we can employ the sets from MixUp in a similar fashion to how we compute consistency in other frameworks.

5. EXPERIMENTS

In this section we detail our findings. This chapter is organized in one section for each method, where we discuss our decisions and show the experiments to validate them. We end with a comparison table, where we place the best results we found from each method.

5.1 Self-ensembling ablations and decisions

In order to understand the behavior of self-ensembling in our context, we conduct three experiments. The first one is to explore the importance of the exponential moving average alone in the framework. The next experiment is to check the behavior of many of the possible consistency losses. Finally, we combine the self-ensembling framework with MixUp to see if we are able to improve results by introducing additional regularization.

5.1.1 Exponential Moving Average model

One reason that could explain improvements in the target set results is that the exponentially averaged (EMA) model is better than training in a standard fashion. This in fact is in the roots of the mean-teacher framework given the Polyak averaging concept for improved optimization. We must then perform an experiment to verify whether the EMA model solely explains the improvements of the framework. We conducted several runs of a standard supervised training while exponetially averaging the model with the same α used in the mean-teacher. We then used a paired t-test over every recorded metric to verify if there is a statistically significant difference between the two models. The results are summarized in Table 5.1. We achieve these results by setting the consistency weight γ to 0.

Table 5.1: Results of the ablation experiment where the baseline model was trained and compared against its Exponential Moving Average (EMA) model without using any unlabeled data. All models were trained with both centers 1 and 2. Center 3 is the validation set. We show the two-tailed p-value from a paired t-test for each metric between the baseline and EMA models.

Evaluation	Version	Dice	mIoU	Recall	Precision	Specificity
Center 3	Baseline	82.94 ± 0.35	71.20 ± 0.41	90.48 ± 0.45	77.39 ± 0.39	99.86 ± 0.00
	EMA	82.97 ± 0.34	71.24 ± 0.40	90.51 ± 0.43	77.42 ± 0.40	99.86 ± 0.00
	p-value	0.0024	0.0013	0.0429	0.0102	0.0201

We can see that the p-value presents significant differences in some cases, but this difference alone between the two models does not explain the mean-teacher gains. This means that the responsible for the improvements in the target dataset is the framework itself. One remarkable characteristic is that, although the EMA model presented just a small improvement

in the evaluated metrics, self-ensembling was able to leverage it to heavily improve target set results.

5.1.2 Consistency loss

Consistency losses must be functions that evaluate the similarity between two distributions. In this case, as defined by the mean-teacher framework, the consistency loss evaluates the similarity between predictions of the teacher and student models. For the following equations we ignore the augmentation, but they follow the same characteristics detailed in Equation 4.1.

Alongside the dice loss, already described in Equation 3.17, we also define two other functions as consistency losses. One of them is the mean-squared error (MSE) of the predictions:

$$\mathcal{L}_{MSE}(m_{stu}, m_{tea}, x) = \frac{1}{N}\sum(m_{stu}(x) - m_{tea}(x))^2 \tag{5.1}$$

where N is the total number of examples in the batch. The student and teacher models are represented as m_{stu} and m_{tea}, respectively. The result of evaluating an arbitrary model $m_{(\cdot)}$ in an instance x is defined as $m_{(\cdot)}(x)$. We also define the cross-entropy loss as another option:

$$\mathcal{L}_{CE}(m_{stud}, m_{teac}, x) = -\sum m_{stud}(x)\log[m_{teac}(x)] \tag{5.2}$$

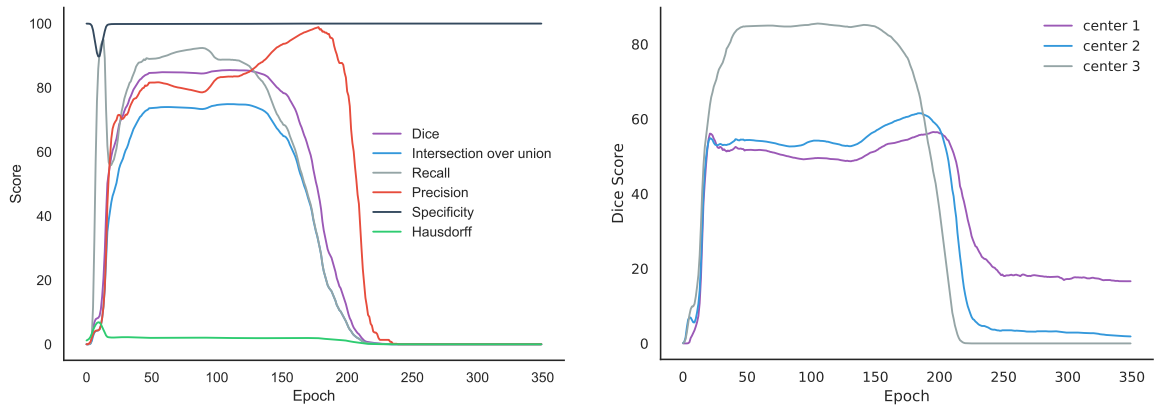
We observed that the dice loss tends to yield worse results when compared with MSE as consistency loss. Cross-entropy also performs worse than MSE. For cross-entropy and dice we have noticed a divergence after several iterations, thus it might not be a reasonable option since sometimes in unsupervised domain adaptation we do not have any labeled examples to validate our model. We show the final results of 350 epochs of training in Table 5.2. We see that the mean-squared error has been robust to changes in its weight on the full loss. Other metrics are susceptible to the choice of the consistency weight, making the process of hyperparameter selection more complex, specially due to the lack of labeled samples for the target set. This instability is also seen by observing the best model achieved with every weight. The best achieved model (the one in parenthesis) using MSE with every consistency weight is very similar to the average, while in both Dice and cross-entropy the models are very inconsistent.

We can observe that cross-entropy consistently fails, even with different weights, potentially due to the class imbalance of this particular task. At the same time, it also achieves high dice values in its best epoch during training. Thus cross-entropy becomes a possible alternative to MSE when a few annotated images are available for validation in the target domain. Figure 5.1.a shows how the training diverges for cross-entropy after several iterations.

One way to alleviate this issue is to conduct an early stopping in the training. However, as we must assume that there are no labeled examples from the target center, the early stopping must be conducted with data from source centers. We investigated whether the epoch when

Table 5.2: Results on evaluating on center 3. The training set includes centers 1 and 2 simultaneously, with unsupervised adaptation for center 3. Values within parenthesis represent the best validation results achieved during training for each metric. The remaining values represent the final result after 350 epochs. Dice is the Dice coefficient and CE is the cross entropy. In bold are the best value for the metric among the experiments (both for the final and the best during training results).

Loss	Weight	Dice	mIoU	Recall	Precision	Specificity
CE	5	0.00 (**85.50**)	0.00 (**74.91**)	0.00 (95.01)	0.00 (**98.90**)	**100.0** (**100.00**)
	10	0.00 (80.73)	0.00 (69.54)	0.00 (83.21)	0.00 (98.78)	**100.0** (**100.00**)
	15	6.43 (37.03)	4.89 (26.06)	5.38 (77.05)	17.34 (65.85)	**100.0** (**100.00**)
	20	2.30 (67.61)	1.86 (52.55)	2.09 (65.00)	7.94 (96.57)	**100.0** (**100.00**)
Dice	5	76.76 (80.74)	62.76 (68.16)	97.88 (99.66)	63.72 (72.50)	99.71 (99.81)
	10	4.77 (10.55)	2.45 (5.64)	96.25 (99.99)	2.45 (5.85)	79.59 (99.75)
	15	2.30 (7.74)	1.16 (4.12)	99.95 (**100.00**)	1.16 (4.62)	55.07 (99.80)
	20	1.79 (4.43)	0.90 (2.27)	**99.99** (**100.00**)	0.90 (2.30)	42.02 (99.84)
MSE	5	83.7 (83.88)	72.2 (72.46)	91.24 (98.19)	78.1 (78.57)	99.87 (99.93)
	10	84.38 (84.38)	73.19 (73.19)	90.15 (99.07)	80.12 (80.12)	99.88 (99.94)
	15	**84.59** (84.59)	**73.49** (73.50)	89.19 (98.52)	**81.28** (81.28)	99.89 (99.89)
	20	84.5 (84.50)	73.36 (73.37)	90.36 (94.63)	80.16 (80.16)	99.88 (99.98)



(a) (b)

Figure 5.1: Per-epoch validation results for the teacher model at center 3 with cross-entropy as consistency loss. Training in both centers 1 and 2 simultaneously, and adapting to center 3 with consistency weight $\gamma = 5$. Best viewed in color. In (a) we present every metric for Center 3 over the training progress. In (b) we present only the Dice coefficient for centers 1, 2, and 3.

drop in scores happen in the target center matches the one in source centers. We found that the drop somewhat appears at the same moment, but, at least in our experiment setting, the target center dropped a bit earlier, as shown in Figure 5.1.b. This means that the early stopping should be conservative, choosing a model of several stages prior to when the drop in scores effectively occurred. A few annotated data from the target set might also mitigate this problem.

Overall, the MSE has outperformed the other consistency losses due to its stability and robustness with several weights. However, there is a clear difference on the best precision value achieved using cross-validation (and also for dice). Depending on the needs of a scenario (e.g. a need for high precision), we can modify the loss in order to fit the model to be most useful.

5.1.3 Beyond standard augmentations in the mean-teacher

We decided to combine the augmentations we used in the mean-teacher experiments with MixUp. We believe that by introducing additional regularization between the student and teacher, the overall results tend to be better. We also use different λ to interpolate instances for the teacher and the student, introducing maximum consistency, which we empirically found to be better than using the same value for both. In our case, the MixUp occurs after data augmentation, much like in MixMatch. We show the results in Table 5.3.

Table 5.3: Mean-teacher with MixUp regularization tested with multiple α from $\beta(\alpha, \alpha)$.

α	**Dice**	**mIoU**	**Recall**	**Precision**	**Specificity**
0.2	83.39	71.75	**91.53**	77.36	99.86
0.4	83.59	72.04	90.90	78.16	**99.87**
0.6	83.57	72.02	90.48	78.44	**99.87**
0.8	83.47	71.84	89.78	**78.81**	**99.87**
1.0	**83.64**	**72.10**	90.36	78.66	**99.87**

Early experiments with a low number of epochs presented an improved result when using MixUp alongside standard augmentation. However, when we conduct our full experiment (350 epochs), the model performed poorer than without MixUp. We argue that this is a promising research path and we will investigate this further in future work.

5.2 Unsupervised data augmentation ablations and experiments

We move on to explore how unsupervised data augmentation performs with respect to changes in its hyperparameters. We investigate two hyperparameters that might influence the model predictive performance. First, we observe the behavior of multiple choices of consistency weight. Furthermore, we employ the consistency ramp-up from the mean-teacher framework to see whether unsupervised data augmentation can benefit from it. Second, we try all the schedules proposed in the original paper for training signal annealing.

5.2.1 Consistency weight

We test the best consistency weight for unsupervised data augmentation. We also compare the behavior of the consistency weight when using the ramp-up as proposed in the mean-teacher. We show the results in Table 5.4.

Table 5.4: Trained in source centers 1 and 2 and adapted in center 3.

Weight	Ramp-up	Dice	mIoU	Recall	Precision	Specificity
1	No	83.71	72.27	90.82	78.44	99.87
1	Yes	83.70	72.25	91.49	77.92	99.86
2	No	84.11	72.80	89.57	80.12	99.88
2	Yes	84.18	72.92	91.22	78.95	99.87
5	**No**	**84.29**	**73.06**	88.99	**80.92**	**99.89**
5	Yes	83.49	71.94	**91.87**	77.28	99.86
10	No	83.94	72.54	90.23	79.29	99.88
10	Yes	83.96	72.59	91.34	78.45	99.87

The framework seems to be fairly stable against changes in the consistency weight. We keep consistency weight as 5 for the rest of the experiments, which is the one that maximized the dice coefficient.

5.2.2 Training signal annealing scheduling

The training signal annealing serves the purpose of removing highly-confident samples from the computation of the gradients, as discussed in Section 3.1.5. The paper proposes three different schedules: linear, logarithmic, exponential. We compare each of them in Table 5.5. We run this experiment with consistency weight $\gamma = 5$ without rampup.

Table 5.5: Training signal annealing with different scheduling. Trained in source centers 1 and 2 and adapted in center 3. Training was conducted with the best weight-rampup combination found in the previous experiment.

Schedule	Dice	mIoU	Recall	Precision	Specificity
No annealing	**84.29**	**73.06**	88.99	**80.92**	**99.89**
Linear	81.71	69.45	**95.47**	72.11	99.81
Logarithmic	82.00	69.85	95.03	72.81	99.81
Exponential	83.06	71.38	94.71	74.69	99.83

Unfortunately, the training signal annealing proposed in unsupervised data augmentation does not translate well to our task. It is possible that changes can improve these results, but we do not explore this further and leave it for future work.

5.3 MixMatch ablations and experiments

We want to observe the impact of each of the most important points of the MixMatch algorithm. We conduct the experiments in this section in the same order we present them,

keeping the hyperparameter with the best results for the remaining experiments. The best set of hyperparameters is used when comparing with other algorithms in Section 5.4.

5.3.1 Consistency weight

We evaluate multiple values for consistency weight. We also employ the consistency weight scheduling proposed in the mean-teacher, much like we did in unsupervised domain adaptation. Full results for validation in center 3 are in Table 5.6.

Table 5.6: Multiple weights for consistency weight, evaluated both with and without rampup.

Weight	**Ramp-up**	**Dice**	**mIoU**	**Recall**	**Precision**	**Specificity**
1	No	83.47	71.95	**93.50**	76.14	99.85
1	Yes	83.42	71.87	93.36	76.15	99.85
2	No	83.43	71.90	93.36	76.18	99.85
2	Yes	83.56	72.09	93.34	76.40	99.85
5	No	67.04	56.46	71.58	68.56	**99.89**
5	**Yes**	**83.84**	**72.49**	93.34	76.89	99.85
10	No	66.30	55.55	70.20	69.01	99.88
10	Yes	83.21	71.94	90.81	**78.06**	99.87

We see a pattern of how important the consistency ramp-up is when we drastically increase the consistency weight. For high values of weight, no ramp-up means that the model uses too much information from poor model predictions instead of using labeled data supervision. Ultimately, training is overall more stable with ramp-up, as we can increase the consistency weight without worrying about underfitting.

5.3.2 β distribution

The MixUp algorithm relies on a $\beta(\alpha, \alpha)$ distribution to sample λ and combine two samples. Lower values of α and, consequently, higher values of λ present more influence in the final sample from one instance than another (λ close to 1). The opposite (high α and, consequently, low λ) causes the final sample to be an average of the two samples ($\lambda = 0.5$). We summarize the results for multiple α in Table 5.7 to see which one works best in our task.

We see three patterns. First, when values of α get closer to 0, the model becomes more similar to the baseline model. Second, as we increase α, the dice coefficient increases. However, high values of α get λ (the value sampled from the β distribution) to get too close to 0.5, inducing an underfitted model (which we see clearly with $\alpha = 5$ and forward).

Table 5.7: Multiple values for α from $\beta(\alpha, \alpha)$.

α	**Dice**	**mIoU**	**Recall**	**Precision**	**Specificity**
0.2	83.62	72.16	**93.52**	76.38	99.85
0.4	83.84	72.49	93.34	76.89	99.85
0.6	**83.88**	**72.54**	93.17	77.06	99.85
0.8	83.70	72.28	93.29	76.68	99.85
1.0	83.85	72.50	93.01	**77.13**	99.86
5.0	78.07	66.55	85.43	74.53	99.86
10.0	67.20	56.40	70.51	69.86	**99.90**

5.3.3 Sharpen τ

The sharpening hyperparameter τ influences how much predictions from the MixUp samples tend to either 0 (when prediction < 0.5) or 1 (when prediction > 0.5) after they are evaluated by the model and averaged. We hypothesize that low values of τ would lead to underfitting, since all predictions tend to 0.5, and high values of τ would lead to suboptimal results due to a nonsmooth loss. We experiment with different values of τ and present the full results in Table 5.8.

Table 5.8: Multiple τ for the sharpen function $sharpen(p, \tau)$. Trained in source centers 1 and 2 and adapted in center 3.

τ	**Dice**	**mIoU**	**Recall**	**Precision**	**Specificity**
1	59.28	49.05	61.07	62.03	**99.89**
3	83.73	72.36	93.37	76.71	99.85
5	83.68	72.26	93.35	76.61	99.85
10	**83.93**	**72.63**	93.10	**77.19**	99.86
20	83.72	72.30	93.34	76.67	99.85
50	83.81	72.45	**93.42**	76.77	99.85

We observed that sharpening predictions was not among the most important parts of our framework. We see a gain of using $\tau = 10$ against no sharpening, but with marginal gains that might vanish at multiple executions. We keep $\tau = 10$ for the rest of the experiments, as it might be more stable when using alongside $K > 1$ augmentations.

5.3.4 K Augmentations

The number of instance augmentations highly influence the amount of consistency for distortions within the model. When $K > 1$, the model must learn to handle multiple augmented samples as they were the same, potentially improving the prediction manifold. However, due to limitations of the MixMatch for segmentation tasks (Section 4.4), increasing the number of

augmented samples might also introduce too much noise in the network. One possible way to alleviate this is to rescale the parameters from the augmentation policy according to the choice of K. We leave this analysis for future work. We present the results for $K = 1$ and $K = 2$ in Table 5.9.

Table 5.9: Number of independent augmentations for each unlabeled sample. Trained in source centers 1 and 2 and adapted in center 3.

K	**Dice**	**mIoU**	**Recall**	**Precision**	**Specificity**
1	**83.93**	**72.63**	**93.10**	**77.19**	99.86
2	70.93	59.87	76.16	69.92	**99.87**

We show that, based on our methodology on how to replicate the K augmentations proposed in MixMatch, the results fall dramatically. We thus keep MixMatch with $K = 1$.

5.4 Comparative results

We want to answer two important questions regarding each of the approaches we evaluated: a) Can the frameworks improve the results with respect to the baseline (just training in centers 1 and 2)? b) Does adapting with unlabeled data from another center improves generalization (for instance, are unlabeled data from center 3 improving results in center 4)?

We see in Table 5.10 the full results of our experimental analysis. We ran 10 executions for each combination: Centers 1 and 2; Centers 1 and 2 adapted with Center 3; Centers 1 and 2 adapted with Center 4. We can see that the response when unlabeled data is included in the training set is consistently better than the baseline in all metrics except for recall.

The most important lines of our analysis are Center 4 evaluation and adaptation with Center 4 compared against Center 4 evaluation baseline. We see a baseline value of 69.41 against a mean-teacher result of 74.67, which represents more than 5 points of dice coefficient gain. Unsupervised data augmentation also surpassed the baseline, while MixMatch, in contrast with the same experiment in Center 3, was not able to improve baseline results.

Furthermore, the best models for both centers 3 and 4 happen when the adaptation data come from their own domain. This shows that, although introducing unlabeled data from other domains to the training produces improvements, better response is achieved when using data from the same center. We show this pattern in Table 5.10 in Center 3 evaluation and Center 3 adaptation rows, and Center 4 evaluation and Center 4 adaptation rows.

Unfortunately we could not surpass the results we set with the mean-teacher framework with the other two methods. However, we were able to effectively adapt both unsupervised data augmentation and MixMatch to our context, taking into consideration that both outperformed the baseline in most cases, with the exception of MixMatch adapting with Center 4. Interestingly, both the mean-teacher and unsupervised data augmentation were able to improve results for the

Table 5.10: Evaluation results in different centers. The evaluation and adaptation columns represent the centers where testing and adaptation data were collected, respectively. Results are averages and standard deviations over 10 runs (with independent initialization of random weights). Values highlighted represent the best results at each center. All experiments were trained in both centers 1 and 2 simultaneously.

Evaluation	Adaptation	Method	Dice	mIoU	Recall	Precision	Specificity
Center 1	-	Baseline	47.25 ± 0.10	31.46 ± 0.08	94.90 ± 0.29	32.18 ± 0.09	99.66 ± 0.00
	Center 3	Mean-teacher	47.71 ± 0.16	31.84 ± 0.14	94.18 ± 0.16	32.69 ± 0.15	99.67 ± 0.00
		UDA	47.48 ± 0.14	31.66 ± 0.12	94.95 ± 0.29	32.40 ± 0.16	99.66 ± 0.00
		MixMatch	46.78 ± 0.06	31.15 ± 0.05	**99.12** ± 0.04	31.27 ± 0.05	99.63 ± 0.00
	Center 4	Mean-teacher	**48.42** ± 0.92	**32.47** ± 0.80	94.51 ± 0.57	**33.33** ± 0.93	**99.68** ± 0.02
		UDA	47.61 ± 0.18	31.77 ± 0.16	95.05 ± 0.20	32.50 ± 0.18	99.66 ± 0.00
		MixMatch	46.94 ± 0.05	31.30 ± 0.05	99.10 ± 0.04	31.43 ± 0.05	99.63 ± 0.00
Center 2	-	Baseline	50.69 ± 0.09	34.44 ± 0.08	94.79 ± 0.24	35.32 ± 0.10	99.61 ± 0.00
	Center 3	Mean-teacher	51.05 ± 0.25	34.76 ± 0.23	93.78 ± 0.42	35.83 ± 0.31	99.62 ± 0.01
		UDA	50.85 ± 0.10	34.59 ± 0.09	94.85 ± 0.21	35.46 ± 0.11	99.61 ± 0.00
		MixMatch	50.58 ± 0.06	34.45 ± 0.06	98.93 ± 0.02	34.65 ± 0.06	99.58 ± 0.00
	Center 4	Mean-teacher	**51.29** ± 0.67	**34.98** ± 0.61	93.87 ± 0.91	**36.06** ± 0.82	**99.63** ± 0.02
		UDA	50.86 ± 0.14	34.60 ± 0.13	95.06 ± 0.27	35.44 ± 0.16	99.61 ± 0.00
		MixMatch	50.73 ± 0.03	34.59 ± 0.03	**98.89** ± 0.02	34.80 ± 0.03	99.58 ± 0.00
Center 3	-	Baseline	82.81 ± 0.33	71.05 ± 0.36	90.61 ± 0.63	77.09 ± 0.34	99.86 ± 0.00
	Center 3	Mean-teacher	**84.72** ± 0.18	**73.67** ± 0.28	87.43 ± 1.90	**83.17** ± 1.62	**99.91** ± 0.01
		UDA	83.99 ± 0.13	72.62 ± 0.19	90.42 ± 0.27	79.23 ± 0.37	99.88 ± 0.00
		MixMatch	83.79 ± 0.09	72.42 ± 0.13	**93.13** ± 0.13	76.96 ± 0.20	99.85 ± 0.00
	Center 4	Mean-teacher	84.45 ± 0.14	73.30 ± 0.19	87.13 ± 1.77	82.92 ± 1.76	**99.91** ± 0.01
		UDA	83.47 ± 0.15	71.93 ± 0.20	90.04 ± 0.47	78.64 ± 0.35	99.87 ± 0.00
		MixMatch	83.38 ± 0.25	71.84 ± 0.33	92.14 ± 0.42	77.00 ± 0.20	99.86 ± 0.00
Center 4	-	Baseline	69.41 ± 0.27	53.89 ± 0.31	97.22 ± 0.11	54.95 ± 0.35	99.70 ± 0.00
	Center 3	Mean-teacher	73.27 ± 1.29	58.50 ± 1.57	94.92 ± 1.48	60.93 ± 2.51	99.77 ± 0.03
		UDA	70.55 ± 0.30	55.24 ± 0.35	96.85 ± 0.11	56.52 ± 0.41	99.72 ± 0.01
		MixMatch	68.08 ± 0.15	52.35 ± 0.18	**97.69** ± 0.04	53.18 ± 0.19	99.68 ± 0.00
	Center 4	Mean-teacher	**74.67** ± 1.03	**60.22** ± 1.24	93.33 ± 1.96	**63.62** ± 2.42	**99.80** ± 0.02
		UDA	70.73 ± 0.40	55.45 ± 0.48	96.64 ± 0.20	56.82 ± 0.57	99.73 ± 0.01
		MixMatch	68.49 ± 0.12	52.82 ± 0.14	97.40 ± 0.04	53.78 ± 0.15	99.69 ± 0.00

source domain as well. By looking at evaluation Center 1 and adaptation Center 4 lines, we see the mean-teacher increasing Dice by over 1 point (48.42 against 47.25). The Dice improvements over Centers 1 and 2 are unintended since the domain adaptation task does not aim at enhancing prediction in source centers. This finding shows that methods from the semi-supervised learning literature applied to the task of domain adaptation do not have to sacrifice their predictive performance on the source domains in favor of the target. Moreover, it seems that the additional data from other domains can even help in modeling the source domains, which paves the way for more generalized models that work across multiple centers.

We also see an overall trade-off of precision and recall. No method achieved simultaneously the best precision and recall in any center. MixMatch achieved the best recall at every center, while the mean-teacher lost to the baseline in all scenarios at the same metric. This shows that MixMatch might be a more suitable method when the model is expected to present more false positives than false negatives, but overall presenting more unreliable predictions. The specificity of all models in all centers is somewhat similar. The high specificity is most likely related to the high class-imbalance of the dataset, favoring the negative class, which consequently is more easily modeled. Interestingly, based on the specificity consistency across experiments, we can attest that the Dice coefficient improvements from all methods over the baseline did not have to sacrifice the quality of negative pixels predictions.

As discussed, the results with unsupervised data augmentation were slightly better than the ones with MixMatch, in contrast with MixMatch presenting a better result in their original task of semi-supervised learning for object classification. We believe that this happens mostly because the adaptation for MixMatch in our task was less straight-forward, leading to decisions that were not entirely validated due to computational limitations and a combinatorial explosion of necessary experiments.

6. CONCLUSIONS

In this work we discussed what is domain adaptation, how it relates to other fields of machine learning and deep learning and its importance for medical imaging. We explained the task of semantic segmentation and detailed some of the nonstandard machine learning paradigms, such as semi-supervised learning, transfer learning, and multitask learning. Finally, we presented our methodology on how to solve the task of domain adaptation for spinal cord grey matter segmentation.

We demonstrated how a state-of-the-art method for domain adaptation, originally from the semi-supervised learning literature, can be used in our scenario. Based on that, we also adapted two other semi-supervised learning algorithms to domain adaptation in medical imaging segmentation to compare their results, namely, unsupervised data augmentation and MixMatch. Both unsupervised data augmentation and MixMatch were released in early 2019 and still had not been used for semantic segmentation, presenting challenges on how to prepare the methods for this new scenario. Furthermore, both methods were only used for semi-supervised learning, so employing them in domain adaptation was also novel.

Retaking what we discussed in Section 3.5, we proposed two hypotheses:

A. Self-ensembling-based methods are suitable for unsupervised domain adaptation in medical imaging segmentation,

B. Other state-of-the-art semi-supervised learning algorithms can be applied to our task and surpass the self-ensembling performance.

We can say, based on the results presented in Section 5.4, we can accept our first hypothesis. The mean-teacher is able to perform better than our baseline in every center, and thus is a suitable method for the task. The mean-teacher outperforms the baseline in every metric we evaluated, with the exception of recall, which is sacrificed in favor of more precision.

In the current state of our work, we are not able to definitely reject nor accept our second hypothesis. Although our results show that mean-teacher consistently outperformed unsupervised data augmentation and MixMatch, we still cannot attest that they are worse for the task. We made several decisions when adapting the algorithms, ranging from the way we model our loss in MixMatch to how we design training signal annealing in the unsupervised data augmentation. This means that, most likely, there are better ways we did not explore of modeling each of our changes, improving the overall results and, eventually, even surpassing the mean-teacher performance.

Furthermore, the baseline decisions such as the learning rate and learning rate schedule might not fit as well for the other methods as they do for the mean-teacher. To overcome this limitation, experiments with random search and grid search for all the hyperparameters, the ones from the baseline and the ones specific for each framework, should be conducted jointly. This is very computationally demanding, and thus, we leave it for future work.

Additionally, we provided two more marginal contributions. We did ablations and experiments for all methods, showcasing the most important aspects of each framework for our scenario. We also performed cross-framework experiments, such as the consistency weight rampup used in the *unsupervised data augmentation* and the MixUp used in the *mean-teacher*.

Regarding the mean-teacher method, it consistently presents improvements over baseline. However, we observe a severe loss of training stability, specially when using cross-entropy or dice as consistency loss. Work on the mean-teacher stability and better ways of modeling the consistency loss need to be conducted so unsupervised domain adaptation can be performed with no additional validation instances. We see a somewhat similar drop of predictive performance for all centers simultaneously, which means the creation of early stopping methods based on instances from the source domain is possible.

On the unsupervised data augmentation, we see a stable method that did not appear to collapse the training at any moment. However, some key characteristics that might be the responsible for the method success in its original field is the training signal annealing, which did not translate well into our context. Overall the method performed well, but the gains were inferior to the mean-teacher in all centers and all metrics, except for recall.

Considering the MixMatch method, we can easily find some points of instability, specially when increasing the consistency weight. Our use of consistency weight ramp-up, originating from the mean-teacher, fixes the instability of the MixMatch. Our version of the MixMatch algorithm is the most different with respect to its original among all three methods, so its worse result in our context, contrasting its superiority in their original task of object classification, might be attributed to this. Most parts of the method, such as the sharpening and sampling from β behaved as expected, with the exception of K augmentations, which did not work properly. We believe that we can achieve better results with this formulation of the MixMatch by finding another way of aligning augmented samples prior to average their results.

Our future work include better understanding of the decisions we made in each framework. We will investigate more task losses that do not suffer from the same problems as the dice loss (when dealing with continuous variables) and try to build a more natural adaptation of each method for domain adaptation in medical imaging segmentation.

REFERENCES

[1] Amukotuwa, S. A.; Cook, M. J. "Spinal disease: neoplastic, degenerative, and infective spinal cord diseases and spinal cord compression". In: *Neurology and Clinical Neuroscience*, Elsevier, 2007.

[2] Arlot, S.; Celisse, A. "A survey of cross-validation procedures for model selection", *Statistics Surveys*, vol. 4, Mar 2010, pp. 40–79.

[3] Barkhof, F.; Filippi, M.; Van Waesberghe, J.; Molyneux, P.; Rovaris, M.; a Nijeholt, G. L.; Tubridy, N.; Miller, D.; Yousry, T.; Radue, E.; et al.. "Improving interobserver variation in reporting gadolinium-enhanced mri lesions in multiple sclerosis", *Neurology*, vol. 49–6, Dec 1997, pp. 1682–1688.

[4] Bengio, Y.; Courville, A.; Vincent, P. "Representation learning: A review and new perspectives", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35–8, Mar 2013, pp. 1798–1828.

[5] Berthelot, D.; Carlini, N.; Goodfellow, I.; Papernot, N.; Oliver, A.; Raffel, C. "Mixmatch: A holistic approach to semi-supervised learning", *ArXiv Preprint*, vol. 1905.02249, May 2019, pp. 15.

[6] Bishop, C. M. "Pattern Recognition and Machine Learning". Springer-Verlag, 2006, 738p.

[7] Brock, A.; Donahue, J.; Simonyan, K. "Large scale gan training for high fidelity natural image synthesis", *ArXiv Preprint*, vol. 1809.11096, Sep 2018, pp. 35.

[8] Carr, T.; Chli, M.; Vogiatzis, G. "Domain adaptation for reinforcement learning on the atari". In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019, pp. 1859–1861.

[9] Caruana, R. "Multitask learning", *Machine Learning*, vol. 28–1, Jul 1997, pp. 41–75.

[10] Chadha, A.; Andreopoulos, Y. "Improving adversarial discriminative domain adaptation", *ArXiv Preprint*, vol. 1809.03625, Sep 2018, pp. 11.

[11] Chen, C.; Dou, Q.; Chen, H.; Heng, P.-A. "Semantic-aware generative adversarial nets for unsupervised domain adaptation in chest x-ray segmentation". In: Proceedings of the International Workshop on Machine Learning in Medical Imaging, 2018, pp. 143–151.

[12] Chua, A. S.; Egorova, S.; Anderson, M. C.; Polgar-Turcsanyi, M.; Chitnis, T.; Weiner, H. L.; Guttmann, C. R.; Bakshi, R.; Healy, B. C. "Handling changes in mri acquisition parameters in modeling whole brain lesion volume and atrophy data in multiple sclerosis subjects: Comparison of linear mixed-effect models", *NeuroImage: Clinical*, vol. 8, Jul 2015, pp. 606–610.

[13] Csurka, G. “Domain adaptation for visual applications: A comprehensive survey”, *ArXiv Preprint*, vol. 1702.05374, Feb 2017, pp. 46.

[14] Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; Le, Q. V. “Autoaugment: Learning augmentation policies from data”, *ArXiv Preprint*, vol. 1805.09501, May 2018, pp. 14.

[15] DeVries, T.; Taylor, G. W. “Improved regularization of convolutional neural networks with cutout”, *ArXiv Preprint*, vol. 1708.04552, Aug 2017, pp. 8.

[16] Dice, L. R. “Measures of the amount of ecologic association between species”, *Ecology*, vol. 26–3, Jul 1945, pp. 297–302.

[17] Dou, Q.; Ouyang, C.; Chen, C.; Chen, H.; Heng, P.-A. “Unsupervised cross-modality domain adaptation of convnets for biomedical image segmentations with adversarial loss”. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 691–697.

[18] French, G.; Mackiewicz, M.; Fisher, M. “Self-ensembling for visual domain adaptation”, *ArXiv Preprint*, vol. 1706.05208, Jun 2017, pp. 13.

[19] French, R. M. “Catastrophic forgetting in connectionist networks”, *Trends in Cognitive Sciences*, vol. 3–4, Apr 1999, pp. 128–135.

[20] Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; Lempitsky, V. “Domain-adversarial training of neural networks”, *The Journal of Machine Learning Research*, vol. 17–1, Apr 2016, pp. 2096–2030.

[21] Gholami, A.; Subramanian, S.; Shenoy, V.; Himthani, N.; Yue, X.; Zhao, S.; Jin, P.; Biros, G.; Keutzer, K. “A novel domain adaptation framework for medical image segmentation”. In: Proceedings of the International Medical Image Computing and Computer Assisted Intervention Brainlesion Workshop, 2018, pp. 289–298.

[22] Glorot, X.; Bengio, Y. “Understanding the difficulty of training deep feedforward neural networks”. In: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[23] Goodfellow, I.; Bengio, Y.; Courville, A. “Deep Learning”. MIT Press, 2016, 775p.

[24] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. “Generative adversarial nets”. In: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[25] He, K.; Girshick, R.; Dollár, P. “Rethinking imagenet pre-training”, *ArXiv Preprint*, vol. 1811.08883, Nov 2018, pp. 10.

[26] He, K.; Zhang, X.; Ren, S.; Sun, J. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: Proceedings of the International Conference on Computer Vision, 2015, pp. 1026–1034.

[27] Higgins, I.; Pal, A.; Rusu, A.; Matthey, L.; Burgess, C.; Pritzel, A.; Botvinick, M.; Blundell, C.; Lerchner, A. “Darla: Improving zero-shot transfer in reinforcement learning”. In: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1480–1490.

[28] Hoffman, J.; Tzeng, E.; Park, T.; Zhu, J.-Y.; Isola, P.; Saenko, K.; Efros, A. A.; Darrell, T. “Cycada: Cycle consistent adversarial domain adaptation”. In: Proceedings of the International Conference on Machine Learning, 2018, pp. 15.

[29] Hu, J.; Shen, L.; Sun, G. “Squeeze-and-excitation networks”. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141.

[30] Hussain, S.; Anwar, S. M.; Majid, M. “Brain tumor segmentation using cascaded deep convolutional neural network”. In: Proceedings of the 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2017, pp. 1998–2001.

[31] Ioffe, S.; Szegedy, C. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, *ArXiv Preprint*, vol. 1502.03167, Feb 2015, pp. 11.

[32] Javanmardi, M.; Tasdizen, T. “Domain adaptation for biomedical image segmentation using adversarial training”. In: Proceedings of the 15th International Symposium on Biomedical Imaging, 2018, pp. 554–558.

[33] Johnson-Roberson, M.; Barto, C.; Mehta, R.; Sridhar, S. N.; Rosaen, K.; Vasudevan, R. “Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?”, *ArXiv Preprint*, vol. 1610.01983, Oct 2016, pp. 8.

[34] Kingma, D. P.; Ba, J. “Adam: A method for stochastic optimization”, *ArXiv Preprint*, vol. 1412.6980, Dec 2014, pp. 15.

[35] Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al.. “Overcoming catastrophic forgetting in neural networks”, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 114–13, Dec 2017, pp. 3521–3526.

[36] Kohl, S.; Romera-Paredes, B.; Meyer, C.; De Fauw, J.; Ledsam, J. R.; Maier-Hein, K.; Eslami, S. A.; Rezende, D. J.; Ronneberger, O. “A probabilistic u-net for segmentation of ambiguous images”. In: Proceedings of the Advances in Neural Information Processing Systems, 2018, pp. 6965–6975.

[37] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. “Imagenet classification with deep convolutional neural networks”. In: Proceedings of the Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[38] Laine, S.; Aila, T. “Temporal ensembling for semi-supervised learning”, *ArXiv Preprint*, vol. 1610.02242, Oct 2016, pp. 13.

[39] LeCun, Y.; Boser, B. E.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. E.; Jackel, L. D. “Handwritten digit recognition with a back-propagation network”. In: Proceedings of the Advances in Neural Information Processing Systems, 1990, pp. 396–404.

[40] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, vol. 86–11, Nov 1998, pp. 2278–2324.

[41] Li, Y.; Wang, N.; Shi, J.; Liu, J.; Hou, X. “Revisiting batch normalization for practical domain adaptation”, *ArXiv Preprint*, vol. 1603.04779, Mar 2016, pp. 12.

[42] Li, Z.; Peng, C.; Yu, G.; Zhang, X.; Deng, Y.; Sun, J. “Light-head r-cnn: In defense of two-stage object detector”, *ArXiv Preprint*, vol. 1711.07264, Nov 2017, pp. 9.

[43] Litjens, G.; Kooi, T.; Bejnordi, B. E.; Setio, A. A. A.; Ciompi, F.; Ghafoorian, M.; van der Laak, J. A.; van Ginneken, B.; Sánchez, C. I. “A survey on deep learning in medical image analysis”, *Medical Image Analysis*, vol. 42, Dec 2017, pp. 60–88.

[44] Liu, M.-Y.; Tuzel, O. “Coupled generative adversarial networks”. In: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 469–477.

[45] Long, J.; Shelhamer, E.; Darrell, T. “Fully convolutional networks for semantic segmentation”. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[46] Maaten, L. v. d.; Hinton, G. “Visualizing data using t-sne”, *Journal of Machine Learning Research*, vol. 9, Nov 2008, pp. 2579–2605.

[47] Madani, A.; Moradi, M.; Karargyris, A.; Syeda-Mahmood, T. “Semi-supervised learning with generative adversarial networks for chest x-ray classification with ability of data domain adaptation”. In: Proceedings of the 15th International Symposium on Biomedical Imaging, 2018, pp. 1038–1042.

[48] Mahmood, F.; Chen, R.; Durr, N. J. “Unsupervised reverse domain adaptation for synthetic medical images via adversarial training”, *IEEE Transactions on Medical Imaging*, vol. 37, Jun 2018, pp. 10.

[49] McClelland, J. L.; McNaughton, B. L.; O’reilly, R. C. “Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures

of connectionist models of learning and memory.", *Psychological Review*, vol. 102–3, Jul 1995, pp. 419.

[50] McCloskey, M.; Cohen, N. J. "Catastrophic interference in connectionist networks: The sequential learning problem". In: *Psychology of Learning and Motivation*, Elsevier, 1989.

[51] Menze, B. H.; Jakab, A.; Bauer, S.; Kalpathy-Cramer, J.; Farahani, K.; Kirby, J.; Burren, Y.; Porz, N.; Slotboom, J.; Wiest, R.; et al.. "The multimodal brain tumor image segmentation benchmark (brats)", *IEEE transactions on medical imaging*, vol. 34–10, Oct 2015, pp. 1993.

[52] Milletari, F.; Navab, N.; Ahmadi, S.-A. "V-net: Fully convolutional neural networks for volumetric medical image segmentation". In: Proceedings of the 2016 Fourth International Conference on 3D Vision (3DV), 2016, pp. 565–571.

[53] Mirza, M.; Osindero, S. "Conditional generative adversarial nets", *ArXiv Preprint*, vol. 1411.1784, Nov 2014, pp. 7.

[54] Oliver, A.; Odena, A.; Raffel, C. A.; Cubuk, E. D.; Goodfellow, I. "Realistic evaluation of deep semi-supervised learning algorithms". In: Proceedings of the Advances in Neural Information Processing Systems, 2018, pp. 3235–3246.

[55] Patel, V. M.; Gopalan, R.; Li, R.; Chellappa, R. "Visual domain adaptation: A survey of recent advances", *IEEE Signal Processing Magazine*, vol. 32–3, Apr 2015, pp. 53–69.

[56] Pereira, S.; Pinto, A.; Alves, V.; Silva, C. A. "Brain tumor segmentation using convolutional neural networks in mri images", *IEEE Transactions on Medical Imaging*, vol. 35–5, Mar 2016, pp. 1240–1251.

[57] Perone, C. S.; Ballester, P.; Barros, R. C.; Cohen-Adad, J. "Unsupervised domain adaptation for medical imaging segmentation with self-ensembling", *NeuroImage*, vol. 194, Jul 2019, pp. 1–11.

[58] Perone, C. S.; Ballester, P.; Barros, R. C.; Cohen-Adad, J. "Unsupervised domain adaptation for medical imaging segmentation with self-ensembling", *ArXiv Preprint*, vol. 1811.06042, Nov 2018, pp. 15.

[59] Polyak, B. T.; Juditsky, A. B. "Acceleration of stochastic approximation by averaging", *SIAM Journal on Control and Optimization*, vol. 30–4, Jul 1992, pp. 838–855.

[60] Prados, F.; Ashburner, J.; Blaiotta, C.; Brosch, T.; Carballido-Gamio, J.; Cardoso, M. J.; et al.. "Spinal cord grey matter segmentation challenge", *NeuroImage*, vol. 152, May 2017, pp. 312–329.

[61] Radford, A.; Metz, L.; Chintala, S. "Unsupervised representation learning with deep convolutional generative adversarial networks", *ArXiv Preprint*, vol. 1511.06434, Nov 2015, pp. 16.

[62] Ratcliff, R. “Connectionist models of recognition memory: constraints imposed by learning and forgetting functions.”, *Psychological Review*, vol. 97–2, Apr 1990, pp. 285.

[63] Razavi, A.; Oord, A. v. d.; Vinyals, O. “Generating diverse high-fidelity images with vq-vae-2”, *ArXiv Preprint*, vol. 1906.00446, Jun 2019, pp. 15.

[64] Redmon, J.; Farhadi, A. “Yolov3: An incremental improvement”, *ArXiv Preprint*, vol. 1804.02767, Apr 2018, pp. 6.

[65] Ronneberger, O.; Fischer, P.; Brox, T. “U-net: Convolutional networks for biomedical image segmentation”. In: Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, 2015, pp. 234–241.

[66] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al.. “Imagenet large scale visual recognition challenge”, *International Journal of Computer Vision*, vol. 115–3, Apr 2015, pp. 211–252.

[67] Salimans, T.; Kingma, D. P. “Weight normalization: A simple reparameterization to accelerate training of deep neural networks”. In: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 901–909.

[68] Santurkar, S.; Tsipras, D.; Ilyas, A.; Madry, A. “How does batch normalization help optimization?” In: Proceedings of the Advances in Neural Information Processing Systems, 2018, pp. 2483–2493.

[69] Simard, P. Y.; Steinkraus, D.; Platt, J. C.; et al.. “Best practices for convolutional neural networks applied to visual document analysis.” In: Proceedings of the International Conference on Document Analysis and Recognition, 2003, pp. 6.

[70] Srinivas, S.; Fleuret, F. “Knowledge transfer with jacobian matching”, *ArXiv Preprint*, vol. 1803.00443, Mar 2018, pp. 11.

[71] Sun, B.; Saenko, K. “Deep coral: Correlation alignment for deep domain adaptation”. In: Proceedings of the European Conference on Computer Vision, 2016, pp. 443–450.

[72] Tarvainen, A.; Valpola, H. “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results”. In: Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 1195–1204.

[73] Torralba, A.; Efros, A. A.; et al.. “Unbiased look at dataset bias.” In: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2011, pp. 7.

[74] Tzeng, E.; Hoffman, J.; Saenko, K.; Darrell, T. “Adversarial discriminative domain adaptation”. In: Proceedings of the Computer Vision and Pattern Recognition, 2017, pp. 4.

[75] Tzeng, E.; Hoffman, J.; Zhang, N.; Saenko, K.; Darrell, T. "Deep domain confusion: Maximizing for domain invariance", *ArXiv Preprint*, vol. 1412.3474, Dec 2014, pp. 9.

[76] Wang, M.; Deng, W. "Deep visual domain adaptation: A survey", *Neurocomputing*, vol. 312, Oct 2018, pp. 135–153.

[77] Weissenbacher, A.; Kasess, C.; Gerstl, F.; Lanzenberger, R.; Moser, E.; Windischberger, C. "Correlations and anticorrelations in resting-state functional connectivity mri: a quantitative comparison of preprocessing strategies", *NeuroImage*, vol. 47–4, May 2009, pp. 1408–1416.

[78] Wu, Y.; He, K. "Group normalization", *ArXiv Preprint*, vol. 1803.08494, Mar 2018, pp. 10.

[79] Xiao, Z.; Huang, R.; Ding, Y.; Lan, T.; Dong, R.; Qin, Z.; Zhang, X.; Wang, W. "A deep learning-based segmentation method for brain tumor in mr images". In: Proceedings of the 6th International Conference on Computational Advances in Bio and Medical Sciences, 2016, pp. 1–6.

[80] Xie, Q.; Dai, Z.; Hovy, E. H.; Luong, M.-T.; Le, Q. V. "Unsupervised data augmentation", *ArXiv Preprint*, vol. 1904.12848, Apr 2019, pp. 15.

[81] Yang, L.; Liang, X.; Wang, T.; Xing, E. "Real-to-virtual domain unification for end-to-end autonomous driving". In: Proceedings of the European Conference on Computer Vision, 2018, pp. 530–545.

[82] Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. "How transferable are features in deep neural networks?" In: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 3320–3328.

[83] Zhang, H.; Cisse, M.; Dauphin, Y. N.; Lopez-Paz, D. "mixup: Beyond empirical risk minimization", *ArXiv Preprint*, vol. 1710.09412, Oct 2017, pp. 13.

[84] Zhou, Z.; Lu, Z.-R. "Gadolinium-based contrast agents for magnetic resonance cancer imaging", *Wiley Interdisciplinary Reviews: Nanomedicine and Nanobiotechnology*, vol. 5–1, Oct 2012, pp. 1–18.

[85] Zhu, J.-Y.; Park, T.; Isola, P.; Efros, A. A. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: Proceedings of the International Conference on Computer Vision, 2017, pp. 2223–2232.

[86] Zhu, X.; Ghahramani, Z. "Learning from labeled and unlabeled data with label propagation", Technical report, School of Computer Science, Carnegie Mellon University, 2002, 19p.

[87] Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q. V. "Learning transferable architectures for scalable image recognition", *ArXiv Preprint*, vol. 1707.07012–6, Jul 2017, pp. 14.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br