Improving Latent Representations of ConvNets for Visual Understanding

Thomas Robert

SCIENCES SORBONNE UNIVERSITÉ

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ
Spécialité **Informatique**
École Doctorale Informatique, Télécommunications et Électronique (Paris)

Improving Latent Representations of ConvNets for Visual Understanding

Amélioration des représentations latentes des ConvNets pour l'interprétation de données visuelles

Présentée par
Thomas ROBERT

Dirigée par
Matthieu CORD et Nicolas THOME

Pour obtenir le grade de
DOCTEUR de SORBONNE UNIVERSITÉ

Présentée et soutenue publiquement le 03 octobre 2019

Devant le jury composé de :

M. Stéphane CANU *Professeur, INSA Rouen Normandie – LITIS*	Rapporteur
M. Greg MORI *Professeur, Simon Fraser University & Research Director, Borealis AI Vancouver*	Rapporteur
Mme Catherine ACHARD *Professeure, Sorbonne Université – ISIR*	Examinatrice
M. Karteek ALAHARI *Chargé de recherche, Inria Grenoble – Thoth*	Examinateur
M. David PICARD *Chercheur HdR, École Nationale des Ponts et Chaussées*	Examinateur
M. Nicolas THOME *Professeur, Conservatoire National des Arts et Métiers – CEDRIC*	Co-directeur de thèse
M. Matthieu CORD *Professeur, Sorbonne Université – LIP6 & Senior Scientist, Valeo.ai*	Directeur de thèse

ABSTRACT

For a decade now, computer vision models based on deep convolutional neural networks (ConvNets) have demonstrated their capability to produce excellent results and are now reaching the stage of "industrialization", being used more and more in commercial products. The most frequent application is probably image classification, for which a model transforms the input image into a series of latent representations until obtaining the prediction of the class to which the image should belong. For this task, recent models such as ResNets are able to obtain impressive human-like performance on ImageNet, a large and complex dataset. In this thesis, we work at improving the "quality" of the latent representations of ConvNets for different tasks. The general goal consists in making those representations more robust to non-relevant variations and structuring the information in the latent space.

First, we focus on making these representations more effective for classification by proposing a new regularization method called SHADE. To do so, using information theory metrics, we propose to minimize the entropy of the representations conditionally to the class label, thus making the representation more robust and invariant to the intra-class variability that is not useful for classification.

Then, we propose to structure the information in two complementary latent spaces with our model called HybridNet. Doing so, we solve a conflict between the objectives of two motivations that seem complementary but are practically incompatible in a regular model: producing more invariant representations for classification and extracting more information for reconstruction. While the first one aims at removing information, the second adds it. By structuring the information in two latent spaces, we are able to release the constraint posed by classical architecture, allowing to obtain better results in the context of semi-supervised learning.

Finally, to go further in structuring the latent space, we propose to address the recent problem of disentangling, *i.e.* explicitly separating and representing independent factors of variation of the dataset. For this, we propose DualDis, an approach that combines a two-branch architecture to structure two complementary types of information, and the use of regular and adversarial classification costs to ensure an effective separation of the information. This approach allows us to obtain better representations compared to existing methods. It also allows to easily perform semantic image editing but also data augmentation by generating new images that can be used to train a classifier.

RÉSUMÉ

Depuis le début de la décennie, les algorithmes de traitement d'images basés sur les réseaux de neurones convolutifs profonds (ConvNets) ont démontré leur capacité à produire d'excellent résultats et permettent désormais d'envisager leur mise en application dans de plus en plus de cas. Ces modèles transforment une image en une succession de représentations latentes jusqu'à obtenir une prediction (classe, segmentation, *etc*.). Dans cette thèse, nous travaillerons à l'amélioration de la qualité des représentations latentes des ConvNets pour diverses tâches. L'objectif général est de rendre ces représentations plus robustes aux variations et de structurer l'espace de resprésentation afin d'améliorer l'organisation de l'information dans cet espace.

Dans un premier temps, nous nous intéressons à rendre ces représentations plus performantes pour la classification en proposant une méthode de régularisation nommée SHADE. Pour se faire, via des métriques liées à la théorie de l'information, nous minimisons l'entropie des représentations conditionnellement à la classe, permettant ainsi de rendre ces représentations plus robustes à la variabilité intra-classe, inutile pour la classification.

Dans un second temps, nous proposons de structurer l'information en deux sous-espaces latents complémentaires avec notre modèle nommé HybridNet. Se faisant, nous résolvons un conflit entre deux motivations complémentaires mais pratiquement incompatibles pour améliorer la qualité des représentations : l'augmentation de l'invariance des représentations et la représentation de davantage d'information via la reconstruction. Alors que la première vise à supprimer de l'information, la seconde en ajoute. La structuration en deux espaces permet ainsi de relâcher la contrainte posée par les architectures classiques, permettant ainsi d'obtenir de meilleurs résultats en classification dans un contexte semi-supervisé.

Enfin, pour aller plus loin dans la structuration de l'espace latent, nous considérons le problème du *disentangling*, c'est-à-dire la séparation et représentation explicite de facteurs sémantiques indépendants. Nous proposons DualDis, une approche qui combine une architecture à deux branches pour structurer deux types d'informations, et l'utilisation de coûts de classification classiques et adverses afin d'assurer une séparation efficace de l'information. Cette approche nous permet d'obtenir des représentations de meilleure qualité comparativement aux méthodes existantes. Elle permet ainsi l'édition sémantique d'images mais aussi l'augmentation de données pour la classification par la génération de nouvelles images.

REMERCIEMENTS

J'aimerai remercier les nombreuses personnes qui ont concouru à l'existence de ce manuscrit.

Pour commencer chronologiquement, j'aimerais remercier mes professeurs de *machine learning* à l'INSA Rouen : Stéphane Canu, Gilles Gasso et Romain Hérault. La grande qualité des cours et leur excellente pédagogie m'a permis de me passionner pour ce sujet et m'a donné envie de me lancer dans cette thèse.

Ensuite, j'aimerais remercier Matthieu Cord et Nicolas Thome, qui m'ont accueilli au LIP6 et m'ont encadré et aidé durant les 4 ans passées à leur côté. Ils ont su me guider dans le sinueux domaine du *deep learning*. Je remercie également David Picard pour ses précieux conseils durant son année au LIP6 ; ainsi que les partenaires du projet ANR DeepVision dans lequel s'inscrit ma thèse, dont les discussions au cours des réunions/workshops d'étapes m'ont également permis de découvrir de nombreux travaux et d'affiner mes idées, en particulier Christian Wolf, Graham Taylor et Greg Mori.

Vient ensuite le tour de mes camarades de bureau, les "Chordettes", initialement Thibaut Durand, Marion Chevalier, Michael Blot, Taylor Mordan, Micael Carvalho, Hedi Ben-Younès, Rémi Cadène et Xin Wang, puis tous ceux qui m'ont rejoins au fur et à mesure des années : Yifu Chen, Arnaud Dapogny, Antoine Saporta, Corentin Dancette, Arthur Douillard. C'est avec ce "petit" groupe soudé que j'ai passé une grande partie de ces dernières années. L'excellente ambiance dans cette équipe a permis à la fois de passer de très bons moments mais aussi de partager, s'entraider et apprendre ensemble. Il est évident que les discussions avec eux sont une part importante de ce que ce doctorat m'a apporté. Plus largement, je remercie bien évidemment tous les thésards et professeurs de l'équipe MLIA (trop nombreux pour tous les citer) pour les exactes même raisons, tant sur la bonne ambiance que sur les discussions constructives que nous avons eu. Je remercie également toutes les personnes du LIP6 dont l'aide a été précieuse, en particulier Nadine Taniou et Christophe Bouder.

Je remercie bien évidemment le jury pour l'intérêt qu'ils ont porté à mes travaux et pour avoir accepté de relire ma thèse, assister à ma soutenance et valider le fruit du travail de ces 3 dernières années.

Enfin pour finir, je remercie bien évidemment ma famille et mes amis pour leur soutien durant ce marathon comportant des hauts et des bas, et tout particulièrement ma fiancé Clara pour son importance vitale à mes côtés au quotidien.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AE	Auto-Encoder
AI	Artificial Intelligence
BN	Batch Normalization
ConvNet	Convolutional Neural Network
CV	Computer Vision
DA	Data Augmentation
DAE	Denoising Auto-Encoder
DL	Deep Learning
DNN	Deep Neural Network
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
IB	Information Bottleneck
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean-Squared Error
NN	Neural Network
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
SHADE	SHAnnon DEcay
SIFT	Scale-Invariant Feature Transform
SSL	Semi-Supervised Learning
SWWAE	Stacked What-Where Auto-Encoder
TPU	Tensor Processing Unit
VAE	Variational Auto-Encoder
VQA	Visual Question Answering

CHAPTER 1

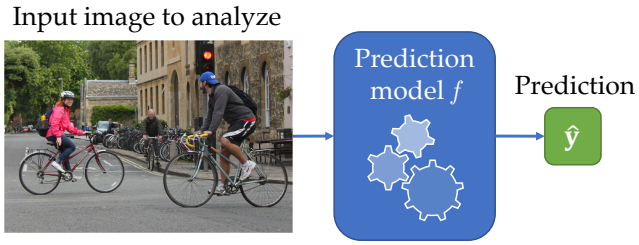
INTRODUCTION

1.1 Context

Artificial Intelligence (AI) has been a subject of great interest for many decades, aiming at making machines reproduce more or less specific human behaviors, ranging from playing chess to producing medical diagnostics. Specifically, in the past decade, this domain has seen a rapid and almost exponential growth, being invested by numerous research labs and companies (like Google, Facebook, Microsoft, Amazon, etc.). Nowadays, applications of AI are varied and show very impressive results. Those include information and image retrieval (web and image search engines), automatic translation, image recognition and classification (*e.g.* Google Photos), face recognition, tracking of objects in videos, speech recognition, making autonomous vehicles drive, interpreting medical imagery, *etc*.

One of the domains of AI that shows the most impressive results is Computer Vision (CV), which focuses on automatic processing of images and videos. This domain is especially important considering the exponential growth of the volume of visual data being generated around the world, often uploaded and published on the Internet. For example, it is said that more than 500 hours of video are uploaded each minute on YouTube (Hale 2019) or 300 million new photos each day on Facebook (Noyes 2019). Considering those figures, it is therefore clear that CV is becoming more and more important to automatically process this large amount of data that would otherwise be impossible to handle by humans. In particular, CV answers the necessity for those platforms to ensure a form of control on this content (*e.g.* to prevent the upload of illegal material), but also the desire to propose new features and services related to this multimedia content (*e.g.* captioning images for visually impaired people, or proposing semantic search in a photo library).

Computer Vision (CV) aims at solving this issue and covers a lot of different tasks. We present the general principle and some of those tasks in Figure 1.1: a predictive model f transforms an input image into a prediction, that can be the class of the image, a list of objects and their localizations, a segmentation assigning a class to each pixel, a text caption, *etc*. One particular problem with image

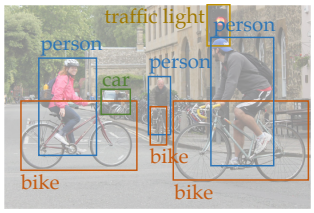


Examples of different tasks (i.e. types of prediction) in Computer Vision:

Classification

✓ Street
✗ Office
✗ Bedroom
✗ …

Object detection



Segmentation

Captioning …

a girl in a pink jacket on a bicycle passes a man in a blue cap on a bicycle.

Figure 1.1. – **General principle of Computer Vision (CV).** A discriminative model transforms an image into a prediction that depends on the task of interest. For example, classification, object detection (which usually implies localization of the object with a bounding box), image segmentation, image captioning, *etc*.

data is what is called the "semantic gap". This describes the wide difference in meaning between what a human sees in a picture, *i.e.* objects, concepts, a scene, *etc*.; and what a machine sees: pixel values, *i.e.* numbers representing quantities of red, green and blue in quantized portions of space. For classification for example, a machine will need to *bridge* this gap and find a mathematical mapping between pixel data and semantic categories. To that end, CV relies heavily on Machine Learning (ML), a broad domain that proposes to create models that learn to reproduce a task. To do so, these models use a dataset of examples to improve themselves until they produce satisfactory predictions. For example, given pictures of cats and dogs and knowing which one is on each picture, we can try to learn a model that distinguishes them.

To see how CV evolved, we can take the example of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), an image classification challenge created in 2010 based on a dataset of more than 1.2 million images (that later grew to 1.3 million) distributed among 1,000 classes used to train the Computer Vision (CV) model. In the years 2010 and 2011, leading methods used a combination of handcrafted CV preprocessing to find compact numerical representations of images, called features; and used ML models to learn a mapping from those features to the possible classes of an image. This first step, called feature engineering, was long developed and refined and was a key component of the CV field, allowing to

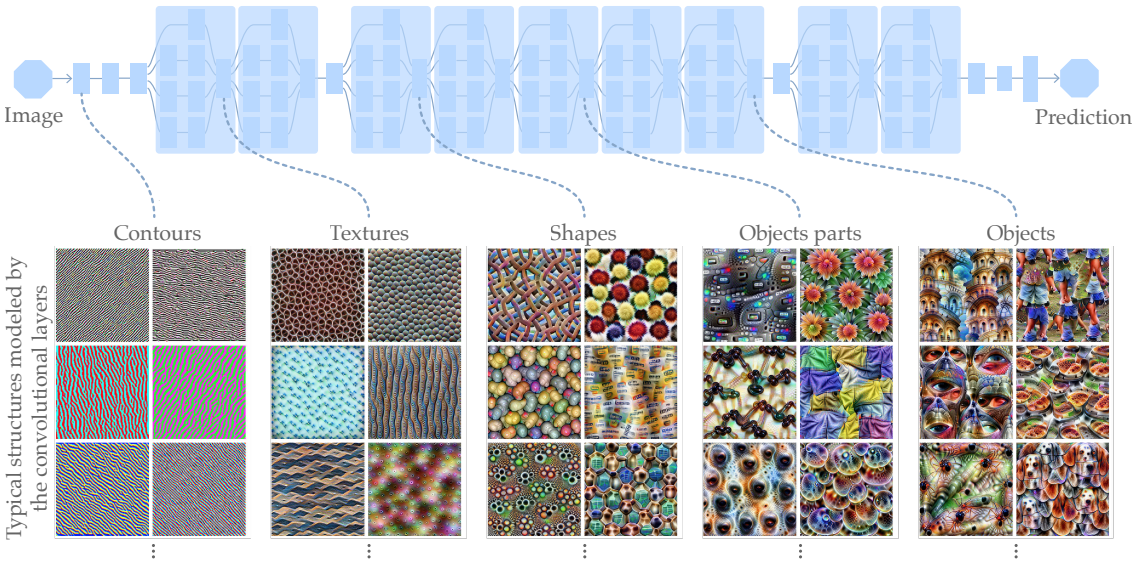


Figure 1.2. – **Illustration of the typical structures extracted by a ConvNet**. The first layers detect simple contours, that are then assembled into textures, more complex shapes, parts of objects and finally complete objects. This property of representing more and more semantic information to finally produce a prediction arises thanks to the training of the ConvNet. Credits: Illustration based on Olah et al. (2017).

partly bridge the semantic gap thanks to human knowledge in how pixels should be interpreted to form more meaningful representations, leaving the machine to learn a much simple task of mapping those meaningful representations to semantic classes. However, this step required specific handcrafting and expertise to find features that would best represent an image, requiring to be both robust and invariant to changes in the different images of a given class, while being different enough from one class to another to be able to correctly discriminate the classes.

In the year 2012, Deep Learning (DL) models – a specific kind of ML models – started replacing this traditional approach and since that year, all winners of ILSVRC are DL models. Deep Learning (DL) proposes to use a Deep Neural Network (DNN) to transform the raw input data – pixels in the case of CV – into the desired semantic prediction, *e.g.* the classes of ILSVRC. This means that the traditional feature engineering step was replaced. With DL, humans no longer design a smart way to transform the pixels, but rather design a framework of mathematical transformations and let the machine determine which features to extract from the raw data. Of course, because of this, DL models need to be much more complex that the ML models used before, with orders of magnitude more parameters that the machine needs to learn.

The type of Deep Learning (DL) model that allowed Krizhevsky et al. (2012) to win ILSVRC in 2012 is a Convolutional Neural Network (ConvNet), a type of architecture especially well fitted for CV. A ConvNet produces a series of transformations of the input image into what is called *latent* representations, until the final transformation produces the prediction. Interestingly, when investigating the information modeled by those representations (Olah et al. 2017) as illustrated in Figure 1.2, we can notice that a ConvNet naturally learns to first detect contours, assembled in textures, shapes and object detections.

After 2012, DL research has led to a succession of ConvNet architectures each outperforming the previous one on ILSVRC. It is also important to note that those deep ConvNets designed for ILSVRC have proven to be extremely versatile for CV tasks, far beyond classification, and are now *de facto* standards in the field. It is of course possible to use the same architecture to classify other datasets, and even take advantage of the features learned on ImageNet to obtain better results on much smaller datasets. But more interestingly, those architectures can be used as backbones for models used in many other CV tasks such as object detection and localization (*e.g.* Faster R-CNN uses VGG-16, *cf.* Ren et al. 2015), segmentation (DeepLab uses VGG-16 or ResNet-101, *cf.* L.-C. Chen et al. 2017), Visual Question Answering (VQA) (MUTAN uses ResNet-152, *cf.* Ben-Younes et al. 2017), *etc*. This demonstrates the great strength and versatility of those models for Computer Vision (CV).

This *revolution* of Deep Learning (DL) was made possible by the combination of multiple factors. First of course, the progress in the architecture of the DNNs and in particular the development of Convolutional Neural Networks (ConvNets), started by Fukushima (1980) and refined over the years (LeCun et al. 1989; Krizhevsky et al. 2012; K. He et al. 2016, *etc*.). Alongside the architecture, the training method of those models was also improved, with backpropagation methods first proposed by Rumelhart et al. (1988) and LeCun et al. (1998) and refined as well to allow the training of those complex models (Srivastava et al. 2014; Ioffe and Szegedy 2016, *etc*.). Finally, as we have seen, DL models are much more complex than previous models. Their training was made possible thanks to large annotated datasets such as ImageNet (Russakovsky et al. 2015), the dataset of ILSVRC; and the improvements of the computing hardware, especially the Graphics Processing Units (GPUs), strongly reducing the compute time required to train those models. Nowadays, pieces of hardware are even developed specifically for DL training such as Google's Tensor Processing Units (TPUs).

1.2 Motivations

We now go over some interesting questions that remain open regarding the learning of ConvNets for Computer Vision (CV) and that we will tackle in the course of this thesis. Namely, regularizing DNNs is still an important problem, in particular because of their very large number of parameters. In this direction, using additional unlabeled data is an interesting solution addressed by SSL for which further developments are interesting. Finally, producing more semantic and rich representations using disentangling is a new and interesting domain to address.

Regularization. A key advantage of DNNs for CV tasks is their large amounts of parameters that can model very complex transformations of the input image, which is necessary for complicated tasks. However, this is also an important problem since these models have orders of magnitude more parameters than there are images, even in large datasets like ImageNet and its 1.2M images. Because of this, deep ConvNets easily suffer from over-fitting and require efficient regularization. In addition, the optimization problem of those DNNs is highly non-convex (Kawaguchi 2016) which makes the training even more difficult. For these reasons, since the beginning of DL, finding ways to make the training of those models possible has always been an important part of the research done by the community.

As such, numerous approaches have been developed over the years which became standard techniques. First, Data Augmentation (DA) (Dyk and Meng 2001) tries to produce new images to artificially increase the size of the dataset; but is limited because it does not really produce new information. Numerous techniques are also based on adding noise at different stages of the training process (*cf.* Kukačka et al. 2017), which have shown to help generalization but have a behavior that is not really interpretable and thus provide no control over it. A popular type of noise injection is dropout (Srivastava et al. 2014) and its variants, which disconnects parts of the model randomly. While some interpretations exist, the exact advantage obtained by disabling parts of the model is not obvious, seems unnatural, and fails on some architectures. Weight decay (Krogh and Hertz 1992) is also a popular regularization technique that is equivalent to L2 regularization of non-DL models. While for linear models it encourages smoother decision function, its effect with deep networks is less clear (C. Zhang et al. 2017). Finally, the very popular Batch Normalization (BN) (Ioffe and Szegedy 2016) has recently been key to train very deep ConvNets like ResNet, but the interpretation of its behavior is currently debated in the community (*cf.* Santurkar et al. 2018). H. Zhang et al. (2019) even recently proposed to simply replace it with a simple specific weight initialization that provides equivalent results. Those numerous limitations in our

understanding of how those existing techniques affect the generalization performance of Deep Neural Networks (DNNs) are studies by C. Zhang et al. (2017). They conclude, "explicit regularization may improve generalization performance, but is neither necessary nor by itself sufficient for controlling generalization error".

Semi-Supervised Learning. Another interesting aspect of the DL literature is Semi-Supervised Learning (SSL) which proposes to use partially labeled datasets. As we have seen, DL usually relies on large labeled datasets which are costly to obtain; something that SSL tries to limit by using a mostly unlabeled dataset, only using a small number of labeled images for each class. To that end, two main categories of techniques exist: invariance-based methods and reconstruction-based methods.

Invariance-based methods rely on producing latent representations that are well fitted for classification. In particular, Sajjadi et al. (2016), Laine and Aila (2017), and Tarvainen and Valpola (2017) propose to enforce the stability of the class prediction for different versions of the same image. Reconstruction-based methods use an unsupervised reconstruction loss to extract additional information from unlabeled images, producing features that are more robust, more general and representative of the full diversity of the data (Ranzato and Szummer 2008). We can see that the first technique aims at improving the invariance of the features while the second aims at improving their representativity.

While both are interesting and could seem complementary, they pose the problem of having conflicting roles. The first idea aims at improving classification directly by increasing the invariance and discriminative nature of the representations. On the other hand, reconstruction proposes to extract features regardless of their ability to discriminate the supervised task that we eventually want to solve. This means that even if reconstruction would improve the quality of the discriminate features, it would also extract numerous features that are not well fitted for classification.

Disentangling. Finally, an interesting domain of research called *disentangling* proposes to look for ways to improve the semantic quality of the latent representations of DNNs. The definition of disentangling varies (Higgins et al. 2018) but it mostly aims at producing representations that model independent factors of variation of the data. For example, considering a dataset of faces, we would like to have independent representations of the hairstyle, the makeup, the smile, the eye colors, the presence of glasses, *etc*. This kind of models try to improve the quality of the features that can then be used for various semantic tasks (classification, retrieval, *etc*.) or to interpret and manipulate the representation of an input.

A first solution is to address disentangling in an unsupervised manner, meaning that no label regarding the factors of variation of the dataset are provided. Those models, in particular β-VAE (Higgins et al. 2017) and its variations (T. Q. Chen et al. 2018; Klys et al. 2018), are effective at providing features that are independent of each other. However, because no label is used, they are not able to provide semantic information about the factors they extracted. In addition, the absence of labels does not encourage the representation of complex semantic factors and is more likely to produce simpler low-level variations that possess limited semantic information.

The second solution is to rely on labels of factors of variation. This can be done using conditional generative models (Perarnau et al. 2016; Lample et al. 2017) but this has the drawback of representing the labeled factors in a binary state, which is usually not sufficient since it is impossible to encode the complex diversity of those factors with only one boolean. Other approaches propose to use latent subspaces to represent and disentangle different kinds of semantic information. Mathieu et al. (2016), Hadad et al. (2018), and Yu Liu et al. (2018) for example all propose to use a latent space split in two parts, each dedicated to an information domain, *e.g.* the identity of a person on one side and its visual attributes (hairstyle, makeup, *etc.*) on the other. Using labels, they can provide semantic information but usually rely on labels for one of the two types of information which is the cause of most of their limitations as we will see.

1.3 Contributions and outline

Because DL now produces classification scores comparable to human performance using very large models on ImageNet, in this thesis, we propose to study ConvNets "beyond ImageNet", investigating and improving the capabilities of those models beyond proposing bigger and deeper architectures on extremely large datasets. In particular, we know that ConvNets produce a succession of representations encoding pieces of information extracted from the input. We propose to investigate in depth different ways to improve the quality of those representations. First, by improving their discriminative quality using invariance-based regularization. Then, dealing with fewer labels and additional unlabeled data, we propose new architectures to structure the latent space for Semi-Supervised Learning (SSL). This structuring of the information is then pursued and reinforced to address the disentangling of complementary information.

- Chapter 2: DEEP NEURAL NETWORKS FOR IMAGE CLASSIFICATION: TRAINING, REGULARIZATION AND INVARIANCE

We first propose an overview of the recent evolution of DL and deep ConvNets and position the work of this thesis regarding regularization, SSL and disentangling of DNNs.

We then propose our contribution, a new regularization method called **SHADE**. This method is inspired by recent work applying the Information Bottleneck (IB) principle to DNNs. With SHADE, we explicitly work on influencing the information represented by a DNN. Because an input image contains a very large amount of information, from which only a small portion is relevant for classification, we propose to encourage the filtering of such information. Representations should indeed be intra-class invariant, filtering out all the inherent variability of each category to focus on discriminate information only. To explicitly encourage this intra-class invariance, SHADE minimizes the entropy of the representations conditionally to the classes. This optimization goal is both interpretable and perfectly aligned with the goal of classification. We demonstrate its effectiveness at producing more robust and invariant features using different standard architectures on CIFAR-10.

- Chapter 3: SEPARATING DISCRIMINATIVE AND NON-DISCRIMINATIVE INFORMATION FOR SEMI-SUPERVISED LEARNING

  In this chapter, we focus on improving standard deep ConvNets using Semi-Supervised Learning (SSL) by working on the way latent information is structured in this context. Current SSL methods seem to have conflicting objectives and cannot work in synergy, with classification and stability methods increasing the invariance of the representations, filtering only the discriminative information; while reconstruction requires to conserve all the information to reach its goal. Thus, when mixing classification with reconstruction arises a dilemma.

  To overcome this conflict and overcome this "incompatibility", we introduce a novel idea to structure the information in two separate and complementary latent spaces. This takes the form of a novel architecture and a dedicated training method called **HybridNet**. This architecture uses an encoder with two branches so that the first branch can encode only the discriminative information and is allowed to remove the rest of the information, which is captured by the second branch and therefore allows correct reconstruction of the image. Thanks to this, we make reconstruction, classification and invariance regularization work in cooperation instead of in opposition. We validate the effectiveness of our method compared to the state of the art on CIFAR-10, SVHN and STL-10. We also propose in-depth analysis of the different terms of the loss as well as visualizations to gain insight regarding the behavior of the model.

- Chapter 4: DUAL-BRANCH STRUCTURING OF THE LATENT SPACE FOR DISENTANGLING AND IMAGE EDITING
  Finally, in an attempt to pursue our work on organizing and separating information in complementary latent spaces, we address the problem of disentangling factors of variation. We propose to design a model that separates two complementary types of information that we call *information domains*; for example, with a dataset of faces, the first domain is the identity (*i.e.* the *class*) of the person and the second is the visual *attributes* (hairstyle, makeup, *etc.*).
  For this, we develop a framework called **DualDis**, which relies on a two-branch architecture, each domain being assigned to one branch. Using adversarial training (Goodfellow et al. 2014), we explicitly train the model to disentangle the two domains so that their information is contained in one branch only. Using classifiers, we structure the representation space in order to be able to efficiently manipulate it, enabling simple semantic image editing and generation of new images. We validate this model on CelebA, Yale-B and NORB with a comparison to the state of the art and examples of possible applications of such an architecture, namely image editing and generation of images for data augmentation.

We conclude this thesis in Chapter 5 and discuss several interesting directions for future work.

1.4 Related publications

This thesis is based on the material published in the following papers:

- Michael Blot, Thomas Robert, Nicolas Thome, and Matthieu Cord (2018b). "SHADE: Information-Based Regularization for Deep Learning". In: *IEEE International Conference on Image Processing (ICIP)*, best paper award;

- Thomas Robert, Nicolas Thome, and Matthieu Cord (2018). "HybridNet: Classification and Reconstruction Cooperation for Semi-Supervised Learning". In: *European Conference on Computer Vision (ECCV)*;

- Thomas Robert, Nicolas Thome, and Matthieu Cord (2019). "DualDis: Dual-Branch Disentangling with Adversarial Learning". In: *Under Review at Advances in Neural Information Processing Systems (NeurIPS)*.

Chapter 2

DEEP NEURAL NETWORKS FOR IMAGE CLASSIFICATION: TRAINING, REGULARIZATION AND INVARIANCE

Chapter abstract

In this chapter, we propose a general overview of the literature regarding the design, training and regularization of Deep Neural Networks (DNNs) and Convolutional Neural Networks (ConvNets) during the past few years. We discuss recent research directions that will be addressed in this thesis, namely invariance-based regularization, Semi-Supervised Learning (SSL) and the disentangling of representations. In particular, in this chapter, we focus on the question of regularizing DNNs to make them produce representations that are well fitted for classification and that generalize well on unseen data. A common direction consists in making latent representations encode information that allows to discriminate between the different classes of interest while being invariant to intra-class variations, which are equivalent to noise regarding classification. To this end, we propose a regularizer called SHADE. This regularization loss is based on a novel idea using information theory metrics to formalize the aforementioned objective of removing the intra-class variance from the latent space. We show that SHADE is able to effectively encourages invariance in many standard ConvNets architectures and provides an interesting gain over usual baselines on CIFAR-10, as well as studies regarding the behavior of ConvNets toward class information.

The work in this chapter, done in collaboration with Michael Blot (Blot 2018), has led to the publication of a conference paper:

- Michael Blot, Thomas Robert, Nicolas Thome, and Matthieu Cord (2018b). "SHADE: Information-Based Regularization for Deep Learning". In: *IEEE International Conference on Image Processing (ICIP)*; best paper award.

Contents

2.1 Introduction

In this chapter, we first propose a general overview of the recent developments in the fields of Deep Learning (DL) and Convolutional Neural Networks (ConvNets) used for classification. In particular, we will see how those models are designed, trained and the evolution that allowed those architectures to become the backbone of almost all state-of-the-art models in Computer Vision (CV) research. While our discussion will focus on image classification, it also applies to many semantic tasks of CV (detection, segmentation, *etc.*).

Key ingredients that make deep ConvNets perform so well is their depth and the number of trainable parameters they contain. Having deeper and deeper models with more and more parameters makes it possible to progressively represent more complex decision functions and extract richer and more semantic information from the input images. However, having such complex models comes with an increased risk of overfitting the training set, especially if it contains a small number of images compared to the number of parameters. For example, a ResNet-101 (K. He et al. 2016) model has 44.5M parameters to train while ImageNet contains "only" 1.3M images. There is thus a clear unfavorable imbalance between the complexity of our models and the quantity of labeled data at our disposal.

Because of this, finding ways to control the training of Deep Neural Networks (DNNs) is a crucial part of the research in DL, both to improve their performance

on very large datasets but also to eventually be able to train deep architectures on small datasets. To do so, many *regularization* methods have been developed over the years, usually introducing prior human knowledge on desired properties to make the model more robust, such as sparsity of the weights, smoothness of the decision boundary or compression and invariance of the representations.

In this chapter, we investigate in depth this last option and propose a new method to encourage the invariance of the representations. This first contribution, done in collaboration with Michael Blot (Blot 2018), consists in a new regularization method called SHADE. Inspired by the Information Bottleneck (IB) principle (Tishby et al. 1999) and based on information theory metrics, we propose to minimize the entropy of the representations of a DNN conditionally to the class label: $\min \mathcal{H}(H \mid Y)$. We show that this corresponds to minimizing the intra-class variance of the features, and therefore encourages the construction of features that are intra-class invariant and are thus more fitted for classification. We validate this idea experimentally on various representative DNN architectures.

We also introduce interesting directions to improve DNNs that will be followed in the next chapters. First, the possibility of improving the generalization ability of DNNs by using additional unlabeled data, which is called Semi-Supervised Learning (SSL). This usually consists in finding a method that extracts robust features on labeled and unlabeled data and uses the labeled data to learn the prediction function. Second, through the design of methods that disentangles, *i.e.* separates into independent representations, the different factors of variation of the dataset, greatly increasing the semantic quality of the latent space for various tasks.

In Section 2.2 we detail the design and training of current deep ConvNets and the existing methods to improve their quality. We then present and validate our first contribution, a novel regularization method called SHADE in Section 2.3.

2.2 Training and Regularizing Deep Neural Networks

In this section, we first introduce the general learning framework of Deep Learning (DL) before a focus on the Convolutional Neural Network (ConvNet) architectures. We will then go over common techniques to improve the generalization abilities of those models.

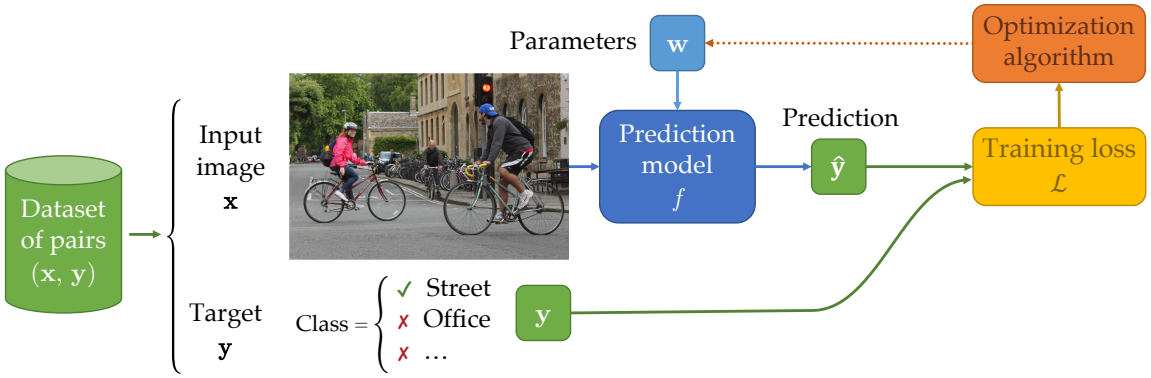


Figure 2.1. – **General overview of Machine Learning (ML) training**. Using examples from a dataset, the predictive model learns to make the correct predictions by minimizing a training loss measuring the error made by the model.

2.2.1 Deep Learning framework

Machine Learning (ML). ML is a broad domain proposing models that learn to solve a task from examples used to improve themselves. In this thesis, we work mostly on models trained to predict a semantic label. For example, given pictures of cats and dogs, we can learn a model that distinguishes them. Let us go over a typical process used to train an ML model, represented by Figure 2.1.

ML proposes to train a **model** f, of **parameters** $\mathbf{w} \in \mathcal{W}$, taking an input $\mathbf{x} \in \mathcal{X}$ to produce a **prediction** $\hat{\mathbf{y}}$. Knowing the **ground-truth label** $\mathbf{y} \in \mathcal{Y}$ associated to $\mathbf{x}$, we can quantify the prediction error of the model by defining a **loss function** $\mathcal{L}_{\text{task}}(\hat{\mathbf{y}}, \mathbf{y})$. Our goal is therefore to find the optimal parameters $\mathbf{w}^*$ that minimizes the expectation of the loss:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})} \left[\mathcal{L}_{\text{task}}(\hat{\mathbf{y}}, \mathbf{y})\right] = \arg\min_{\mathbf{w}} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})} \left[\mathcal{L}_{\text{task}}\big(f_{\mathbf{w}}(\mathbf{x}), \mathbf{y}\big)\right]. \tag{2.1}$$

To solve this optimization problem, we use a **dataset** $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}), i = 1 \dots N_{\text{train}}\}$ on which we can sample pairs $(\mathbf{x}, \mathbf{y})$ used to estimate the expectation with Monte-Carlo sampling. We then use an **optimization algorithm** to minimize this empirical loss over the dataset:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{N_{\text{train}}} \left[\mathcal{L}_{\text{task}}\big(f_{\mathbf{w}}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}\big)\right]. \tag{2.2}$$

Deep Learning (DL). DL is a subset of ML models using Deep Neural Networks (DNNs), initially inspired by a simple modeling of the neurons proposed

by McCulloch and Pitts (1943). Most of the time, we use *feed-forward* Neural Networks (NNs), where the model f is a succession (more precisely a directed acyclic graph) of mathematical transformations called *layers* transforming $\mathbf{x}$ in a succession of *representations* $\mathbf{h}_\ell$ for each layer ℓ. The most common layers are **dense** layers, which consist in a linear transformation of the input $\mathbf{h}_\ell = \mathbf{w}_\ell \mathbf{h}_{\ell-1} + b_\ell$; and **non-linear activation** layers, that can be any function making the model non-linear. Nowadays we mostly use Rectified Linear Unit (ReLU) ($\max(0, \mathbf{h})$) activation, but hyperbolic tangent ($\tanh$) or sigmoid ($e^{\mathbf{h}}/e^{\mathbf{h}+1}$) remain popular options, and many more exist (Nwankpa et al. 2018). Thanks to their depth, *i.e.* number of layers, DNNs are able to transform raw input data into more and more complex representations, and thus perform *representation learning* (Bengio et al. 2013), where the model learns by itself what are the most interesting features to model the input data for the task at hand.

Neural Networks (NNs) are trained using gradient back-propagation (Rumelhart et al. 1988). This allows to compute progressively, using the chain-rule, the gradient $\nabla_{\mathbf{w}}\mathcal{L}$ of the loss $\mathcal{L}$ with respect to all the weights $\mathbf{w}$. Using a gradient descent algorithm, we can then update the weights in a direction that decrease the value of the loss, so that progressively, over the course of the training, we finally reach a minimum of the objective function:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L} \,. \tag{2.3}$$

Numerous gradient descent algorithms exist, the simplest one being Stochastic Gradient Descent (SGD) (Léon Bottou 2010), with variants designed to improve the speed of the convergence as well as finding a *better* minimum, since DNNs training losses are non-convex and lots of local minima exist. Famous methods include SGD with momentum (Rumelhart et al. 1988), RMSProp (Hinton et al. 2012), AdaDelta (Duchi et al. 2011) or Adam (Kingma and Ba 2015).

2.2.2 Convolutional architectures

As we have seen, Deep Learning (DL) became particularly popular for Computer Vision (CV) in 2012 when AlexNet (Krizhevsky et al. 2012) won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). This model is a Convolutional Neural Network (ConvNet), a type of NN that is specially designed for CV tasks. A typical ConvNet, such as VGG-16 (Simonyan and Zisserman 2015) represented in Figure 2.2, is composed of **convolutional layers** used in place of most or even all of the dense layers of a traditional DNN. Indeed, applying a 2D convolution to an image allows to process only small and local patches of information, regardless of their position in the image. Thus, at the beginning of the network, convolutions

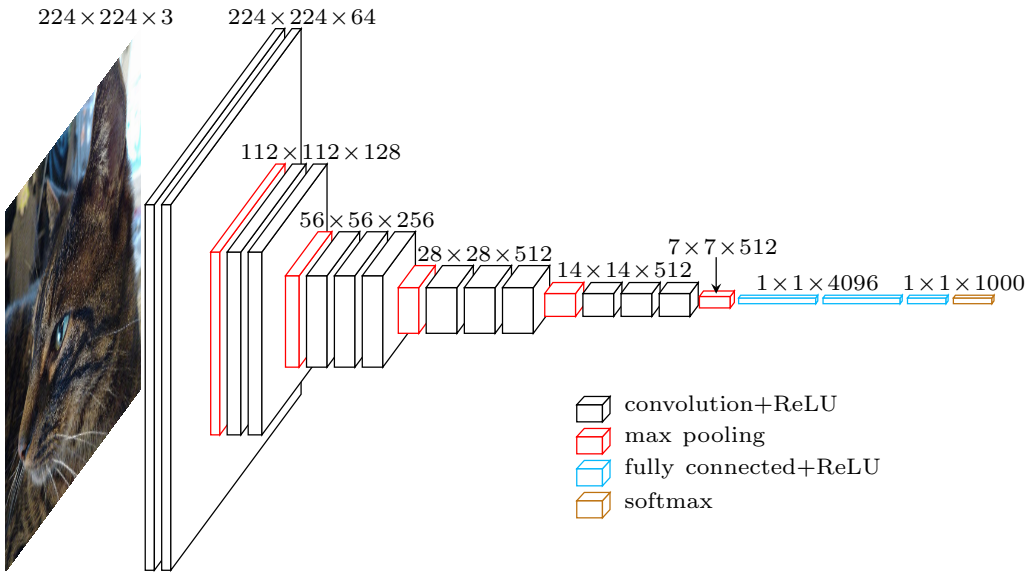


Figure 2.2. – **Architecture of a typical ConvNet**. We show the architecture of a VGG-16 (Simonyan and Zisserman 2015), interlacing convolutional layers with max-poolings. Figure by Durand (2017).

only look for small patterns in the input image. When going deeper in the network, the use of **pooling layers** (or by adding stride in a convolution) progressively aggregates the spatial information, bringing closer the information of the patterns found by previous layers. Thus, the next convolutions have a larger receptive field (Luo et al. 2016) and can assemble the small patterns into bigger and more semantic ones. This behavior of convolutions can be observed by investigating how trained ConvNets represent the information, as studied by Olah et al. (2017) and previously illustrated in Figure 1.2 (page 3). We can see that the model first detects contours, assembled into textures, shapes and objects to finally produce the semantic prediction. Interestingly, it can also be noted that the visual cortex is said to work in a similar fashion to this succession of convolutions and pooling (Hubel and Wiesel 1962).

The first ConvNets trained by back-propagation and designed for CV dates back decades ago, for example with LeNet-5 (LeCun et al. 1998) which classifies hand-written digits. However, as we mentioned, it is only recently that they became the state-of-the-art approach for CV. The first notable network is AlexNet (Krizhevsky et al. 2012), designed to classify natural images of ImageNet (Russakovsky et al. 2015), that won ILSVRC. In the next years, numerous new ConvNet architectures were proposed and won this competition. Popular architectures include VGG-16

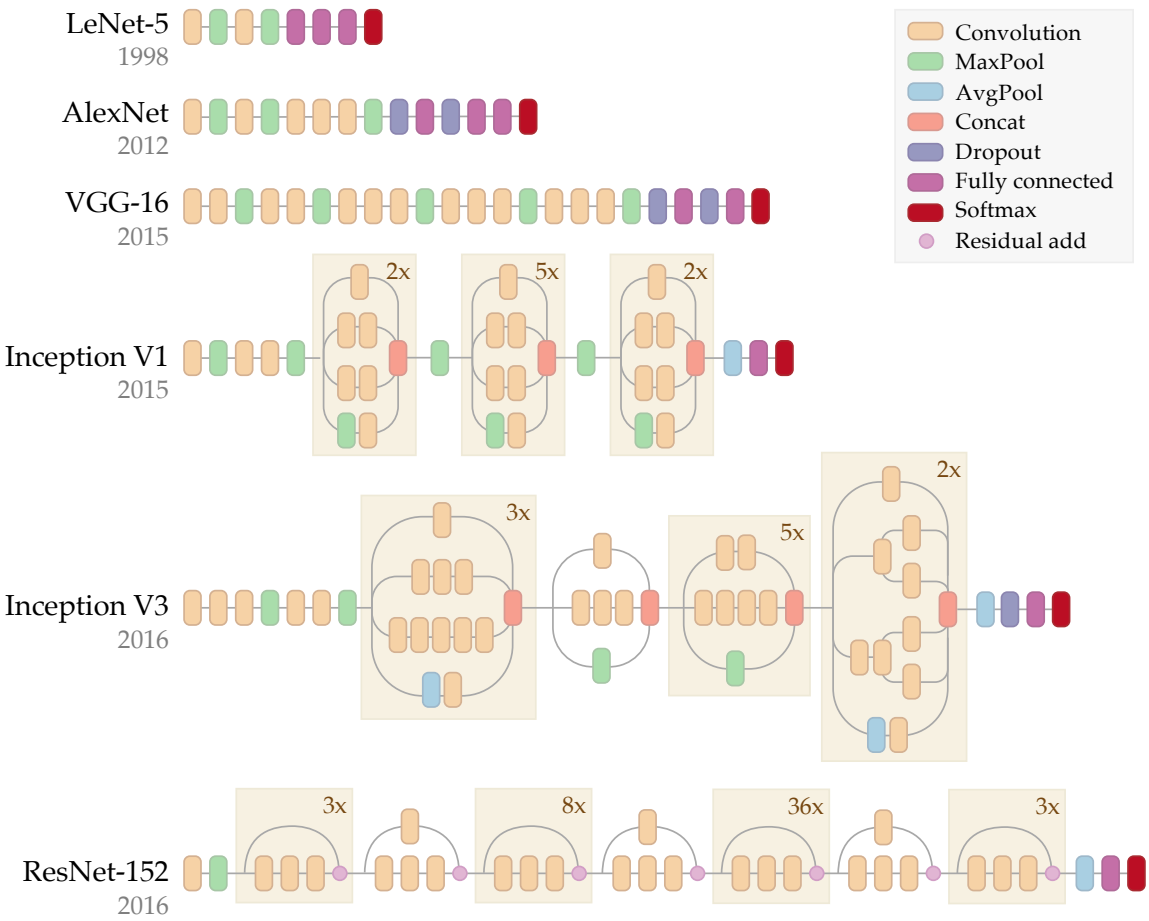


Figure 2.3. – **Visualizations of the evolution of standard ConvNets architectures** over the years. Style inspired by Alemi (2016).

Network	Top-5 error on ImageNet	Number of layers	Number of parameters
AlexNet (Krizhevsky et al. 2012)	16.4%	8	62M
VGG-16 (Simonyan and Zisserman 2015)	9.3%	16	138M
Inception V1 (Szegedy et al. 2015)	9.2%	22	6M
Inception V3 (Szegedy et al. 2016)	5.6%	48	23M
ResNet-50 (K. He et al. 2016)	6.7%	50	26M
ResNet-101 (K. He et al. 2016)	6.0%	101	45M
ResNet-152 (K. He et al. 2016)	5.7%	152	60M

Table 2.1. – **Overview of popular ConvNet architectures** with their results on ImageNet (ILSVRC dataset) in a 10-crop setting (lower than the model ensembling scores submitted to the challenge), along with the number of layers and parameters.

(Simonyan and Zisserman 2015), Inception-v1 also called GoogleNet (Szegedy et al. 2015) and ResNet (K. He et al. 2016).

Visualizations of those architectures are presented in Figure 2.3 and their results on ImageNet are presented in Table 2.1 for an easy comparison of their architectures. A global trend that we can see is that adding depth to the architecture was one of the key factors for improving the results. Indeed, having more layers allows the model to construct progressively more and more semantically rich features in order to better bridge the "semantic gap" between pixels and categories.

As we mentioned, those ConvNet architectures we presented have shown to be very versatile regarding the type of CV problems they are able to address (classification, detection, segmentation, VQA, *etc.*) and are thus now used as standard building blocks in many CV models. This is why we propose to study in depth how those models can be used and improved, first using necessary regularization techniques that allow them to work so efficiently.

2.2.3 Regularizing DNNs with priors

We have seen that DNNs are trained to find the optimal parameters in order to best predict the labels of the samples in the training dataset $\mathcal{D}$. However, a model that perfectly predicts those labels does not necessarily produce the best results on unseen data. For example, if $f_{\mathbf{w}}$ can model a very complex function compared to the number of samples in $\mathcal{D}$, the model can learn *by heart* the labels $\mathbf{y}^{(i)}$ associated to $\mathbf{x}^{(i)}$ without being able to **generalize** to new samples (Vapnik and Chervonenkis 1972), which is called **overfitting**.

Because of their complexity, DL models are highly subject to this risk of overfitting the training set while lacking generalization capabilities on the test set. Indeed, they are known to be universal approximators (Lu et al. 2017) and can possibly produce overly complex decision boundaries. Since the beginning of the development of DNNs, techniques to control the training of these models were developed.

To overcome this issue, we use **regularization** which can take multiple forms, a common one being to add a new loss term $\Omega_{\text{regul}}(\mathbf{w}, \mathbf{x}, \mathbf{y})$ describing preferred solutions, for example, simpler or smoother decision functions (Vapnik 1992). Instead of using Equation 2.2, we thus have:

$$\min_{\mathbf{w}} \mathcal{L}(\mathcal{D}, \mathbf{w}) = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \underbrace{\left[\mathcal{L}_{\text{task}}\left(f_{\mathbf{w}}(\mathbf{x}), \mathbf{y}\right) + \Omega_{\text{regul}}(\mathbf{w}, \mathbf{x}, \mathbf{y})\right]}_{\text{complete loss } \mathcal{L}} . \tag{2.4}$$

For a model f of parameters $\mathbf{w}$, with a dataset $\mathcal{D}$ of pairs image-label $(\mathbf{x}, \mathbf{y})$, we try to find the best parameters so as to minimize the target loss $\mathcal{L}_{\text{task}}(\hat{\mathbf{y}}, \mathbf{y})$, and using an optional regularization penalty Ω_{regul} that can take different forms. To regularize the training, we can therefore influence:

- $\mathcal{D}$, by using additional data (*e.g.* Data Augmentation (DA), noise injection, Semi-Supervised Learning (SSL)...);
- f, by influencing the architecture of the neural network and introducing layers that can favorably influence its behavior (*e.g.* convolutions, dropout, Batch Normalization (BN)...);
- Ω_{regul}, by adding loss terms to the optimization objective of the model, to penalize complex models over simpler ones or produce more robust features (*e.g.* weight decay / L2 normalization, invariance and reconstruction costs...).

Kukačka et al. (2017) present an in-depth review of the techniques used for Deep Learning (DL). We propose to put into perspective the most important ones in the context of this thesis regarding the improvement of the quality of DNNs' representations.

Most regularization techniques can have multiple interpretations and have many possible connections with one another (*cf.* Goodfellow et al. 2016, chapter 7). Here, we organized those methods following the points presented above. Regularization techniques are usually based on the idea of introducing prior human knowledge of types of models or behaviors that would eventually produce better predictions. Common priors include sparsity (having fewer active parameters or neurons) and smoothness of the decision function, both producing a simpler model that would overfit less; and compression, and invariance which aim at producing features that are more general and correspond to a larger number of examples.

2.2.3.1 Using more data

A first and logical way to improve the quality of the model is to use more data in $\mathcal{D}$, directly or indirectly as we will see.

Invariance through Data Augmentation (DA). A simple solution to both add invariance to a model and reduce overfitting is to artificially generate new images by producing random variants of existing images of the train set. This is done by changing factors that are considered to have no effect on the semantic content of the image. This technique is called Data Augmentation (DA) and is described

in details by Dyk and Meng (2001). In practice, we usually generate a new random variation of each input $\mathbf{x}$ each time it is used for training. Those random variations can be chosen among a large set of transformations: translations and rotations of the image, horizontal reflection, rescaling (zoom in or out), aspect ratio deformations, selection of random patches in the image, elastic transformations, jittering of the RGB color planes, changes in hue, contrast, brightness, random noise, *etc*. This technique is standard for the training of deep ConvNets (Simard et al. 2003; Cireşan et al. 2012; Krizhevsky et al. 2012, *etc*.). For example, for AlexNet, Krizhevsky et al. (2012) clearly state that DA is necessary to train their model. While effective, DA has limits, since it produces "new" images that are in fact highly correlated to the original images from which they were generated.

Other analogous ideas exist, like DeVries and Taylor (2017) who propose to apply the augmentation in the latent space by interpolating and extrapolating between samples' representations; or Goodfellow et al. (2015) who propose Adversarial Training, adding to the training set misclassified variations of the input found by gradient descent on the pixels, making the decision function more stable in the neighborhood of existing images.

Noise. Adding noise in the training process can be seen as an indirect way to add new artificial data that would be slightly different from the original input (Grandvalet et al. 1997). It also encourages the model to produce a smooth decision function around existing data points and their representation. Noise can be added to the input (Plaut 1986), representations (DeVries and Taylor 2017), weights (Kang et al. 2016), gradients (Neelakantan et al. 2016) or targets (called "label smoothing") (Szegedy et al. 2016). The effect of noise on generalization was already noted by An (1996) and more recently reviewed by Noh et al. (2017) and Kukačka et al. (2017). Methods based on *dropout*, discussed below, can also be interpreted under this angle of adding noise to the training (Li and F. Liu 2016). Finally, the noise introduced in the gradients by the stochasticity of the optimization algorithms (SGD and variants) and its optional momentum are also said to play a role in helping the model converge to a better solution.

Semi-Supervised Learning (SSL). It is also possible to take advantage of additional real unlabeled data that is much cheaper to obtain than labeled data. This approach called Semi-Supervised Learning (SSL) will be detailed in Section 2.2.4 and will be a particular focus of this thesis in Chapter 3.

2.2.3.2 Architectural changes

By changing the structure of the model f, it is possible to introduce many priors and make the model behave in more desired ways. This can be done through the choice of the architecture's building blocks, the design of explicitly invariant models, and the addition of particular layers like dropout and BN.

Architecture design. The choice of basic layers used in a model is a first way to introduce prior knowledge in the model. The type of layers, their number, organization, sizes, *etc.*, are all factors that are chosen based on prior knowledge about the complexity of the learning problem and how to solve it. In particular, convolutional layers can be seen as an "infinitely strong prior" (Goodfellow et al. 2016, chapter 9) because they force very sparse connections between the neurons of the input and output representations of the layer. Convolution and max-pooling also add respectively equivariance and local invariance properties toward translation.

If we know factors to which representations should be invariant, this knowledge can also be explicitly embedded in the architecture. For example, Mallat (2012) and Bruna and Mallat (2013) use properties of the wavelet scattering to obtain invariance toward various types of transformations; Dieleman et al. (2015) propose a ConvNet that is invariant to rotations using an approach similar to DA done in parallel; Cohen and Welling (2016) define convolutional and pooling layers that are equivariant to mathematical groups of geometric transformation; Mehr et al. (2018) propose an Auto-Encoder (AE) that is explicitly invariant to the pose of a 3D object, *etc*. Of course, those approaches can be very powerful when factors to which invariance is important are well known, but this is rarely the case. For example, in the context of natural image recognition, lots of variability exist in the shape, texture, positions, scales of the objects; variations that are complicated to model explicitly.

Masking connections. In order to better deal with the large number of connections, *i.e.* weights, that a DNN has, Srivastava et al. (2014) propose to randomly remove some connections, sampled differently for each batch during training. This method is called *dropout* and was shown to be effective on various DNNs and data (image, text, speech). The interpretation of this is that dropout prevents the co-adaptation of the neurons, encouraging their independence and producing more robust representations. Another interpretation is that this random effect makes the model behave as an ensemble of many models that are averaged when using the model for predictions. Variations of this were also proposed such as DropConnect (Wan et al. 2013) or DropBlock (Ghiasi et al. 2018) to refine the idea. Similar ideas also propose to add sparsity to connections of existing archi-

tecture, such as L. Zhu et al. (2018) who propose to remove residual connections in ResNets.

Normalizing representations. Another recent and very effective technique to improve the training and generalization of DNNs is to add normalization layers. First proposed by (Ioffe and Szegedy 2016), the Batch Normalization (BN) method proposes to normalize the intermediate representations of the network so that each neuron has a zero mean and unit variance on the batch. This method is said to reduce the internal covariate shift, *i.e.* the variation of the distribution of the inputs of a given layer because of the training of the previous layer. This mode of action is still discussed in the community, as Santurkar et al. (2018) shows that it might be due to a smoothing of the optimization landscape instead. In any case, the effectiveness of the BN is undeniable, and new variants have been proposed to solve the limits of BN such as Layer Norm (Lei Ba et al. 2016), Instance Norm (Ulyanov et al. 2016) or Group Norm (Wu and K. He 2018).

2.2.3.3 Loss based regularization

Finally, it is also frequent to add a loss term Ω_{regul} to encourage the model toward certain prior objectives that would balance the objective of exactly predicting the ground truth training labels.

Weights regularization. A very common prior in ML consists in influencing the weights $\mathbf{w}$ of the model using L1 or L2 penalty on them, *e.g.* $\Omega(\mathbf{w}) = 1/2||\mathbf{w}||_2^2$. Often called weight decay in the literature of DNNs, this term is added to the training loss with a penalization term λ (usually very small), and penalizes weights of strong magnitude that are shown to generalize less (Krogh and Hertz 1992). Similar regularizations can also be applied to the gradients (Seck et al. 2019) or on the Jacobian of the model to smooth the decision function: $\Omega(\mathbf{w}) = ||J_{f_\mathbf{w}}(\mathbf{x})||_F^2$ as used by Rifai et al. (2011) and studied in depth by Sokolić et al. (2017).

Adding stability for invariance. To encourage invariance of the classifier, Sajjadi et al. (2016) propose to encourage stability. The idea is that, considering an image and regardless of its class, when we apply various stochastic transformations (DA, dropout, noise, *etc*.) to the image that preserve its semantic information, the output of the model should remain the same for any variation of this input image. Sajjadi et al. (2016) propose to penalize the Euclidian distance between all the pairs of outputs produced from a similar initial image of the dataset. While very similar in the intuition to other methods presented (adding artificial variability

and noise to reinforce the model), this idea has the advantage of not requiring any label, and can thus be used in the context of SSL as we will see in Section 2.2.4.3.

Reconstruction. Using a reconstructing cost as part of the training of DNNs has been used since the early steps of DL (Hinton and Salakhutdinov 2006) to encourage compression. To do so, we design an encoder-decoder architecture, *i.e.* the model produces a representation of interest and then reconstructs the input from it. The intuition behind this objective is that reconstruction will force the model to represent the important patterns that compose the dataset. By forcing the model to compress input data into a more compact representation space, we make it find relevant compressed features. Indeed, the best way to compress the information is to produce more complex and semantic features, which are then interesting to use for semantic applications. Its exact effect is studied by Erhan et al. (2010). Reconstruction can be used to pre-train a model to improve its performance (Bengio et al. 2007), and was a key element to allow the training of DNN around those years to make those models converge well. Even if this regularization was less used within discriminative models in the recent years, Y. Zhang et al. (2016) does show that it is still an effective method of regularization for large ConvNets trained on ImageNet, as they improve the original results of VGG-16 (Simonyan and Zisserman 2015) using reconstruction. Because reconstruction has always been a key element for Semi-Supervised Learning (SSL) techniques, we propose to detail this regularization method more thoroughly below in Section 2.2.4.2.

Information Bottleneck (IB). Finally, more recently, regularization of DNNs was proposed through the use of the Information Bottleneck (IB) framework (Tishby et al. 1999). This technique proposes to use information theory measures to describe a desired behavior of a model. It first states that the generalization of a model should increase if a model is able to remove more information from the input while still providing enough information about the class label it should predict. This corresponds to both adding compression and invariance properties to the model. This is expressed mathematically by minimizing the mutual information $\mathcal{I}(X, H)$ between the input and the representation at constant mutual information $\mathcal{I}(H, Y)$ between the representation and the target class. Generalization bounds of IB have been studied (Shamir et al. 2010), and differentiable implementations adapted for DNNs have been proposed (Achille and Soatto 2016; Alemi et al. 2017; Pereyra et al. 2017).

These approaches based on the Information Bottleneck (IB) framework seem to be the most interesting and promising ones to develop new regularization methods for DNNs. As we have seen, they have the advantage, in their intuition, to combine both the ideas of compression and invariance but without any restriction

on the exact nature of the factors to which the model should be invariant. However, current methods that extend this idea to DNNs have strong limitations in their tractability. For this reason, we propose in Section 2.3 a new regularization method called SHADE, inspired by this IB framework, that encourages invariance in the representations.

2.2.4 Regularizing DNNs with and for Semi-Supervised Learning

To regularize a model, we can also use Semi-Supervised Learning (SSL). This subfield of Machine Learning (ML) proposes models that are trained on partially labeled dataset $\mathcal{D} = \mathcal{D}_{\text{sup}} \cup \mathcal{D}_{\text{unsup}}$ with labeled pairs $\mathcal{D}_{\text{sup}} = \{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})\}_{k=1..N_{\text{s}}}$ and unlabeled images $\mathcal{D}_{\text{unsup}} = \{\mathbf{x}^{(k)}\}_{k=1..N_{\text{u}}}$. We usually consider that all the images in $\mathcal{D}_{\text{unsup}}$ belong to one of the classes of interest and that the images in $\mathcal{D}_{\text{unsup}}$ have the same distribution as the images of $\mathcal{D}_{\text{sup}}$. Many techniques exist as presented by X. Zhu (2005). We propose to go over the most important ones, based on label propagation, reconstruction and stability techniques.

2.2.4.1 Label propagation methods

A first idea to address SSL is to *bootstrap* the classification model, *i.e.* using it to improve itself. This is done by training it on labeled images and using it to produce labels for the unlabeled images, predictions that can then be considered as ground truth and progressively added to $\mathcal{D}_{\text{sup}}$ for training (Blum and Mitchell 1998; X. Zhu and Ghahramani 2002).

This approach can of course be applied to DNNs as shown by Lee (2013), and is said to be similar to Entropy Regularization (Grandvalet and Bengio 2005). More recently, Shi et al. (2018) and Iscen et al. (2019) proposed some improvements to these methods, adding confidence levels to the labels they assign to unlabeled images, and using metric learning to improve the separability of the features produced to represent the images. Similar methods can also be used in the context of partially labeled problems to predict the missing labels (Durand et al. 2019).

While effective when correctly tuned, the risk with these kinds of approaches is of course to assign the wrong labels to the unlabeled images, which can cause a "vicious circle" where the model converges toward worse and worse predictions.

2.2.4.2 Reconstruction based methods

Because we have unlabeled images, it is natural to use an unsupervised loss as part of the semi-supervised training. Reconstruction is typically used to extract robust features that are relevant representations of elements in the full dataset. Thus, a mix of classification and reconstruction costs has been used for a long time for SSL, *e.g.* Ranzato and Szummer (2008). This way, the classification decision can be learned on labeled samples $\mathcal{D}_{\text{sup}}$ while representations used for the decision are learned on all the images in $\mathcal{D}$.

For this, an auto-encoding architecture is usually used, with an encoder E producing $\mathbf{h}$ from the input $\mathbf{x}$, and then feeding $\mathbf{h}$ to a decoder D to produce a reconstruction $\hat{\mathbf{x}}$. The reconstruction loss can vary but is usually a Mean-Squared Error (MSE):

$$\mathcal{L}(\mathbf{x}) = \mathcal{L}_{\text{rec}}\big(\mathbf{x}, D(E(\mathbf{x}))\big) = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2 \,. \tag{2.5}$$

As we mentioned, this loss makes the encoder produce features that represent the complete distribution of the dataset, without any consideration of labels. By constraining the size of the vector $\mathbf{h}$, compression can be encouraged, filtering only the most significant and frequent patterns and producing more complex and richer features. Compression can also be further encouraged by penalizing the simple copy of $\mathbf{x}$ in $\mathbf{h}$ – in particular if $\mathbf{h}$ has more capacity than $\mathbf{x}$. For this, sparse AE can be used (Ranzato et al. 2007b; Ranzato et al. 2008; Glorot et al. 2011) in order to produce compact representations that are likely to be more "complex" and closer to semantic meaning – to solve the semantic gap problem – thus better fitted for discriminative tasks.

Another common technique to improve AEs is to use Denoising Auto-Encoders (DAEs) (Vincent et al. 2008), where we add a random noise ε to the input but keep the reconstruction target intact:

$$\mathcal{L}(\mathbf{x}) = \mathcal{L}_{\text{rec}}\big(\mathbf{x}, D(E(\tilde{\mathbf{x}}))\big) \text{ with } \tilde{\mathbf{x}} = \mathbf{x} + \varepsilon \,. \tag{2.6}$$

Alain and Bengio (2014) shows that this encourages the encoder to explicitly learn the shape of the data distribution manifold, since the models need to find the closest point of $\tilde{\mathbf{x}}$ that lies on the data manifold, which corresponds to $\mathbf{x}$.

Finally, generative models also fall into this category of models that learn the distribution of the input data. This includes the famous Generative Adversarial Network (GAN) (Goodfellow et al. 2014) and its numerous derivatives, but also the Variational Auto-Encoder (VAE) (Kingma and Welling 2013) literature.

Overall, this idea of modeling the distribution of the data is at the core of many Deep Learning (DL) methods like Restricted Boltzmann Machines (Larochelle and Bengio 2008), Deep Belief Networks (Goh et al. 2013), Deep Generative Models

(Kingma et al. 2014) and discriminative Deep Neural Networks (DNNs) (Ranzato and Szummer 2008; Larochelle and Bengio 2008). It is also a key component of many SSL models based on AE using reconstruction (Weston et al. 2008; Turian et al. 2010), VAE (Kingma et al. 2014) or GANs (Springenberg 2016; Denton et al. 2017; Bodla et al. 2018).

In Chapter 3, we will discuss the way information can be encoded and decoded to both extract information from unlabeled data and be helpful for classification. To that end, we will propose a new architecture design.

2.2.4.3 Stability techniques

Another unsupervised criterion designed for SSL relies on the stability and smoothness of the prediction function. For example, Virtual Adversarial Training (Miyato et al. 2016), an extension of Adversarial Training (Goodfellow et al. 2015) for SSL, encourages the smoothness of the decision function around know data points regardless of the label.

Another major direction is the idea proposed by Sajjadi et al. (2016), making the prediction vector $\hat{\mathbf{y}}$ stable with regard to DA (translation, rotation, shearing, noise, *etc.*) and model stochasticity (dropout) for a given input. This consists in saying that all the outputs $\hat{\mathbf{y}}^{(i,k)}$ should be the same for $k = 1..K$, each k representing a random variation for the same input $\mathbf{x}^{(i)}$. This is measured by MSE between all the pairs, for an image i:

$$\mathcal{L} = \sum_{j=1}^{K} \sum_{k=1}^{K} ||\hat{\mathbf{y}}^{(i,k)} - \hat{\mathbf{y}}^{(i,j)}||_2^2 \,. \tag{2.7}$$

Following work by Laine and Aila (2017) and Tarvainen and Valpola (2017) propose variants of this idea, with refined ways to enforce stability.

All those techniques can be used for SSL because they do not rely on any label to produce stability but simply enforce local smoothness in the decision function to produce invariance.

Those strategies of adding stability in DNNs to improve the quality of the features in the context of SSL will be integrated in our proposed SSL framework in Chapter 3.

2.2.5 Improving the semantic quality of representations

When trying to further improve the quality of the representations of DNNs arises the question of *disentangling*. This problem can encompass a large range

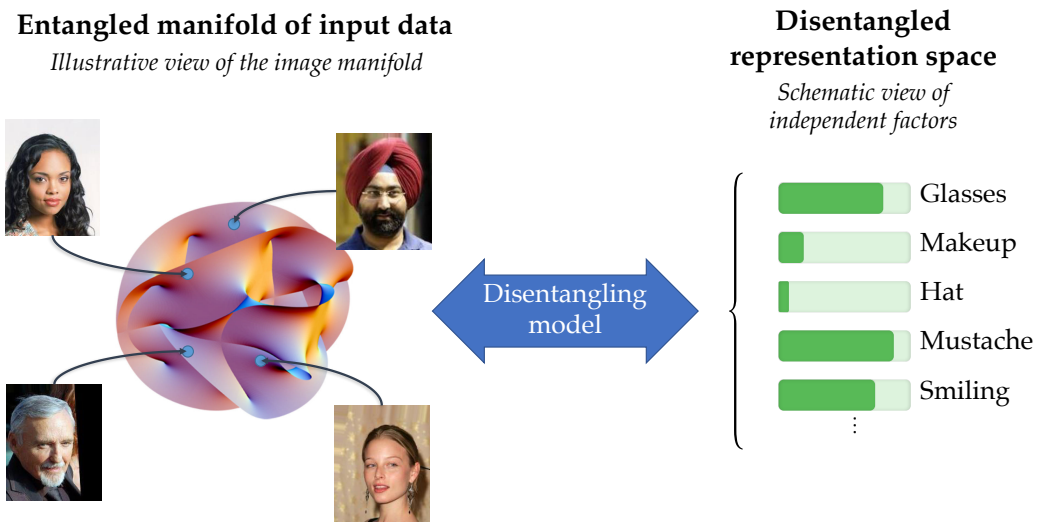


Figure 2.4. – **Illustration of the purpose of disentangling**. Here, the disentangling model provides a bijective mapping between a complex entangled manifold of faces and a series of independant semantic attributes.

of techniques, but the general principle is to propose models that produce representations that separate well the different factors of variation of the dataset. We illustrate this idea in Figure 2.4. Since the definition of disentangling (Higgins et al. 2018) and what the "factors of variation" are for a given dataset are both vague, they can vary widely from one method to the other which thus makes for a very broad literature.

A first category of techniques are simple generative models ($G(\mathbf{y}, \mathbf{h}_z)$) that combine and separate a binary class information $\mathbf{y}$ from the rest of the information (non-class related) stored in $\mathbf{h}_z$. While initial work relied on a natural tendency of the model to separate the information (Cheung et al. 2015; Perarnau et al. 2016), more recent models explicitly try to make the model remove class information from $\mathbf{h}_z$ (Lample et al. 2017; Yang Liu et al. 2018).

However, representing each factor by a binary representation is very constraining and limiting because factors often comprise internal variability that we would like to model. Thus a sort of variation on those generative models propose to better represent and separate the information by learning two complementary latent representations $\mathbf{h}_y$ and $\mathbf{h}_z$, one for the class $\mathbf{y}$ and one for non-class related attributes $\mathbf{z}$. A famous contribution by Mathieu et al. (2016) propose this sort of technique, using adversarial learning (Goodfellow et al. 2014) to separate the information. It was later followed by many extensions of this idea to better represent those factors of variation (Peng et al. 2017; Jaiswal et al. 2018; Yu Liu et al. 2018).

Finally, another category of methods propose to find and separate factors of variation without any label, an approach we could call unsupervised disentangling. In this case, the training objective mostly focuses on finding statistically independent representations that can also be used to reconstruct the input data. Many of those methods are based on the β-VAE (Higgins et al. 2017; T. Q. Chen et al. 2018), which proposes to increase the independence of the neurons in the latent space $\mathbf{h}$ of a VAE. The main issue with this type of models is that without any labels, it is impossible to ensure that semantic factors of interest will in fact be represented and separated as we would hope they do.

Being able to disentangle independent factors of variation is an important research direction for DL, producing more interpretable and reliable latent representations. This is why we will address this problem more deeply in Chapter 4 and propose a new approach that separates and structures the information in the latent space.

2.3 SHADE: Encouraging Invariance in DNNs

As we saw in Section 2.2.3, many solutions exist to regularize DNNs and improve their generalization performance. In particular, adding invariance in the representations of DNNs is a promising and well addressed solution, especially for classification purposes. Indeed, finding invariant representations has long been an important goal in Computer Vision (CV), as illustrated by the famous Scale-Invariant Feature Transform (SIFT) descriptors used before Deep Learning (DL). We have seen that recent solutions using invariance try to model the transformations a representation should be invariant to, for example, Data Augmentation (DA) will add invariance toward handcrafted factors like rotation, scale, Gaussian noise, *etc.*; and this is also the case for architectural blocks (*e.g.* translation invariance for max-pooling) and invariant architectures (*e.g.* rotation invariance for (Dieleman et al. 2015)).

However, for image recognition, it is very difficult to design an explicit modeling of all transformations a model should be invariant to. Thus, for our first contribution, we propose to work on the discriminative quality of the representations produced by a DNN by working on the invariance properties of those representations. In particular, we want to let the network find what *kind* of invariance should be produced by the model, using an entropy-based measure as a surrogate for invariance.

Indeed, as we have seen at the end of Section 2.2.3.3, the Information Bottleneck (IB) framework is a recent and interesting source of inspiration to design new regularization methods for DL. Based on this idea, we propose a regularizer called

SHAnnon DEcay (SHADE) that aims at improving the intra-class invariance of the representations produced by the model. Our main motivation is to design a model that is robust to variations in the training data while preserving class information. Because of this, our regularizer minimizes the entropy of the representations H conditionally to a class Y:

$$\min \mathcal{H}(H \mid Y)\,. \tag{2.8}$$

In this section, we will detail the motivations of using conditional entropy to measure the invariance of the representations. We then develop a differentiable and tractable expression of this SHADE criterion. Finally, we evaluate its ability to regularize a large variety of architectures, on CIFAR-10 and ImageNet; and when using datasets with a small number of samples.

2.3.1 Context

Notations and definitions. For our work using information theory, we use the following random variables and notations. We consider a random variable input $X \in \mathcal{X}$ associated to a target class variable $Y \in \mathcal{Y} = \{1, 2, .., N_{\text{cls}}\}$. X is fed to a DNN named E (for encoder) of parameters $\mathbf{w}$ producing a succession of intermediate representations H_ℓ after each layer ℓ; H designating any of those representations. We will also use information theory measures. First, the Shannon entropy noted $\mathcal{H}$ (*cf.* Cover and Thomas 1991). Second, the mutual information noted $\mathcal{I}$, which quantifies the amount of information shared between two random variables. Considering $U \in \mathcal{U}$ and $V \in \mathcal{V}$ with respective marginal probability distributions P_U and P_V and mutual distribution $P_{(U,V)}$, we have:

$$\mathcal{I}(U, V) = \int_{\mathcal{U}\times\mathcal{V}} P_{(U,V)}(u, v) \log \left[\frac{P_{(U,V)}(u, v)}{P_U(u)P_V(v)}\right] \mathrm{d}u\,\mathrm{d}v \tag{2.9}$$

Useful properties between the two measures are:

$$\mathcal{I}(U, V) = \mathcal{I}(V, U) = \mathcal{H}(U) - \mathcal{H}(U \mid V) = \mathcal{H}(V) - \mathcal{H}(V \mid U) \tag{2.10}$$

Information Bottleneck (IB). The Information Bottleneck (IB) framework proposes to regularize the encoder E by optimizing the following objective:

$$\begin{cases} \max_{\mathbf{w}} & \mathcal{I}(H, Y) \\ \text{s.t.} & \mathcal{I}(X, H) < D \end{cases} \tag{2.11}$$

This can be rewritten with a Lagrange multiplier β as:

$$\max_{\mathbf{w}} \mathcal{I}(H, Y) - \beta \mathcal{I}(X, H)\,. \tag{2.12}$$

In this optimization problem, the first term $\mathcal{I}(H, Y)$ maximizes the mutual information between the representation and the target, which can be interpreted as equivalent to the usual classification training objective minimizing cross-entropy between $\hat{\mathbf{y}}$ and $\mathbf{y}$. The second term $-\mathcal{I}(X, H)$ minimizes the mutual information between the input and the representation, making the model forget most of the information. Indeed, ideally, the model should filter out most of the input information that is useless and focus only on the information that is related to the target Y.

2.3.2 Measuring Invariance with Conditional Entropy

Inspired by this IB framework, we propose to analyze in more details why we propose to use entropy, and more precisely conditional entropy, as a measure of the invariance that we want to encourage.

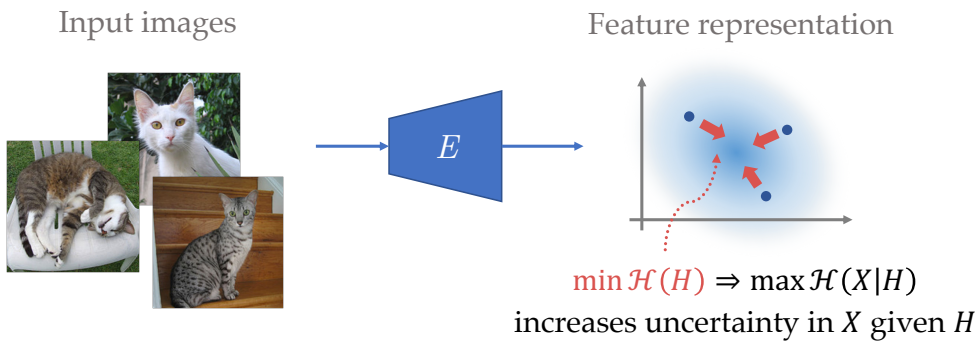
Entropy as a measure of invariance. First, to study how the entropy of a representation $\mathcal{H}(H)$ can be interpreted as a measure of the invariance of the representations of a model, let's write:

$$\mathcal{H}(H) = \mathcal{I}(X, H) + \mathcal{H}(H \mid X)\,. \tag{2.13}$$

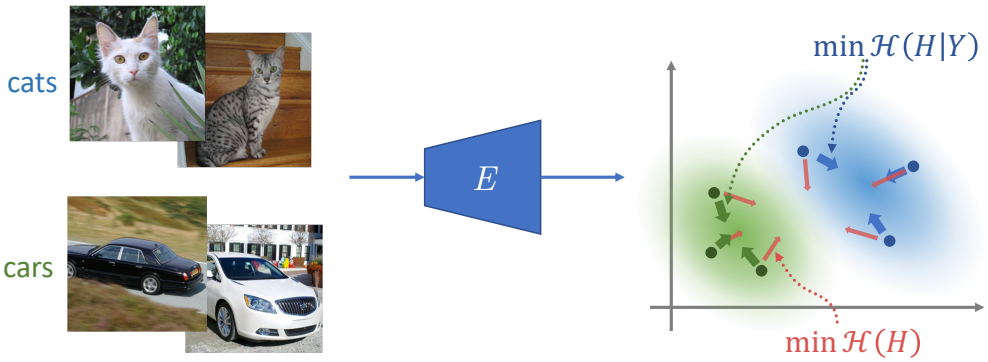
Considering that our encoder E is a deterministic mapping of X into H (*i.e.* a model without stochastic noise), we know that $\mathcal{H}(H \mid X) = 0$, therefore:

$$\mathcal{H}(H) = \mathcal{I}(X, H) = \mathcal{H}(X) - \mathcal{H}(X \mid H)\,. \tag{2.14}$$

$\mathcal{H}(X)$ is the entropy of the data and is fixed, therefore, $\mathcal{H}(H)$ is inversely related to $\mathcal{H}(X \mid H)$. This entropy $\mathcal{H}(X \mid H)$ is a good measure to quantify how invariant a representation is. Indeed, if a representation is invariant to many changes in the image, this means that many inputs have the same representation. Consequently, given a representation sample, it will be difficult to guess from which input it has been computed. These properties are perfectly captured by $\mathcal{H}(X \mid H)$, representing the uncertainty in the input X knowing a representation H. The bigger the uncertainty, the harder it is to predict X. The schematic behavior of a model on which we minimize $\mathcal{H}(H)$ is represented in Figure 2.5a, where representations are made more and more similar. Formally, when trying to guess X knowing H, we



(a) **Effect of minimizing** $\mathcal{H}(H)$. Representations get more similar, increasing the entropy of the input given the representation and thus the invariance.



(b) **Effect of minimizing** $\mathcal{H}(H \mid Y)$. Representations get more similar for each class independently, making the representations more invariant *intra-class*.

Figure 2.5. – **Illustration of the effect of entropy minimization** and the difference between normal and conditional entropy as a regularizer.

can lower bound the error made in the best case, with an increasing function of the conditional entropy (Blot et al. 2018a, Appendix D). Therefore, it seems that $\mathcal{H}(H)$ is a good measure of the invariance of the model.

In the particular case of Deep Learning (DL), K. He et al. (2016), proposing ResNet, explain that the stacking of multiple layers is responsible for improving the generalization of DNNs. This fact can be explained by the data processing inequality (Cover and Thomas 1991). This states that in the case of finite input space, each additional computation layer can only remove a certain amount of information and cannot add any. As clearly illustrated in Tishby and Zaslavsky (2015), for each stage, the representation has a lower entropy than the representation of the preceding layer. Increasing the depth increases the capacity of the

network to reduce the overall entropy of the DNN representation thus increasing their invariance.

Importance of conditional entropy. We have seen that lowering the entropy of the representation enables to make it more invariant. However, another fact reported by K. He et al. (2016) is that stacking layers increases the difficulty to train the network. Indeed, when reducing too much the entropy, there is a risk that the information about the label is filtered as well. The representation is so invariant that it is no longer possible to distinguish between the classes. In K. He et al. (2016), they solve this issue by forcing the transmission of additional information through skip-connections while IB prescribes to maximize the compression rate at constant information about the label. All this highlights the fact that having invariant representations is interesting if it is intra-class invariant.

Indeed, in our case above, we focused on a too broad meaning of invariance that could lead to some issue. We stated that we want to be invariant to any kind of information from the input X, which is actually not true. In fact, we want to keep information regarding the label of the input. To illustrate, we do not want two inputs from different classes to have the same representation, but do not matter if inputs from the same class have the same representation. This is why we prefer to focus on a criterion quantifying the intra-class compression rate in order to maximize intra-class invariance: $\mathcal{H}(H \mid Y)$. This is schematized in Figure 2.5b that shows the difference between conditional and non-conditional entropy on representations.

Behavior of SHADE and comparison with IB. Our criterion $\mathcal{H}(H \mid Y)$ therefore differs from standard IB regularizers based on $\mathcal{I}(X, H)$ (*e.g.* Achille and Soatto 2016; Alemi et al. 2017). In fact, the mutual information minimized in IB can be rewritten:

$$\mathcal{I}(X, H) = \mathcal{H}(H \mid Y) + \mathcal{I}(Y, H)\,. \tag{2.15}$$

Thus, we see that our choice of optimizing only $\mathcal{H}(H \mid Y)$ ignores the term $\mathcal{I}(Y, H)$. In fact, we argue that removing $\mathcal{I}(Y, H)$ from the optimization is a good thing since this term is analogous to the classification loss and should not be minimized since it would be detrimental to the actual objective. When minimizing $\mathcal{I}(X, H)$, there is no control over how both terms $\mathcal{H}(H \mid Y)$ and $\mathcal{I}(Y, H)$ are affected. Our regularizer is therefore beneficial in the sense that minimizing $\mathcal{H}(H \mid Y)$ does not conflict with the mutual information $\mathcal{I}(Y, H)$ between the representation and the label, information useful for classification that should not be penalized.

The behavior of SHADE is illustrated in Figure 2.6, where we show for each layer of a DNN this decomposition of the information $\mathcal{H}(H)$ into the desired information $\mathcal{I}(Y, H)$ used for classification, and the information SHADE intends to

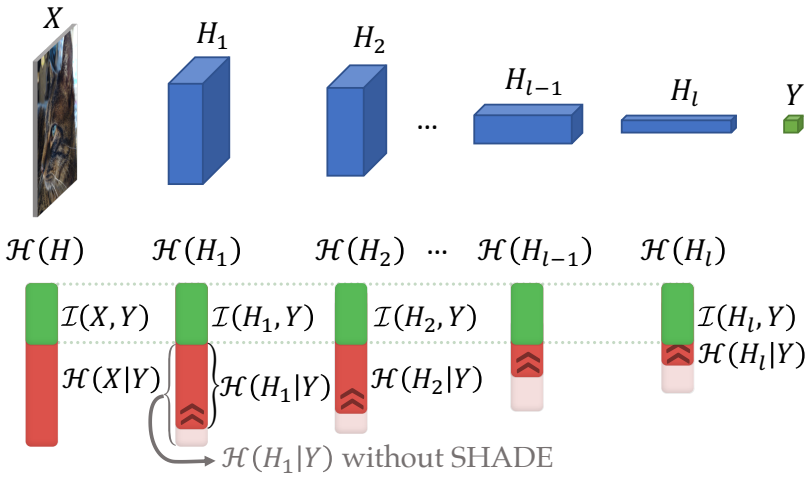


Figure 2.6. – **Illustration of the effect of SHADE.** Considering a DNN architecture with corresponding layers' entropies, we show the layer-wise action of SHADE. Given that $\mathcal{H}(H_\ell) = \mathcal{I}(H_\ell, Y) + \mathcal{H}(H_\ell \mid Y)$, SHADE minimizes $\mathcal{H}(H_\ell \mid Y)$ without affecting $\mathcal{I}(H_\ell, Y)$.

minimize $\mathcal{H}(H \mid Y)$. Next, we will further describe the development SHADE, our regularization term based on the conditional entropy $\mathcal{H}(H \mid Y)$ designed to drive the optimization toward more intra-class invariant representations.

2.3.3 Entropy-based Regularization for Deep Neural Networks

We now propose to describe how we develop and instantiate our SHADE regularizer. This section will present the important choices that we make to obtain the loss function for SHADE, all the details about this process are given in Appendix A.

Toward a unit-wise regularization. Considering a DNN composed of L layers that transform sequentially the input, we first propose to regularize all the layers independently and minimize the sum of entropies $\Omega_{\text{layers}} = \sum_{\ell=1}^{L} \mathcal{H}(H_\ell \mid Y)$.

Noting $H_{\ell,i}$ the unit neurons of H_ℓ, we propose to use the upper bound $\mathcal{H}(H_\ell \mid Y) \leq \sum_{i=1}^{D_\ell} \mathcal{H}(H_{\ell,i} \mid Y)$ to define a unit-wise criterion that SHADE will seek to minimize. Since we only work on individual and independent neurons, we will use the notation H instead of $H_{\ell,i}$ for simplicity in the rest of the chapter. We thus seek to minimize:

$$\Omega_{\text{units}} = \sum_{l=1}^{L} \sum_{i=1}^{D_\ell} \underbrace{\mathcal{H}(H \mid Y)}_{\omega_{\text{unit}}(H|Y)} . \tag{2.16}$$

Limitations. Finding a tractable and differentiable expression of $\mathcal{H}(H \mid Y)$ is not obvious for multiple reasons. The conditional entropy $\mathcal{H}(H \mid Y)$ requires to compute N_{cls} different entropies $\mathcal{H}(H \mid Y_k)$, which, when working on batches, reduces the number of samples used to compute each entropy down to problematic levels. Entropy estimators are also very inaccurate with few samples and require a discretization of the space using a histogram, an operation that raises new issues on how to do it and to keep it tractable since bins for each neuron would lead to important memory usages.

Reducing complexity with a binary latent code. Considering a neuron H prior to the non-linearity, the ReLU can be interpreted as making it act as a detector, returning a signal when a certain pattern is present in the input. We thus propose to associate a binomial variable Z to each unit variable H (before ReLU). This variable Z indicates if a particular pattern is present in the input ($Z = 1$ when $H \gg 0$) or not ($Z = 0$ when $H \ll 0$).

We then assume that this variable Z is a sufficient statistic (see definition by Cover and Thomas 1991) of H for Y, *i.e.* that it contains the necessary information from H to predict the class Y. This means that we have $\mathcal{I}(H, Y) = \mathcal{I}(H, Z)$ and we get the equivalent equality $\mathcal{H}(H \mid Y) = \mathcal{H}(H \mid Z)$. This has the advantage of requiring to compute only 2 entropies ($Z = 0$ and 1) instead of N_{cls}. We finally obtain:

$$\omega_{\text{unit}}(H \mid Y) = \mathcal{H}(H \mid Y) = \mathcal{H}(H \mid Z) = \sum_{z \in \{0,1\}} p(z)\mathcal{H}(H \mid Z = z). \tag{2.17}$$

This variable Z represents a semantically meaningful factor about the class Y and from which the input X is generated, and H is a quantification of the possibility for this semantic attribute Z to be present in the input or not. Interpreting $p(Z \mid H)$ as the probability of presence of the semantic attribute in the input, we choose to define it as:

$$\begin{cases} p(Z = 1 \mid H) = \sigma(H) \\ p(Z = 0 \mid H) = 1 - \sigma(H) \end{cases} \quad \text{using the sigmoid function } \sigma(H) = \frac{1}{1 + e^{-H}}. \tag{2.18}$$

Using a variance bound for tractability. To solve the problem of the complexity of estimating the entropies, we propose to use a simple bound on $\mathcal{H}(H \mid Z)$ using the variance, which does not require the definition of a histogram:

$$\mathcal{H}(H \mid Z) \leq \frac{1}{2} \ln \big(2\pi e \mathcal{V}\text{ar}(H \mid Z)\big). \tag{2.19}$$

This bound converges well and should be very tight if we consider that activations approximately follow a Gaussian distribution, which is often the case. We then propose to simplify the expression and only keep the simpler term $\mathcal{V}\mathrm{ar}(H \mid Z)$. We then estimate our loss using Monte-Carlo sampling on K inputs, and use a moving average on the neurons to estimate the expectancy $\mu^z = \mathbb{E}(H \mid z)$ to compute this variance. We thus obtain our final loss for SHADE:

$$\Omega_{\text{SHADE}} = \sum_{\ell=1}^{L} \sum_{i=1}^{D_\ell} \sum_{z \in \{0,1\}} p(Z_{\ell,i} = z \mid H) \mathcal{V}\mathrm{ar}(H \mid Z_{\ell,i} = z)\,; \tag{2.20}$$

$$\boxed{\Omega_{\text{SHADE}} = \sum_{\ell=1}^{L} \sum_{i=1}^{D_\ell} \sum_{k=1}^{K} \sum_{z \in \{0,1\}} p\left(Z_{\ell,i} = z \,\middle|\, H_{\ell,i}^{(k)}\right) \left(H_{\ell,i}^{(k)} - \mu_{\ell,i}^{z}\right)^2 .} \tag{2.21}$$

We obtain a regularizer that is applied on each neuron of the network, that is differentiable, tractable and that can be integrated into the usual optimization process of a DNN. Notably, SHADE requires only a small amount of additional computation and memory usage (computation and storage of two moving averages per neuron). For comparison, SHADE adds only half as many parameters as Batch Normalization (BN) does.

Comparison to Non-conditional Entropy and relation to Weight Decay

We propose to compare SHADE to a variant based on optimizing the entropy of the representations $\mathcal{H}(H)$ instead of $\mathcal{H}(H \mid Y)$. For this, one can use a variance bound similar to Equation 2.19: $\mathcal{H}(H) \leq {}^1\!/_2 \ln(2\pi e \mathcal{V}\mathrm{ar}(H))$ to derive a loss in order to minimize the representations' entropy $\mathcal{H}(H)$. Thus, minimizing the variance of the representations is an upper bound of the entropy. This can be done by penalizing the empirical variance, using an alternative loss called *VarEntropy*, constructed the same way SHADE has been derived, but avoiding the introduction of a latent variable Z:

$$\Omega_{\text{VarEntropy}} = \frac{1}{K} \sum_{k=1}^{K} \left(H^{(k)} - \mathbb{E}(H)\right)^2 . \tag{2.22}$$

Interestingly, we can show that the weight decay actually reduces this variance $\mathcal{V}\mathrm{ar}(H)$ and therefore $\mathcal{H}(H)$ under some conditions. In fact when $H = \mathbf{w}^\top X + \boldsymbol{b}$, by estimating $\Lambda = \mathcal{C}\mathrm{ov}(X)$, the variance takes the immediate form $\mathcal{V}\mathrm{ar}(H) = \mathbf{w}^\top \Lambda \mathbf{w}$. If $\Lambda = Id$ – the identity matrix –, meaning that the input coordinates are considered independent and with unit variance, then $\mathcal{V}\mathrm{ar}(H) = ||\mathbf{w}||_2^2$. It corresponds to the weight decay regularization or L_2 penalty. Even if within a DNN layer, Batch Normalization (BN) tends to enforce this unit variance hypothesis and the depth

	MLP	AlexNet	ResNet	Inception
No regul.	62.38	83.25	89.84	90.97
Weight decay	62.69	83.54	91.71	91.87
VarEntropy	63.70	83.61	91.72	91.83
Dropout	65.37	85.95	89.94	91.11
SHADE	66.05	85.45	**92.15**	**93.28**
SHADE + Dropout	**66.12**	**86.71**	92.03	92.51

Table 2.2. – **Classification accuracy (%) on CIFAR-10** test set.

of DNN tends to ensure the independence hypothesis the weight decay remains poor at improving generalization as illustrated in C. Zhang et al. (2017).

2.3.4 Evaluation

We now propose different experiments to validate the capability of SHADE to regularize common Computer Vision (CV) architectures. First, classification results on CIFAR-10 for different architectures to show that SHADE is able to regularize a broad range of models. We also validate that our regularization can be applied on large scale models and datasets by applying it on ImageNet. We then investigate the behavior of SHADE when using few images before some more in-depth analysis of the behavior of SHADE and its intuitions.

2.3.4.1 Image Classification with Various Architectures on CIFAR-10

First, we perform image classification on the CIFAR-10 dataset, which contains 50k training images and 10k test images of 32×32 RGB pixels, fairly distributed within 10 classes (Krizhevsky and Hinton 2009). Following the architectures used by C. Zhang et al. (2017), we use a three-layer Multi-Layer Perceptron (MLP), and an AlexNet-like model with 3 convolutional and 2 fully connected layers and a small Inception model. We also use a ResNet architecture from Zagoruyko and Komodakis (2016). Those architectures represent a large family of DNN and some have been well studied in C. Zhang et al. (2017) regarding their generalization ability. For training, we use randomly cropped images of size 28×28 with random horizontal flips. For testing, we simply center-crop 28×28 images. We use momentum SGD for optimization (same protocol as C. Zhang et al. 2017).

We compare SHADE with three regularization methods, namely *weight decay*, *dropout* and *VarEntropy* presented in Equation 2.22. For all architectures, the reg-

	Accuracy (%)	
	Top-1	Top-5
ResNet-101	77.56%	93.89%
WELDON	78.51%	94.65%
WELDON + SHADE	**80.14%**	**95.35%**

Table 2.3. – **Classification accuracy (%) on ImageNet** validation set.

ularization parameters have been cross-validated to find the best ones for each method and the obtained accuracies on the test set are reported in Table 2.2.

We obtain the same trends as C. Zhang et al. (2017), which get a small improvement of 0.31% with weight decay on AlexNet. The improvement with weight decay is slightly more important with ResNet and Inception (0.87% and 0.90%) probably thanks to the use of Batch Normalization (BN). In our experiments, dropout improves generalization performances only for AlexNet and MLP. It is known that the use of BN lowers the benefit of dropout, which is in fact not used in K. He et al. (2016).

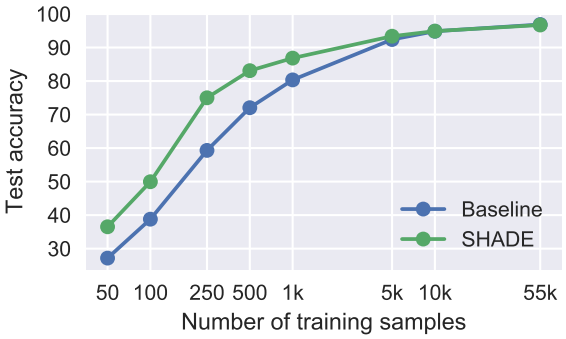
We first notice that for all kind of architectures the use of SHADE significantly improves the generalization performance. It demonstrates the ability of SHADE to regularize the training of deep architectures. Moreover, SHADE systematically outperforms other regularizations of the same type such as weight decay or Var-Entropy, illustrating the advantage of minimizing the conditional entropy instead of the entropy directly.

Finally, SHADE shows better performances than dropout on all architecture except on AlexNet, for which they seem to be complementary, probably because of the very large number of parameters in the fully-connected layers, with best performances obtained with SHADE coupled with dropout. This association is also beneficial for MLP. On Inception and ResNet, even if dropout and SHADE independently improve generalization performances, their association is not as good as SHADE alone, probably because it enforces too much regularization.

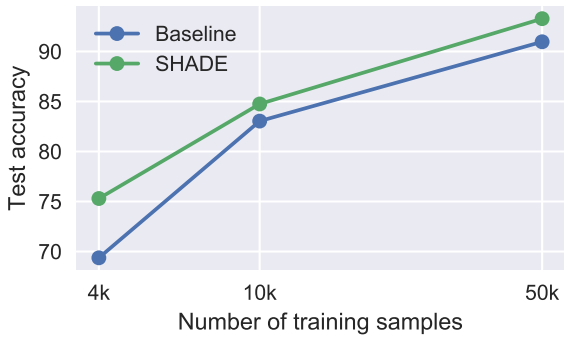
2.3.4.2 Large Scale Classification on ImageNet

In order to experiment SHADE regularization on very large scale dataset, we train on ImageNet (Russakovsky et al. 2015) a WELDON network from Durand et al. (2016) adapted from ResNet-101. This architecture changes the forward and pooling strategy by using the network in a fully-convolutional way and adding a max+min pooling, thus improving the performance of the baseline network.

There are two differences between the two networks: first, for WELDON the images are simply re-scaled to size 448×448 before being forwarded into the



(a) Results for MNIST-M.



(b) Results for CIFAR-10.

(c) Examples of MNIST-M images misclassified by the baseline and correctly classified using SHADE, both trained with 250 samples.

Figure 2.7. – **Results when training with a limited number of samples** in the training set for MNIST-M and CIFAR-10 with and without SHADE.

network; second, compared to ResNet-101 architecture, the final average pooling and fully connected layer are replaced with a 1×1 convolutional layer and a particular max+min pooling described in Durand et al. (2016). This layer averages the 50 highest and 50 lowest activations of the 14×14 output feature map to give a prediction.

Results are summarized in Table 2.3. We report the results of a pre-trained ResNet-101 and the WELDON architecture using those weights. After fine-tuning the network using SHADE we obtain an improvement over the WELDON baseline, demonstrating the ability to apply SHADE on very large scale image classification successfully.

2.3.4.3 Training with a Limited Number of Samples

When datasets are small, DNNs tend to overfit quickly and regularization becomes essential. zBecause it tends to filter out information and make the network more invariant, SHADE seems to be well fitted for this task. To investigate this, we propose to train DNNs with and without SHADE on CIFAR-10 and MNIST-M with different numbers of samples in the training set.

First, we tested this approach on the digits dataset MNIST-M (Ganin and Lempitsky 2015). This dataset consists of the MNIST digits where the background and digit have been replaced by colored and textured information (see Figure 2.7c for examples). The interest of this dataset is that it contains lots of unnecessary

information that should be filtered out, and is therefore well adapted to measure the effect of SHADE.

We train a simple ConvNet (3 convolutional layers with pooling and 1 fully connected layer) with different numbers of samples of MNIST-M. The samples, chosen uniformly in the classes for each value of N, are kept the same for the baseline and SHADE. The results can be seen in Figure 2.7a. We can see that especially for small numbers of training samples (< 1000), SHADE provides an important gain of 10 to 15% points over the baseline. This shows that SHADE helped the model in finding invariant and discriminative patterns using fewer data samples.

Additionally, Figure 2.7c shows samples that are misclassified by the baseline model but correctly classified when using SHADE. These images contain a large amount of intra-class variance (color, texture, etc.) that is not useful for the classification tasks. By encouraging the model to discard information, SHADE obtains an important performance gain on this dataset and especially when only few training samples are given.

Finally, to confirm this behavior, we also applied the same procedure in a more conventional setting by training an Inception model on CIFAR-10. Figure 2.7b shows the results in that case. We can see that once again SHADE helps the model gain in performance and that this behavior is more noticeable when the number of samples is limited, allowing a gain of 6% when using 4000 samples.

2.3.5 Discussion of SHADE

With SHADE, we introduced a new regularization method for DNNs training that focuses on minimizing the entropy of the representation conditionally to the labels.

Inspired by the Information Bottleneck (IB) framework, we proposed to increase the invariance of the representations of DNNs without any prior model on the factors to which those representations should be invariant and let the model find those factors. Thus, SHADE is able to increase the intra-class invariance of the model while keeping class information.

To develop SHADE, we make some hypotheses that we validate in additional experiments described in Appendix A. First, we diverge from the regular IB framework as described in Equation 2.15 because we show that part of the mutual information $\mathcal{I}(X, H)$ is important for classification, as validated in Section A.2.1. We also assume that neurons of a DNN act as a binary detector of factors that encode the class information, a hypothesis that we validate in Section A.2.2.

We also showed that SHADE significantly outperforms standard approaches such as weight decay or dropout with various DNN architectures on CIFAR-10. We also validated the scalability of SHADE by applying it on ImageNet. The invariance potential brought out by SHADE is further illustrated by its ability to ignore irrelevant visual information (texture, color) on MNIST-M. We also highlight the increasing benefit of our regularizer when the number of training examples becomes small.

2.4 Conclusion

In this chapter, we introduced the recent development in the domain of Deep Learning (DL) and in particular with Convolutional Neural Networks (ConvNets). We saw that deep convolutional architectures are now ubiquitous in Computer Vision (CV) research and produce impressive results. To reach these performances however, because of their complexity, a key aspect of research focus on proposing regularization techniques that are required to make those architectures generalize well to unseen data. As we have seen, many possible regularization approaches exist, in particular by adding new data, changing the structure of the model or adding loss terms to enforce constraints on the model.

Our first contribution in this thesis consisted in proposing a new regularization method called SHADE, that takes the form of a new loss. Inspired by the Information Bottleneck (IB) framework, SHADE influences the representations of a ConvNets to be more invariant to the variance in visual inputs conditionally to the class. SHADE thus makes the representations more intra-class invariant and allows to obtain better classification results, as we demonstrated by using many DNN architectures with SHADE on CIFAR-10 but also on ImageNet.

Another important direction to improve DNNs lies in the possibility of using additional unlabeled data, which could greatly help the generalization performances for a low cost compared to producing new labeled data. Semi-Supervised Learning (SSL) techniques address this issue, often by mixing regularization ideas adapted to discriminative models with motivations similar to the ones of SHADE; and generative properties with models that are able to encode an image and decode its representation back into the image. In this case, an invariance regularization like SHADE would conflict with this second goal. To solve this conflict, it is possible to work on the architecture of the model and go beyond the usual basic one-branch encoder-decoder architecture. This idea will be further developed in Chapter 3 where we propose a new type of architecture to address this issue.

Finally, we also saw that designing ConvNets that produce disentangled representations of the different factors of variation of the data is an interesting direction

to improve the semantic quality of those models, making them both more powerful for other tasks and more interpretable. To further work on this idea of structuring the information of deep ConvNets, in Chapter 4, we address the problem of disentangling by proposing a new architecture that separates the information in a structured dual latent space for image editing and data generation.

CHAPTER 3

SEPARATING DISCRIMINATIVE AND NON-DISCRIMINATIVE INFORMATION FOR SEMI-SUPERVISED LEARNING

Chapter abstract

Regularizing Deep Neural Networks (DNNs) by encouraging invariance or by encouraging reconstruction are two popular methods with conflicting behaviors toward classification. In this chapter, we study those methods in more details and propose a new framework to make them cooperate in the context of Semi-Supervised Learning (SSL), leveraging unlabeled data to improve generalization performances of image classifiers. To do so, we investigate ways to organize the information in the latent space and introduce a two-branch encoder-decoder architecture called HybridNet. The first branch of HybridNet receives supervision signal and is dedicated to the extraction of invariant class-related representations. The second branch is fully unsupervised and dedicated to model information discarded by the first branch to reconstruct input data. To further support the expected behavior of our model, we propose an original training objective. It favors stability in the discriminative branch and complementarity between the learned representations in the two branches. At the time of publication, HybridNet was able to outperform state-of-the-art results on CIFAR-10, SVHN, and STL-10 in various semi-supervised settings.

The work in this chapter has led to the publication of a conference paper:

- Thomas Robert, Nicolas Thome, and Matthieu Cord (2018). "HybridNet: Classification and Reconstruction Cooperation for Semi-Supervised Learning". In: *European Conference on Computer Vision (ECCV)*.

Contents

3.1 Introduction

We have seen that DNNs and Convolutional Neural Networks (ConvNets) now show impressive state-of-the-art results on many Computer Vision (CV) tasks (image classification (K. He et al. 2016), object localization (Dai et al. 2016), multi-modal embedding (Engilberge et al. 2018; Carvalho et al. 2018), *etc.*). To achieve these results, important progress has been made to provide ways to regularize the huge number of parameters of DNNs (*e.g.* weight decay, dropout, Batch Normalization (BN), *etc.*). In the previous chapter, we saw that these techniques mostly consist in introducing prior knowledge of models that should perform well; thus influencing the model's architecture, the smoothness of its decision boundary, its invariance capability, *etc*. In this regard, we introduced SHADE which proposes to encourage intra-class invariance. However, we also saw that another very interesting direction to regularize and improve ConvNets is through Semi-Supervised Learning (SSL).

Indeed, supervised learning of DNNs requires very large labeled datasets like ImageNet and its now 1.3 million images. Annotating such large quantities of data is very expensive and remains an obstacle to the application of those models to new tasks. Thus, in this chapter, we propose to tackle the problem of training DNNs using Semi-Supervised Learning (SSL). In this context, only a small portion of our large dataset has labels, making it much less expensive since the cost of the dataset usually come from the expensive human annotation of the ground

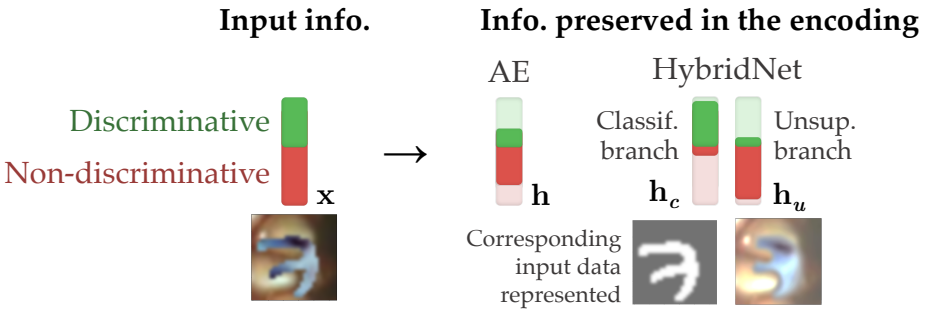


Figure 3.1. – **Illustration of how an AE and HybridNet encode the information differently**. Considering an image contains discriminative and non-discriminative information represented on the left. On the right, we show the information preserved in the latent space of an AE and of HybridNet. The auto-encoder will retain as much information as possible for reconstruction, possibly keeping lots of non-discriminative information necessary to optimize the MSE loss. In HybridNet, we are able to preserve more information and organize it in the two branches, specializing $\mathbf{h}_c$ on discriminative information and leaving $\mathbf{h}_u$ to encode the non-discriminative one.

truth. This problem is particularly true in domains where the annotation process requires rare expertise such as the annotation of medical imagery. In such domains, annotated data is very rare and effective SSL methods could be crucial. Indeed, the idea of SSL is to take advantage of all the images (labeled or not) so that the model can grasp the diversity of the images it should be able to recognize, and simply use a few labeled images to effectively learn the classification rule, which will generalize better thanks to this more general knowledge of the dataset learned by the model.

As we saw in Section 2.2.4, the two main types of approaches for SSL are as follows. The first one is by improving the quality of the model toward classification, introducing more invariance, often by enforcing stable predictions with regard to artificial sources of variability introduced by different stochastic perturbations (Sajjadi et al. 2016; Laine and Aila 2017; Tarvainen and Valpola 2017). Compared to SHADE which encourages intra-class invariance, those methods encourage intra-instance invariance to those stochastic perturbations. This criterion, not requiring any label, can be used on the unlabeled samples of the dataset to improve the quality of the features represented by the model.

The second major direction is reconstruction- or generation-based methods (Bengio et al. 2007). In this case, the idea is that by making the model able to encode and decode all the images of the dataset, we obtain representations that model the whole distribution of images, even the ones for which we do not have a label. Thanks to this, we can produce more robust representations that will

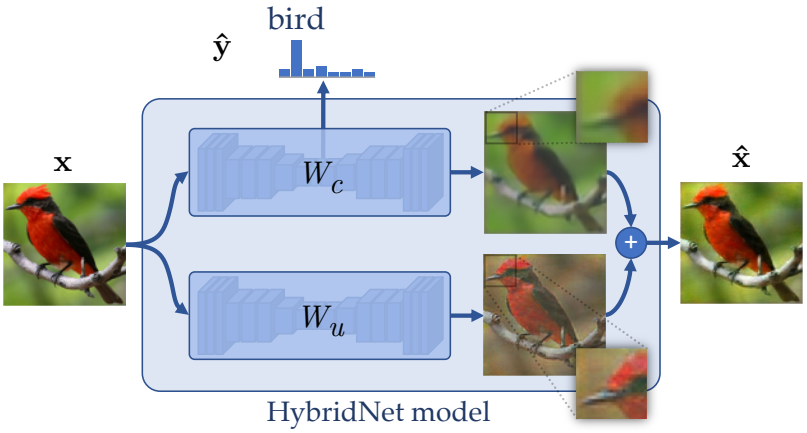


Figure 3.2. – **Illustration of HybridNet's behavior**. The input image is processed by two network paths of weights W_c and W_u; each path produces a partial reconstruction, and both are summed to produce the final reconstruction, while only one path is used to produce a classification prediction. Thanks to the joint training of both tasks, the weights W_c and W_u influence each other to cooperate.

generalize better (Le et al. 2018). This strategy has been followed by historical deep learning approaches (Hinton and Salakhutdinov 2006), but also in some promising recent results with modern ConvNets (Y. Zhang et al. 2016).

Those two directions seem to be complementarity since invariance improves classification and reconstruction should improve the generalization capabilities of the features. However, when we look more closely at the behavior of those techniques, they actually seem to have conflicting goals. Indeed, by definition, to reconstruct an image, all the information from the input is necessary. This information can be transformed, reshaped, but must be retained. On the other hand, classification and intra-class invariance regularizers tend to explicitly remove the information to achieve their goal.

In this chapter, we address this problem of conflict between classification and reconstruction, previously investigated by Ladder Networks (Rasmus et al. 2015) for example. To solve it, we propose a new type of architecture and training that we call HybridNet. The general idea is illustrated in Figure 3.1. Instead of having an auto-encoding model that will try to represent all the information in a single latent space $\mathbf{h}$, we propose to split this information into two complementarity latent spaces $\mathbf{h}_c$ and $\mathbf{h}_u$. To do so, we design a "hybrid" Auto-Encoder (AE) with a feature extraction path decomposed into two branches as represented in Figure 3.2. Thus, compared to a traditional AE, the presence of the two branches allows separating discriminative and non-discriminative information. The first branch (W_c) producing $\mathbf{h}_c$ is responsible for extracting discriminative information and to produce a class prediction, which is its main goal; it should thus extract invariant

class-specific patterns as we have seen in Chapter 2. The second branch (W_u) is then here to capture the non-discriminative information that is not useful for classification. Combining both branches allows performing exact reconstruction without compromising the invariance properties of the discriminative branch.

In Section 3.2, we present existing techniques to address SSL that are the most related to HybridNet and present their limitations. We then present our HybridNet framework in Section 3.3. We then investigate its detailed behavior and capability to improve on state-of-the-art techniques (at time of publication) in Section 3.4.

3.2 Reconstruction and Stability for Semi-Supervised Learning

We have seen in Section 2.2.4 that Semi-Supervised Learning (SSL) is an interesting approach for regularizing Deep Neural Networks (DNNs) and improving the discriminative quality of their representations. We now propose to go over recent SSL techniques that are most in relation to HybridNet.

As we described in Chapter 2, the usual framework of SSL assumes that we have a partially labeled dataset $\mathcal{D} = \mathcal{D}_{\text{sup}} \cup \mathcal{D}_{\text{unsup}}$ with labeled pairs $\mathcal{D}_{\text{sup}} = \{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})\}_{k=1..N_{\text{s}}}$ and unlabeled images $\mathcal{D}_{\text{unsup}} = \{\mathbf{x}^{(k)}\}_{k=1..N_{\text{u}}}$ and consider that the images of $\mathcal{D}_{\text{unsup}}$ and $\mathcal{D}_{\text{sup}}$ are drawn from the same distribution and all correspond to one of the known classes of $\mathcal{D}_{\text{sup}}$. An SSL training usually consists in mixing a supervised classification loss trained on $\mathcal{D}_{\text{sup}}$ with an unsupervised regularization trained on the full dataset $\mathcal{D}$. Here, we focus on two main types of techniques: stability-based methods and reconstruction-based methods.

3.2.1 Stability based methods

A first unsupervised criterion for SSL relies on increasing the stability and smoothness of the prediction function around the data points. As we have seen in Section 2.2.3, this objective is fairly common to regularize DNNs, however, the goal here is to find a way to also take advantage of additional unlabeled images.

In particular, an original idea by Sajjadi et al. (2016) proposes to force the model to produce output prediction $\hat{\mathbf{y}}$ that are stable toward many sources of variability to which the model should be invariant. The paper proposes to use heavy Data Augmentation (DA) (translation, rotation, shearing, noise, *etc.*) and use a stochastic model containing dropout. The loss is designed so that all the outputs $\hat{\mathbf{y}}^{(i,k)}$ should be the same for $k = 1..K$, each k representing a random variation of the same

input $\mathbf{x}^{(i)}$. This is measured by Mean-Squared Error (MSE) between all the pairs, for each image i:

$$\mathcal{L}_{\text{stability}} = \sum_{j=1}^{K} \sum_{k=1}^{K} ||\hat{\mathbf{y}}^{(i,k)} - \hat{\mathbf{y}}^{(i,j)}||_2^2 \,. \tag{3.1}$$

This, however, has the drawback of requiring many passes of the same image to provide an effective regularization. To solve this, Laine and Aila (2017) and Tarvainen and Valpola (2017) both propose variants where the loss becomes, for an image i:

$$\mathcal{L}_{\text{stability}} = ||\hat{\mathbf{y}}^{(i)} - \tilde{\mathbf{y}}^{(i)}||_2^2 \,, \tag{3.2}$$

which can be seen as a special case of Sajjadi et al. (2016) where $K = 2$ (called the Π model) but now using a *virtual* target $\tilde{\mathbf{y}}^{(i)}$. Laine and Aila (2017) and Tarvainen and Valpola (2017) propose different solutions to obtain this virtual target, with the idea that this target will be more "stable" and thus more useful that the multiple targets of Sajjadi et al. (2016).

Laine and Aila (2017) propose Temporal Ensembling, defining $\tilde{\mathbf{y}}^{(i)}$ as the exponential moving average of the previous outputs $\hat{\mathbf{y}}^{(i)}$. If we consider the model constant, this would mean that the virtual target is an average of the outputs for the sample i with many different random variants of $\mathbf{x}^{(i)}$. Of course, because the model is changing by being trained, the behavior of this averaging is more complex because it also "contains" outputs produced by the model during past epochs.

To overcome this problem of averaging outputs over many epochs, Tarvainen and Valpola (2017) propose Mean Teacher, where instead of averaging outputs, the weights of the classification model $f_{\mathbf{w}}$ are averaged over the previous batches to obtain the teacher model $f_{\mathbf{w}}^{\text{MT}}$. The virtual target is the output of this model: $\tilde{\mathbf{y}}^{(i)} = f_{\mathbf{w}}^{\text{MT}}(\mathbf{x}^{(i)})$. While the interpretation regarding stability becomes less clear, this relies on the property that averaged models are more stable and accurate. This model also retains the stability objective of the Π model because $f_{\mathbf{w}}$ and $f_{\mathbf{w}}^{\text{MT}}$ use independent random sources of variability.

3.2.2 Reconstruction based methods

The main limit of stability approaches is that their regularization effect only rely on the smoothing of the prediction function, encouraging the robustness and invariance of their features. However, they do not encourage the extraction of new and more general patterns using the appearance of the unlabeled images. This question is addressed by other approaches which explore the use of reconstruction in the SSL context. By adding a decoder to the classifier and learning to reconstruct,

it is possible to extract new and more robust features that more broadly model the full dataset, as we described in length in Section 2.2.4.2.

Mixing a classification and a reconstruction cost has been used for a long time as a way to perform SSL (*e.g.* Ranzato and Szummer 2008). This way, the classification decision can be learned on labeled samples $\mathcal{D}_{\text{sup}}$ while features used to make the decision are learned on all the images in $\mathcal{D}$. Existing SSL approaches can be based on AE using reconstruction (Weston et al. 2008; Turian et al. 2010) or on generative models that learn the data distribution, namely Variational Auto-Encoder (VAE) (Kingma et al. 2014) and Generative Adversarial Networks (GANs) (Springenberg 2016; Denton et al. 2017; Bodla et al. 2018).

However, as we mentioned, this strategy of mixing classification and reconstruction is questionable since they play contradictory roles. Classification arguably aims at extracting invariant class-specific features which induce an information loss detrimental for an effective reconstruction. To overcome this issue, solutions have been proposed. A decade ago, Ranzato et al. (2007a) already proposed to address a specific component of this issue which is related to the localization of the visual information. Indeed, because encoders usually include max-pooling layers, this produces a loss of the spatial information that the decoder cannot guess. To overcome this issue, they provide the position of the max-poolings to the decoder to restore this information. This idea was applied more recently by Zhao et al. (2016) in the model Stacked What-Where Auto-Encoder (SWWAE), designed for SSL and based on an AE.

Ladder Networks (Rasmus et al. 2015) are another AE-based attempt to overcome this, designing an encoder allowed to discard information thanks to noisy *skip* connections to the decoder. Reconstruction at each layer of the decoder is produced using upper-layer representation and a noisy version of the reconstruction target. However, it is not obvious that providing a noisy version of the target and training the network to remove the noise allows the encoder to effectively remove information and produce invariant features, since it must be able to correct "low-level" errors that require specific information about the image instance being reconstructed.

The limit of those two approaches is that they explicitly provide *by-passed* information to the decoder so that the encoder can discard this information, however, it is not clear if this is sufficient to allow the encoder to effectively produce intra-class invariant features. For example, while providing spatial information (in SWWAE) is useful to enable translation invariance, the encoder still needs to represent lots of information that is not relevant for classification like detailed textures, shapes, *etc*. For the Ladder Networks, as we have seen, it is not clear if providing a noisy version of the reconstruction target can allow the model to only encode intra-class invariant features. For example, the decoder must be able to correct a

"by-passed" corrupted texture information by using "normal" information that must be represented by the encoder.

Notes on decoder design

Finally, we can note that when using large modern ConvNets, the problem of designing decoders able to invert an encoder still is an open question (Wojna et al. 2017). The usual solution is to mirror the architecture of the encoder by using *transposed convolutions* (Dumoulin and Visin 2016), but this does not ensure that we will be able to reconstruct perfectly the original information. This problem is exacerbated with irreversible pooling operations such as max-pooling that must be reversed by an upsampling operation. In Zhao et al. (2016) and Y. Zhang et al. (2016), they use unpooling operations to bring back spatial information from the encoder to the decoder, reusing pooling switches locations for upsampling. Another interesting option is to explicitly create models which are reversible by design. This is the option followed by recent works such as RevNet (Gomez et al. 2017) and i-RevNet (Jacobsen et al. 2018), being inspired by the second generation of bi-orthogonal multi-resolution analysis and wavelets (Sweldens 1995) from the signal processing literature.

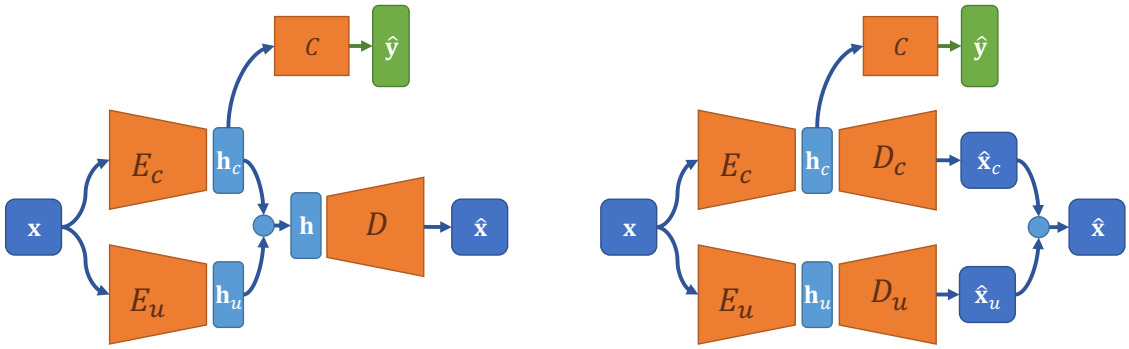
3.3 HybridNet framework

In this section, we detail our contributions with the HybridNet framework, that are twofold: first, in Section 3.3.1, we propose an architecture designed to efficiently make reconstruction and classification losses cooperate; second, in Section 3.3.2, we design a training loss adapted to it that includes reconstruction, stability in the discriminative branch and a branch complementarity technique. This framework will then be validated on CIFAR-10, SVHN and STL-10 datasets in Section 3.4.

3.3.1 Designing the HybridNet architecture

3.3.1.1 Separating visual information in two latent spaces

As we have seen, a model designed for both classification and reconstruction faces a conflict in the features it should encode. For classification, a model will tend to filter out non-discriminative information, removing possibly a large part of the visual data to produce intra-class invariance features. This is of course especially true when using regularization that adds invariance. On the other hand,



(a) **Early fusion** merging the representations and using a single decoder.

(b) **Late fusion** using two decoders and merging the reconstructions.

Figure 3.3. – **Overview of the architecture of HybridNet**, with the two main possible fusion process for merging the two branches. The blue dot ● symbolizes the fusion operation, like concatenation or addition, but more complex fusion methods could be used.

reconstruction needs to preserve all the information in a compact representation, and then decompress this representation into the reconstruction. If the same latent space is used for both tasks, a clear competition between the two losses arise.

Previous methods addressing this problem (Rasmus et al. 2015; Zhao et al. 2016; Y. Zhang et al. 2016) mostly propose to add direct skip connections from the encoder to the decode to provide pooling locations information or a noisy version of the reconstruction target. Unlike those approaches, with HybridNet, we choose to propose an explicit separation of the information in two distinct learned spaces, which should allow for a finer and learned separation of discriminative and non-discriminative features, and is thus expected to have a more favorable impact on producing discriminative features. HybridNet also differs from reversible models (Gomez et al. 2017; Jacobsen et al. 2018) by having an explicit separation between the two latent spaces, which is complicated to achieve in reversible models due to the their particular design. Another difference is that HybridNet is allowed to discard information. Reversible models, by design, cannot discard any information but can only deform the input manifold into a new one, as opposed to the usual idea of introducing invariance in the representations.

As we illustrated in Figure 3.1 (page 45), the idea of HybridNet consists in using a "hybrid" Auto-Encoder (AE) with the feature extraction path decomposed in two complementarity branches. Compared to a traditional AE, the presence of the two-branches allows separating discriminative and non-discriminative information.

Taking the example of the architecture in Figure 3.3a, the *discriminative encoder* E_c (top) produces $\mathbf{h}_c$ and is connected to a classification block C that produces class predictions $\hat{\mathbf{y}}$, which is the main goal of this branch. This encoder should

thus ideally focus on discriminative features that are expected to extract invariant class-specific patterns for a better generalization, as we have seen in Chapter 2. In this case, part of the information is lost in this branch and exact reconstruction from it should not be possible.

To complement this encoder and capture the missing information, a second *unsupervised encoder* E_u (bottom) is added and produces $\mathbf{h}_u$. It is only by combining the information in $\mathbf{h}_c$ and $\mathbf{h}_u$ that we are able to produce a complete reconstruction $\hat{\mathbf{x}}$. During training, the supervised classification cost impacts the weights of E_c while an unsupervised reconstruction cost is applied to both E_c and E_u to properly reconstruct the input image. The main assumption behind HybridNet is that this two-path architecture helps in making classification and reconstruction cooperate.

An important goal of HybridNet is to produce encoders with complementary roles. The discriminative path must extract discriminative features $\mathbf{h}_c$ that should eventually be well crafted to effectively perform a classification task, without being able to produce a full reconstruction since preserving all the information is not a behavior we want to encourage. Consequently, the role of the unsupervised path is to be complementary to the discriminative branch by retaining in $\mathbf{h}_u$ the information lost in $\mathbf{h}_c$. The general HybridNet architecture can thus be described with the following equations:

$$\mathbf{h}_c = E_c(\mathbf{x}) \qquad \hat{\mathbf{y}} = C(\mathbf{h}_c) \qquad \mathbf{h}_u = E_u(\mathbf{x}) \qquad \hat{\mathbf{x}} = D(\mathbf{h}_c, \mathbf{h}_u)\,. \tag{3.3}$$

It is important to note that the end-role of reconstruction here is just to act as a regularizer for the discriminative encoder E_c. However, as we have seen, complete reconstruction is a too strong regularization since it requires to retain too much information. Our unsupervised path via E_u is the key element that gives freedom to the discriminative branch to take advantage of additional data while still being able to filter out information and to perform the target task of classification.

Interestingly, this idea of separating the information in two subspaces and combining it for reconstruction also has conceptual connections to wavelet decomposition (Mallat and Peyré 2009): the first branch can be seen as extracting discriminative low-pass features from input images, and the second branch acting as a high-pass filter to restore the lost information.

3.3.1.2 Merging the information for reconstruction

After separating the information in two latent spaces, arise the question of how to merge back the information to produce the final reconstruction. Indeed, the decoding part of our model $D(\mathbf{h}_c, \mathbf{h}_u)$ (*cf.* Equation 3.3) must produce the reconstruction by merging the information from the two latent spaces. To do so,

we have two main possibilities: an early fusion (*cf.* Figure 3.3a), merging directly $\mathbf{h}_c$ and $\mathbf{h}_u$ and then producing the reconstruction; or a late fusion (*cf.* Figure 3.3b), where each representation is decoded independently by a specific decoder, giving partial reconstructions $\hat{\mathbf{x}}_c$ and $\hat{\mathbf{x}}_u$ that are then merged into $\hat{\mathbf{x}}$.

With early fusion, the equations describing the architecture are:

$$\mathbf{h} = \text{merge}(\mathbf{h}_c, \mathbf{h}_u)\,, \qquad \hat{\mathbf{x}} = D(\mathbf{h})\,. \tag{3.4}$$

Using an early fusion has the advantage of using a single decoder D that has all the information available and can combine it freely to produce the reconstruction. Thus, this decoder can find complex combinations and interactions between the two latent spaces to solve its task. In addition, we can introduce prior knowledge in the choice of the merge function, which are numerous. The simplest solutions are a merge by addition, concatenation or element-wise product. Each of these solutions has its own semantic: addition enforces that representations have additive values in a shared latent space, concatenation would be an extension of addition where the first layer learns linear combinations of both representations, and multiplication can be interpreted as a weighting or attention mechanism between the two. More complex merging strategies could also be used, such as using bilinear combinations of $\mathbf{h}_c$ and $\mathbf{h}_u$ (Ben-Younes et al. 2017; Paumard et al. 2018). This merging method could model complex relations between the two spaces and would give the ability to the encoders to produce strongly semantic latent representations that could interact with each other. For example, semantic information about the class of the image (*e.g.* a car) could be combined with semantic information regarding the style of the object (*e.g.* the design and color) to effectively reconstruct a specific object on the input.

With late fusion, the equations are:

$$\hat{\mathbf{x}}_c = D_c(\mathbf{h}_c)\,, \qquad \hat{\mathbf{x}}_u = D_u(\mathbf{h}_u)\,, \qquad \hat{\mathbf{x}} = \text{merge}(\hat{\mathbf{x}}_c, \hat{\mathbf{x}}_u)\,. \tag{3.5}$$

In this case, a first advantage is that each decoder can be designed symmetrically to its corresponding encoder, as is common when designing AE. In addition to simplifying the design of the architecture, it also provides additional control over the behavior of the model. For example, we can use the location of the max-pooling layers in the encoder to use in the corresponding upsampling layer of the decoder (Zhao et al. 2016). We can also introduce intermediate reconstruction costs as we will see. Finally, we can ensure that both branches are contributing to the final reconstruction, which is much more complex to do with an early fusion. Using late fusion, we also have the question of how to merge the two partial reconstructions. The easiest solution is to merge them by addition in the pixel space, similarly to what is done with wavelet decomposition. But more

complex merge techniques could be used to enforce specific semantics to partial reconstructions through a specific merge strategy, as is proposed by some methods (Shu et al. 2017; Shu et al. 2018; Luvizon et al. 2017). For example, it could be interesting to investigate the use of a sort of soft attention or masking mechanism, with one branch creating the textures and the other creating the shape. Such an handcrafted mechanism should however be carefully designed to be aligned with the goal of E_c and the production of discriminative features which is not obvious.

In this situation, the drawbacks of one solution are the absence of the advantages proposed by the other. On one side, early fusion seems to enable more complex complementarity between the two spaces. On the other side, late fusion provides much more control over the behavior of the model. And actually, an important and interesting challenge of HybridNet is to find a way to ensure that the two branches will, in fact, behave in this desired way. The two main issues that we tackle are the fact that we want the discriminative branch to focus on discriminative features, and that we want both branches to cooperate and contribute to the reconstruction. Indeed, in both cases, we can end up with two paths that work independently: a classification path being $\hat{\mathbf{y}} = C(E_c(\mathbf{x}))$; and a reconstruction path being $\hat{\mathbf{x}} = D(\mathbf{h}_c)$ with $\mathbf{h}_u$ not intervening in D for early fusion or $\hat{\mathbf{x}} = \hat{\mathbf{x}}_u = D_u(E_u(\mathbf{x}))$ and $\hat{\mathbf{x}}_c = 0$ for late fusion.

Solving those two issues is achieved both through the choice of the architecture of HybridNet and the development of a well-designed training loss to control the behavior of HybridNet. While we tried both fusion strategies, late fusion is more adapted to solve this problem. For this reason, the rest of the architecture will be explained for this kind of fusion, using a merge by addition in the pixel space. On the other hand, early fusion will be further explored in Chapter 4 in the context of disentangling.

3.3.1.3 Designing a dual decoder HybridNet

Let us now describe more thoroughly the architecture of a HybridNet using late fusion, *i.e.* with two complete separate encoding-decoding paths, merging partial reconstructions by addition. An example of HybridNet architecture is presented in Figure 3.4, described with the following equations:

$$\mathbf{h}_c = E_c(\mathbf{x}) \qquad \hat{\mathbf{x}}_c = D_c(\mathbf{h}_c) \qquad \hat{\mathbf{y}} = C(\mathbf{h}_c) \tag{3.6}$$

$$\mathbf{h}_u = E_u(\mathbf{x}) \qquad \hat{\mathbf{x}}_u = D_u(\mathbf{h}_u) \qquad \hat{\mathbf{x}} = \hat{\mathbf{x}}_c + \hat{\mathbf{x}}_u \tag{3.7}$$

To design the HybridNet architecture, we start with a convolutional architecture adapted to the targeted dataset, for example a state-of-the-art ResNet architecture for CIFAR-10. This architecture is split into two modules: the discriminative encoder E_c and the classifier C. On top of this model, we add the discriminative

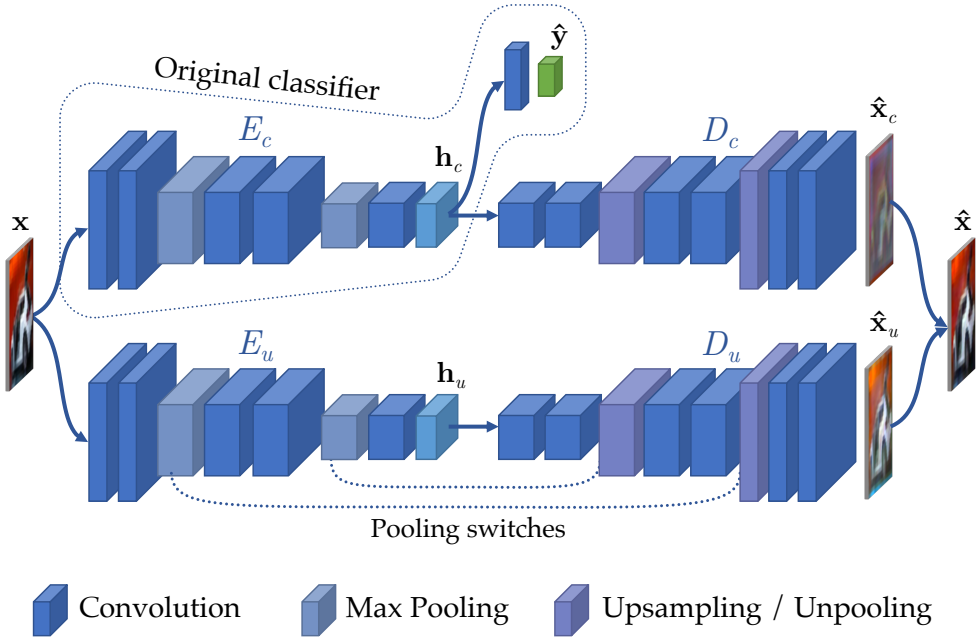


Figure 3.4. – **Example of a detailed HybridNet architecture using late fusion**. We represent the output feature maps of convolutions, max-pooling and unpooling/upsampling layers.

decoder D_c. The location of the splitting point in the original network is free, but C will not be directly affected by the reconstruction loss. In our experiments, we choose $\mathbf{h}_c$ (E_c's output) to be the last intermediate representation before the final pooling that aggregates all the spatial information, leaving in C a global average pooling followed by one or more fully-connected layers.

The decoder D_c is designed to be a "mirror" of the encoder's architecture, as commonly done in the literature, *e.g.* (Zhao et al. 2016; Rasmus et al. 2015; Zeiler and Fergus 2014). This means that all the convolutional layers (with unit stride) are replaced by similar convolutions with swapped input/output planes number and in the reverse order. Pooling layers or strides convolutions, that reduce the spatial size, can be reversed using upsampling (or *transposed* convolutions (*cf.* Dumoulin and Visin 2016) that can be seen as a special upsampling followed by a regular convolution) or unpooling as we will see.

After constructing the discriminative branch, we add an unsupervised complementary branch. To ensure that both branches are "balanced" and behave in a similar way, the internal architecture of E_u and D_u is mostly the same as for E_c and D_c. The only difference remains in the mirroring of pooling layers, using either upsampling or unpooling. An upsampling will increase the spatial size of a feature map without any additional information while an unpooling, used by Zhao et al. (2016) and Y. Zhang et al. (2016), will use spatial information (*pooling switches*) from the corresponding max-pooling layer to do the upsampling.

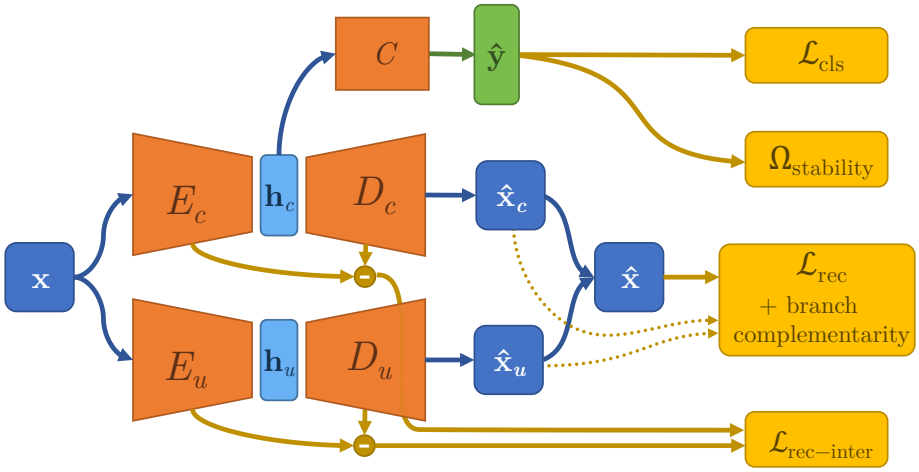


Figure 3.5. – **General description of the HybridNet framework**. E_c and C correspond to a classifier, E_c and D_c form an autoencoder that we call *discriminative path*, and E_u and D_u form a second autoencoder called *unsupervised path*. The various loss functions used to train HybridNet are also represented in yellow.

In our architecture, we propose to use upsampling in the discriminative branch because we want to encourage spatial invariance, and use unpooling in the unsupervised branch to compensate this information loss and favor the learning of spatial-dependent low-level information.

As mentioned previously, one key problem to tackle is to ensure that this model will behave as expected, *i.e.* by learning discriminative features in the discriminative encoder and non-discriminative features in the unsupervised one. This is encouraged by different ways in the design of the architecture. First, the fact that only $\mathbf{h}_c$ is used for classification means that E_c will be pushed by the classification loss to produce discriminative features. Thus, the unsupervised branch will naturally focus on information lost by E_c. Using upsampling in D_c and unpooling in D_u also encourages the unsupervised branch to focus on low-level information. In addition to this, the design of an adapted loss and training protocol is a major contribution to the efficient training of HybridNet.

3.3.2 Training HybridNet

The HybridNet architecture has two information paths with only one producing a class prediction and both producing partial reconstructions that should be combined. In this section, we address the question of training this architecture efficiently. The complete loss is composed of various terms as illustrated in Figure 3.5. It comprises terms for classification with $\mathcal{L}_{\text{cls}}$; final reconstruction

with $\mathcal{L}_{\text{rec}}$; intermediate reconstructions with $\mathcal{L}_{\text{rec}-\text{inter}_{b,l}}$ (for layer l and branch b); and stability with $\Omega_{\text{stability}}$. It is also accompanied by a branch complementarity training method. Each term is weighted by a corresponding parameter λ:

$$\mathcal{L} = \lambda_{\text{c}}\mathcal{L}_{\text{cls}} + \lambda_{\text{r}}\mathcal{L}_{\text{rec}} + \sum_{b\in\{c,u\},l} \lambda_{\text{r}b,l}\mathcal{L}_{\text{rec}-\text{inter}_{b,l}} + \lambda_{\text{s}}\Omega_{\text{stability}} \,. \tag{3.8}$$

HybridNet can be trained on a partially labeled dataset, *i.e.* that is composed of labeled pairs $\mathcal{D}_{\text{sup}} = \{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})\}_{k=1..N_{\text{s}}}$ and unlabeled images $\mathcal{D}_{\text{unsup}} = \{\mathbf{x}^{(k)}\}_{k=1..N_{\text{u}}}$. Each batch is composed of n samples, divided into n_{s} image-label pairs from $\mathcal{D}_{\text{sup}}$ and n_{u} unlabeled images from $\mathcal{D}_{\text{unsup}}$.

3.3.2.1 Classification

The classification term is a regular cross-entropy term, that is applied only on the n_s labeled samples of the batch and averaged over them:

$$\ell_{\text{cls}}(\hat{\mathbf{y}}, \mathbf{y}) = \ell_{\text{CE}}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i \mathbf{y}_i \log \hat{\mathbf{y}}_i \,, \qquad \mathcal{L}_{\text{cls}} = \frac{1}{n_s}\sum_k \ell_{\text{cls}}(\hat{\mathbf{y}}^{(k)}, \mathbf{y}^{(k)}) \,. \tag{3.9}$$

3.3.2.2 Reconstruction losses

We saw that in HybridNet, we mostly choose to keep discriminative and unsupervised paths separate so that they produce two complementary reconstructions $(\hat{\mathbf{x}}_u, \hat{\mathbf{x}}_c)$ that we combine with an addition into $\hat{\mathbf{x}} = \hat{\mathbf{x}}_u + \hat{\mathbf{x}}_c$. Keeping the two paths independent until the reconstruction in pixel space, as well as the merge-by-addition strategy, allows us to apply different treatments to them and influence their behavior efficiently. The reconstruction loss that we use is a simple MSE between the input and the sum of the partial reconstructions:

$$\ell_{\text{rec}} = ||\hat{\mathbf{x}} - \mathbf{x}||_2^2 = ||\hat{\mathbf{x}}_u + \hat{\mathbf{x}}_c - \mathbf{x}||_2^2 \,, \qquad \mathcal{L}_{\text{rec}} = \frac{1}{n}\sum_k \ell_{\text{rec}}(\hat{\mathbf{x}}^{(k)}, \mathbf{x}^{(k)}) \,. \tag{3.10}$$

In addition to the final reconstruction loss, we also add reconstruction costs between intermediate representations in the encoders and the decoders which is possible since encoders and decoders have mirrored structure. We apply these costs to the representations $\mathbf{h}_{b,l}$ (for branch b and layer l) produced just after pooling layers in the encoders and reconstructions $\hat{\mathbf{h}}_{b,l}$ produced just before the corresponding upsampling or unpooling layers in the decoders. This is common in the literature (Zhao et al. 2016; Y. Zhang et al. 2016; Rasmus et al. 2015) but is particularly important in our case: in addition to guiding the model to produce the right final reconstruction, it pushes the discriminative branch to produce a reconstruction and avoid the undesired situation where only the unsupervised branch

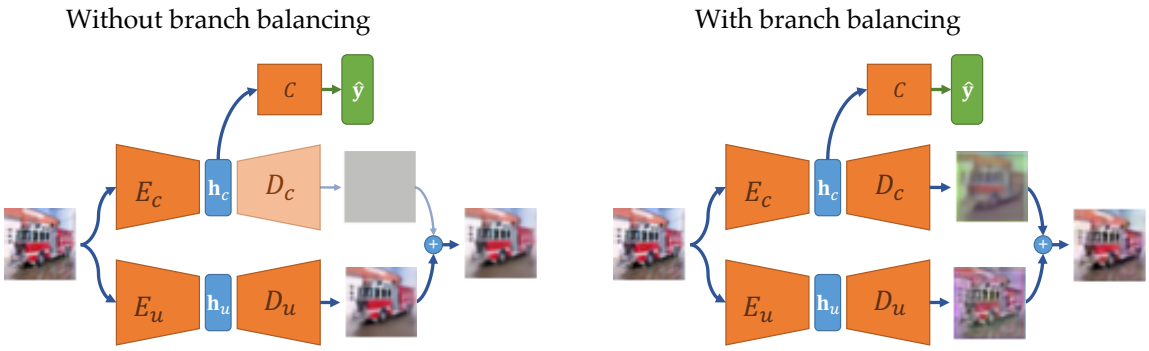


Figure 3.6. – **Illustration of the effect of branch balancing on reconstructions**. Without branch balancing, the model can learn two separate paths, classifying with E_c and reconstructing with $D_u \circ E_u$ without using $\mathbf{h}_c$. The branch balancing forces both branches to produce a reconstruction and ensures that E_c is regularized by the reconstruction loss.

would contribute to the final reconstruction. This is applied in both branches ($b \in \{c, u\}$):

$$\mathcal{L}_{\text{rec-inter}_{b,l}} = \frac{1}{n} \sum_k ||\hat{\mathbf{h}}_{b,l}^{(k)} - \mathbf{h}_{b,l}^{(k)}||_2^2 \,. \tag{3.11}$$

3.3.2.3 Branch cooperation

As described previously, we want to ensure that both branches contribute to the final reconstruction, otherwise, this would mean that the reconstruction is not helping to regularize E_c, which is our end-goal. Having both branches produce a partial reconstruction and using intermediate reconstructions already help with this goal. In addition, to balance their training even more, we propose a training technique such that the reconstruction loss is only backpropagated to the branch that contributes less to the final reconstruction of each sample. This is done by comparing $||\hat{\mathbf{x}}_c - \mathbf{x}||_2^2$ and $||\hat{\mathbf{x}}_u - \mathbf{x}||_2^2$ and only applying the final reconstruction loss to the branch with the higher error. The expected effect of this branch balancing strategy is shown in Figure 3.6.

This can be implemented either in the gradient descent or simply by preventing gradient propagation in one branch or the other using features like `tf.stop_gradient` in Tensorflow or `.detach()` in PyTorch:

$$\ell_{\text{rec-balanced}} = \begin{cases} ||\hat{\mathbf{x}}_u + \text{stopgrad}(\hat{\mathbf{x}}_c) - \mathbf{x}||_2^2 & \text{if } ||\hat{\mathbf{x}}_u - \mathbf{x}||_2^2 \geq ||\hat{\mathbf{x}}_c - \mathbf{x}||_2^2 \\ ||\text{stopgrad}(\hat{\mathbf{x}}_u) + \hat{\mathbf{x}}_c - \mathbf{x}||_2^2 & \text{otherwise} \end{cases} . \tag{3.12}$$

3.3.2.4 Encouraging invariance in the discriminative branch

We have seen that an important issue that needs to be addressed when training this model is to ensure that the discriminative branch will filter out information and learn invariant features. For now, the only signal that pushes the model to do so is the classification loss. However, in a semi-supervised context, when only a small portion of our dataset is labeled, this signal can be fairly weak and might not be sufficient to make the discriminative encoder focus on invariant features.

A first option to encourage invariance would be to use SHADE. As presented in Chapter 2, SHADE is designed to maximize the intra-class invariance of the features. As such, it is a good candidate for our purpose and could be applied to the discriminative encoder E_c.

The second option is to use a *stability regularizer*. Such a regularizer is currently at the core of the models that give state-of-the-art results in a semi-supervised setting on the most common datasets (Sajjadi et al. 2016; Laine and Aila 2017; Tarvainen and Valpola 2017). The principle is to encourage the classifier's output prediction $\hat{\mathbf{y}}^{(k)}$ for sample k to be invariant to different sources of randomness applied on the input (translation, horizontal flip, random noise, *etc.*) and in the network (*e.g.* dropout). This is done by minimizing the MSE between the output $\hat{\mathbf{y}}^{(k)}$ and a "stability" target $\tilde{\mathbf{y}}^{(k)}$. Multiple methods have been proposed to compute such a target (Sajjadi et al. 2016; Laine and Aila 2017; Tarvainen and Valpola 2017), for example by using a second pass of the sample in the network with a different draw of random factors that will therefore produce a different output. We have:

$$\Omega_{\text{stability}} = \frac{1}{n} \sum_{k} ||\hat{\mathbf{y}}^{(k)} - \tilde{\mathbf{y}}^{(k)}||_2^2 \,. \tag{3.13}$$

By applying this loss on $\hat{\mathbf{y}}$, we encourage E_c to find invariant patterns in the data, patterns that have more chances of being discriminative and useful for classification. Furthermore, this loss has the advantage of being applicable to both labeled and unlabeled images.

3.4 Experiments

In this section, we study and validate the behavior of our novel framework. After some preliminary experiments, we perform detailed ablation studies to validate the architecture and loss terms of the model. We also propose visualizations of the behavior of the model in various configurations, before demonstrating the capability of HybridNet to obtain state-of-the-art results.

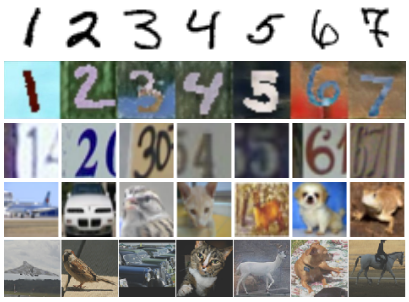
Dataset	Image size	# Train	# Test	# Extra	Samples
MNIST	32×32	50,000	10,000		
MNIST-M	32×32	50,000	10,000		
SVHN	32×32	73,257	26,032	531,131	
CIFAR-10	32×32	50,000	10,000		
STL-10	96×96	1,000	8,000	100,000	

Table 3.1. – **Overview of the datasets used**. We report the size of the images, the number of samples used for training (with only a subset of them for which we will keep the labels), the number of test samples and the number of extra images that are used only as labeled samples.

3.4.1 Datasets and data processing

In our experiments, we use different datasets of image classification among 10 classes. Those datasets are as follows, with additional numerical details and samples presented in Table 3.1:

- **MNIST** (LeCun et al. 1998), is a simple and historical dataset that contains black and white hand-written digits.
- **MNIST-M** (Ganin et al. 2016), is an interesting variant of MNIST constructed artificially, initially to test domain adaptation models. In this dataset, textures and colors taken from natural images are added to MNIST digits to introduce variability. This additional information is non-discriminative and should be represented by E_u.
- **SVHN** (Street View House Numbers, by Netzer et al. 2011), contains cropped photos of house plate numbers taken by Google Street View cars. It is thus a digits dataset with a much larger variability (camera angle, colors, textures, noise, *etc.*) than MNIST, and is thus more difficult.
- **CIFAR-10** (Krizhevsky and Hinton 2009) is a dataset of natural images that cover 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). It is more complicated than the previous ones due to the high variability of the natural images. It can be seen as a small ImageNet dataset since CIFAR-10 pictures are taken from ImageNet and have been resized to 32×32 pixels.
- **STL-10** (Coates et al. 2011) is the most challenging dataset. It uses the same classes as CIFAR-10 but with different images of higher resolution. Designed for SSL, only a small number of labeled images are provided, with a large

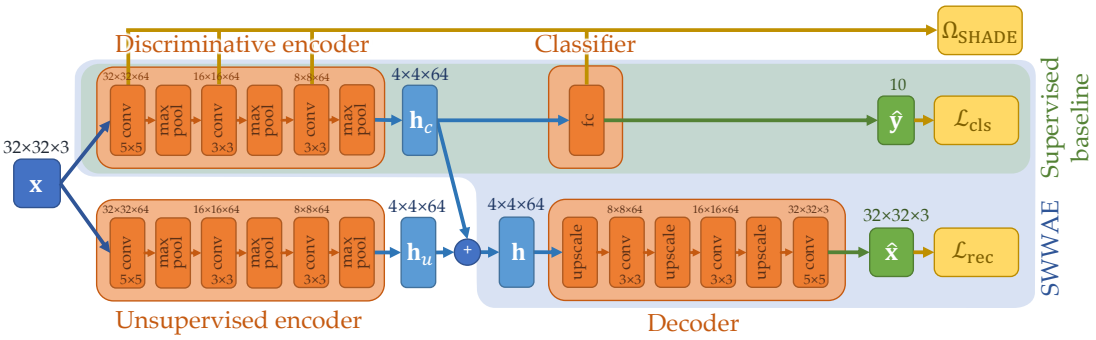


Figure 3.7. – **Detailed architecture of an HybridNet trained with SHADE**. We indicate the supervised baseline (delimited in green), SWWAE (delimited in blue) and HybridNet (whole figure) used for MNIST-M.

number of additional images that are unlabeled and taken from subclasses of the real classes (*e.g.* animals and vehicles that are not in the 10 classes) which increases the difficulty.

For our semi-supervised experiments, we will keep N_s labeled training samples (with $N_s/10$ samples per class) while the rest of the data is kept unlabeled, as is commonly done. The value for N_s varies over the experiments and will always be indicated.

3.4.2 Preliminary results using SHADE

We first conduct preliminary experiments of HybridNet in conjunction with SHADE in order to evaluate the effectiveness of our approach compared to existing SSL baselines, and especially validate the idea of using a two-branch architecture. For this, we especially compare our HybridNet framework to SWWAE by Zhao et al. (2016), which proposes to address the same motivations as HybridNet. SWWAE is an AE-based architecture for SSL that proposes to address the problem of decoding from features made invariant because of max-pooling layers. To do so, SWWAE uses unpooling layers in the decoder to reintroduce the pooling location and provide this missing information.

For this experiment, we start with the architecture and protocol of Zhao et al. (2016), illustrated in Figure 3.7 (delimited in blue). We develop an HybridNet version of the architecture used by SWWAE by adding a second encoder and choose to use an early fusion strategy by merging the information in the latent space ($\mathbf{h} = \mathbf{h}_c + \mathbf{h}_u$) by addition. We also do not use the unpooling strategy of SWWAE and replace those with simple upsampling layers. This architecture is detailed in Figure 3.7 (full figure).

Dataset	MNIST		MNIST-M		SVHN
Nb labeled samples N_s	100	1000	100	1000	1000
Supervised baseline	83.26	95.51	47.14	83.09	75.03
SWWAE*	86.38	95.72	45.83	82.89	75.27
HybridNet no regul.	84.13	96.01	48.07	84.86	75.63
HybridNet + weight decay	87.71	95.98	48.62	83.69	76.13
HybridNet + SHADE	**89.15**	**97.18**	**52.58**	**88.23**	**79.12**

* The results reported correspond to our reimplementation of this architecture based on the information provided in Zhao et al. 2016.

Table 3.2. – **Results of a first version of HybridNet with SHADE compared to SWWAE**. We compare supervised and semi-supervised baselines to HybridNet in different stability regularization setups. We report the accuracy (%) measured on the test set of MNIST, MNIST-M and SVHN for different sizes of labeled dataset.

We train the supervised baseline (without a decoder), SWWAE and three variants of HybridNet with no stability regularization, with weight decay and with SHADE for stability. We apply them on MNIST, MNIST-M and SVHN (see Table 3.1 for image samples) and report the results in Table 3.2. Among the three, we can note that MNIST-M is particularly interesting to demonstrate the capabilities of HybridNet. As we can see in the image samples, it contains a lot of variability in the visual information (colors, textures) and very little semantic information useful for classification (the overall shape of the digit). HybridNet, thanks to its two latent spaces, should allow efficient separation of the two types of information.

First, SWWAE, our SSL baseline, improves over the supervised-only baseline except on MNIST-M which could be explained by the fact that the reconstruction loss might produce too many features that need to encode the wide variety of textures in the dataset and does not learn features useful for classification. HybridNet without regularization usually does improve slightly over SWWAE, which is normal since stability regularization is a key element to make this model work. Indeed, adding weight decay –which can be seen as adding invariance *cf.* Section 2.3.3 *"Non-conditional Entropy"*–, provides an interesting gain to HybridNet, increasing the accuracy on MNIST ($N_s = 100$) from 84.1% to 87.7%. On MNIST-M however, the gain is negligible, probably due to the non-class-conditional nature of weight decay. Indeed, using SHADE for stability provides an important gain, especially on MNIST-M where we gain ∼4 pts over the HybridNet baseline ∼6 pts over the supervised and SSL baselines.

With Figure 3.8, we propose to visualize the effect of using SHADE in HybridNet on MNIST-M. We show, for four different images, the final reconstruction $\hat{\mathbf{x}} = D(\mathbf{h}) = D(\mathbf{h}_c + \mathbf{h}_u)$ and partial reconstructions $\hat{\mathbf{x}}_c = D(\mathbf{h}_c)$ and $\hat{\mathbf{x}}_u = D(\mathbf{h}_u)$. This is

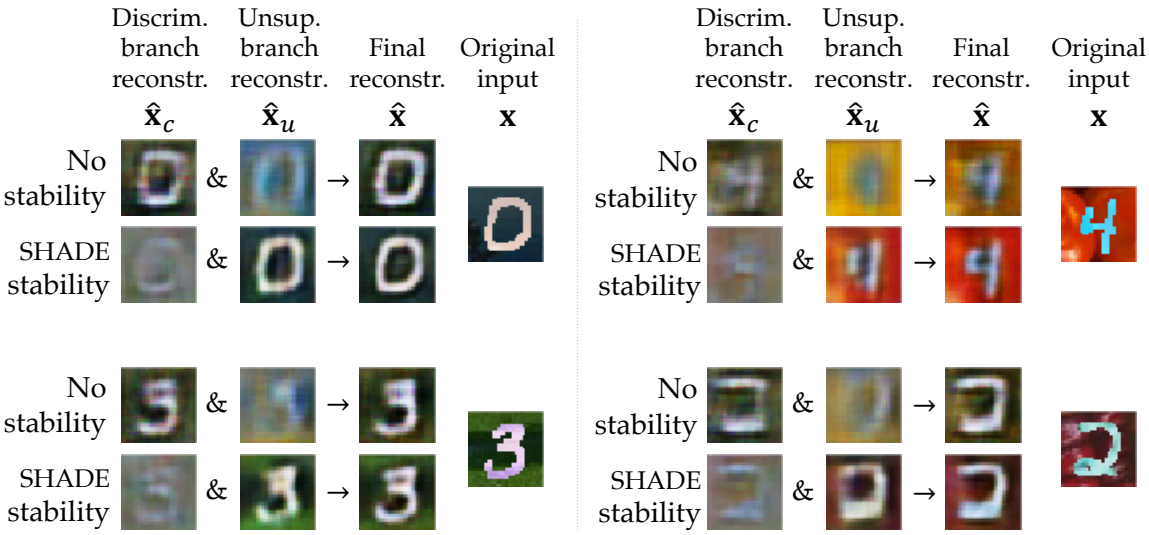


Figure 3.8. – **Reconstructions of samples from MNIST-M by HybridNet with and without SHADE.** We show the final reconstruction $\hat{\mathbf{x}} = D(\mathbf{h}) = D(\mathbf{h}_c + \mathbf{h}_u)$ and the "partial" reconstructions $\hat{\mathbf{x}}_c = D(\mathbf{h}_c)$ and $\hat{\mathbf{x}}_u = D(\mathbf{h}_u)$ produced when feeding the decoder with the features from a single branch.

shown for two HybridNet models, trained with and without SHADE as a stability regularizer. One can see that the full reconstructions $\hat{\mathbf{x}}$ with *SHADE* are quite better than the ones with *no stability* regularization, particularly in terms of color integrity and texture quality. This can be analyzed by looking at the contribution of each branch through $\hat{\mathbf{x}}_c$ and $\hat{\mathbf{x}}_u$. For the HybridNet with SHADE, we observe that the details (color, texture, *etc.*) are encoded in the unsupervised branch $\mathbf{h}_u$ as desired, while the discriminative branch $\mathbf{h}_c$ barely contains the shape of the digit. This is not the case without regularization where the information does not seem to be organized, which is particularly visible on the bottom digits 2 and 3 where the green and red backgrounds are almost lost without stability.

In this experiment, we validated the relevance of using a two-branch architecture to improve on existing techniques based on AE, in particular SWWAE. We also show how adding a regularization encouraging invariance, namely SHADE, can work in concert with the two-branch architecture to effectively separate the information.

Discussion. While it made sense to reuse our work on invariance with SHADE in HybridNet, we now propose to discuss its viability as a regularizer in this context. During this first experiment, we observed that, while effective, SHADE was too strong of a regularizer toward invariant representations, making it difficult for representations from E_c to cooperate with reconstruction. Furthermore, SHADE is

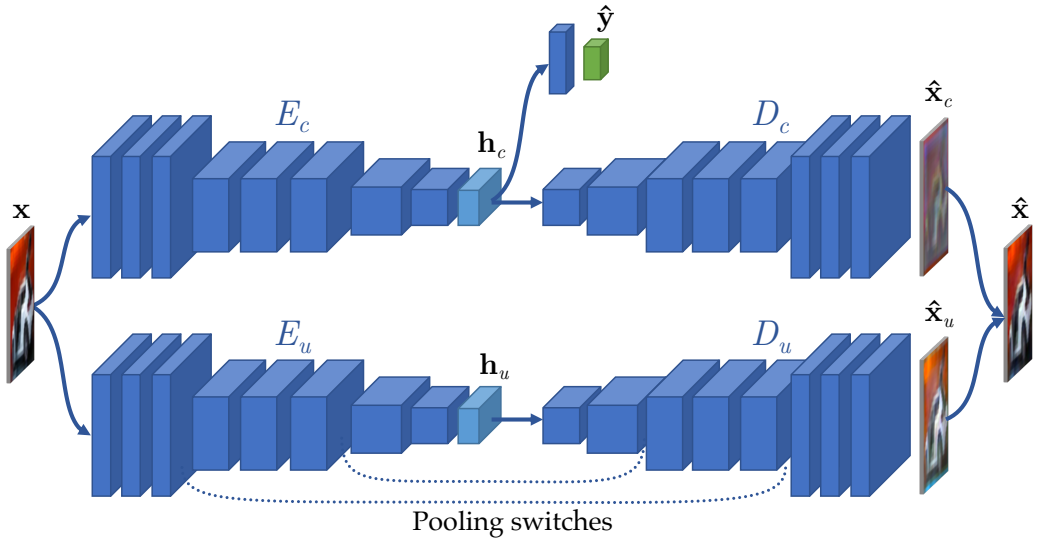


Figure 3.9. – **Example of a HybridNet architecture**. The original classifier (ConvLarge) constitutes E_c and has been mirrored to create D_c and duplicated for E_u and D_u, with the addition of unpooling in the discriminative branch.

designed with the idea that the model trained with a supervision signal is able to produce relevant class-related representations that we model with a binary latent code $\mathbf{z}$, *cf.* Section 2.3.3, $\mathbf{z}$ being a sort of surrogate of $\mathbf{y}$ for class information, to compute a replacement of $\mathcal{H}(\mathbf{h}_c \mid \mathbf{y})$. In the context of SSL where we only have very few labels to learn those latent codes $\mathbf{z}$, it is not certain how SHADE will behave. For this reason and in order to more easily compare HybridNet with state-of-the-art methods, we choose to replace SHADE with stability methods like Mean Teacher (Tarvainen and Valpola 2017).

In addition, these experiments confirmed that controlling the behavior of an early fusion HybridNet was complicated and very sensitive to hyperparameters values. This is why the rest of the experiments will use a late fusion strategy, replacing the single decoder by two decoders, one for each branch, to provide more control over the behavior of the model as explained in Section 3.3.2, *"Branch cooperation"*.

3.4.3 HybridNet framework validation

We now propose a thorough analysis of the behavior of our model at two different levels: first by comparing it to baselines that we obtain when disabling parts of the architecture, and second by analyzing the contribution of the different terms of the training loss of HybridNet both quantitatively and through visualizations.

Encoders E_c and E_u		
Input	$\tilde{\mathbf{x}}$	$32 \times 32 \times 3$
Convolution	128 filters, 3×3, padding 1	$32 \times 32 \times 128$
Convolution	128 filters, 3×3, padding 1	$32 \times 32 \times 128$
Convolution	128 filters, 3×3, padding 1	$32 \times 32 \times 128$
Pooling	Maxpool 2×2	$16 \times 16 \times 128$
Dropout	$p = 0.5$	$16 \times 16 \times 128$
Convolution	256 filters, 3×3, padding 1	$16 \times 16 \times 256$
Convolution	256 filters, 3×3, padding 1	$16 \times 16 \times 256$
Convolution	256 filters, 3×3, padding 1	$16 \times 16 \times 256$
Pooling	Maxpool 2×2	$8 \times 8 \times 256$
Dropout	$p = 0.5$	$8 \times 8 \times 256$
Convolution	512 filters, 3×3, padding 0	$6 \times 6 \times 512$
Convolution	256 filters, 1×1, padding 1	$6 \times 6 \times 256$
Convolution	128 filters, 1×1, padding 1	$6 \times 6 \times 128$
Output	$\mathbf{h}_c$ or $\mathbf{h}_u$	$6 \times 6 \times 128$
Classifier C		
Input	$\mathbf{h}_c$	$6 \times 6 \times 128$
Pooling	Global average pool	$1 \times 1 \times 128$
Fully connected	with Softmax	10
Output	$\hat{\mathbf{y}}$	10
Decoders D_c and D_u		
Input	$\mathbf{h}_c$ or $\mathbf{h}_u$	$6 \times 6 \times 128$
TConvolution	256 filters, 1×1, padding 1	$6 \times 6 \times 256$
TConvolution	512 filters, 1×1, padding 1	$6 \times 6 \times 512$
TConvolution	256 filters, 3×3, padding 0	$8 \times 8 \times 256$
Upsampling	2×2 (unpooling in D_u)	$16 \times 16 \times 256$
TConvolution	256 filters, 3×3, padding 1	$16 \times 16 \times 256$
TConvolution	256 filters, 3×3, padding 1	$16 \times 16 \times 256$
TConvolution	128 filters, 3×3, padding 1	$16 \times 16 \times 128$
Upsampling	2×2 (unpooling in D_u)	$32 \times 32 \times 128$
TConvolution	128 filters, 3×3, padding 1	$32 \times 32 \times 128$
TConvolution	128 filters, 3×3, padding 1	$32 \times 32 \times 128$
TConvolution	3 filters, 3×3, padding 1	$32 \times 32 \times 3$
Output	$\hat{\mathbf{x}}_c$ or $\hat{\mathbf{x}}_u$	$32 \times 32 \times 3$

Table 3.3. – **Architecture of HybridNet version of ConvLarge for CIFAR-10**. This architecture is also presented visually in Figure 3.9. The original ConvLarge model corresponds to $C \circ E_c$. TConvolution stands for "transposed convolution" (Dumoulin and Visin 2016). Each Convolution or TConvolution is followed by a Batch Normalization layer and a LeakyRELU of parameter $\alpha = 0.1$.

This study is performed based on the ConvLarge architecture introduced by Rasmus et al. (2015) on CIFAR-10. This is the most common setup used in recent SSL experiments (Sajjadi et al. 2016; Laine and Aila 2017; Tarvainen and Valpola 2017). This is a fairly simple ConvNet architecture, resembling VGG (Simonyan and Zisserman 2015), with blocks of convolutions followed by max-poolings.

We detail this architecture in Table 3.3 and represent it schematically in Figure 3.9, in its HybridNet version. The design of the HybridNet version of ConvLarge follows Section 3.3 and uses Temporal Ensembling (Laine and Aila 2017) to produce stability targets $\tilde{y}$. We also use an adapted version of ConvLarge for STL-10 with added blocks of convolutions and pooling to obtain additional visualizations and quantitative results.

Models are trained with Adam with a learning rate of 0.003 for 600 epochs with batches of 20 labeled images and 80 unlabeled ones. The various loss-weighting terms λ of the general loss (Equation 3.8) were set so that the different loss terms have values of the same order of magnitude. Thus, all λ were set to either 0 or 1 if activated or not, except λ_s set to 0 or 100. The details of the architecture and hyperparameters values are provided in Appendix B.

3.4.3.1 Ablation study of the architecture

We start this analysis by validating our architecture with an ablation study on CIFAR-10 with different number of labeled samples. By disabling parts of the model and training terms, we compare HybridNet to different baselines and validate the importance of combining both contributions of the paper: the architecture and the training method.

Results are presented in Table 3.4. The classification and auto-encoder results are obtained with the same code and hyperparameters by simply disabling different losses and parts of the model: the classifier only uses E_c and C; and the auto-encoder (similar to Zhao et al. 2016) only E_c, D_c and C. For both, we can add the stability loss. The HybridNet architecture only uses the classification and reconstructions loss terms while the second result uses the full training loss.

First, we can see that the HybridNet architecture alone already yields an improvement over the baseline and the auto-encoder, except at 1000 labels. This could be explained by the fact that with very few labels, the model fails to correctly separate the information between the two branches because of the faint classification signal, and the additional loss terms that control the training of HybridNet are even more necessary. Overall, the architecture alone does not provide an important gain since it is not guided to efficiently take advantage of the two branches, indeed, we see that the addition of the complete HybridNet loss allows the model to provide much stronger results, with an improvement of 6-7 pts over

Model	Labeled samples N_s		
	1000	2000	4000
Classification	63.4	71.5	79.0
Classification and stability	65.6	74.6	81.3
Auto-encoder	65.0	73.6	79.8
Auto-encoder and stability	71.8	80.4	84.9
HybridNet architecture	63.2	74.0	80.3
HybridNet architecture and full training loss	**74.1**	**81.6**	**86.6**

Table 3.4. – **Ablation study** performed on CIFAR-10 with ConvLarge architecture.

the architecture alone, around 5-6 pts better than the stability or auto-encoding baseline, and 7-10 pts more than the supervised baseline. The most challenging baseline is the stabilized auto-encoder that manages to take advantage of the stability loss but from which we still improve by 1.2-2.8 pts.

This ablation study demonstrates the capability of the HybridNet framework to surpass the different architectural baselines, and shows the importance of the complementarity between the two-branch architecture and the complete training loss.

3.4.3.2 Importance of the various loss terms

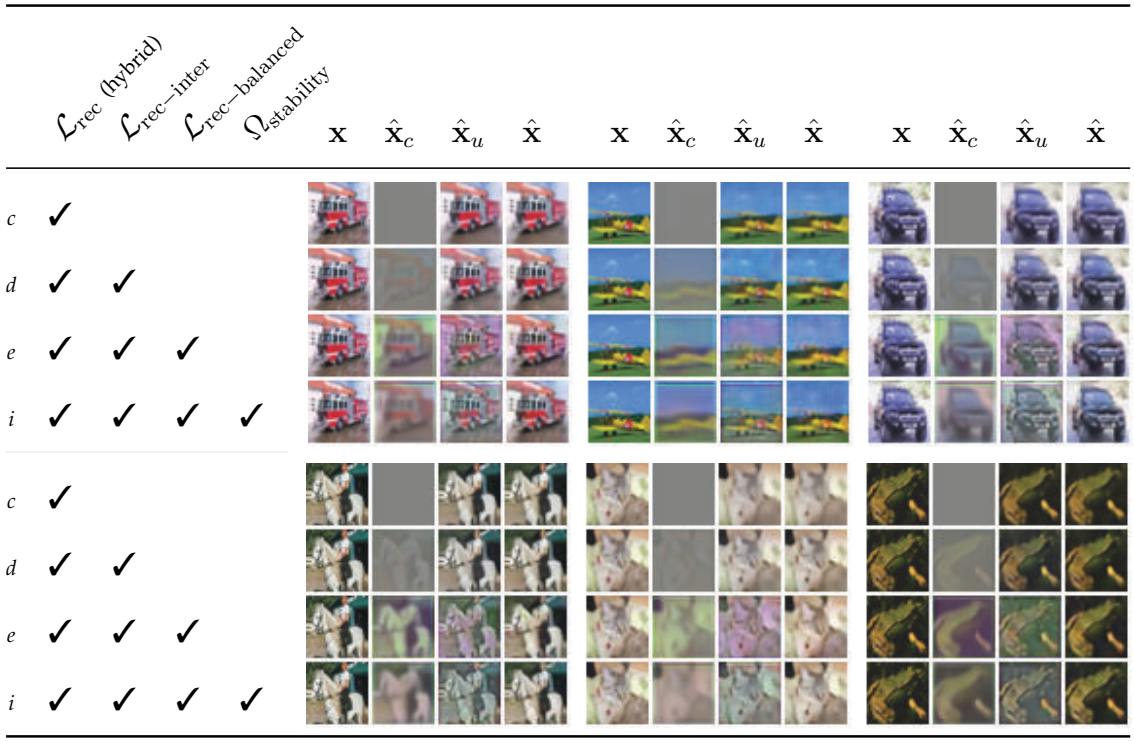
We now propose a more fine-grain study to look at the importance of each loss term of the HybridNet training described in Section 3.3.2, both through classification results and visualizations.

First, in Table 3.5a we show the classification accuracy on CIFAR-10 with 2000 labels and STL-10 with 1000 labels for numerous combinations of loss terms. These results demonstrate that each loss term has its importance and that all of them cooperate in order to reach the final best result of the full HybridNet model. In particular, the stability loss is an important element of the training but is not sufficient as shown by lines *b* and *f-h*, while the other terms bring an equivalent gain as shown by lines *c-e*. Both those $\sim$5 pts gains can be combined to work in concert and reach the final score line *i* of a $\sim$10 pts gain.

Second, to interpret how the branches behave we propose to visualize the different reconstructions $\hat{\mathbf{x}}_c$, $\hat{\mathbf{x}}_u$ and $\hat{\mathbf{x}}$ for different combinations of loss terms in Table 3.5b. With only the final reconstruction term (lines *c*), the discriminative branch does not contribute to the reconstruction and is thus barely regularized by the reconstruction loss, showing little gain over the classification baseline. The addition of the intermediate reconstruction terms helps the discriminative

		$\mathcal{L}_{\text{classif}}$	$\Omega_{\text{stability}}$	$\mathcal{L}_{\text{rec (hybrid)}}$	$\mathcal{L}_{\text{rec-inter}}$	$\mathcal{L}_{\text{rec-balanced}}$	CIFAR-10	STL-10
cls.	*a*	✓					71.5	65.6
	b	✓	✓				74.6	69.8
cls. + rec.	*c*	✓		✓			72.4	67.8
	d	✓		✓	✓		74.0	–
	e	✓		✓	✓	✓	75.2	–
cls. + rec. + stab.	*f*	✓	✓	✓			77.7	71.5
	g	✓	✓	✓		✓	77.4	–
	h	✓	✓	✓	✓		80.8	72.2
	i	✓	✓	✓	✓	✓	**81.6**	**74.1**

(a) **Quantitative results** obtained with ConvLarge on CIFAR-10 with 2000 labeled samples and ConvLarge-like on STL-10 with 1000 labeled samples.



(b) **Visual results on CIFAR-10 for four model variants.** We show the input x, full ($\hat{x}$) and partial ($\hat{x}_c$ and $\hat{x}_u$) reconstructions. With only reconstruction, the visual information by-passes the discriminative branch and is completely captured by $\hat{x}_u$. The different terms improve the organization of the information between both branches.

Table 3.5. – **Detailed ablation studies** when activating different terms and techniques of the HybridNet learning.

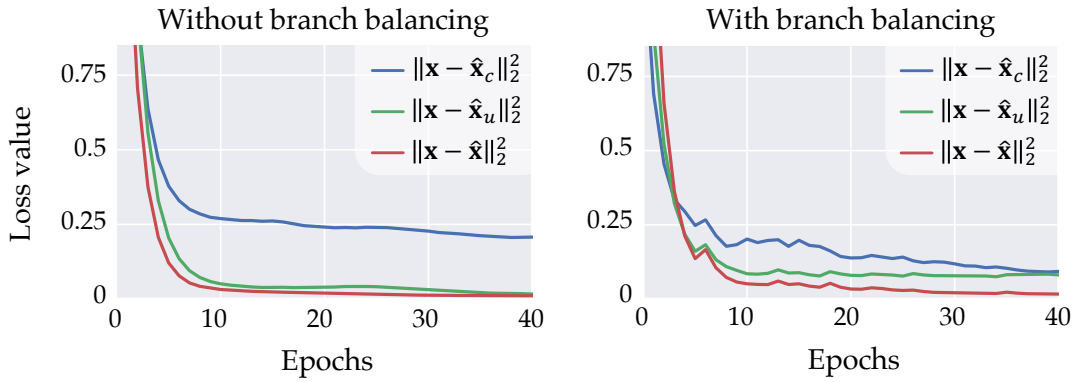


Figure 3.10. – **Evolution of reconstruction losses with and without branch balancing**. Without branch balancing, $\hat{\mathbf{x}}_u$ reconstructs $\mathbf{x}$ alone and E_c receives little regularization signal. Using branch balancing makes both branches contribute to the reconstruction in comparable magnitude.

branch to produce a weak reconstruction (lines *d*) and is complemented by the branch balancing technique (lines *e*) to produce balanced reconstructions in both branches. The stability loss (lines *i*) adds little visual impact on $\hat{\mathbf{x}}_c$, it has probably more impact on the quality of the latent representation $\mathbf{h}_c$ and seems to help in making the discriminative features and classifier more robust with a large improvement of the accuracy.

Another way to see the effect of the branch balancing term is by looking at the evolution of the partial reconstruction losses ($||\mathbf{x} - \hat{\mathbf{x}}_c||_2^2$ and $||\mathbf{x} - \hat{\mathbf{x}}_u||_2^2$) that measure how much each branch is contributing to the final reconstruction $||\mathbf{x} - \hat{\mathbf{x}}||_2^2$. We do so during the training of a HybridNet with and without branch balancing and show the result in Figure 3.10: without it, we can see that $||\hat{\mathbf{x}}_u - \mathbf{x}||_2^2 \approx 0$ so the unsupervised branch is reconstructing almost alone; with it, the two branches do not reconstruct perfectly which enables their cooperation, as expected.

3.4.3.3 Visualization of information separation on CIFAR-10 and STL-10

Overall, we can see in Table 3.5b lines *i* that thanks to the full HybridNet training loss, the information is correctly separated between $\hat{\mathbf{x}}_c$ and $\hat{\mathbf{x}}_u$ than both contribute somewhat equally while specializing on different types of information. For example, for the blue car, $\hat{\mathbf{x}}_c$ produces a blurry car with approximate colors, while $\hat{\mathbf{x}}_u$ provides both shape details and exact color information. For nicer visualizations, we also show reconstructions of the full HybridNet model trained on STL-10, which has larger images, in Figure 3.11. These confirm the observations on CIFAR-10. HybridNet is able to produce a very good final reconstruction composed of a rough reconstruction that lacks texture and color details from the

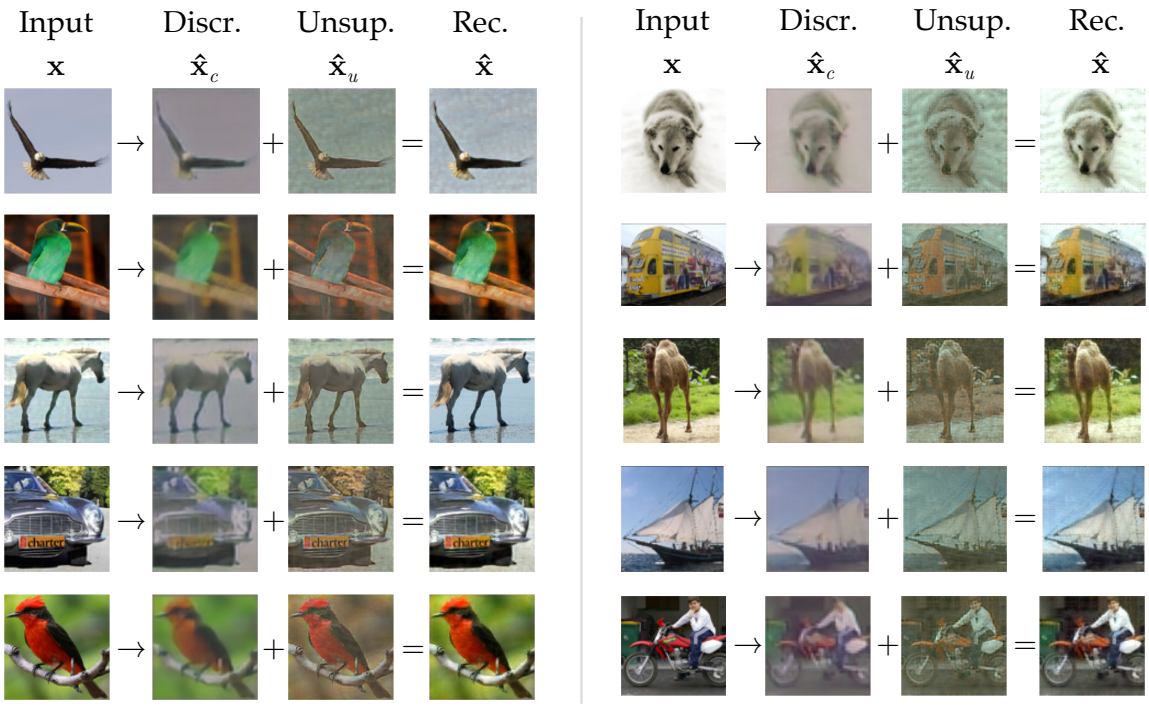


Figure 3.11. – **Visualizations of input, partial and final reconstructions** of STL-10 images using a HybridNet model derived from a ConvLarge-like architecture.

discriminative branch, completed by low-level details of shape, texture, writings, color correction and background information from the unsupervised branch.

3.4.4 State-of-the-art comparison

After studying the behavior of this novel architecture, we propose to demonstrate its effectiveness and capability to produce state-of-the-art results for SSL on three datasets: SVHN, CIFAR-10 and STL-10.

We use ResNet architectures to constitute the supervised encoder E_c and classifier C; and augment them with a mirror decoder D_c and an unsupervised second branch containing an encoder E_u and a decoder D_u using the same architecture. For SVHN and CIFAR-10, we use the small ResNet from (Gastaldi 2017), which is used in Mean Teacher (Tarvainen and Valpola 2017) and currently achieves state-of-the-art results on CIFAR-10. For STL-10, we upscale the images to 224×224 px and use a regular ResNet-50 pretrained on the Places dataset.

We trained HybridNet with the training method described in Section 3.3.2, using Mean Teacher to produce stability targets $\tilde{\mathbf{y}}^{(k)}$. The training protocol follows exactly the protocol of Mean Teacher (Tarvainen and Valpola 2017) for CIFAR-10

		Dataset	CIFAR-10			SVHN		STL-10
Model type	Model	Nb. labeled images N_s	1000	2000	4000	500	1000	1000
Reconstruction / Generation based	SWWAE (Zhao et al. 2016)						23.56	25.67
	Ladder Network (Rasmus et al. 2015)				20.40			
	Improved GAN (Salimans et al. 2016)		21.83	19.61	18.63	18.44	8.11	
	CatGAN (Springenberg 2016)				19.58			
Stability based	Stability regularization (Sajjadi et al. 2016)				11.29	6.03		
	Temporal Ensembling (Laine and Aila 2017)				12.16	5.12	4.42	
	Mean Teacher ConvLarge (Tarvainen and Valpola 2017)		21.55	15.73	12.31	4.18	3.95	
	Mean Teacher ResNet (Tarvainen and Valpola 2017)		10.10		6.28	*2.33	*2.05	*16.8
Rec. & Stability	ResNet baseline (Gastaldi 2017)		45.2	24.3	15.45	12.27	9.56	18.0
	HybridNet [ours]		**8.81**	**7.87**	**6.09**	**1.85**	**1.80**	**15.9**

Table 3.6. – **Results using a ResNet-based HybridNet**. trained on CIFAR-10, STL-10 and SVHN. "Mean Teacher ResNet" is our classification & stability baseline; results marked with * are not reported in the original paper and were obtained ourselves.

and a similar one for SVHN and STL-10 for which (Tarvainen and Valpola 2017) does not report results with ResNet. The hyperparameters added in HybridNet, *i.e.* the weights of the reconstruction terms (final and intermediate), were coarsely adjusted on a validation set. The details of the architecture used and the values of the hyperparameters are provided in Appendix B.

The results of these experiments are presented in Table 3.6. We can see the huge performance boost obtained by HybridNet compared to the ResNet baselines, in particular with CIFAR-10 with 1000 labels where the error rate goes from 45.2% to 8.81%, which demonstrates the large benefit of our regularizer. HybridNet also improves over the strong Mean Teacher baseline (Tarvainen and Valpola 2017), with an improvement of 1.29 pt with 1000 labeled samples on CIFAR-10, and 0.9 pt on STL-10. We also significantly improve over other stability-based approaches (Sajjadi et al. 2016; Laine and Aila 2017), and over the Ladder Networks (Rasmus et al. 2015) and GAN-based techniques (Springenberg 2016; Salimans et al. 2016).

These results demonstrate the capability of HybridNet to apply to large residual architectures – that are very common nowadays – and to improve over baselines that already provided very good performance.

3.5 Conclusion

In this chapter, we proposed to go further in the development of new ways to represent the information in a Convolutional Neural Network (ConvNet), tackling the issue of Semi-Supervised Learning (SSL). In this context, one can usually use regularization techniques like SHADE or stability techniques which propose to increase the invariance of the representation, but this comes in conflict with a reconstruction task that is often used to leverage unlabeled data in SSL.

To overcome this incompatibility and encourage efficient cooperation between the two tasks, we proposed HybridNet, an auto-encoder-based architecture with two distinct paths that separate the discriminative information useful for classification from the remaining information that is only useful for reconstruction. By adding this new unsupervised branch, we are able to release the constraints imposed by reconstruction and structure the information between the two latent spaces. This is achieved by the loss terms and training technique that accompany the architecture and allow it to behave in the desired way. In the experiments, we validated the significant performance boost brought by HybridNet in comparison with several other common architectures that use reconstruction losses and stability. We also showed that HybridNet can produce state-of-the-art results on CIFAR-10, STL-10 and SVHN.

While HybridNet was developed for the particular case of SSL, this idea of a two-branch architecture that makes classification and reconstruction cooperate can also be interesting in a wider variety of contexts. In the next chapter, we consider the problem of using a two-branch architecture to encode two complementary *semantic* information – thus using early fusion for a more complex combination – from which we can reconstruct an image.

CHAPTER 4

DUAL-BRANCH STRUCTURING OF THE LATENT SPACE FOR DISENTANGLING AND IMAGE EDITING

Chapter abstract

To complete our work on latent representations of images, we address the problem of producing disentangled latent representations, which means finding representations that model individual factors of variation in the data. For this, we propose DualDis, a new auto-encoder-based framework that separates and linearizes two complementary types of information that we call class and attributes, improving the semantic quality of the representations and their complementarity. This is achieved thanks to a two-branch architecture forcing the separation of the two kinds of information, accompanied by a decoder for image reconstruction and generation. To effectively separate the information, we propose to use a combination of regular and adversarial classifiers to guide the two branches in specializing for class and attribute information respectively. We also investigate the possibility of using semi-supervised learning for an effective disentangling even using few labels. We leverage the linearization property of the latent spaces for semantic image editing and generation of new images. We validate our approach on CelebA, Yale-B and NORB by measuring the efficiency of information separation via classification metrics, visual image manipulation and data augmentation.

The work in this chapter has led to the submission of a conference paper currently under review:

- Thomas Robert, Nicolas Thome, and Matthieu Cord (2019). "DualDis: Dual-Branch Disentangling with Adversarial Learning". In: *Under Review at Advances in Neural Information Processing Systems (NeurIPS)*.

Contents

4.1 Introduction

In the previous chapter, we proposed to separate the visual information contained in an image in two latent spaces, one for the discriminative information related to the category, and one for the non-discriminative information. This was done in order to improve the accuracy of a classifier in the context of Semi-Supervised Learning (SSL) by solving a conflict between two popular ways of regularizing Deep Neural Networks (DNNs) using HybridNet. One interpretation of the behavior of HybridNet is that its first branch encodes discriminative information that is related to the general pattern of the image (related to the class) while the second and complementary unsupervised branch focuses on details independent of the class such as local textures, detailed shape, *etc*.

In this chapter, we propose to pursue further this interpretation of information separation and tackle the problem of *disentangling*. In Deep Learning (DL) and especially in the Computer Vision (CV) community, this problem of disentangling factors of variation is a very active field of research, *cf.* Higgins et al. (2018). The

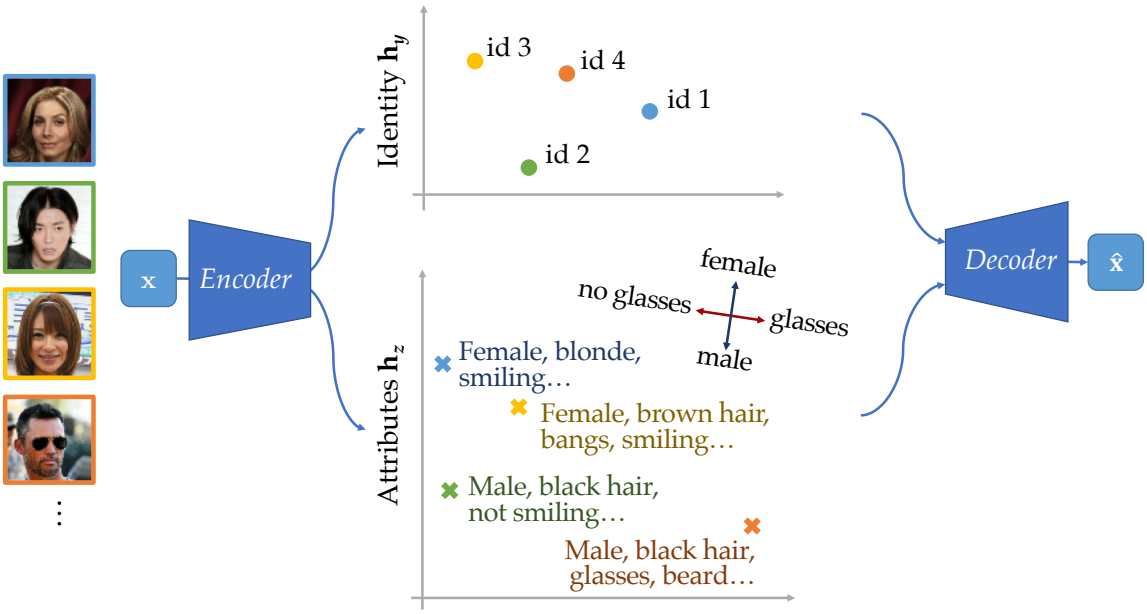
exact objective of those model vary but the overall idea of disentangling is to increase the quality of the latent representations so that they represent independent factors of variation in the data. We will see that this can be done in various ways.

By improving the semantic quality of the representations and their independence, disentangling can be used to improve many applications such as transfer learning (Ruiz et al. 2019), domain adaptation (Chang et al. 2019; Louizos et al. 2016), information retrieval (Mathieu et al. 2016), image generation (Perarnau et al. 2016), *etc*. In addition, since these models are usually based on encoder-decoder architectures, they can combine *visual understanding* and *image generation*. For the particular application of image generation, while disentangling models do not yet compete with powerful generative models (*e.g.* Karras et al. 2019) regarding the quality of generated images, they are an interesting direction for controlling latent conditional factors regarding what is being generated, which remains a challenging task in this literature.

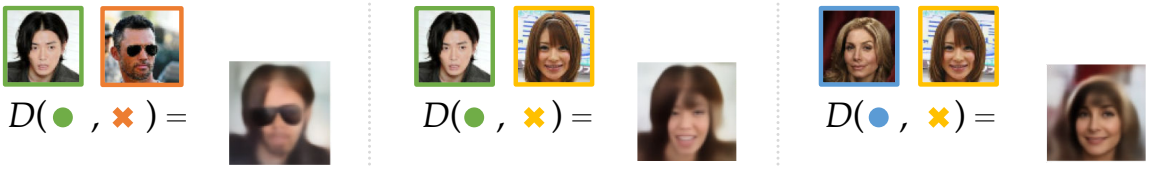
To address this problem, we propose to further explore structured latent representations of images, in continuation of our work in Chapter 3, but this time designed for both *image classification* and *visual attribute detection*. We are interested in modeling two complementary kinds of information that we will call *information domains*. For example, with a face dataset, we would like to represent the identity (*i.e.* the class) of the person and various visual attributes (hairstyle, makeup, facial expression, *etc*.). This direction thus addresses the question of producing complementary representation spaces that improve their cooperation to structure the information, generalize better and produce good reconstructions and generations.

While HybridNet focused on separating discriminative and non-discriminative information, with a clear asymmetry between the two, here we separate two similarly interesting types of information. Leveraging the insights of the previous chapter, we propose **DualDis**, a dual-branch deep Auto-Encoder (AE) that explicitly separates the information domains in two distinct latent subspaces as schematized in Figure 4.1a: one space $\mathbf{h}_y$ for class-related information and other $\mathbf{h}_z$ for attribute-related information. A decoder $D(\mathbf{h}_y, \mathbf{h}_z)$ is then used to reconstruct images and generate new ones. An important contribution lies in the learning strategy that we propose. Using adversarial training, we are able to explicitly find and remove wrongly organized information and effectively separate and "orthogonalize" the two information domains. This disentangling behavior is illustrated in Figure 4.1b where we show that it allows mixing representations of different images.

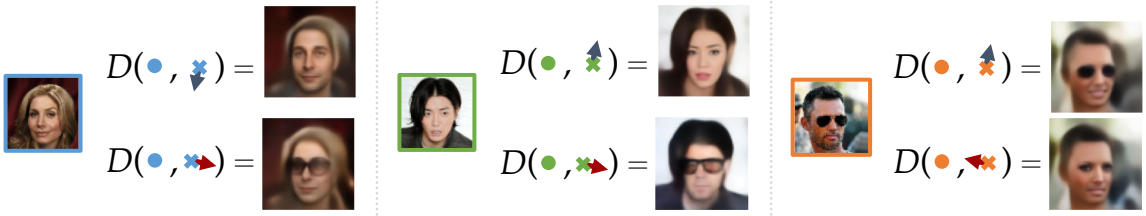
In addition, our architecture is also designed to linearize the factors of variation in each latent space. This reinforces the semantic quality of the representations and means that simple linear shifts of a latent representation in given directions are directly linked to known semantic factors. Taking advantage of this property,



(a) **Behavior of our encoder-decoder**, learned to explicitly separate complementary representations of identity (top) and attributes (bottom) in dual latent subspaces.



(b) **Illustration of the *disentangling* ability of DualDis**, mixing the identity of a first image and the attributes of a second. In the middle example, the man framed in green takes the attributes of the women framed in yellow, becoming a smiling woman with brown bangs.



(c) **Illustration of the *image editing* ability of DualDis**, provided by the linearization of the factors of variation. For the first example (woman framed in blue), we move the representation × along the directions male (first line) and glasses (second line) to add those attributes.

Figure 4.1. – **Overview of our DualDis framework**. We present the general behavior of the model (a) and illustrate its abilities at disentangling (b) and editing images (c).

we can perform image editing by modifying the representations of a given image. This is illustrated in Figure 4.1c, where we change the gender and eyeglasses attributes of images while conserving the identity and the other attributes. This is done by following the linear directions provided by the model, as shown on the left of the figure. Thanks to this, we can also perform *guided* Data Augmentation (DA) by generating variations of images with semantic changes instead of low-level changes (flip, translation, color jitter, *etc.*) as usually done.

In this chapter, we first present the state of the art in Section 4.2, then we present DualDis in depth in Section 4.3 and we validate the effectiveness of our approach in the next sections quantitatively (Section 4.5), using SSL (Section 4.6), and for editing (Section 4.7).

4.2 Related work

In this chapter, we try to go one step closer to eventually "bridging the gap" between discriminative and generative models. Indeed, our objective is to produce a model that both represents highly semantic information and is able to generate new data. We thus propose to go over some generative and disentangling models that exist in the literature.

Notations. Because the standard notations vary from one domain to another, and to avoid any misunderstanding, we choose to unify the notations and use the ones that follow:

$\mathbf{y}, \mathbf{z}$ Labels of the primary and secondary semantic information are designated respectively with $\mathbf{y}$ and $\mathbf{z}$. The primary information ($\mathbf{y}$) corresponds to the class or identity of the subject of the image, while the secondary information $\mathbf{z}$ corresponds to general visual attributes (*e.g.* for faces it represent the expression, hairstyle, *etc.*).

$\mathbf{h}$ All latent vectors are designed with $\mathbf{h}$, including inputs of generative models.

$\tilde{\square}$ Vectors with a tilde correspond to elements that were randomly drawn from a prior distribution or generated from one (*e.g.* $\tilde{\mathbf{h}} \sim p(\mathbf{h}), \tilde{\mathbf{x}} = G(\tilde{\mathbf{h}})$)

$\hat{\square}$ Vectors with a hat correspond to elements that are estimated and usually correspond to a known value (*e.g.* $\hat{\mathbf{x}} = D(E(\mathbf{x}))$ is the reconstruction of a known input image $\mathbf{x}$, $\hat{\mathbf{y}}$ is a prediction of a ground truth $\mathbf{y}$)

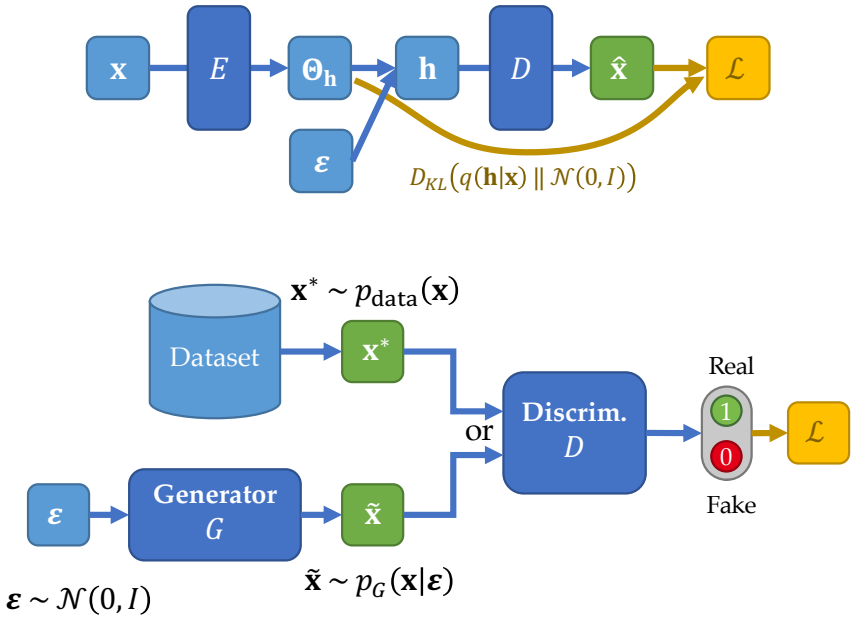


Figure 4.2. – Schematic representation of a VAE (top) and a GAN (bottom).

4.2.1 Generative models

First, we propose a quick overview of some modern generative models and how they represent the information. In this context, the term *generative models* refers to models that are able to generate new images $\tilde{\mathbf{x}}$ from a randomly sampled input noted $\tilde{\mathbf{h}}$. In the recent Deep Learning (DL) literature, there are two main types of models for this, Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs).

Variational Auto-Encoder. VAEs were introduced by Kingma and Welling (2013) and can be seen as an extension of an AE, represented in Figure 4.2 (top). To summarize, after some modeling assumptions and lower-bounds of the data likelihood maximization $\max \log p(\mathbf{x})$, the training loss ends up consisting in adding a regularization term on the latent representations $\mathbf{h}$ of an AE so that they match a prior distribution $p_{\text{prior}}(\mathbf{h})$:

$$\mathcal{L}(\mathbf{x}) = \mathcal{L}_{\text{rec}}(\mathbf{x}, \hat{\mathbf{x}}) + \mathcal{D}_{\text{KL}}(\mathbf{h} \,||\, p_{\text{prior}}(\mathbf{h})) \tag{4.1}$$

During the training, the model does not generate samples, it simply learns to reconstruct and make sure that the representations $\mathbf{h}$ respect the prior distribution. However, thanks to this prior distribution $p_{\text{prior}}(\mathbf{h})$, after the training, it is possible to sample values $\tilde{\mathbf{h}} \sim p(\mathbf{h})$ and use the decoder D of the VAE to obtain a new

image $\tilde{\mathbf{x}} = D(\tilde{\mathbf{h}})$ that should belong to the distribution of real images. As we mentioned, this is ensured through the maximization of the data likelihood.

Generative Adversarial Network. GANs were proposed by Goodfellow et al. (2014) and rely on a different and original approach that directly addresses the question of producing realistic images, as represented in Figure 4.2 (bottom). They propose to use a generator G transforming an input noise $\tilde{\mathbf{h}} \sim p_{\text{prior}}(\mathbf{h})$ (with a chosen fixed prior $p_{\text{prior}}(\mathbf{h})$) into an image $\tilde{\mathbf{x}}$. This generator is accompanied by a discriminator D that learns to detect if its input $\mathbf{x}$ is a real image from the dataset ($\mathbf{x}^*$) or a generated image from G ($\tilde{\mathbf{x}}$). The discriminator is trained to make correct predictions; and the generator is trained to fool the discriminator, *i.e.* to produce images $\tilde{\mathbf{x}}$ that D classify as real images. They have therefore opposite goals and are often said to be trained toward a Nash equilibrium.

$$\tilde{\mathbf{x}} = G(\tilde{\mathbf{h}}),\ \tilde{\mathbf{h}} \sim p_{\text{prior}}(\mathbf{h}), \quad D(\mathbf{x}) \in [0,1],\ \text{ideally,}\ \begin{cases} D(\tilde{\mathbf{x}}) = 0, & \tilde{\mathbf{x}} = G(\tilde{\mathbf{h}}) \\ D(\mathbf{x}^*) = 1, & \mathbf{x}^* \in \mathcal{D}_{\text{real}} \end{cases} \tag{4.2}$$

Because of their particular training, the exact behavior of GANs has been widely discussed. Very unstable especially in its early stages, solutions were proposed (Kodali et al. 2017; Roth et al. 2017) and GANs can now produce very high-quality images (Karras et al. 2017; Karras et al. 2019) and can also be used for image-to-image translations (*e.g.* changing maps into satellite images) (J.-Y. Zhu et al. 2017; Choi et al. 2018), super-resolution (Ledig et al. 2017), *etc*.

A difference that is often discussed between GANs and VAEs is the "sharpness" of the images produced. This phenomenon is studied and quantified more thoroughly by Blau and Michaeli (2018) in the context of super-resolution or image restoration and is described as the perception-distortion tradeoff. This shows that models that produce "sharp" looking images (*i.e.* with good perceptual quality) are actually adding incorrect local information that does not correspond to the reality. Doing so, they actually produce more distortion (*i.e.* difference with the ground truth) than models that produce blurry images. This may or may not be problematic depending on the application, *i.e.* if matching a specific *ground truth* is relevant or if artificial details are welcome.

Information representation in generative models. First, we can note that unlike VAEs that come with an encoding model, GAN-based models usually do not, making it impossible to know the representation $\mathbf{h}$ of an existing image, which limits the possible applications to pure generation. To solve this, models propose variations of the original GAN framework. They introduce an encoder $E(\mathbf{x})$ producing representations $\mathbf{h}$ that the generator $G(\mathbf{h})$ must be able to decode back

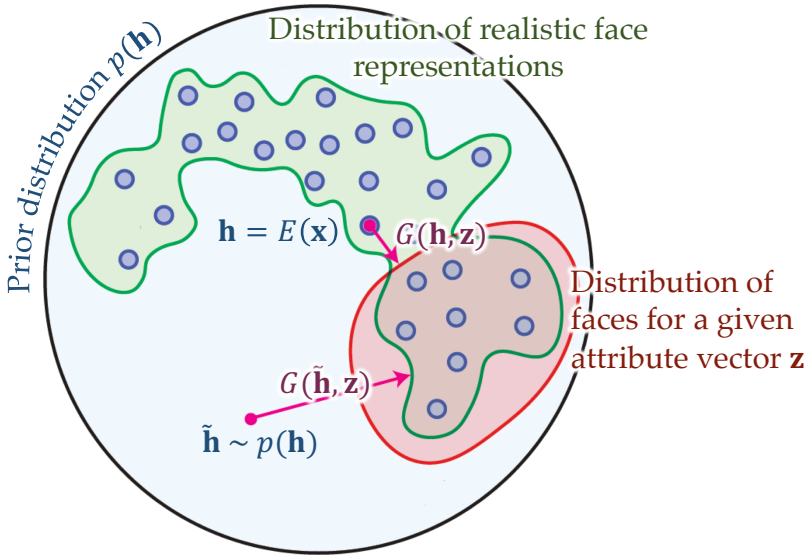


Figure 4.3. – **Behavior of Latent Constraint VAE by Engel et al. (2018).** They learn a model $G(\mathbf{z}, \mathbf{z})$ that is capable of transforming an existing representation $\mathbf{h}$ (encoded from an image or sampled randomly) into a realistic image that is valid regarding the provided attributes $\mathbf{z}$. Credits: Illustration based on Engel et al. (2018)

into the original image (Dumoulin et al. 2017; Donahue et al. 2017; Brock et al. 2019), which also has the advantage of being a way to help prevent mode collapse (Rosca et al. 2017; Bang and Shim 2018).

Second, we can see that those generative models define a prior distribution of the latent information (the embedding of the VAE and the noise of the GAN), which is usually very simple, like a Gaussian $\mathcal{N}(0, I)$ or a uniform distribution $U_{[-1,1]}$. This means that the factors of variation of the dataset must be somehow encoded in this unstructured space. The fact that there is no structure though, is limiting if we wanted to manipulate or interpret this information. While Makhzani et al. (2016) proposed to use more complex and structured latent spaces (*e.g.* a mixture of Gaussians), it had little followup, possibly because choosing and enforcing a structure *a priori* is not an easy task.

To overcome this and control semantic factors, the most simple way is to use conditional generation, which consists in generating from a noise $\tilde{\mathbf{h}}$ and a label vector $\mathbf{z}$, as proposed by Perarnau et al. (2016), T.-C. Wang et al. (2018), Bodla et al. (2018), and Z. He et al. (2019) and many others. While effective, it seems clear that all the complex information related to a semantic factor (*e.g.* the glasses on a picture of a face) cannot be fully represented by a binary representation. Thus, that part of this information must leak into $\tilde{\mathbf{h}}$. This is why trying to find richer representations of the semantic factors is an important direction.

An interesting idea in this direction to model more complex representations of semantic factors is proposed by Engel et al. (2018) and is illustrated in Figure 4.3. They start with a regular trained VAE model that encoded the dataset without supervision. Making the hypothesis that the information about semantic factors has been somewhat organized in the latent space $\mathbf{h}$ of the AE, they add a model $G(\mathbf{h}, \mathbf{z})$ which role will be to represent how the semantic information is organized in $\mathbf{h}$. More precisely, G is trained to transform any representation $\mathbf{h}$ into a representation $\mathbf{h}'$ that is "valid" with regard to the label vector $\mathbf{z}$: $\mathbf{h}' = G(\mathbf{h}, \mathbf{z})$. For example, if $\mathbf{h}$ represents someone without glasses, the glasses can be added using a vector $\mathbf{z}$ with glasses set to true. G should produce a new vector $\mathbf{h}'$ keeping the same identity but adding glasses. The role of G is therefore to model how each semantic factor $\mathbf{z}_i$ has been represented and organized in the latent space $\mathbf{h}$ by the original VAE. Unlike a conditional model, the information about $\mathbf{z}$ is represented directly in $\mathbf{h}$ and G allows finding regions of $\mathbf{h}$ corresponding to factors $\mathbf{z}$.

4.2.2 Unsupervised disentangling

The disentangling literature proposes to go further on this idea of structuring the information in the latent space. "Disentangling" can have more or less constraining definitions depending on the articles, but overall it consists in explicitly separating and representing independent factors of variation.

A first approach is unsupervised disentangling, where no labels about the factors of variation are provided. Those approaches are usually based on VAEs and try to produce a model in which latent units are all independent of each other. A simple solution is for example β-VAE by Higgins et al. (2017), which adds a weighting parameters β on the prior constraint, increasing how independent the latent neurons are. T. Q. Chen et al. (2018) and Kim and Mnih (2018) both study this phenomenon more thoroughly by decomposing the prior term into multiple terms and increasing only the importance of the *total correlation* term in their models FactorVAE and TC-VAE. Finally, Dupont (2018) proposes a variant which allows discrete intermediate representation in $\mathbf{h}$.

Those models are based on the assumption that each latent unit encodes one variation factor and that all the units are independent of each other. To actually quantify this phenomenon, they all propose their own metrics to, *in fine*, measure how well labeled variation factors (used only for evaluation) are represented, each in a single neuron of $\mathbf{h}$. This definition (and metric) of disentangling is of course widely discussed (Higgins et al. 2018). For example, it is not clear why a semantic factor should only be encoded by a single neuron, and whether it is realistic to consider only independent neurons and factors.

Other models differ from this "disentangling by regularization" approaches based on VAE. For example, Kulkarni et al. (2015) and Hu et al. (2018) work on latent chunks, *i.e.* groups of neurons of $\mathbf{h}$ that should each encode a semantic piece of information. In particular, Hu et al. (2018) propose to mix latent chunks of different images and adversarially enforcing similarity and difference of images generated from the mixes to obtain disentangling and independent chunks.

However, unsupervised disentangling has two important issues: first, without any supervision, it is not easy to enforce that a model actually learns complex semantic factors (*e.g.* facial expression, pose, hairstyle, *etc.*) and not for example a decomposition of the factor into an ensemble of simpler and less semantic ones. This is especially true when enforcing that each neuron encodes a different factor. Second, because we don't have labels, it is impossible to interpret the latent representations, the only solution being to ask a human to look at the effect of each neuron and visually interpret it, which is risky regarding the generalization on the full dataset of the interpretation made on a few samples. Besides, if a human needs to label the representations, it seems more reasonable to use this time to label the factors in the dataset and use them in the training.

4.2.3 Supervised disentangling

Numerous approaches also propose *supervised disentangling*, using labels in addition to images to disentangle factors of variation. In this literature, the definition of disentangling is even broader than for *unsupervised disentangling*, focusing on separating different pieces of information.

A first approach for this is based on conditional generative models, where the decoder takes $(\mathbf{h}, \mathbf{z})$ as input, where $\mathbf{z}$ are the labeled factors and $\mathbf{h}$ represents the rest of the information. In this regard, we can cite Perarnau et al. (2016), Tran et al. (2017), and Yang Liu et al. (2018). A good example is Fader Network (Lample et al. 2017), which uses an AE-based model: the encoder producing $\mathbf{h}$ is supposed to represent everything (mostly the identity and the background) except the visual attributes (makeup, hairstyle, glasses, *etc.*); the decoder takes $\mathbf{h}$ and the attributes as a binary vector $\mathbf{z}$ as input to reconstruct. The disentangling between attributes and non-attributes is achieved by forcing the encoder to remove $\mathbf{z}$-related information in $\mathbf{h}$ using adversarial training.

Numerous approaches are also based on an approach similar to the intuition of HybridNet, separating the information in two latent spaces. This is the case of Mathieu et al. (2016), Peng et al. (2017), Klys et al. (2018), Hadad et al. (2018), Jaiswal et al. (2018), and Yu Liu et al. (2018). The general idea of those models is to find a way to separate two information domains (*e.g.* information related

to a person's identity $\mathbf{y}$ and information related to visual attributes $\mathbf{z}$ of a face) in two latent spaces. Mostly, it consists in using classification and adversarial training to represent $\mathbf{y}$-related information in one subspace and remove $\mathbf{y}$-related information in the other subspace. These methods can be reproduced using parts of our DualDis framework and will be discussed in depth in Section 4.4.

4.3 DualDis approach

In this chapter, we propose to build upon the intuitions and interpretations of HybridNet to address supervised disentangling. Our objective is to separate two *information domains* that are complementary: the first represents the class or person's identity $\mathbf{y}$, *i.e.* a category that contains a lot of intra-class variability, and the second domain represents semantic attributes $\mathbf{z}$ (*e.g.* hairstyle, makeup, pose, lighting, *etc.*) related to this variability.

To this end, we propose an approach called DualDis presented in Figure 4.4. On the left, we show the architectural part of our contribution, using a two-branch model and disentangling to separate the two information domains. Those domains have classification labels $\mathbf{y}$ and $\mathbf{z}$ that we want to predict ($\hat{\mathbf{y}}$, $\hat{\mathbf{z}}$), along with a reconstruction $\hat{\mathbf{x}}$ of the input $\mathbf{x}$. In the center of the figure, we describe the second part of our approach which is the training process designed to successfully disentangle the two domains, using adversarial classifiers and multiple loss terms. On the right, to put our model in perspective, we indicate how some related works can be reproduced using the same kind of architectures with variations in the losses used as described in Section 4.4.

In this section, we will detail how DualDis is designed to separate the information domains and linearize semantic factors. For information separation, each latent space is connected to an adversarial classifier of the opposite information domain which learns to find the information that belongs to the wrong domain. The encoder will then learn to remove this information from the latent space, making classification impossible and thus filtering only the relevant information. DualDis is also designed to linearize the labeled factors. We use linear classifiers that both guide the linearization process and provide us with the linear directions that are associated with known labeled semantic factors. This allows us to semantically navigate the latent spaces since moving in one of those linear directions will increase the presence of the associated factor.

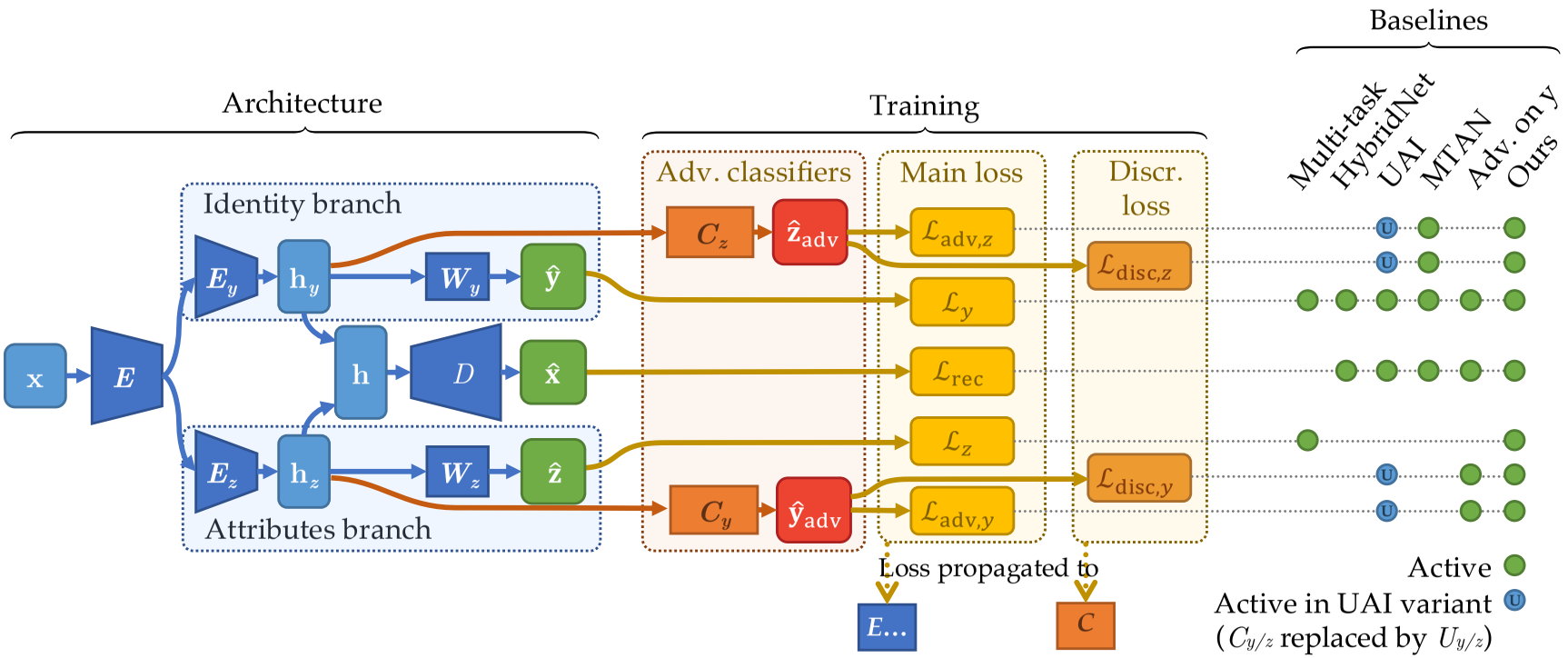


Figure 4.4. – **Architecture and learning of the DualDis framework**. We use a two-branch encoder-decoder architecture with classifiers (left). For the training, in addition to classical reconstruction and classification losses, we use adversarial classifiers and loss terms to force the two latent spaces to encode complementary "orthogonal" information (middle). x We also indicate (right) how subsets of components of DualDis can be used to reproduce existing baselines described in Section 4.4, in particular Multi-task models (Kokkinos 2017), HybridNet, UAI (Jaiswal et al. 2018), MTAN (Yang Liu et al. 2018), models with an adversarial loss on y only (Hadad et al. 2018; Yu Liu et al. 2018; Klys et al. 2018).

4.3.1 Dual branch Auto-Encoder

We propose an encoder-decoder architecture with a latent space split into two parts, $\mathbf{h}_y$ and $\mathbf{h}_z$. Each representation is produced by a deep encoder E_y or E_z so that the features are explicitly separated. These representations are concatenated into $\mathbf{h}$ and fed to a decoder D, producing a reconstruction $\hat{\mathbf{x}}$. Having a decoder enables image generation and also ensures that the model extracts robust features (Le et al. 2018) as we have seen in the previous chapters.

While it would be possible to encode all the information in a single latent space, having two branches encourages the model to encode two complementary kinds of information (Mathieu et al. 2016; Hadad et al. 2018; Yang Liu et al. 2018). Taking the example of a face dataset, we want the identity branch ($E_y \circ E$) to capture information related to the identity $\mathbf{y}$ with invariance toward other factors of variation (hairstyle, makeup, pose, *etc.*); and we want the attribute branch ($E_z \circ E$) to model this ignored information, since this branch needs to capture factors of variation linked to visual attributes $\mathbf{z}$. Having two separate deep encoders E_y and E_z is key to an effective disentangling, and they should be designed deep enough to produce "orthogonal" latent representations that encode very different information. Since the low-level features represented by the first convolutional layers are likely common to both domains, we use a single common encoder E before specializing the information in our two branches.

This auto-encoding backbone is trained using a simple Mean-Squared Error (MSE), $\mathcal{L}_{\text{rec}} = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2$. Using a different loss to train the decoder could be possible, like using a GAN discriminator (Goodfellow et al. 2014) or a perceptual loss (Dosovitskiy and Brox 2016) which are both known to produce sharper images. However, we considered that it was out of the scope of this work since we are not interested in producing high-quality generations but focus on the disentangling ability of the model.

4.3.2 Modeling factors of variation

We want our architecture to produce robust representations of each information domain as well as provide classification predictions. To that end, we have seen in the previous chapter that having a two-branch encoder can improve classification performance by encouraging representations $\mathbf{h}_y$ and $\mathbf{h}_z$ to be invariant toward intra-class variations. To the encoders, we add linear classifiers $\mathbf{W}_y$ and $\mathbf{W}_z$, one for each branch, that predicts respectively $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$. These classifiers guide the auto-encoding backbone to organize the information extracted for reconstruction in the right branch between our two latent spaces $\mathbf{h}_y$ and $\mathbf{h}_z$ so that it allows

predicting the class/identity and the attributes. To train those classifiers, we use regular classification losses. We have:

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_y \mathbf{h}_y)\,, \qquad \hat{\mathbf{z}} = \text{sigmoid}(\mathbf{W}_z \mathbf{h}_z)\,; \tag{4.3}$$

$$\mathcal{L}_y = \text{CrossEntropy}(\mathbf{y}, \hat{\mathbf{y}})\,, \qquad \mathcal{L}_z = \text{BinaryCrossEntropy}(\mathbf{z}, \hat{\mathbf{z}})\,. \tag{4.4}$$

Linearization of the factors for manipulation. In addition, the design of those classifiers is chosen in order to encourage the linearization of the representations of labeled factors of variation. By choosing linear classifier, the presence of the i^{th} attribute $\mathbf{z}_i$ in an input image is estimated by a linear predictor:

$$\hat{\mathbf{z}}_i = \text{sigmoid}(\mathbf{w}_{z_i} \cdot \mathbf{h}_z) \text{ with } \mathbf{w}_{z_i} \text{ the } i^{\text{th}} \text{ row of the matrix } \mathbf{W}_z\,. \tag{4.5}$$

This means that we can manipulate the latent space to artificially increase or decrease the presence of this attribute by moving $\mathbf{h}_z$ in the direction of this vector:

$$\mathbf{h}'_z = \mathbf{h}_z \pm \varepsilon \mathbf{w}_{z_i}^\top\,. \tag{4.6}$$

4.3.3 Disentangling information domains and factors of variation

We also want our architecture to explicitly disentangle the two domains. To this end, we add classifiers $C_y(\mathbf{h}_z)$ and $C_z(\mathbf{h}_y)$ that predict the target of the opposite domain ($\mathbf{y}$ from $\mathbf{h}_z$ and vice versa). We call those classifiers "adversarial" since their role is to find information that should not be present, *e.g.* information about attributes encoded in the identity branch. By training a classifier to find this information, we are then able to make the encoder remove it. Those classifiers are designed as non-linear multi-layer classifiers to find the information even if it is not linearly separable in the latent space.

4.3.3.1 Global loss and adversarial training

To train our model, we rely on adversarial training, with parts of the model that have different training losses but are connected. Concretely, it means that the parameters of the model are not all updated using the same losses, as is represented in Figure 4.4.

We define θ_{main} as the weights of the *main* model, *i.e.* the model that is eventually used at the end of the training, which includes the encoders, the decoder and the normal classifiers: $E, E_y, E_z, D, \mathbf{W}_y, \mathbf{W}_z$. We also define θ_{disc} as the weights of the two adversarial classifiers C_y and C_z (similarly to GANs denomination).

To train the model, we have two losses: *main loss* $\mathcal{L}_{\text{main}}$ describes the expected behavior of the model and is propagated in the encoders, decoder and classifiers $\mathbf{W}$. The *discriminative loss* $\mathcal{L}_{\text{disc}}$ is used to train the adversarial classifiers C_y and C_z only; helping the main loss by searching for information that should be removed.

$$\mathcal{L}_{\text{main}} = \lambda_r \mathcal{L}_{\text{rec}} + \lambda_y \mathcal{L}_y + \lambda_z \mathcal{L}_z + \lambda_{a,y} \mathcal{L}_{\text{adv},y} + \lambda_{a,z} \mathcal{L}_{\text{adv},z} + \lambda_o \mathcal{L}_{\text{orth}} \,, \tag{4.7}$$

$$\mathcal{L}_{\text{disc}} = \lambda_{d,y} \mathcal{L}_{\text{disc},y} + \lambda_{d,z} \mathcal{L}_{\text{disc},z} \,. \tag{4.8}$$

Each loss term in these global losses can be weighted using various λ to control their importance.

To optimize our model, we apply a gradient descent using backpropagation, which for SGD writes:

$$\theta_{\text{main}} \leftarrow \theta_{\text{main}} - \eta \nabla_{\theta_{\text{main}}} \mathcal{L}_{\text{main}} \,, \tag{4.9}$$

$$\theta_{\text{disc}} \leftarrow \theta_{\text{disc}} - \eta \nabla_{\theta_{\text{disc}}} \mathcal{L}_{\text{disc}} \,. \tag{4.10}$$

Interestingly, we can see that to compute the gradients, we apply backpropagation to layers that are not necessarily updated by the optimizer, which is unusual in DL outside of adversarial learning.

4.3.3.2 Disentangling information domains

To effectively remove information where it should not be encoded, we thus start by defining discriminative loss terms $\mathcal{L}_{\text{disc},y}$ and $\mathcal{L}_{\text{disc},z}$, applied to C_y and C_z, to make the adversarial classifiers achieve the correct classification of their target. These will make the adversarial classifiers model the information we want to remove.

We then define the adversarial terms $\mathcal{L}_{\text{adv},y}$ and $\mathcal{L}_{\text{adv},z}$ in the main loss to make the encoders produce features that prevent the classifiers C to achieve their goal and ideally have the accuracy of a random classifier. This means that the encoders will need to remove the information that is used by the classifiers C, which is unwanted. This goal can be expressed in different possible ways:

- $\max \mathcal{H}[\hat{\mathbf{y}}_{\text{adv}}]$
  By maximizing the entropy of the adversarial prediction, which is optimal when the classifier assigns the same probability to all the classes.

- $\min \text{CrossEntropy}(p_{\text{prior}}(\hat{\mathbf{y}}_{\text{adv}}), \hat{\mathbf{y}}_{\text{adv}})$
  By minimizing the cross-entropy with a prior distribution $p_{\text{prior}}(\hat{\mathbf{y}}_{\text{adv}})$ for which we have different options: a uniform distribution among the classes (which has the same optimum as $\max H[\hat{\mathbf{y}}_{\text{adv}}]$ but different gradients); a distribution that

matches the prior distribution of the classes in the dataset if it is unbalanced; *etc*.

- $\max \text{CrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}_{\text{adv}})$
  By simply encouraging the prediction of the "inverse" of the ground truth, *i.e.* maximizing the cross-entropy with the ground truth. While being less mathematically correct since it means the optimum would be to predict a probability score of the ground truth class lower than the other classes, we found this solution to be the most effective, probably because it has better gradients and since in practice it is actually never able to reach the point where the probability of the real class is lower than the others.

Thus, we choose to use the following loss terms:

$$\mathcal{L}_{\text{disc},y} = \text{CrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}_{\text{adv}}) \,, \tag{4.11}$$

$$\mathcal{L}_{\text{adv},y} = -\text{CrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}_{\text{adv}}) \,; \tag{4.12}$$

$$\mathcal{L}_{\text{disc},z} = \text{BinaryCrossEntropy}(\mathbf{z}, \hat{\mathbf{z}}_{\text{adv}}) \,, \tag{4.13}$$

$$\mathcal{L}_{\text{adv},z} = \text{BinaryCrossEntropy}(1 - \mathbf{z}, \hat{\mathbf{z}}_{\text{adv}}) \,. \tag{4.14}$$

4.3.3.3 Intra-branch disentangling

Using adversarial classifiers, we disentangled class and attribute features. We now propose to disentangle the different labeled factors of variation by making the rows of the matrix $\mathbf{W}_z$ orthogonal, meaning each factor detector is independent of the others. We do so by minimizing the dot products of the pairs of normalized row vectors $\mathbf{w}'_{z_i}$ of $\mathbf{W}_z$:

$$\mathcal{L}_{\text{orth}} = \sum_i \sum_j \mathbf{w}'_{z_i} \mathbf{w}'^{\top}_{z_j} \quad \text{with} \quad \mathbf{w}'_{z_i} = \frac{\mathbf{w}_{z_i}}{||\mathbf{w}_{z_i}||_2} \,. \tag{4.15}$$

4.4 Discussion

Let us now discuss our DualDis approach in regard to related works and existing state-of-the-art models of supervised disentangling. Interestingly, as shown in Figure 4.4 (right), several existing state-of-the-art disentangling models can be reproduced and evaluated using the same architecture and loss components. By disabling parts of our architecture and/or with small additions and variations, we can thus produce a complete and fair comparison to them. We assign letters (A) to (E) to those models as we will refer to them in the experiments.

Class-only adversarial disentangling (E). First, by starting with DualDis and removing losses related to the attribute labels $\mathbf{z}$, we obtain *model (E)* that learns to predict the class $\mathbf{y}$ in one branch and dispel the information related to $\mathbf{y}$ in the other. This is notably the case for the recent models by Hadad et al. (2018), Yu Liu et al. (2018), and Klys et al. (2018).

The advantage compared to DualDis is that fewer annotations are required. However, this means that the disentangling is asymmetrical: while identity-related information is constrained, the attribute-related information remains unconstrained and can thus be encoded freely in $\mathbf{h}_y$ and $\mathbf{h}_z$ and remain entangled. It is therefore very difficult to ensure proper separation of the two information domains with this asymmetry. Besides, without labels $\mathbf{z}$, we lack a way to semantically navigate in the latent space $\mathbf{h}_z$. This is why we choose strong supervision with labels for both $\mathbf{y}$ and $\mathbf{z}$.

Symmetric latent adversarial disentangling (D). To solve this asymmetry, UAI (Jaiswal et al. 2018) proposes a new adversarial training used as *model (D)*, still without attribute labels. In their model, adversarial predictors U_y and U_z replace C_y and C_z. Those predictors learn to predict $\hat{\mathbf{h}}_y$ from $\mathbf{h}_z$ and vice versa, *i.e.* the opposite latent space. The encoders try to fool those predictors and make them fail at their prediction. Because no labels are needed for this task, this can be trained in an unsupervised manner. Probably due to the complexity of the predictor's task, all the modules (encoders, predictors, decoder) in Jaiswal et al. (2018) are linear.

This idea has the advantage of being symmetrical even without attribute labels, but it only "orthogonalizes" $\mathbf{h}_y$ and $\mathbf{h}_z$, making them independent of each other, but without guarantee of semantic disentangling. In particular, nothing ensures attribute information is removed from $\mathbf{h}_y$, since for it to happen, the model would need to encoded attribute information in $\mathbf{h}_z$ without labels. In addition, identity information can even remain in $\mathbf{h}_z$ if absent from $\mathbf{h}_y$. This is why we choose an explicit disentangling of labeled information.

Attribute-conditional decoder (C). Some models like Fader Networks (Lample et al. 2017), IcGAN (Perarnau et al. 2016), or MTAN (Yang Liu et al. 2018) *(model (C))* propose to use a *conditional generator* approach, which a single latent space $\mathbf{h}_y$ to encode the identity and a decoder fed with a binary vector $\mathbf{z}$: $D(\mathbf{h}_y, \mathbf{z})$. MTAN in particular applies a classification loss on $\mathbf{y}$ and an adversarial loss on $\mathbf{z}$ to remove the attribute information from $\mathbf{h}_y$.

Simplifying the disentangling since $\mathbf{z}$ is by definition purely attribute information and we only need to remove attribute information from $\mathbf{h}_y$, this approach has other drawbacks. Notably, it makes the strong assumption that all the attribute

information of a specific image is encoded in the binary vector $\mathbf{z}$, which is unlikely for complex semantic attributes, like a pair of glasses, a hat, facial expression, age, *etc.* This means that attributes information will necessarily leak in $\mathbf{h}_y$, and that our control on the attribute is very simple since we cant only control a binary value. This is why we prefer to use two latent spaces to let the model encode complex information in both information domains.

Multi-Task Learning (A & B). Finally, while not explicitly designed for it, two-branch multi-task models could also lead to disentangling thanks to the specialization induced by classification, even without adversarial training. In particular, we consider two possible setups. First, a model that learns to classify $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ like UberNet (Kokkinos 2017) *(model (A))*. And second, a simple variation of HybridNet *(model (B))*, that both learn to classify $\hat{\mathbf{y}}$ while encoding additional information in $\mathbf{h}_z$, and simultaneously learn to reconstruct to extract additional information not present explicitly in the labels. Compared to those methods, we choose to integrate an explicit disentangling process to effectively separate the information domains.

4.5 DualDis evaluation

We now propose to validate the effectiveness of our DualDis approach to efficiently disentangle two information domains. In this section, we describe the datasets used for this as well as the evaluation metrics that we use to quantitatively measure the disentangling and linearization performance of the models. We evaluate DualDis and compare it to state-of-the-art techniques previously described.

In Section 4.6, we present and evaluate own DualDis can be adapted to a semi-supervised context. In Section 4.7, we perform a qualitative evaluation of the possibility to perform image editing and use this capability to generate new images for Data Augmentation (DA).

4.5.1 Datasets and architecture

For the experiments, we use three datasets (*cf.* Table 4.1) preprocessed to fit our training protocol:

- **CelebA** (Z. Liu et al. 2015) is a face dataset of celebrities, for which we use 60k images with 2000 identities $\mathbf{y}$ and 40 attributes $\mathbf{z}$ (hairstyle, makeup, expression, *etc.*).

Dataset	Image size	# Train	# Test	$\lvert\mathcal{Y}\rvert$	$\lvert\mathcal{Z}\rvert$	Samples
CelebA	256×256	48,000	12,000	2,000	40	
Yale-B	64×64	1,200	1,200	38	14	
NORB	64×64	24,000	24,000	5	8	

Table 4.1. – **Overview of the datasets used**. We report the size of the images, the number of samples used for training and testing, the number of classes $|\mathcal{Y}|$ in $\mathbf{y}$ and the number of attributes $|\mathcal{Z}|$ in $\mathbf{z}$.

- **Yale-B** (Georghiades et al. 2001) is also a face dataset of 2.4k images with 38 identities $\mathbf{y}$ and 14 attributes $\mathbf{z}$ (light source position). More precisely, each of the 38 persons has been photographed once with 64 possible lighting combinations which we grouped into 14 categories of lightning.
- **NORB** (LeCun et al. 2004) is a dataset of 3D object renderings. 50 objects are rendered with different camera position and lighting type (960 combinations per object), making a dataset of 48k images with 5 categories $\mathbf{y}$ (10 objects per category) and 8 attributes $\mathbf{y}$ (camera position and lighting, created by us by grouping similar angles and lightings).

For CelebA and Yale-B, we create a test set by using respectively 20% and 50% of the images of each class, regardless of the attributes. For NORB, the test set is provided and consists of 25 of the 50 3D objects which represent 50% of the images. The validation set represents 20% of the training set, defined with the same process, and is used for the few hyper-parameters tests we do.

For the architecture, we use simple Convolutional Neural Networks (ConvNets) inspired by exiting models used in the literature. Encoders and decoders are composed of convolutional layers interlaced with Batch Normalization (BN) and ReLU activations. Spatial information is aggregated using strided convolutions in the encoders, and is recreated in the decoded with nearest neighbor upsampling. The depth of the encoders and decoders depends on the dataset: the encoders range from 6 to 8 layers, decoders between 7 and 13 layers and latent spaces $\mathbf{h}_y$ and $\mathbf{h}_z$ are 80 to 196 units.

Finally, we can note that our model is not really sensitive to the hyperparameters λ, which were set with logical values (depending on the importance of the loss terms) and with validation tests. Also, the adversarial training does not present particular instabilities like can be encountered with GANs, since we did not had

issues during our experiments. The exact details about the data preprocessing, the architectures used and the training information is provided in Appendix C.

4.5.2 Quantitative evaluation

We now study the ability of DualDis to successfully disentangle and linearize the factors of variation of the two information domains (class/identity and attributes) in the latent spaces. To do so, we compare our model to baselines described in Section 4.4 that are reproduced similarly to an ablation study by deactivating components of our model or by making small changes, *cf.* Figure 4.4. We can therefore compare our model to fair reimplementations of the state of the art. We also evaluate variants of the baselines where labels are available for both information domains, like for DualDis. All models we train have the same architecture (in E, E_y, *etc.*) and the same hyperparameters' values whenever those are common between models. Only UAI (Jaiswal et al. 2018) has a slightly different architecture since it requires shallow encoders E_y and E_z. [1]

Evaluation metrics. First, to evaluate the quality of the representations, we measure the *accuracy* of the linear classifiers $\mathbf{W}_y$ and $\mathbf{W}_z$, which indicates both if the information regarding the labels is correctly extracted and if this information has been linearized to allow its manipulation. Second, to evaluate the disentangling of the information domains, we measure the *error rate* (100 - accuracy) of the adversarial classifiers C_y and C_z, which measures how much of the "undesired" information has been removed through disentangling. [2] Of course, because a random classifier would not have an error rate of 100%, this metric cannot reach this value. Finally, since both these metrics increase when they improve on our objective, we summarize the overall quality of a model with an *aggregated metric* that averages those 4 values for a quicker interpretation of the results.

Note that the disentangling metric cannot go higher than a certain value that depends on the dataset. For example, if we want to remove information about a binary attribute $\mathbf{z}$ present for 80% of the examples, with a perfect disentangling, C_z would predict this attribute randomly 80% of the time, would have an accuracy of 80% and thus a disentangling metric of 20%. The best disentangling metric value is the error rate of a random classifier considering the distribution of the dataset. We indicate this value in the results for each dataset.

1. When using this architecture with shallow encoders with the other methods, we obtain the same trends as the ones we report, only with a small degradation to all the models.

2. For models that do not include classifiers C, we add and train them to obtain disentangling metrics, without impacting the behavior of the rest of the model, *cf.* details in the appendix.

		Model	Labels used	Aggr. metric	Accuracy $\mathbf{h}_y \to \mathbf{y}$	Accuracy $\mathbf{h}_z \to \mathbf{z}$	Disentangling $\mathbf{h}_z \to \mathbf{y}_{\text{adv}}$	Disentangling $\mathbf{h}_y \to \mathbf{z}_{\text{adv}}$
CelebA		*Dataset prior*					*99.5%*	*19.5%*
	(A)	Multi-task classif.	$\mathbf{y}$, $\mathbf{z}$	61.1	**77.6%**	**91.8%**	65.5%	9.5%
	(B)	HybridNet-like	$\mathbf{y}$	65.1	73.0%	82.4%	95.5%	9.4%
	(B')	HybridNet-like + attr	$\mathbf{y}$, $\mathbf{z}$	65.2	72.7%	90.1%	88.5%	9.5%
	(C)	MTAN	$\mathbf{y}$, $\mathbf{z}$, $\mathbf{z}_{\text{eval}}$	–	68.9%	–	–	13.8%
	(D)	UAI adv. loss	$\mathbf{y}$	63.7	67.9%	80.3%	**97.3%**	9.3%
	(D')	UAI adv. loss + attr	$\mathbf{y}$, $\mathbf{z}$	65.0	68.0%	89.4%	92.9%	9.5%
	(E)	Adv. on $\mathbf{y}$ only	$\mathbf{y}$	64.7	69.2%	83.6%	96.4%	9.6%
		DualDis	$\mathbf{y}$, $\mathbf{z}$	**68.0**	71.1%	88.6%	**97.3%**	**14.9%**
Yale-B		*Dataset prior*					*97.4%*	*92.9%*
	(A)	Multi-task classif.	$\mathbf{y}$, $\mathbf{z}$	81.5	98.5%	97.2%	85.3%	45.1%
	(B)	HybridNet-like	$\mathbf{y}$	65.3	97.6%	93.7%	23.3%	46.5%
	(B')	HybridNet-like + attr	$\mathbf{y}$, $\mathbf{z}$	80.5	**99.0%**	96.9%	80.0%	46.1%
	(C)	MTAN	$\mathbf{y}$, $\mathbf{z}$, $\mathbf{z}_{\text{eval}}$	–	98.4%	–	–	70.3%
	(D)	UAI adv. loss	$\mathbf{y}$	60.0	98.6%	65.5%	28.1%	48.0%
	(D')	UAI adv. loss + attr	$\mathbf{y}$, $\mathbf{z}$	65.1	96.1%	95.8%	44.4%	24.1%
	(E)	Adv. on $\mathbf{y}$ only	$\mathbf{y}$	79.8	98.3%	84.1%	92.5%	44.4%
		DualDis	$\mathbf{y}$, $\mathbf{z}$	**92.0**	98.6%	**97.3%**	**98.8%**	**73.4%**
NORB		*Dataset prior*					*80.0%*	*31.3%*
	(A)	Multi-task classif.	$\mathbf{y}$, $\mathbf{z}$	53.7	93.0%	84.2%	13.5%	24.0%
	(B)	HybridNet-like	$\mathbf{y}$	51.1	93.3%	76.8%	12.2%	22.1%
	(B')	HybridNet-like + attr	$\mathbf{y}$, $\mathbf{z}$	52.5	92.9%	84.1%	10.7%	22.2%
	(C)	MTAN	$\mathbf{y}$, $\mathbf{z}$, $\mathbf{z}_{\text{eval}}$	–	92.2%	–	–	**30.5%**
	(D)	UAI adv. loss	$\mathbf{y}$	51.8	92.8%	76.0%	13.7%	24.7%
	(D')	UAI adv. loss + attr	$\mathbf{y}$, $\mathbf{z}$	52.5	93.2%	82.8%	8.0%	26.0%
	(E)	Adv. on $\mathbf{y}$ only	$\mathbf{y}$	67.3	92.2%	76.9%	78.9%	21.1%
		DualDis	$\mathbf{y}$, $\mathbf{z}$	**72.3**	**93.5%**	**84.5%**	**80.7%**	**30.5%**

Table 4.2. – **Comparison to state-of-the-art models**. We indicate the labels necessary in train $(\mathbf{y}, \mathbf{z})$ and to use the model $(\mathbf{z}_{\text{eval}})$. We measure the *accuracy* of the classifiers $\mathbf{W}_y(\mathbf{h}_y)$ and $\mathbf{W}_z(\mathbf{h}_z)$ to predict the classes and attributes; and the *disentangling quality* as the error rate (100 - accuracy) of the classifiers $C_y(\mathbf{h}_z)$ and $C_z(\mathbf{h}_y)$ indicating if the information was correctly removed. Our *aggregated metric* is an average of the four scores to indicate the overall performance. For all the scores, *higher is better*. The *dataset prior* scores show the highest disentangling score that we can obtain considering the prior on the distribution of the dataset.

Comparison to the state of the art. Results are presented in Table 4.2 with baselines described in Section 4.4 labeled (A) to (E). DualDis provides the best performances on the three datasets with a gain of 3 to 10 pts in the aggregated metric compared to the best baseline. Overall, we obtain strong disentangling results while having similar or better classification accuracies.

Looking at the baselines, first, the multi-task classifier (A) (Kokkinos 2017) provides good accuracies but completely fails at disentangling the information. This is probably because classification is made easier by not needing to compromise with reconstruction, that is not used in this model, also making generation impossible. Adding reconstruction gives (B & B'), which regularizes the model and slightly helps the disentangling metrics by grounding latent features to specific visual patterns in the reconstruction. In comparison, DualDis provides an explicit disentangling mechanism to greatly improve disentangling metrics.

MTAN (C) (Yang Liu et al. 2018), using a conditional generation approach, produces good results but uses a single latent space $\mathbf{h}_y$ and thus cannot be directly compared to DualDis. In particular, it is complicated to use in a real setup because it always requires labels $\mathbf{z}$ as inputs of its decoder to be used, even in "test" (here meaning the use of the model after training). Besides, this constraint of encoding the attribute information in a binary vector causes lower quality reconstructions because this information cannot be represented in such a compact space.

Finally, we can see that the orthogonalization mechanism applied to latent representations proposed by UAI (D & D') only produces weak disentangling results because of the strong limitations that we discussed. The most competitive disentangling results are provided by the asymmetrical disentangling of (E) (Yu Liu et al. 2018), only applied to $\mathbf{y}$. However, as we discussed, because of this asymmetry, it lacks the ability to correctly remove the attribute information in the identity space, with a disentangling metric on $\mathbf{z}$ that is similar to the results of models that do not have any disentangling mechanism.

In comparison, we combine reconstruction for generation and features regularization; classification and linearization to improve the semantic quality of the features; and leverage symmetrical label-guided adversarial disentangling; and effectively separate the two domains in both latent spaces while having competitive classification results.

4.6 Semi-Supervised Learning

The main drawback of DualDis is that it requires labels for both information domains, which can be expensive to annotate and is rarely the case in existing datasets. As we just saw in the previous section, using labels in both domains is

Nb. attr. labels	Aggr. metric	Accuracy $\mathbf{h}_y \to \mathbf{y}$	Accuracy $\mathbf{h}_z \to \mathbf{z}$	Disentangling $\mathbf{h}_z \to \mathbf{y}_{\text{adv}}$	Disentangling $\mathbf{h}_y \to \mathbf{z}_{\text{adv}}$
400	63.9	65.2%	81.2%	**97.7%**	11.6%
1000	65.5	68.4%	84.3%	97.4%	11.9%
2000	66.8	71.0%	85.0%	**98.4%**	12.7%
4000	67.6	**72.6%**	85.8%	**98.3%**	13.8%
48000	**68.0**	71.1%	**88.6%**	97.3%	**14.9%**

Table 4.3. – **Results of disentangling on CelebA using Semi-Supervised Learning (SSL) on attribute labels**. "48k labels" is the fully supervised baseline.

critical for an effective disentangling. However, ideally, only a few labeled samples would be sufficient to guide the disentangling process of DualDis and provide semantic information on how the information is organized in the latent space. Thus, to strongly minimize the need for annotations, we propose to explore the possibility of using Semi-Supervised Learning (SSL) in DualDis. As we saw in Chapter 3, those methods can efficiently alleviate the need for labeled data with a relatively low cost. This would relax the constraint of having a fully labeled dataset for both domains, making the "cost" of applying this idea much lower.

A first and very simple "SSL" strategy is to just ignore unlabeled samples whenever the label is necessary for a loss term. To go further and really apply state-of-the-art SSL method to DualDis, the most relevant solutions are probably the techniques producing "virtual" targets like Temporal Ensembling (Laine and Aila 2017) or Mean Teacher (Tarvainen and Valpola 2017), or label propagation-based methods (Iscen et al. 2019). Those methods would provide ground-truth-like vectors that could be used as classification targets whenever needed in the various loss terms.

We propose some preliminary experiments to investigate the possibility of obtaining disentangling and linearization of the domains using few attribute labels on CelebA. Using the same architecture as previously, we applied SSL first by simply ignoring unlabeled samples when labels were needed, and second using the more advanced Mean Teacher strategy (Tarvainen and Valpola 2017). We found, however, only a negligible difference in the results between the two. This is likely due to a sub-optimal tuning of the hyper-parameters of Mean Teacher which are very sensitive, as we noticed when working on HybridNet and confirmed by Oliver et al. (2018).

Even with this simple strategy, we obtain interesting results that we report in Table 4.3. Interestingly, we can see that even 1000 or 2000 labels – corresponding to 2 and 4% of the train set size – produce an aggregated score of 65.5 and 66.8,

which is better than the baselines in Table 4.2 and reasonably close to the model trained with full supervision. We also confirmed via a qualitative analysis of the model with 1000 labels that it can produce image edition and we obtained results visually similar to the ones of the fully supervised model in Figure 4.6. Applying SSL to DualDis thus looks like a very promising idea.

4.7 Image Editing and Data Augmentation

We now propose to evaluate two possible applications of this architecture provided by our encoder-decoder architecture and the ability to manipulate its latent representations: image editing and image generation for Data Augmentation (DA) on Yale-B.

4.7.1 Semantic image editing

Using the disentangling and linearization abilities of DualDis, we perform semantic image manipulation by applying linear changes to $\mathbf{h}_z$ along attribute directions $\mathbf{w}_{z_i}$. As explained in Section 4.3.2, we can add or remove an attribute like so:

$$\mathbf{h}'_z = \mathbf{h}_z \pm \varepsilon \mathbf{w}_{z_i}^\top . \tag{4.16}$$

In Figure 4.5, we show the visual effect of moving in positive and negative directions with different magnitudes for different attributes. This shows that we can finely control the importance of an attribute, increasing or decreasing its original presence on the image, until the attribute can eventually be considered removed or added. For example, we can add a small or big smile, choose among different shades of blonde hair, *etc*.

For the next visualization in Figure 4.6, we fix a reasonable magnitude threshold for when an attribute is considered as changed (*i.e.* added / removed). We then apply the method used above with this magnitude, in a direction chosen to flip the ground truth labels of different images for different attributes. Using this, we show that we can switch the gender, add and remove eyeglasses, hair bangs, *etc*. A similar protocol was also used on Yale-B and NORB as shown in Figure 4.7. On those datasets though, because of the small size and variability in the datasets, even if the effect mostly remains, it is less clear than on CelebA.

Overall, these two visualizations show that we effectively modeled and linearized attribute information in $\mathbf{h}_z$. In order to confirm that we also effectively separated the two domains, we propose another protocol on CelebA, where we mix representations of two images and see the resulting reconstruction.

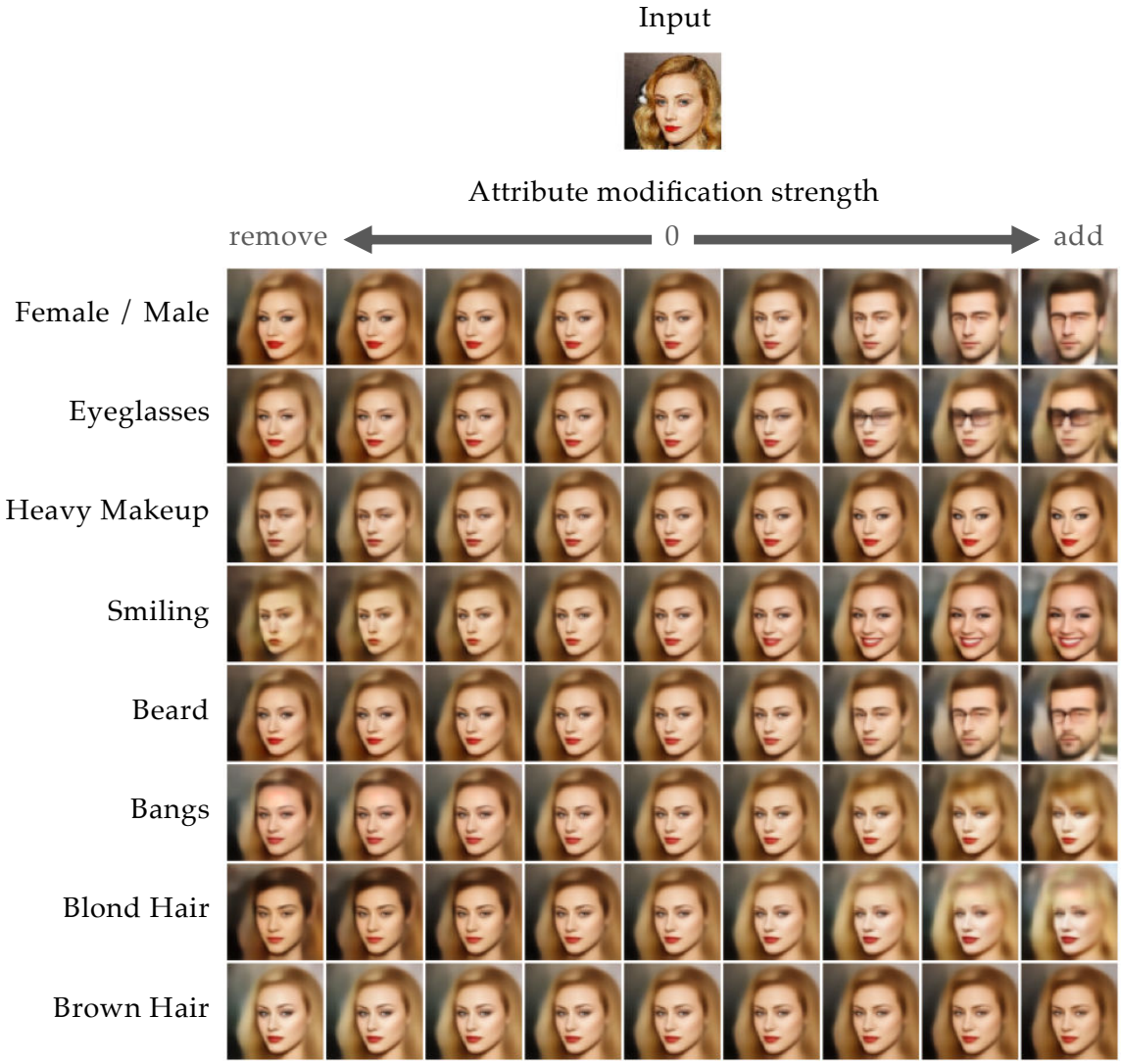


Figure 4.5. – **Visualizations of progressive attribute editing on CelebA**. We do so by moving $\mathbf{h}_z$ with different strength ε and in positive and negative directions along each vector $\mathbf{w}_{z_i}$.

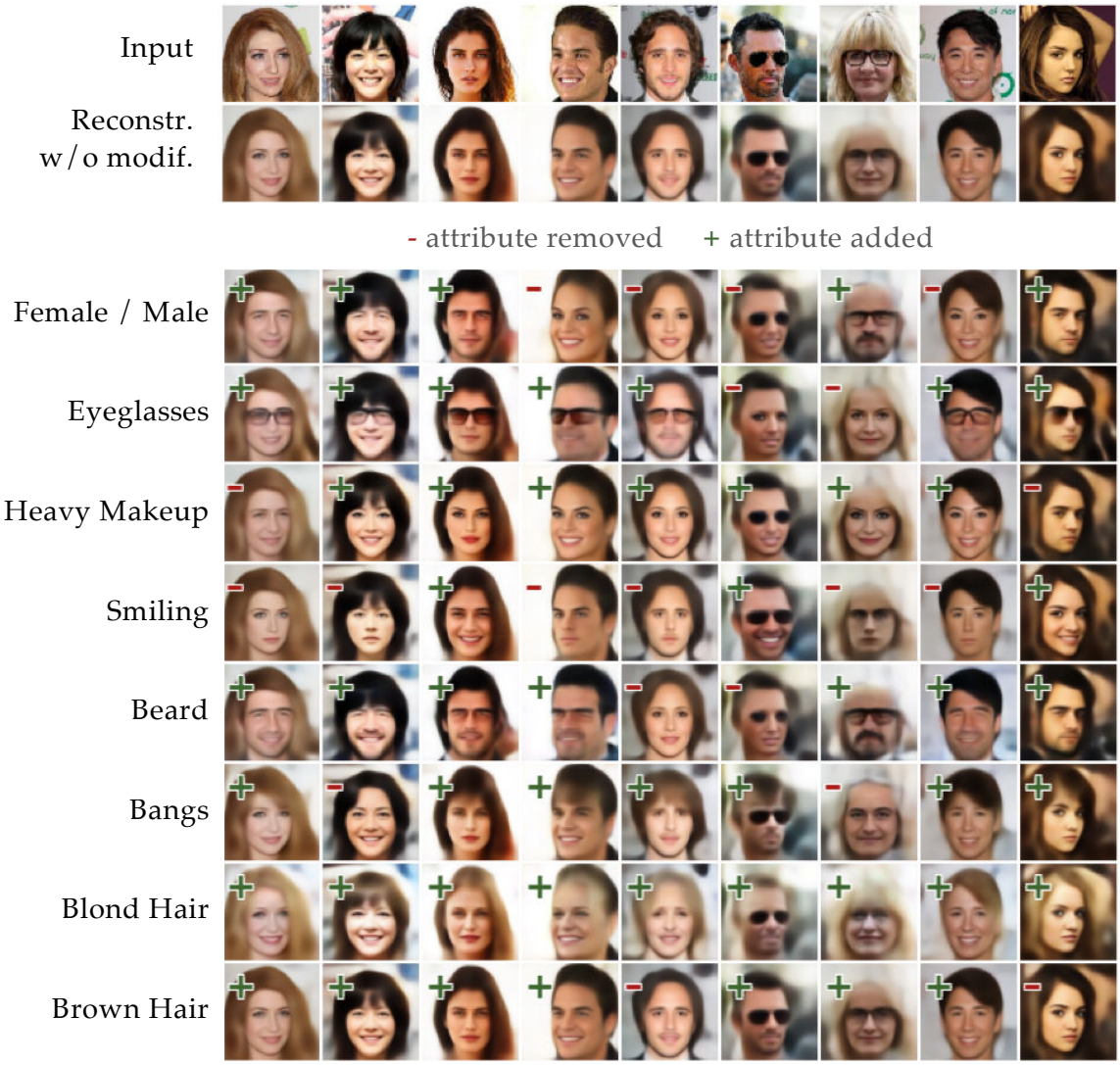
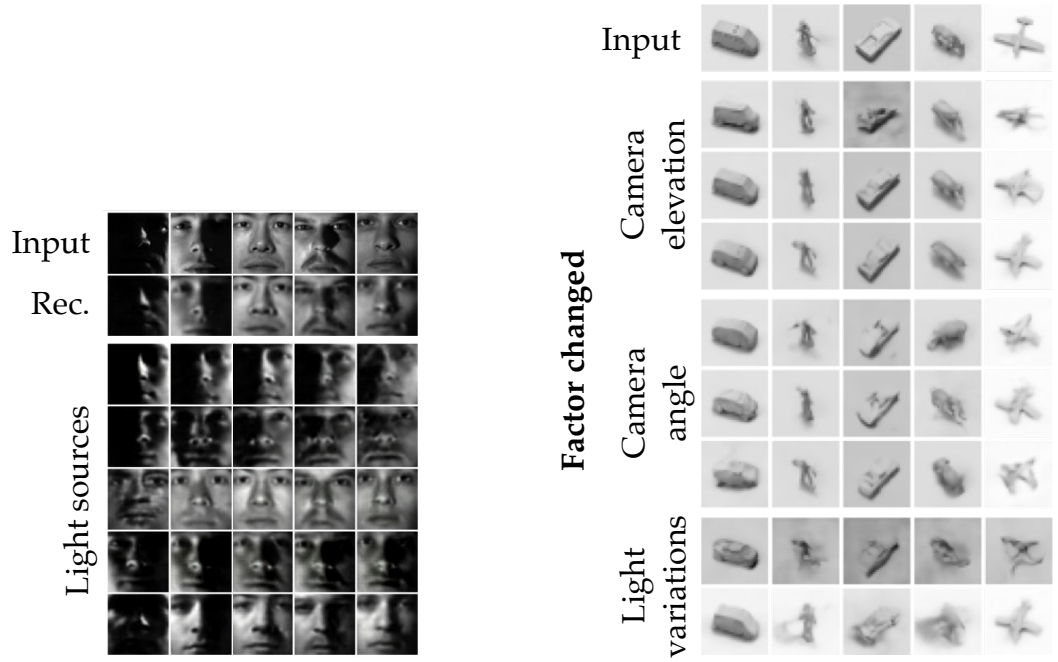


Figure 4.6. – **Visualizations of attribute inversion on CelebA**. For each image k and each attribute i, we modify $\mathbf{h}_z^{(k)}$ along the vector $\mathbf{w}_{z_i}$ in the direction opposite to the ground truth label.



(a) **Yale-B**. Our attributes correspond to the source of the light (from far right to far left when going down).

(b) **NORB**. We change the camera elevation (from close to the ground to a view from above), the camera angle around the object and the intensity of light.

Figure 4.7. – **Visualizations of image editing on Yale-B and NORB**.

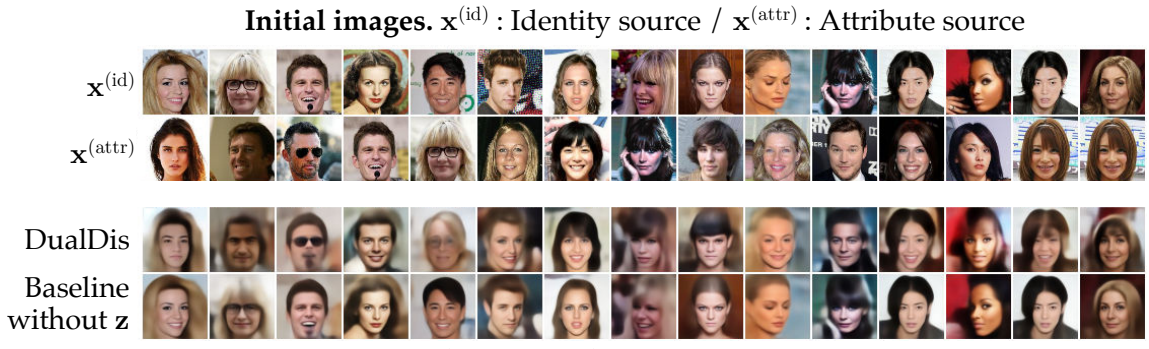


Figure 4.8. – **Visualizations of the information separation through attribute and identity mix**. We show reconstructions using identity $\mathbf{h}_y$ from an image $\mathbf{x}^{(\text{id})}$ and attributes $\mathbf{h}_z$ from an image $\mathbf{x}^{(\text{attr})}$. We do this with our model and a baseline that do not use attribute labels for training (Yu Liu et al. 2018). We see that our model does use attributes from image $\mathbf{x}^{(\text{attr})}$ while keeping the identity of image $\mathbf{x}^{(\text{id})}$, whereas the baseline did not manage to remove attribute information from $\mathbf{h}_y$ and still uses attributes of image $\mathbf{x}^{(\text{id})}$.

More precisely, we propose to use $\mathbf{h}_y$ from a first image noted $\mathbf{x}^{(\text{id})}$ and $\mathbf{h}_z$ from a second image $\mathbf{x}^{(\text{attr})}$, and reconstruct an image from those. We do this for DualDis and the baseline not using attributes (E) (Hadad et al. 2018; Yu Liu et al. 2018). Results are presented in Figure 4.8 and show that our model is able to mix the identity of the first image and the attributes of the second, while the baseline could not correctly separate the two types of information and uses attributes from the first image. For example, for the first pair, we keep the original identity but change the smile and hair color; for the second, we change the gender and glasses; for the third, we add glasses, *etc*. This confirms that our symmetrical disentangling on both domains manage to separate the two types of information that otherwise remain entangled for a complex dataset like CelebA.

4.7.2 Image generation for Data Augmentation

Finally, we propose to use DualDis for semantic guided Data Augmentation (DA). Classical DA usually consists in producing variants of an existing image by applying low-level changes to it, like mirroring, translation, changes in the color space, *etc*. While efficient as a regularizer, this approach is limited since no new semantic information is added. Using DualDis, we are able to manipulate the semantic content of an image, making it possible to, for example, change the attributes $\mathbf{z}$ of an image while keeping the class $\mathbf{y}$ fixed, which produces a new image for the class $\mathbf{y}$ with new semantic information.

We propose to apply this method to Yale-B since is particularly well adapted to illustrate this idea. Indeed, Yale-B contains different lighting for each identity. So if we use a restricted number N_{init} of images in train, each class contains only a small portion of the possible lighting variations for each identity. Using DualDis, we propose to generate the missing variants.

To do so, we first train a DualDis model with N_{init} training images. Based on the linearization and editing properties of our model, we generate variations in attributes $\mathbf{z}$ (similarly to Figure 4.7a) for each training image to obtain new images with the same identity but a new and known attribute label $\mathbf{z}'$. For each identity, we generate N_{gen} new images, obtaining images with attributes missing in the original train set as well as increasing the number of instances of existing attributes for robustness. This provides a more representative dataset of the variations in attributes for each identity.

We apply this procedure for different values of N_{init} and N_{gen}. We then learn a classifier C that has the architecture $C = \mathbf{W}_y \circ E_y \circ E$ on a train set that contains original and generated images and evaluates its accuracy on the test set. The results are reported in Table 4.4. Adding generated images to the train set with

Initial	Nb. generated images per class				
train size	0	10	20	30	60
480	78.9%	79.3%	80.1%	81.6%	**82.8%**
360	69.1%	70.5%	72.6%	73.1%	**75.6%**
240	48.9%	51.8%	55.5%	56.8%	**58.6%**

Table 4.4. – **Accuracy of identity prediction on Yale-B using generated images as Data Augmentation (DA).**

new attribute variations for each identity provides a gain in our 3 setups. When the initial train set is small, the gain is larger, with a gain of almost 10 pts between the baseline without DA and a DA of 60 images per class. Adding more than 60 images per class does not yield a significant improvement.

4.8 Conclusion

In this chapter, we presented DualDis, a disentangling model designed to effectively separate two information domains using a two-branch architecture, one branch for each domain. This is made possible by the use of classifiers and adversarial training in order to organize the information and guide the training process. Thanks to this, we obtain improved classification and disentangling results on CelebA, Yale-B and NORB while linearizing and modeling semantic factors of variation. Using our structured latent space, we perform image editing, making it possible to change the attributes of an image. We also carry out semantic Data Augmentation (DA) on Yale-B and obtain important identity classification improvements.

For future work, an interesting direction would be to go further on the application of Semi-Supervised Learning (SSL) to DualDis. Our preliminary experiments on this subject indeed show interesting results, but further investigations on applying efficient state-of-the-art SSL methods could provide much better results. In addition, it would be interesting to bring this model closer to the techniques used in generative models. This could provide better quality generations which in turn could strongly improve the results in semantic Data Augmentation (DA). Another interesting idea would be to allow our model to generate new images from noise, completely bridging the gap with generative models and enabling both image generation and latent space manipulations.

CHAPTER 5

CONCLUSION

We first summarize the contributions that we propose in this thesis before discussing interesting directions for future work.

5.1 Summary of Contributions

In this thesis, we work on improving the quality of the latent spaces of deep Convolutional Neural Networks (ConvNets) in three different contexts: regularization, Semi-Supervised Learning (SSL) and disentangling.

Regularizing through intra-class invariance. Because powerful ConvNets are easily subject to overfitting, we propose SHADE in Chapter 2, a novel regularization method that encourages intra-class invariance in the representations. While the idea of adding invariance is not new, the novelty of our contribution lies in the way to enforce it. Indeed, we show that it is possible to use conditional entropy $\mathcal{H}(H \mid Y)$ as a measure of the intra-class invariance, which does not limit the types of variance that are targeted. We then derive a tractable and differentiable criterion from it. Applying this SHADE criterion to many standard Deep Neural Network (DNN) architectures, we are able to outperform other common forms of regularization and improve classification results on CIFAR-10 and ImageNet. We also demonstrate that SHADE has the ability to significantly outperform the baseline on small datasets. This first contribution is a step in the direction of making ConvNets usable in more practical cases, that we continue to explore in the following chapter with SSL.

Information separation for Semi-Supervised Learning (SSL). While SHADE address the problem of scarce labeled data when overfitting becomes a preponderant issue, we then propose to use additional inexpensive unlabeled data to further make those models usable in more cases. We thus address Semi-Supervised Learning (SSL) in Chapter 3 to greatly improve the quality of a ConvNet. In this context, we propose a two-branch architecture called HybridNet, accompanied by an adapted training loss, that separates the information in two complemen-

tary latent spaces. We show that this novel idea was able to efficiently address a conflict between classification and reconstruction that is detrimental to their cooperation, which is not the case with HybridNet. We demonstrate that HybridNet is able to integrate existing stability techniques and produce state-of-the-art results with large ResNet architectures on standard SSL datasets, obtaining impressive performances with very few labeled samples. HybridNet is thus an interesting idea to make it possible to train large ConvNets in contexts where labeled data is particularly expensive such as the medical domain.

Disentangling of two information domains. Finally, pursuing our idea of separating the information in complementary latent spaces, we also address the problem of producing highly semantic disentangled representations in Chapter 4. To that end, we propose DualDis, an approach that allows to effectively separate and linearize two labeled and complementary information domains in two latent spaces. This study allows to underline some limitations of approaches using asymmetrical labeling of the domains (*i.e.* only one information domain having labels) and shows that DualDis is able to produce interesting domain-specific representations that could then be used for other tasks such as information retrieval. Using the linearization properties of DualDis, it is possible to control the presence of the semantic attributes in the representation of an image, and thus perform semantic image editing. We show that this property can be leveraged for semantic data augmentation, producing new semantic variations of existing images.

5.2 Perspectives for Future Work

Let us now discuss interesting directions that could be addressed in future work in relation to our contributions.

Handling data from different domains with domain adaptation

In this thesis, we had a particular focus on making ConvNets usable in more practical cases by working on regularization and SSL, reducing the need for large and expensive datasets. This goal can be pushed further by designing models that can handle data coming from different domains, which is called *domain adaptation*.

With *unsupervised* domain adaptation (Ben-David et al. 2010), the goal is to transfer knowledge from a labeled source dataset to an unlabeled target dataset of interest, both sharing the same labels but having different image distributions. The goal being to leverage the knowledge of an existing or inexpensive (*e.g.*

synthetic) source dataset instead of getting expensive annotations for the target dataset. This problem is of particular interest nowadays for applications such as autonomous driving, where each city and country have different styles of cars, roadsigns, architecture, weather, *etc*. To address domain adaptation, ideas from both HybridNet and DualDis could be used.

Domain adaptation through information separation. Indeed, a first possible solution for domain adaptation is to remove the information that is domain dependent and keep only the domain independent information. There is thus a clear link with the separation of discriminative and non-discriminative information in HybridNet. A solution in this direction has been proposed by Bousmalis et al. (2016), using three encoders, one that is common to both domains and should extract domain invariant features (used for classification), and one for each domain for domain specific features. The features are then merged and decoded for reconstruction. This shows that the idea of HybridNet to produce complementary reconstructions could be exploited further for domain adaptation. In addition to HybridNet, it would mostly require to design techniques to ensure the structuring of domain specific and domain agnostic information in the latent spaces, through adversarial training for example, and work on the architecture and merge strategy for this specific context.

Domain adaptation through generation. Another recent approach of domain adaptation consists in transforming the source images to target-like images, effectively removing the *domain shift*. This allows to obtain a classifier adapted to the target domain, trained on those labeled target-like images. Following this idea, an extension of DualDis could help, offering the possibility of editing the image from one domain to the other. For example, Chang et al. (2019) recently proposed to disentangle domain invariant and domain specific features in order to perform this image editing. This approach is based on assumptions that the changes must be performed at the texture level without changing the structure of the image, which is not always the case. An extension of DualDis using some labels about domain specific factors could for example be developed to follow this idea.

Bridging the gap between discriminative and generative models

An important motivation during this thesis was to develop models that could bridge the gap between discriminative and generative models. One the one hand, discriminative models aim at extracting discriminative and relevant patterns for a semantic task $\mathbf{y}$ from input data $\mathbf{x}$ (*i.e.* model $p(\mathbf{y} \mid \mathbf{x})$). On the other hand, recent generative models try to model the distribution of the data with no or small rela-

tion with semantic information (*i.e.* model $p(\mathbf{x})$ or $p(\mathbf{x}, \mathbf{y})$). With HybridNet and DualDis, we proposed models that are able to, at the same time, model all the information about the data and provide powerful discriminative and semantic features. However, some work still needs to be done to reach this goal. The objective would be to have a model that transforms an input $\mathbf{x}$ into a representation $\mathbf{h}$ in which many semantic factors $\mathbf{y}$ can be analyzed, interpreted, manipulated; that can be decoded back into an image $\hat{\mathbf{x}}$; and that also allows to generate new images $\tilde{\mathbf{x}}$ from scratch with fine control over the semantic factors. We propose to go over some possible directions for this.

Semi-Supervised Learning (SSL) of semantic factors. As studied in DualDis, we show that using labels related to semantic factors is important and has a great advantage over fully unsupervised models. However, to actually make those models usable in a large variety of cases, we believe that integrating SSL in such models is a key direction. Indeed, the ideal situation would be to have a model that can learn to capture semantic factors of a dataset with only a few examples of each factor. This way, real world datasets that usually have many semantic factors could be semantically represented by this hybrid discriminative/generative model at a reasonable labeling expense. We showed preliminary results using SSL in DualDis and we believe this should be further explored to make such a model viable.

Modeling factors diversity in separate subspaces. We also believe that it would be important to model the internal diversity of each semantic factor. Indeed, simple modelings of semantic factors consider they are represented by a binary value (*e.g.* Perarnau et al. 2016) or a real value (*e.g.* Lample et al. 2017). Many semantic factors however have a large variability (*e.g.* all the possible hairstyles, makeup styles, *etc.* that exist) that cannot be realistically represented in such limited spaces. With DualDis, we relieve this constraint and simply require that they be linearly separable, however, we cannot explore their internal variability. Klys et al. (2018) proposes the idea of representing each factors in delimited latent subspaces, but only show limited experiments on this idea. We thus believe this kind of work should be pursued.

Semantic generation and reversible architectures. Finally, the question of finding models that are both able to represent the information and produce sharp reconstructions and generations remains open. In DualDis, we did not address this problem, however, one goal of a good discriminative and generative model would be to eventually produce clear images (*e.g.* Karras et al. 2019). To achieve this, apart from the usual GAN and perceptual losses that are known to help in this matter, reversible models (Gomez et al. 2017; Jacobsen et al. 2018) are also an

interesting direction to address such an issue. Those models are, by design, able to encode and decode an image without loss of information. However, because of the particular layers they use to make this possible, manipulation in the latent information is made much more difficult. Some recent research shows promising results in this direction, such as Kingma and Dhariwal (2018) and Lucas et al. (2019). We believe this idea should be pursued and could be an important part of this goal of building a powerful model that could address both discriminative and generative tasks.

BIBLIOGRAPHY

Achille, A. and S. Soatto (2016). "Information Dropout: learning optimal representations through noisy computation". In: *arXiv preprint libary* (cit. on pp. 23, 32).

Alain, Guillaume and Yoshua Bengio (2014). "What regularized auto-encoders learn from the data-generating distribution". In: *Journal of Machine Learning Research (JMLR)* (cit. on p. 25).

Alemi, Alexander (2016). *Improving Inception and Image Classification in TensorFlow*. URL: https://ai.googleblog.com/2016/08/improving-inception-and-image.html (cit. on p. 17).

Alemi, Alexander, Ian Fischer, Joshua V. Dillon, and Kevin Murphy (2017). "Deep Variational Information Bottleneck". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 23, 32).

An, Guozhong (1996). "The effects of adding noise during backpropagation training on a generalization performance". In: *Neural computation* (cit. on p. 20).

Bang, Duhyeon and Hyunjung Shim (2018). "Mggan: Solving mode collapse using manifold guided training". In: *arXiv preprint libary* (cit. on p. 82).

Ben-David, Shai, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan (2010). "A theory of learning from different domains". In: *Machine learning* (cit. on p. 106).

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (2013). "Representation learning: A review and new perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (cit. on p. 15).

Bengio, Yoshua, Pascal Lamblin, Dan Popovici, and Hugo Larochelle (2007). "Greedy layer-wise training of deep networks". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 23, 45).

Ben-Younes, Hedi, Rémi Cadène, Nicolas Thome, and Matthieu Cord (2017). "MUTAN: Multimodal Tucker Fusion for Visual Question Answering". In: *IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 4, 53).

Blau, Yochai and Tomer Michaeli (2018). "The perception-distortion tradeoff". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 81).

Blot, Michael (2018). "Study on training methods and generalization performance of deep learning for image classification". PhD thesis. Sorbonne Université (cit. on pp. 11, 13).

Blot, Michael, Thomas Robert, Nicolas Thome, and Matthieu Cord (2018a). "SHADE: Information Based Regularization for Deep Learning - Extended version". In: *arXiv preprint libary* (cit. on p. 31).

Blot, Michael, Thomas Robert, Nicolas Thome, and Matthieu Cord (2018b). "SHADE: Information-Based Regularization for Deep Learning". In: *IEEE International Conference on Image Processing (ICIP)* (cit. on pp. 9, 11).

Blum, Avrim and Tom Mitchell (1998). "Combining labeled and unlabeled data with co-training". In: *Annual Conference on Computational Learning Theory* (cit. on p. 24).

Bodla, Navaneeth, Gang Hua, and Rama Chellappa (2018). "Semi-supervised FusedGAN for Conditional Image Generation". In: *European Conference on Computer Vision (ECCV)* (cit. on pp. 26, 49, 82).

Bottou, Léon (2010). "Large-scale machine learning with stochastic gradient descent". In: *COMPSTAT*. Springer (cit. on p. 15).

Bousmalis, Konstantinos, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan (2016). "Domain separation networks". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 107).

Brock, Andrew, Jeff Donahue, and Karen Simonyan (2019). "Large scale gan training for high fidelity natural image synthesis". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 82).

Bruna, Joan and Stephane Mallat (2013). "Invariant Scattering Convolution Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (cit. on p. 21).

Carvalho, Micael, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord (2018). "Cross-Modal Retrieval in the Cooking Context: Learning Semantic Text-Image Embeddings". In: *Special Interest Group on Information Retrieval (SIGIR)* (cit. on p. 44).

Chang, Wei-Lun, Hui-Po Wang, Wen-Hsiao Peng, and Wei-Chen Chiu (2019). "All about Structure: Adapting Structural Information across Domains for Boosting Semantic Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 77, 107).

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille (2017). "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (cit. on p. 4).

Chen, Tian Qi, Xuechen Li, Roger B Grosse, and David K Duvenaud (2018). "Isolating sources of disentanglement in variational autoencoders". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 7, 28, 83).

Chen, Xi, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel (2016). "InfoGAN: Interpretable Representation Learning by Informa-

tion Maximizing Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 129).

Cheung, Brian, Jesse A Livezey, Arjun K Bansal, and Bruno A Olshausen (2015). "Discovering hidden factors of variation in deep networks". In: *International Conference on Machine Learning Workshop (ICML-W)* (cit. on p. 27).

Choi, Yunjey, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo (2018). "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 81).

Cireşan, Dan, Ueli Meier, and Jürgen Schmidhuber (2012). "Multi-column deep neural networks for image classification". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 20).

Coates, Adam, Andrew Ng, and Honglak Lee (2011). "An analysis of single-layer networks in unsupervised feature learning". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)* (cit. on p. 60).

Cohen, Taco and Max Welling (2016). "Group equivariant convolutional networks". In: *International Conference on Machine Learning (ICML)* (cit. on p. 21).

Cover, T. and J. Thomas (1991). "Elements of information theory". In: *Wiley New York* (cit. on pp. 29, 31, 34, 128, 129).

Dai, Jifeng, Yi Li, Kaiming He, and Jian Sun (2016). "R-FCN: Object detection via region-based fully convolutional networks". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 44).

Denton, Emily, Sam Gross, and Rob Fergus (2017). "Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 26, 49).

DeVries, Terrance and Graham W Taylor (2017). "Dataset augmentation in feature space". In: *International Conference on Learning Representations Workshop (ICLR-W)* (cit. on p. 20).

Dieleman, Sander, Kyle W Willett, and Joni Dambre (2015). "Rotation-invariant convolutional neural networks for galaxy morphology prediction". In: *Monthly notices of the royal astronomical society* (cit. on pp. 21, 28).

Donahue, Jeff, Philipp Krähenbühl, and Trevor Darrell (2017). "Adversarial Feature Learning". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 82).

Dosovitskiy, Alexey and Thomas Brox (2016). "Generating images with perceptual similarity metrics based on deep networks". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 87).

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research (JMLR)* (cit. on p. 15).

Dumoulin, Vincent, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville (2017). "Adversarially Learned Inference". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 82).

Dumoulin, Vincent and Francesco Visin (2016). "A guide to convolution arithmetic for deep learning". In: *Technical Report* (cit. on pp. 50, 55, 65).

Dupont, Emilien (2018). "Learning disentangled joint continuous and discrete representations". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 83).

Durand, Thibaut (2017). *Deep Architectures in LaTeX*. URL: https://github.com/durandtibo/deep_archi_latex (cit. on p. 16).

Durand, Thibaut, Nazanin Mehrasa, and Greg Mori (2019). "Learning a Deep ConvNet for Multi-label Classification with Partial Labels". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 24).

Durand, Thibaut, Nicolas Thome, and Matthieu Cord (2016). "WELDON: Weakly Supervised Learning of Deep Convolutional Neural Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 37, 38).

Dyk, David A. van and Xiao-Li Meng (2001). "The Art of Data Augmentation". In: *Journal of Computational and Graphical Statistics* (cit. on pp. 5, 20).

Engel, Jesse, Matthew Hoffman, and Adam Roberts (2018). "Latent Constraints: Learning to Generate Conditionally from Unconditional Generative Models". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 82, 83).

Engilberge, Martin, Louis Chevallier, Patrick Pérez, and Matthieu Cord (2018). "Finding beans in burgers: Deep semantic-visual embedding with localization". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 44).

Erhan, Dumitru, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio (2010). "Why does unsupervised pre-training help deep learning?" In: *Journal of Machine Learning Research (JMLR)* (cit. on p. 23).

Fukushima, K. (1980). "Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position". In: *Biological Cybernetics* (cit. on p. 4).

Ganin, Yaroslav and Victor Lempitsky (2015). "Unsupervised Domain Adaptation by Backpropagation". In: *International Conference on Machine Learning (ICML)* (cit. on p. 38).

Ganin, Yaroslav, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky (2016). "Domain-Adversarial Training of Neural Networks". In: *Journal of Machine Learning Research (JMLR)* (cit. on p. 60).

Gastaldi, Xavier (2017). "Shake-Shake regularization of 3-branch residual networks". In: *International Conference on Learning Representations Workshop (ICLR-W)* (cit. on pp. 70, 71, 141).

Georghiades, A.S., P.N. Belhumeur, and D.J. Kriegman (2001). "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (cit. on p. 93).

Ghiasi, Golnaz, Tsung-Yi Lin, and Quoc V Le (2018). "DropBlock: A regularization method for convolutional networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 21).

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Deep sparse rectifier neural networks". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)* (cit. on p. 25).

Goh, Hanlin, Nicolas Thome, Matthieu Cord, and Joo-Hwee Lim (2013). "Top-down regularization of deep belief networks". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 25).

Gomez, Aidan N, Mengye Ren, Raquel Urtasun, and Roger B Grosse (2017). "The Reversible Residual Network: Backpropagation Without Storing Activations". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 50, 51, 108).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press (cit. on pp. 19, 21).

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 9, 25, 27, 81, 87).

Goodfellow, Ian, Jonathon Shlens, and Christian Szegedy (2015). "Explaining and harnessing adversarial examples". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 20, 26).

Grandvalet, Yves and Yoshua Bengio (2005). "Semi-supervised learning by entropy minimization". In: *Advances in neural information processing systems*, pp. 529–536 (cit. on p. 24).

Grandvalet, Yves, Stéphane Canu, and Stéphane Boucheron (1997). "Noise injection: Theoretical prospects". In: *Neural Computation* (cit. on p. 20).

Hadad, Naama, Lior Wolf, and Moni Shahar (2018). "A Two-Step Disentanglement Method". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 7, 84, 86, 87, 91, 102).

Hale, James Loke (2019). *More Than 500 Hours Of Content Are Now Being Uploaded To YouTube Every Minute*. URL: https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/ (cit. on p. 1).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 4, 12, 17, 18, 31, 32, 37, 44).

He, Zhenliang, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen (2019). "Attgan: Facial attribute editing by only changing what you want". In: *IEEE Transactions on Image Processing (TIP)* (cit. on p. 82).

Higgins, Irina, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner (2018). "Towards a Definition of Disentangled Representations". In: *arXiv preprint libary* (cit. on pp. 6, 27, 76, 83).

Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2017). "beta-vae: Learning basic visual concepts with a constrained variational framework". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 7, 28, 83).

Hinton, Geoffrey and R R Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks". In: *Science* (cit. on pp. 23, 46).

Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky (2012). *Neural networks for machine learning lecture 6a overview of mini-batch gradient descent* (cit. on p. 15).

Hu, Qiyang, Attila Szabó, Tiziano Portenier, Paolo Favaro, and Matthias Zwicker (2018). "Disentangling Factors of Variation by Mixing Them". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 84).

Hubel, David H and Torsten N Wiesel (1962). "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of physiology* (cit. on p. 16).

Ioffe, Sergey and Christian Szegedy (2016). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Journal of Machine Learning Research (JMLR)* (cit. on pp. 4, 5, 22).

Iscen, Ahmet, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum (2019). "Label Propagation for Deep Semi-supervised Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 24, 97).

Jacobsen, Jörn-Henrik, Arnold Smeulders, and Edouard Oyallon (2018). "i-RevNet: Deep Invertible Networks". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 50, 51, 108).

Jaiswal, Ayush, Rex Yue Wu, Wael Abd-Almageed, and Prem Natarajan (2018). "Unsupervised Adversarial Invariance". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 27, 84, 86, 91, 94).

Kang, Guoliang, Jun Li, and Dacheng Tao (2016). "Shakeout: A new regularized deep neural network training scheme". In: *Conference on Artificial Intelligence (AAAI)* (cit. on p. 20).

Karras, Tero, Timo Aila, Samuli Laine, and Jaakko Lehtinen (2017). "Progressive growing of gans for improved quality, stability, and variation". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 81).

Karras, Tero, Samuli Laine, and Timo Aila (2019). "A style-based generator architecture for generative adversarial networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 77, 81, 108).

Kawaguchi, Kenji (2016). "Deep learning without poor local minima". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 5).

Kim, Hyunjik and Andriy Mnih (2018). "Disentangling by factorising". In: *arXiv preprint libary* (cit. on p. 83).

Kingma, Diederik P and Jimmy Ba (2015). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 15).

Kingma, Diederik P and Prafulla Dhariwal (2018). "Glow: Generative flow with invertible 1x1 convolutions". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 109).

Kingma, Diederik P, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling (2014). "Semi-supervised learning with deep generative models". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 26, 49).

Kingma, Diederik P and Max Welling (2013). "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 25, 80, 129).

Klys, Jack, Jake Snell, and Richard Zemel (2018). "Learning Latent Subspaces in Variational Autoencoders". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 7, 84, 86, 91, 108).

Kodali, Naveen, Jacob Abernethy, James Hays, and Zsolt Kira (2017). "On convergence and stability of gans". In: *arXiv preprint libary* (cit. on p. 81).

Kokkinos, Iasonas (2017). "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 86, 92, 96).

Krizhevsky, Alex and Geoffrey Hinton (2009). "Learning multiple layers of features from tiny images". In: *Technical Report* (cit. on pp. 36, 60).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 4, 15–17, 20).

Krogh, Anders and John A. Hertz (1992). "A Simple Weight Decay Can Improve Generalization". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 5, 22).

Kukačka, Jan, Vladimir Golkov, and Daniel Cremers (2017). "Regularization for deep learning: A taxonomy". In: *arXiv preprint libary* (cit. on pp. 5, 19, 20).

Kulkarni, Tejas D, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum (2015). "Deep Convolutional Inverse Graphics Network". In: *Advances in Neu-*

ral Information Processing Systems (NIPS). Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (cit. on p. 84).

Laine, Samuli and Timo Aila (2017). "Temporal Ensembling for Semi-Supervised Learning". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 6, 26, 45, 48, 59, 66, 71, 72, 97, 140).

Lample, Guillaume, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato (2017). "Fader Networks:Manipulating Images by Sliding Attributes". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 7, 27, 84, 91, 108).

Larochelle, Hugo and Yoshua Bengio (2008). "Classification using discriminative restricted Boltzmann machines". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 25, 26).

Le, Lei, Andrew Patterson, and Martha White (2018). "Supervised autoencoders: Improving generalization performance with unsupervised regularizers". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 46, 87).

LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel (1989). "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* (cit. on p. 4).

LeCun, Yann, Leon Bottou, Yoshua Bengio, and P. Haffner (1998). "Gradient based learning applied to document recognition". In: *Proceedings of the IEEE* (cit. on pp. 4, 16, 60).

LeCun, Yann, Fu Jie Huang, Leon Bottou, et al. (2004). "Learning methods for generic object recognition with invariance to pose and lighting". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Citeseer (cit. on p. 93).

Ledig, Christian, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. (2017). "Photo-realistic single image super-resolution using a generative adversarial network". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 81).

Lee, Dong-Hyun (2013). "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks". In: *International Conference on Machine Learning Workshop (ICML-W)* (cit. on p. 24).

Lei Ba, Jimmy, Jamie Ryan Kiros, and Geoffrey Hinton (2016). "Layer normalization". In: *arXiv preprint libary* (cit. on p. 22).

Li, Yinan and Fang Liu (2016). "Whiteout: Gaussian Adaptive Noise Regularization in FeedForward Neural Networks". In: *arXiv preprint libary* (cit. on p. 20).

Liu, Yang, Zhaowen Wang, Hailin Jin, and Ian Wassell (2018). "Multi-Task Adversarial Network for Disentangled Feature Learning". In: *IEEE Conference on*

Computer Vision and Pattern Recognition (CVPR) (cit. on pp. 27, 84, 86, 87, 91, 96).

Liu, Yu, Fangyin Wei, Jing Shao, Lu Sheng, Junjie Yan, and Xiaogang Wang (2018). "Exploring Disentangled Feature Representation Beyond Face Identification". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 7, 27, 84, 86, 91, 96, 101, 102).

Liu, Ziwei, Ping Luo, Xiaogang Wang, and Xiaoou Tang (2015). "Deep Learning Face Attributes in the Wild". In: *IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 92).

Louizos, Christos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel (2016). "The variational fair autoencoder". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 77).

Lu, Zhou, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang (2017). "The expressive power of neural networks: A view from the width". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 18).

Lucas, Thomas, Konstantin Shmelkov, Karteek Alahari, Cordelia Schmid, and Jakob Verbeek (2019). "Adversarial training of partially invertible variational autoencoders". In: *arXiv preprint libary* (cit. on p. 109).

Luo, Wenjie, Yujia Li, Raquel Urtasun, and Richard Zemel (2016). "Understanding the effective receptive field in deep convolutional neural networks". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 16).

Luvizon, Diogo C, Hedi Tabia, and David Picard (2017). "Human pose regression by combining indirect part detection and contextual information". In: *arXiv preprint libary* (cit. on p. 54).

Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow (2016). "Adversarial Autoencoders". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 82).

Mallat, Stephane (2012). "Group Invariant Scattering". In: *Communications on Pure and Applied Mathematics (CPAM)* (cit. on p. 21).

Mallat, Stephane and Gabriel Peyré (2009). *A wavelet tour of signal processing : the sparse way*. Academic Press (cit. on p. 52).

Mathieu, Michael F, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun (2016). "Disentangling factors of variation in deep representation using adversarial training". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 7, 27, 77, 84, 87).

McCulloch, Warren S and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* (cit. on p. 15).

Mehr, Éloi, André Lieutier, Fernando Sanchez Bermudez, Vincent Guitteny, Nicolas Thome, and Matthieu Cord (2018). "Manifold Learning in Quotient Spaces". In: (cit. on p. 21).

Miyato, Takeru, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii (2016). "Distributional smoothing with virtual adversarial training". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 26).

Neelakantan, Arvind, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens (2016). "Adding gradient noise improves learning for very deep networks". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 20).

Netzer, Yuval, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng (2011). "Reading digits in natural images with unsupervised feature learning". In: *Advances in Neural Information Processing Systems Workshop (NIPS-W)* (cit. on p. 60).

Noh, Hyeonwoo, Tackgeun You, Jonghwan Mun, and Bohyung Han (2017). "Regularizing deep neural networks by noise: Its interpretation and optimization". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 20).

Noyes, Dan (2019). *The Top 20 Valuable Facebook Statistics*. URL: https://zephoria.com/top-15-valuable-facebook-statistics/ (cit. on p. 1).

Nwankpa, Chigozie, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall (2018). "Activation functions: Comparison of trends in practice and research for deep learning". In: *arXiv preprint libary* (cit. on p. 15).

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). "Feature Visualization". In: *Distill* (cit. on pp. 3, 4, 16).

Oliver, Avital, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow (2018). "Realistic evaluation of deep semi-supervised learning algorithms". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 97).

Paninski, Liam (2003). "Estimation of Entropy and Mutual Information". In: *Neural Computation* (cit. on p. 129).

Paumard, Marie-Morgane, David Picard, and Hedi Tabia (2018). "Jigsaw Puzzle Solving Using Local Feature Co-Occurrences in Deep Neural Networks". In: *IEEE International Conference on Image Processing (ICIP)* (cit. on p. 53).

Peng, Xi, Xiang Yu, Kihyuk Sohn, Dimitris N Metaxas, and Manmohan Chandraker (2017). "Reconstruction-based disentanglement for pose-invariant face recognition". In: *IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 27, 84).

Perarnau, Guim, Joost Van De Weijer, Bogdan Raducanu, and Jose M Álvarez (2016). "Invertible conditional gans for image editing". In: *Advances in Neural Information Processing Systems Workshop (NIPS-W)* (cit. on pp. 7, 27, 77, 82, 84, 91, 108).

Pereyra, Gabriel, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton (2017). "Regularizing Neural Networks by Penalizing Confident Output

Distributions". In: *International Conference on Learning Representations Workshop (ICLR-W)* (cit. on p. 23).

Plaut, David C et al. (1986). *Experiments on Learning by Back Propagation.* Tech. rep. CMU (cit. on p. 20).

Ranzato, Marc'Aurelio, Y-lan Boureau, and Yann L. Cun (2008). "Sparse Feature Learning for Deep Belief Networks". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 25).

Ranzato, Marc'Aurelio, F. J. Huang, Y. L. Boureau, and Y. LeCun (2007a). "Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 49).

Ranzato, Marc'Aurelio, Christopher Poultney, Sumit Chopra, and Yann Lecun (2007b). "Efficient Learning of Sparse Representations with an Energy-Based Model". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 25).

Ranzato, Marc'Aurelio and Martin Szummer (2008). "Semi-supervised learning of compact document representations with deep networks". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 6, 25, 26, 49).

Rasmus, Antti, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko (2015). "Semi-supervised learning with ladder networks". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 46, 49, 51, 55, 57, 66, 71, 72).

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (2015). "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 4).

Rifai, Salah, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio (2011). "Contractive auto-encoders: Explicit invariance during feature extraction". In: *International Conference on Machine Learning (ICML)*. Omnipress (cit. on p. 22).

Robert, Thomas, Nicolas Thome, and Matthieu Cord (2018). "HybridNet: Classification and Reconstruction Cooperation for Semi-Supervised Learning". In: *European Conference on Computer Vision (ECCV)* (cit. on pp. 9, 43, 157).

Robert, Thomas, Nicolas Thome, and Matthieu Cord (2019). "DualDis: Dual-Branch Disentangling with Adversarial Learning". In: *Under Review at Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 9, 75).

Rosca, Mihaela, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed (2017). "Variational approaches for auto-encoding generative adversarial networks". In: *arXiv preprint libary* (cit. on p. 82).

Roth, Kevin, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann (2017). "Stabilizing training of generative adversarial networks through regulariza-

tion". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 81).

Ruiz, Adrià, Oriol Martinez, Xavier Binefa, and Jakob Verbeek (2019). "Learning Disentangled Representations with Reference-Based Variational Autoencoders". In: *arXiv preprint libary* (cit. on p. 77).

Rumelhart, David E, Geoffrey Hinton, Ronald J Williams, et al. (1988). "Learning representations by back-propagating errors". In: *Cognitive modeling* (cit. on pp. 4, 15).

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* (cit. on pp. 4, 16, 37).

Sajjadi, Mehdi, Mehran Javanmardi, and Tolga Tasdizen (2016). "Regularization With Stochastic Transformations and Perturbations for Deep Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 6, 22, 26, 45, 47, 48, 59, 66, 71, 72, 140, 157).

Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen (2016). "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc. (cit. on pp. 71, 72).

Santurkar, Shibani, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry (2018). "How does batch normalization help optimization?" In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 5, 22).

Seck, Ismaïla, Gaëlle Loosli, and Stephane Canu (2019). "L1-norm double backpropagation adversarial defense". In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)* (cit. on p. 22).

Shamir, O., S. Sabato, and Naftali Tishby (2010). "Learning and generalization with the information bottleneck". In: *Theoretical Computer Science* (cit. on p. 23).

Shi, Weiwei, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng (2018). "Transductive semi-supervised deep learning using min-max features". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 299–315 (cit. on p. 24).

Shu, Zhixin, Mihir Sahasrabudhe, Riza Alp Guler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos (2018). "Deforming Autoencoders: Unsupervised Disentangling of Shape and Appearance". In: *European Conference on Computer Vision (ECCV)* (cit. on p. 54).

Shu, Zhixin, Ersin Yumer, Sunil Hadap, Kalyan Sunkavalli, Eli Shechtman, and Dimitris Samaras (2017). "Neural face editing with intrinsic image disentan-

gling". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 54).

Simard, Patrice Y, David Steinkraus, John C Platt, et al. (2003). "Best practices for convolutional neural networks applied to visual document analysis." In: *Icdar* (cit. on p. 20).

Simonyan, Karen and Andrew Zisserman (2015). "Very deep convolutional networks for large-scale image recognition". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 15–18, 23, 66).

Sokolić, Jure, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues (2017). "Robust large margin deep neural networks". In: *IEEE Transactions on Signal Processing* (cit. on p. 22).

Springenberg, Jost Tobias (2016). "Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 26, 49, 71, 72).

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research (JMLR)* (cit. on pp. 4, 5, 21).

Sweldens, W. (1995). "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions". In: *Wavelet Applications in Signal and Image Processing III* (cit. on p. 50).

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2015). "Going deeper with convolutions". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 17, 18).

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna (2016). "Rethinking the inception architecture for computer vision". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 17, 20).

Tarvainen, Antti and Harri Valpola (2017). "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on pp. 6, 26, 45, 48, 59, 64, 66, 70–72, 97, 140–142, 157).

Tishby, Naftali, F. C. Pereira, and W. Bialek (1999). "The information bottleneck method". In: *Annual Allerton Conference on Communication, Control and Computing* (cit. on pp. 13, 23).

Tishby, Naftali and Noga Zaslavsky (2015). "Deep learning and the information bottleneck principle". In: *Information Theory Workshop (ITW)*. IEEE (cit. on pp. 31, 128).

Tran, Luan, Xi Yin, and Xiaoming Liu (2017). " Disentangled Representation Learning GAN for Pose-Invariant Face Recognition ". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 84).

Turian, Joseph, Lev Ratinov, and Yoshua Bengio (2010). "Word representations: a simple and general method for semi-supervised learning". In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics (cit. on pp. 26, 49).

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky (2016). "Instance normalization: The missing ingredient for fast stylization". In: *arXiv preprint libary* (cit. on p. 22).

Vapnik, Vladimir (1992). "Principles of risk minimization for learning theory". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 18).

Vapnik, Vladimir and A Ya Chervonenkis (1972). "On the uniform convergence of relative frequencies of events to their probabilities". In: *Measures of Complexity*. Springer (cit. on p. 18).

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol (2008). "Extracting and composing robust features with denoising autoencoders". In: *International Conference on Machine Learning (ICML)* (cit. on p. 25).

Wan, Li, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus (2013). "Regularization of Neural Networks using DropConnect". In: *International Conference on Machine Learning (ICML)* (cit. on p. 21).

Wang, Ting-Chun, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro (2018). "High-resolution image synthesis and semantic manipulation with conditional gans". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 82).

Weston, Jason, Frédéric Ratle, and Ronan Collobert (2008). "Deep Learning via Semi-supervised Embedding". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 26, 49).

Wojna, Zbigniew, Vittorio Ferrari, Sergio Guadarrama, Nathan Silberman, Liang-Chieh Chen, Alireza Fathi, and Jasper Uijlings (2017). "The Devil is in the Decoder". In: *British Machine Vision Conference (BMVC)* (cit. on p. 50).

Wu, Yuxin and Kaiming He (2018). "Group normalization". In: *European Conference on Computer Vision (ECCV)* (cit. on p. 22).

Zagoruyko, Sergey and Nikos Komodakis (2016). "Wide Residual Networks". In: *arXiv preprint libary* (cit. on p. 36).

Zeiler, Matthew D and Rob Fergus (2014). "Visualizing and understanding convolutional networks". In: *European Conference on Computer Vision (ECCV)* (cit. on p. 55).

Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals (2017). "Understanding deep learning requires rethinking generalization". In:

International Conference on Learning Representations (ICLR) (cit. on pp. 5, 6, 36, 37).

Zhang, Hongyi, Yann N Dauphin, and Tengyu Ma (2019). "Fixup Initialization: Residual Learning Without Normalization". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 5).

Zhang, Yuting, Kibok Lee, and Honglak Lee (2016). "Augmenting Supervised Neural Networks with Unsupervised Objectives for Large-scale Image Classification". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 23, 46, 50, 51, 55, 57).

Zhao, Junbo, Michael Mathieu, Ross Goroshin, and Yann LeCun (2016). "Stacked What-Where Auto-encoders". In: *International Conference on Learning Representations Workshop (ICLR-W)* (cit. on pp. 49–51, 53, 55, 57, 61, 62, 66, 71).

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A Efros (2017). "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 81).

Zhu, Ligeng, Ruizhi Deng, Michael Maire, Zhiwei Deng, Greg Mori, and Ping Tan (2018). "Sparsely aggregated convolutional networks". In: *European Conference on Computer Vision (ECCV)* (cit. on p. 22).

Zhu, Xiaojin (2005). *Semi-Supervised Learning Literature Survey*. Tech. rep. Computer Sciences, University of Wisconsin-Madison (cit. on p. 24).

Zhu, Xiaojin and Zoubin Ghahramani (2002). *Learning from labeled and unlabeled data with label propagation*. Tech. rep. CMU (cit. on p. 24).

DETAILS AND ADDITIONAL EXPERIMENTS ON SHADE

Contents

In this appendix, we first provide details about the development of SHADE that were summarized in Chapter 2, and provide results of additional experiments conducted to validate the hypotheses made during this development.

A.1 Detail on the development of SHADE

In this section, we provide more details about the development of SHADE described in Section 2.3, from the original idea of minimizing the entropy $\mathcal{H}(Y \mid Y)$ to the final loss.

A.1.1 Unit-wise Entropy Regularizer

Layer-wise regularization. A Deep Neural Network (DNN) is composed of a number L of layers that transform sequentially the input. Each one can be seen as an intermediate representation variable, noted H_ℓ for layer ℓ, that is determined by the output of the previous layer and a set of parameters $\mathbf{w}_\ell$. Each layer filters out a certain part of the information from the initial input. Thus, from the data

processing inequality in Cover and Thomas (1991) can be derived the following inequalities for any layer ℓ:

$$\mathcal{H}(H_\ell \mid Y) \leq \mathcal{H}(H_{\ell-1} \mid Y) \leq \cdots \leq \mathcal{H}(H_1 \mid Y) \leq \mathcal{H}(X \mid Y). \tag{A.1}$$

This shows that each layer can only remove some entropy from the previous layer. What we want is to encourage the decrease in entropy. Thus, and following the recommendation of Tishby and Zaslavsky (2015), we apply our regularization to all the layers, using a layer-wise criterion $\mathcal{H}(H_\ell \mid Y)$, and producing a global criterion to minimize:

$$\Omega_{\text{layers}} = \sum_{\ell=1}^{L} \mathcal{H}(H_\ell \mid Y). \tag{A.2}$$

Unit-wise regularization. Examining one layer ℓ, its representation variable is a random vector of D_ℓ coordinates $H_{\ell,i}$: $H_\ell = [H_{\ell,1}, \ldots, H_{\ell,D_\ell}]^\top$. The upper bound [1] $\mathcal{H}(H_\ell \mid Y) \leq \sum_{i=1}^{D_\ell} \mathcal{H}(H_{\ell,i} \mid Y)$ enables to define a unit-wise criterion that SHAnnon DEcay (SHADE) seeks to minimize. For each unit i of every layer ℓ we design a loss $\omega_{\text{unit}}(H_{\ell,i} \mid Y) = \mathcal{H}(H_{\ell,i} \mid Y)$ that will be part of the global regularization loss:

$$\Omega_{\text{layers}} \leq \Omega_{\text{units}} = \sum_{l=1}^{L} \sum_{i=1}^{D_\ell} \underbrace{\mathcal{H}(H_{\ell,i} \mid Y)}_{\omega_{\text{unit}}(H_{\ell,i}|Y)}. \tag{A.3}$$

Later in the chapter, we use the notation H instead of $H_{\ell,i}$ for simplicity since the coordinates are all considered independently to define our criterion based on $\omega_{\text{unit}}(H_{\ell,i} \mid Y)$.

A.1.2 Estimating Conditional Entropy with a Latent Code

In this section, we describe how to define a loss based on the measure $\mathcal{H}(H \mid Y)$ with H being one coordinate variable of one layer. Defining this loss is not obvious as the gradient of $\mathcal{H}(H \mid Y)$ with respect to the layer's parameters may be computationally intractable. H has an unknown distribution and without modeling it properly it is not possible to compute $\mathcal{H}(H \mid Y)$ precisely for the following reasons.

Since $\mathcal{H}(H \mid Y) = \sum_{k=1}^{N_{\text{cls}}} p(Y)\mathcal{H}(H \mid Y_k)$ it is necessary to compute N_{cls} different entropies $\mathcal{H}(H \mid Y_k)$. This means that, given a batch, the number of samples used

1. This upper bound is well justified in deep learning as the neurons of a layer tend to be more and more independent of each other as we go deeper within the network.

to estimate one of these entropies is divided by N_{cls} on average which becomes particularly problematic when dealing with a large number of classes such as the 1,000 classes of ImageNet. Furthermore, entropy estimators are extremely inaccurate considering the number of samples in a batch. For example, LME estimators of entropy described by Paninski (2003) converge in $\mathcal{O}(^{(\log K)^2}/_K)$ for K samples. Finally, most estimators such as LME require discretizing the space in order to approximate the distribution *via* a histogram. This raises issues on the bins definition considering that the variable distribution is unknown and varies during the training in addition to the fact that having a histogram for each neuron of the model is computationally and memory consuming.

To tackle these drawbacks we investigate the two following workarounds: the introduction of a binary latent representation that enables to use more examples to estimate the entropy; and a bound on the entropy of the variable by an increasing function of its variance to avoid the issue of entropy estimation with a histogram and make the computation tractable and scalable.

Binary latent code. First, inspecting a neuron H prior to the non-linearity, the ReLU activation makes it act as a detector, returning a signal when a certain pattern is present on the input. If the pattern is absent the signal is zero, otherwise, it quantifies the resemblance with it. We therefore propose to associate a binomial variable Z to each unit variable H (before ReLU). This variable Z indicates if a particular pattern is present on the input ($Z = 1$ when $H \gg 0$) or not ($Z = 0$ when $H \ll 0$). It acts like a latent code in variational models (*e.g.* Kingma and Welling 2013) or in generative models (*e.g.* X. Chen et al. 2016). In our implementation, we chose a binomial distribution $p(Z = 1 \mid H) = \text{sigmoid}(H)$ that matches this intuition.

Furthermore, it is very likely that most intermediate features of a DNN can take similar values for inputs of different classes – this is especially true for low-level features. The semantic information provided by a feature is thus more about a particular pattern than about the class itself. Only the association of features allows determining the class. So Z represents a semantically meaningful factor about the class Y and from which the input X is generated. The feature value H is then a quantification of the possibility for this semantic attribute Z to be present in the input or not.

We assume the Markov chain $Y \to Z \to X \to H$. During the training, the distribution of H varies in order to get as close as possible to a sufficient statistic of X for Y (see definition by Cover and Thomas 1991). Therefore, we expect Z to be such that H draws near a sufficient statistic of Z for Y as well. By assuming

the sufficient statistic relation $\mathcal{I}(H, Y) = \mathcal{I}(H, Z)$ we get the equivalent equality $\mathcal{H}(H \mid Y) = \mathcal{H}(H \mid Z)$, and finally obtain:

$$\omega_{\text{unit}}(H \mid Y) = \mathcal{H}(H \mid Y) = \mathcal{H}(H \mid Z) = \sum_{z \in \{0,1\}} p(z)\mathcal{H}(H \mid Z = z). \tag{A.4}$$

This modeling of Z as a binomial variable (one for each unit) has the advantage of enabling good estimators of conditional entropy since we only divide the batch into two sets for the estimation ($z = 0$ and 1) regardless of the number of classes.

Variance bound. Using a binomial latent code allows computing fewer entropy estimates to obtain the global conditional entropy, thus increasing the sample size used for each entropy estimation. Unfortunately, it does not solve the bin definition issue. To address this, we propose to use the following bound on $\mathcal{H}(H \mid Z)$, that does not require the definition of bins:

$$\mathcal{H}(H \mid Z) \leq \frac{1}{2} \ln \big(2\pi e \mathcal{V}\text{ar}(H \mid Z)\big). \tag{A.5}$$

This bound holds for any continuous distributions H and there is equality if the distribution is Gaussian. For many other distributions such as the exponential one, the entropy is also equal to an increasing function of the variance. In addition, one main advantage is that variance estimators are much more robust than entropy estimators, converging in $\mathcal{O}(1/K)$ for K samples instead of $\mathcal{O}(\log(K)^2/K)$.

Finally, the $\ln$ function being one-to-one and increasing, we only keep the simpler term $\mathcal{V}\text{ar}(H \mid Z)$ to design our final loss:

$$\Omega_{\text{SHADE}} = \sum_{\ell=1}^{L} \sum_{i=1}^{D_\ell} \sum_{z \in \{0,1\}} p(Z_{\ell,i} = z \mid H)\mathcal{V}\text{ar}(H \mid Z_{\ell,i} = z). \tag{A.6}$$

In next section, we detail the definition of the differentiable loss using $\mathcal{V}\text{ar}(H \mid Z)$ as a criterion computed on a mini-batch.

Algorithm A.1 Moving average updates: for $z \in \{0, 1\}$, p^z estimates $p(Z = z)$ and μ^z estimates $\mathbb{E}(H \mid Z = z)$

Initialize: $\mu^0 = -1$, $\mu^1 = 1$, $p^0 = p^1 = 0.5$, $\lambda = 0.8$
for each mini-batch $\{h^{(k)}, k \in 1..K\}$ **do**
  for $z \in \{0, 1\}$ **do**
    $p^z \leftarrow \lambda p^z + (1 - \lambda) \frac{1}{K} \sum_{k=1}^{K} p(z \mid h^{(k)})$
    $\mu^z \leftarrow \lambda \mu^z + (1 - \lambda) \frac{1}{K} \sum_{k=1}^{K} \frac{p(z \mid h^{(k)})}{p^z} h^{(k)}$
  end for
end for

A.1.3 Instantiating SHADE

For one unit of one layer, the previous criterion writes:

$$\mathcal{V}\mathrm{ar}(H \mid Z) = \int_{\mathcal{H}} p(h) \int_{\mathcal{Z}} p(z \mid h) \big(h - \mathbb{E}(H \mid z)\big)^2 \, \mathrm{d}z \, \mathrm{d}h \tag{A.7}$$

$$\approx \frac{1}{K} \sum_{k=1}^{K} \left[\int_{\mathcal{Z}} p(z \mid h^{(k)}) \big(h^{(k)} - \mathbb{E}(H \mid z)\big)^2 \, \mathrm{d}\boldsymbol{z} \right] ; \tag{A.8}$$

estimating the quantity $\mathcal{V}\mathrm{ar}(H \mid Z)$ with Monte-Carlo sampling on a mini-batch of input-target pairs $\left\{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})\right\}_{1 \leq k \leq K}$ of intermediate representations $\left\{h^{(k)}\right\}_{1 \leq k \leq K}$.

$p(Z \mid H)$ interpreted as the probability of presence of attribute Z on the input, it should clearly be modeled such that $p(Z = 1 \mid H)$ increases with H. The more similarities between the input and the pattern represented by H, the higher the probability of presence for Z. We suggest using:

$$\begin{cases} p(Z = 1 \mid h) = \sigma(h) \\ p(Z = 0 \mid h) = 1 - \sigma(h) \end{cases} \quad \text{with sigmoid function } \sigma(h) = \frac{1}{1 + e^{-h}}. \tag{A.9}$$

For the expected values $\mu^z = \mathbb{E}(H \mid z)$ we use a classic moving average that is updated after each batch as described in Algorithm A.1. Note that the expected values are not changed by the optimization since they have no influence on the entropy $\mathcal{H}(H \mid Z)$.

For this proposed instantiation, our SHADE regularization penalty takes the form:

$$\boxed{\Omega_{\text{SHADE}} = \sum_{\ell=1}^{L} \sum_{i=1}^{D_\ell} \sum_{k=1}^{K} \sum_{z \in \{0,1\}} p\left(Z_{\ell,i} = z \,\middle|\, h_{\ell,i}^{(k)}\right) \left(h_{\ell,i}^{(k)} - \mu_{\ell,i}^{z}\right)^2 .} \tag{A.10}$$

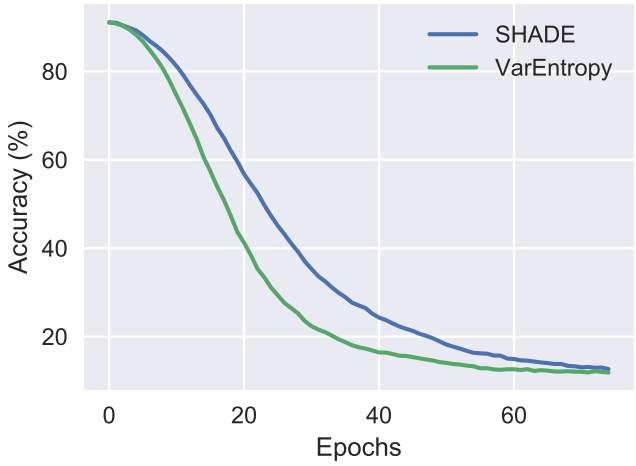


Figure A.1. – **Illustration of the effect of the regularization alone.**. Evolution of the accuracy of a pre-trained Inception model on CIFAR-10 when only applying regularization.

A.2 Additional experiments with SHADE

Through these additional experiments, we validate the hypotheses made during the development of SHADE, first regarding the preservation of class information by conditional entropy compare to non-conditional entropy, and second by validating of modeling of the behavior of neurons as binary detectors.

A.2.1 Class-Information Preservation by Conditional Entropy

We now propose to validate our hypothesis that the introduction of our variable Z is able to capture class information used for our conditional entropy.

Indeed, the main difference between SHADE and VarEntropy is the introduction of the latent variable Z, supposed to contain the information about the label Y. The motivation of this extension is that minimizing $\mathcal{H}(H \mid Y)$ instead of $\mathcal{H}(H)$ enables to save the class information during the optimization of the regularization loss, as explained in Section 2.3.2. To illustrate this benefit, we compare the impact of the two regularization losses on the classification performances of a pre-trained model. To do so, we fine tune a pre-trained Inception model only with a regularization loss, either based on $\mathcal{V}\mathrm{ar}(H)$ or $\mathcal{V}\mathrm{ar}(H \mid Z)$; without any label data or classification loss. The network performance obviously declines for both regularizers as we can see in Figure A.1. However, this decline is slower with SHADE than VarEntropy. This confirms the intuition that Z contains class-information and that SHADE produces less class-information filtering. This is explained by the

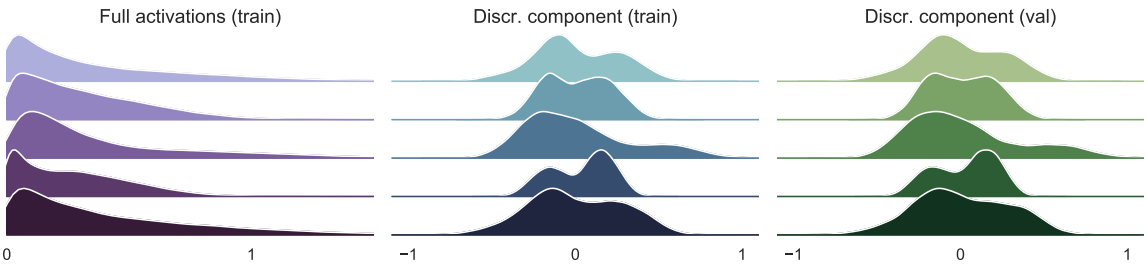


Figure A.2. – **Visualization of 5 neurons from the penultimate activations** (*i.e.* the input of the last fully-connected layer) of an Inception model trained on CIFAR-10. On the left is the distribution of the values taken by each neuron H. In the middle and right is the distribution of the discriminiative component H^* of the neuron (the part that does not belong to the kernel of the layer weights).

optimization of the metric $\mathcal{V}\mathrm{ar}(H \mid Z)$ that uses implicitly-learned information encoded in the network.

A.2.2 Exploration of the latent representations

Finally, we propose to validate our hypothesis that neurons behave as binary detectors, which motivated our binomial modeling Z. For this, we propose some investigations on the behavior of the activations of trained Convolutional Neural Networks (ConvNets).

Two modes neuron variable. First, we propose to experimentally show that DNN optimization drives the neurons distribution toward a bi-modal distribution.

Focusing on the input neurons H_{L-1} of the last layer of a trained network (before the class projection), the output of the network is obtained by applying a fully connected layer on H_{L-1} plus a softmax activation: $\hat{Y} = \mathrm{Softmax}(\mathbf{W} \cdot H_{L-1} + b)$, with $\mathbf{W}$ and b the weights of this layer. By plotting a histogram of any coordinate (one neuron) of H_{L-1}, it will not be possible to identify two modes. This can be seen on the purple distribution in Figure A.2 (left), which represents the distributions of $H_{L-1,i}$ on CIFAR-10 training set for five random coordinates i of an Inception model.

However H_{L-1} contains a lot of information that will not be exploited for the prediction. Indeed, lets rewrite $H_{L-1} = H^{\perp} + H^*$ with $H^{\perp}$ in the kernel of $\mathbf{W}$ such that $\mathbf{W} \cdot H^{\perp} = 0$ and Y^* is in the supplementary of $\mathbf{W}$'s kernel. The class space has generally much fewer dimensions than the space of H_{L-1}, thus the kernel of $\mathbf{w}$ is not reduced to zero and some information will be filtered.

Architecture	Original score	Binarized layer		
		Before $\hat{\mathbf{y}}$ ($\mathbf{h}_{L-1}$)	Middle ($\mathbf{h}_{L/2}$)	After input ($\mathbf{h}_1$)
MLP	64.68	64.92	62.45	61.13
AlexNet	83.25	82.71	82.38	82.01
Inception	91.34	91.41	90.88	90.21
ResNet	93.24	92.67	92.09	91.99

Table A.1. – **Classification accuracy (%) using binarized activation** on CIFAR-10 test set.

In Figure A.2 (middle and right) is the distribution of H^* on the training set (blue, middle) and on the validation set (green right), for the same neurons of the same network as the activations on the left. H^* is the information effectively used for the prediction and its distribution look very much like a mixture of two Gaussians. We clearly identify two modes, one negative and one positive. This confirms the intuition of a binary latent variable Z whose values correspond to the two modes. The fact that the distribution look like a mixture of two Gaussians support the use of the inequality at Equation 2.19 in the definition of SHADE.

The distributions are obtain via a kernel density estimator using as data the neuron variable output by a forward pass on the totality of the CIFAR-10 training and test set. The three distributions are for the same coordinates taken randomly among all H units. Note that the experiment could have been done on other layers but the computation of H^* would be more complicated as the following transformations up to the top of the network are not linear.

Binary activation. To further demonstrate that activations in the models can be represented as a binary information, we propose to transform the ReLU activation function of a layer into a binary activation function that can only take two values. By exhibiting that such a binary activation does not affect the accuracy, we show that we can summarize the class information of a neuron into a binary variable and still get the same prediction accuracy as with the continuous ReLU activation. The experiment have been done on CIFAR-10 dataset with the same networks used in Section 2.3.4.1.

To do this, we replace the ReLU activation of a trained model with a binary activation function:

$$\mathrm{BinAct}(H) = \begin{cases} 0 & \text{if } H \leq 0 \\ \overline{H^+} & \text{if } H > 0 \end{cases} \tag{A.11}$$

with $\overline{H^+}$ a constant value defined as the average value of the positive activations of the unit H.

After replacing the activation function we fine tune the layers on the top of the chosen layer, in order to adapt the top of the network to the new values and we report the obtained accuracies in Table A.1 for different architecture and different layers. We note that the differences in accuracy are very small. This confirms that for a given layer, the information that is used for the class prediction can be compressed in a binary variable confirming the existence of a binary latent variable containing most of the class information that is exploited by the rest of the network. The fact that this apply for all layers of the network is consistent with the application of SHADE loss to all layers. Note that this binary activation could be further researched to improve the modeling integrated in SHADE.

APPENDIX B

EXPERIMENTAL DETAILS FOR HYBRIDNET

Contents

B.1 Additional visualizations of HybridNet

Additional visualizations of HybridNet behavior are presented in Figure B.1 for CIFAR-10 and Figure B.2 for STL-10.

B.1.1 Additional results on CIFAR-10

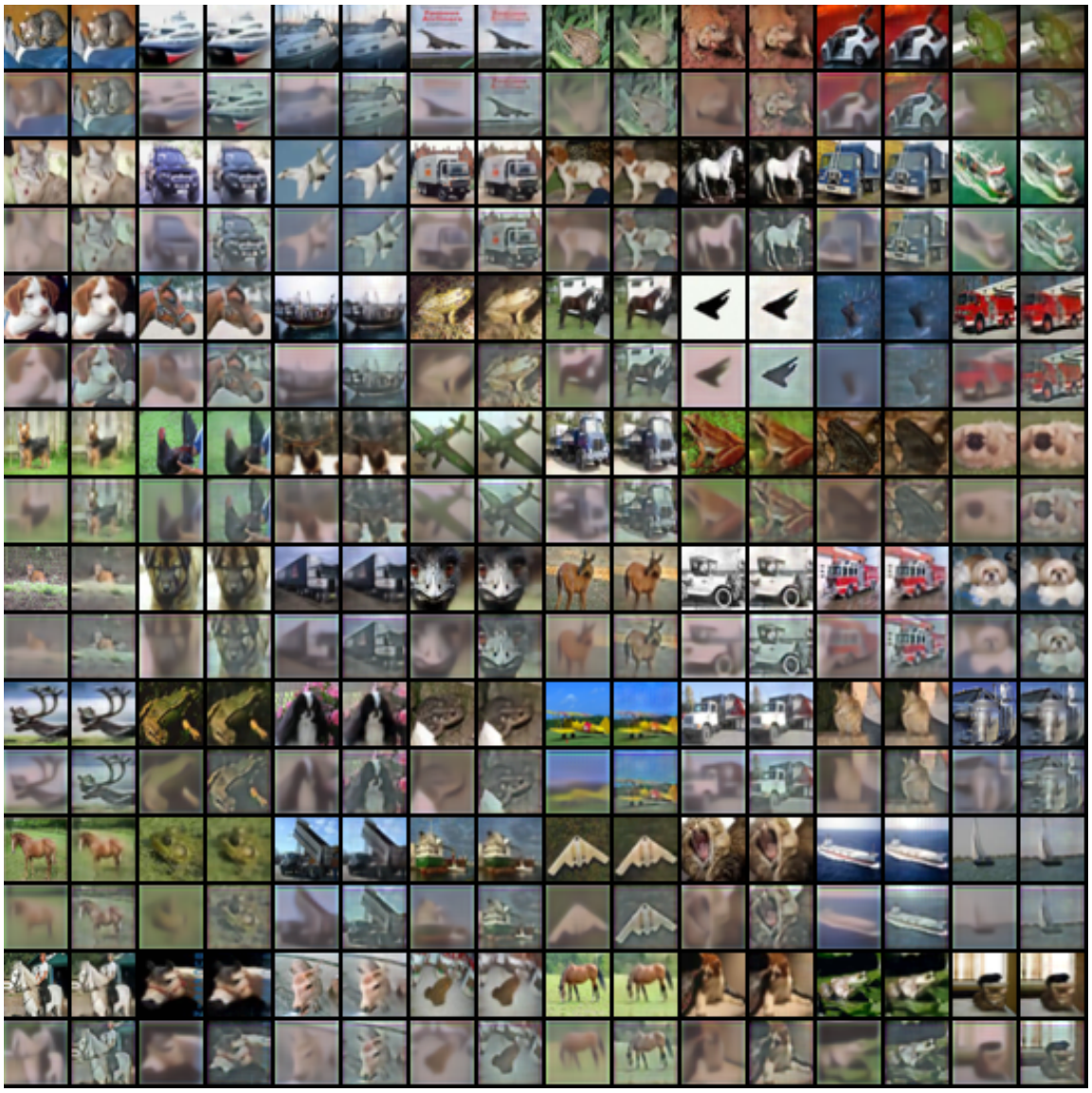


Figure B.1. – **Example of visualizations for a ConvLarge-based HybridNet on CIFAR-10**. For each input image, there is block of 4 images on the figure with the following organization: $\begin{bmatrix} \mathbf{x} & \hat{\mathbf{x}} \\ \hat{\mathbf{x}}_c & \hat{\mathbf{x}}_u \end{bmatrix}$.

B.1.2 Additional results on STL-10

Figure B.2. – **Example of visualizations for a ConvLarge-like-based HybridNet on STL-10**. For each input image, there is block of 4 images on the figure with the following organization: $\begin{bmatrix} \mathbf{x} & \hat{\mathbf{x}} \\ \hat{\mathbf{x}}_c & \hat{\mathbf{x}}_u \end{bmatrix}$.

B.2 Experiment details

B.2.1 Experimental setup for ConvLarge on CIFAR-10

B.2.1.1 Data preprocessing and model architecture

Input images are data-augmented with a random translation of a maximum of 2 pixels with mirror padding to fill-in the missing pixels, and randomly flipped. This constitutes the input $\mathbf{x}$. We also add a Gaussian noise on $\mathbf{x}$ ($\sigma = 0.15$) to obtain $\tilde{\mathbf{x}}$ that is fed into the model. The model's architecture is described in Table B.1.

B.2.1.2 Training details

The training method is similar to the one presented in recent paper using ConvLarge (Sajjadi et al. 2016; Laine and Aila 2017; Tarvainen and Valpola 2017).

The model is optimized with Adam during 60,000 batches (which corresponds to various number of epochs depending on the number of labeled images), with batches of 80 unlabeled samples and 20 labeled samples.

The weights of the various loss terms and the optimizer's parameters have base values and are varied over the training similarly to previous work using this model. The parameters' values and variations are summarized in Table B.2. For the ablation study, parts of the model are removed and/or some weights are set to 0.

B.2.2 Experimental setup for ConvLarge-like on STL-10

B.2.2.1 Model architecture

Input images are data-augmented with a random translation of a maximum of 12 pixels with mirror padding to fill-in the missing pixels, and randomly flipped. This constitutes the input $\mathbf{x}$. We also add a Gaussian noise on $\mathbf{x}$ ($\sigma = 0.15$) to obtain $\tilde{\mathbf{x}}$ that is fed into the model. The model's architecture is detailed in Table B.3.

B.2.2.2 Training details

The model is optimized with Adam during 150,000 batches (corresponding to 300 epochs over the labeled images, 48 epochs over the unlabeled images), with

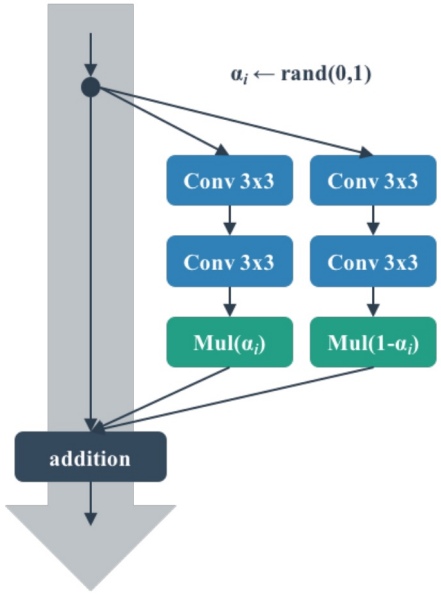


Figure B.3. – **Shake-Shake building block**.

batches of 30 unlabeled samples and 2 labeled samples. Hyperparameters' values and scheduling over training are detailed in Table B.4.

B.2.3 Experimental setup for ResNet on CIFAR-10 and SVHN

The experimental setup described below follows the one described by Tarvainen and Valpola (2017) for a fair comparison.

B.2.3.1 Model architecture

The data preprocessing simply consists in a classic per-color-channel mean-variance standardization. Images are data-augmented using random translation of a maximum of 4 pixels with mirror padding to fill-in the missing pixels and random flip. For SVHN, we disable image mirroring for obvious reasons.

We use the ResNet architecture with Shake-Shake building blocks described by Gastaldi (2017). A Shake-Shake building block consists of 2 similar branches each containing 2 convolutions, with the first one possibly having a stride greater that 1. The two branches are averaged with a weight α (see Figure B.3 for illustration and the original paper for details) before being added to the result of a residual connection.

A "layer" is constituted of 4 blocks with possibly the first one having a stride in its first convolution and all convolutions having the same number of channels.

To reverse a layer, we apply the same strategy as before. This means that only the last transposed convolution of a decoding layer will have a smaller number of channels and a "stride" larger than 1 to reverse the first convolution of the corresponding layer in the encoder.

The architecture of the HybridNet based on this ResNet is described in Table B.5.

B.2.3.2 Training details for CIFAR-10

The model's training is based on the settings of Tarvainen and Valpola (2017). It is trained with Nesterov SGD with a base learning rate of 0.04 with a momentum of 0.9 over 300 epochs (one epoch correspond to one pass over the unlabeled images) with batches of 61 unlabeled images and 19 labeled images. Hyperparameters values and scheduling over training are detailed in Table B.6.

B.2.3.3 Training details for SVHN

The model is trained with Nesterov SGD with a base learning rate of 0.04 with a momentum of 0.9 over 150 epochs (one epoch correspond to one pass over the unlabeled images) with batches of 265 unlabeled images and 15 labeled images. Hyperparameters values and scheduling over training are detailed in Table B.7 for SVHN.

B.2.4 Experimental setup for ResNet on STL-10

B.2.4.1 Model architecture

The model is a ResNet-50 pretraind on the Places dataset available at `https://github.com/CSAILVision/places365`. We did not use a model trained on ImageNet since the images of STL-10 have been extracted from ImageNet.

The data preprocessing simply consists in a classic per-color-channel mean-variance standardization. Images are data-augmented using random translation of a maximum of 30 pixels with mirror padding to fill-in the missing pixels and random flip.

B.2.4.2 Training details

The model is trained with Nesterov SGD with a base learning rate of 0.01 with a momentum of 0.9 over 350 epochs (one epoch correspond to one pass over the

unlabeled images) with batches of 11 unlabeled images and 5 labeled images. Hyperparameters values and scheduling over training are detailed in Table B.8.

Table B.1. – Architecture of the HybridNet ConvLarge for CIFAR-10

Encoders E_c and E_u		
Input	$\tilde{\mathbf{x}}$	$32 \times 32 \times 3$
Convolution	128 filters, 3×3, *same* padding	$32 \times 32 \times 128$
Convolution	128 filters, 3×3, *same* padding	$32 \times 32 \times 128$
Convolution	128 filters, 3×3, *same* padding	$32 \times 32 \times 128$
Pooling	Maxpool 2×2	$16 \times 16 \times 128$
Dropout	$p = 0.5$	$16 \times 16 \times 128$
Convolution	256 filters, 3×3, *same* padding	$16 \times 16 \times 256$
Convolution	256 filters, 3×3, *same* padding	$16 \times 16 \times 256$
Convolution	256 filters, 3×3, *same* padding	$16 \times 16 \times 256$
Pooling	Maxpool 2×2	$8 \times 8 \times 256$
Dropout	$p = 0.5$	$8 \times 8 \times 256$
Convolution	512 filters, 3×3, *valid* padding	$6 \times 6 \times 512$
Convolution	256 filters, 1×1, *same* padding	$6 \times 6 \times 256$
Convolution	128 filters, 1×1, *same* padding	$6 \times 6 \times 128$
Output	$\mathbf{h}_c$ or $\mathbf{h}_u$	$6 \times 6 \times 128$
Classifier C		
Input	$\mathbf{h}_c$	$6 \times 6 \times 128$
Pooling	Global average pool	$1 \times 1 \times 128$
Fully connected	with Softmax	10
Output	$\hat{\mathbf{y}}$	10
Decoders D_c and D_u		
Input	$\mathbf{h}_c$ or $\mathbf{h}_u$	$6 \times 6 \times 128$
TConvolution	256 filters, 1×1, *same* padding	$6 \times 6 \times 256$
TConvolution	512 filters, 1×1, *same* padding	$6 \times 6 \times 512$
TConvolution	256 filters, 3×3, *valid* padding	$8 \times 8 \times 256$
Upsampling	2×2 (unpooling in D_u)	$16 \times 16 \times 256$
TConvolution	256 filters, 3×3, *same* padding	$16 \times 16 \times 256$
TConvolution	256 filters, 3×3, *same* padding	$16 \times 16 \times 256$
TConvolution	128 filters, 3×3, *same* padding	$16 \times 16 \times 128$
Upsampling	2×2 (unpooling in D_u)	$32 \times 32 \times 128$
TConvolution	128 filters, 3×3, *same* padding	$32 \times 32 \times 128$
TConvolution	128 filters, 3×3, *same* padding	$32 \times 32 \times 128$
TConvolution	3 filters, 3×3, *same* padding	$32 \times 32 \times 3$
Output	$\hat{\mathbf{x}}_c$ or $\hat{\mathbf{x}}_u$	$32 \times 32 \times 3$

TConvolution stands for "transposed convolution".
Each Convolution or TConvolution is followed by a Batch Normalization layer and a LeakyRELU of parameter $\alpha = 0.1$

Table B.2. – Evolution of weights for ConvLarge on CIFAR-10

	Value	Scheduling
η	0.003	Linear decrease to 0 over the last $1/3$ of the training
β_1	0.9	Exponential decrease to 0.5 over the last $1/5$ of the training
λ_c	1	Exponential increase from 0 over 800 first batches
λ_s	100	Exponential increase from 0 over first $1/4$ of the training and exponential decrease to 0 over the last $1/5$ of the training
λ_r	1	Exponential decrease over the last 5% of the training

η is the learning rate, β_1 the first momentum of Adam, λ_c the classification weight, λ_s the stability weight, λ_r the reconstructions weights. Exponential decrease follows the function $\exp(-5t^2)$ with $t \in [0, 1]$ from the start to the end of the decreasing interval. When increasing, t goes from 1 to 0.

Table B.3. – Architecture of the HybridNet ConvLarge-like architecture for STL-10

Encoders E_c and E_u		
Input	$\tilde{\mathbf{x}}$	$96 \times 96 \times 3$
Convolution	64 filters, 3×3, *same* padding	$96 \times 96 \times 64$
Convolution	64 filters, 3×3, *same* padding	$96 \times 96 \times 64$
Pooling	Maxpool 2×2	$48 \times 48 \times 64$
Convolution	128 filters, 3×3, *same* padding	$48 \times 48 \times 128$
Convolution	128 filters, 3×3, *same* padding	$48 \times 48 \times 128$
Pooling	Maxpool 2×2	$24 \times 24 \times 128$
Convolution	256 filters, 3×3, *same* padding	$24 \times 24 \times 256$
Convolution	256 filters, 3×3, *same* padding	$24 \times 24 \times 256$
Pooling	Maxpool 2×2	$12 \times 12 \times 256$
Convolution	256 filters, 3×3, *same* padding	$12 \times 12 \times 256$
Pooling	Maxpool 2×2	$6 \times 6 \times 256$
Output	$\mathbf{h}_c$ or $\mathbf{h}_u$	$6 \times 6 \times 256$
Classifier C		
Input	$\mathbf{h}_c$	$6 \times 6 \times 256$
Convolution	512 filters, 4×4, *valid* padding	$3 \times 3 \times 512$
Dropout	$p = 0.5$	$3 \times 3 \times 512$
Convolution	512 filters, 1×1, *same* padding	$3 \times 3 \times 512$
Dropout	$p = 0.5$	$3 \times 3 \times 512$
Convolution	10 filters, 1×1, *same* padding	$3 \times 3 \times 10$
Pooling	Global average pool	$1 \times 1 \times 10$
Softmax		10
Output	$\hat{\mathbf{y}}$	10
Decoders D_c and D_u		
Input	$\mathbf{h}_c$ or $\mathbf{h}_u$	$6 \times 6 \times 256$
Upsampling	2×2 (unpooling in D_u)	$12 \times 12 \times 256$
TConvolution	256 filters, 3×3, *same* padding	$12 \times 12 \times 256$
Upsampling	2×2 (unpooling in D_u)	$24 \times 24 \times 256$
TConvolution	256 filters, 3×3, *same* padding	$24 \times 24 \times 256$
TConvolution	128 filters, 3×3, *same* padding	$24 \times 24 \times 128$
Upsampling	2×2 (unpooling in D_u)	$48 \times 48 \times 128$
TConvolution	128 filters, 3×3, *same* padding	$48 \times 48 \times 128$
TConvolution	64 filters, 3×3, *same* padding	$48 \times 48 \times 64$
Upsampling	2×2 (unpooling in D_u)	$96 \times 96 \times 64$
TConvolution	64 filters, 3×3, *same* padding	$96 \times 96 \times 64$
TConvolution	3 filters, 3×3, *same* padding	$96 \times 96 \times 3$
Output	$\hat{\mathbf{x}}_c$ or $\hat{\mathbf{x}}_u$	$96 \times 96 \times 3$

TConvolution stands for "transposed convolution".
Each Convolution or TConvolution is followed by a Batch Normalization layer and a ELU activation.

Table B.4. – Evolution of weights for ConvLarge-like on STL-10

	Value	Scheduling
η	0.001	Linear decrease to 0 over the last $1/10$ of the training
β_1	0.9	Constant
λ_c	1	Exponential increase from 0 over 4000 first batches
λ_s	300	Exponential increase from 0 over first $1/4$ of the training and exponential decrease to 0 over the last $1/4$ of the training
λ_r	1	Exponential decrease over the last 5% of the training

η is the learning rate, β_1 the first momentum of Adam, λ_c the classification weight, λ_s the stability weight, λ_r the reconstructions weights. Exponential decrease follows the function $\exp(-5t^2)$ with $t \in [0, 1]$ from the start to the end of the decreasing interval. When increasing, t goes from 1 to 0.

Table B.5. – Architecture of the HybridNet ResNet architecture for CIFAR-10 and SVHN

Encoders E_c and E_u		
Input	$\tilde{\mathbf{x}}$	$32 \times 32 \times 3$
Convolution	16 filters, 3×3, *same* padding	$32 \times 32 \times 16$
Shake Shake layer	4 blocks, 96 filters, 3×3, stride 1	$32 \times 32 \times 96$
Shake Shake layer	4 blocks, 192 filters, 3×3, stride 2	$16 \times 16 \times 192$
Shake Shake layer	4 blocks, 384 filters, 3×3, stride 2	$8 \times 8 \times 384$
Output	$\mathbf{h}_c$ or $\mathbf{h}_u$	$8 \times 8 \times 384$
Classifier C		
Input	$\mathbf{h}_c$	$8 \times 8 \times 384$
Pooling	Global average pool	$1 \times 1 \times 384$
Fully connected	with Softmax	10
Output	$\hat{\mathbf{y}}$	10
Decoders D_c and D_u		
Input	$\mathbf{h}_c$ or $\mathbf{h}_u$	$8 \times 8 \times 384$
Shake Shake dec layer	4 blocks, 384 filters, 3×3, stride 2	$8 \times 8 \times 192$
Shake Shake dec layer	4 blocks, 192 filters, 3×3, stride 2	$16 \times 16 \times 96$
Shake Shake dec layer	4 blocks, 96 filters, 3×3, stride 1	$32 \times 32 \times 16$
Output	$\hat{\mathbf{x}}_c$ or $\hat{\mathbf{x}}_u$	$32 \times 32 \times 3$

Table B.6. – Evolution of weights for HybridNet ResNet architecture for CIFAR-10

	Value	Scheduling
η	0.04	Cosine decrease over the full training
λ_c	1	Constant
λ_s	300	Constant
λ_r	0.25	Exponential increase over the first 5 epochs
$\lambda_{rb,l}$	0.5	Exponential increase over the first 2 epochs

η is the learning rate, λ_c the classification weight, λ_s the stability weight, λ_r the final reconstruction weight, $\lambda_{rb,l}$ the intermediate reconstructions weights.
Exponential decrease follows the function $\exp(-5t^2)$ with $t \in [0, 1]$ from the start to the end of the decreasing interval. When increasing, t goes from 1 to 0.
Cosine decrease follows the function $\cos(\pi t) + 1$ with $t \in [0, 1]$ from the start to the end of the decreasing interval.

Table B.7. – Evolution of weights for HybridNet ResNet architecture for SVHN

	Value	Scheduling
η	0.04	Cosine decrease over the full training
λ_c	1	Constant
λ_s	100	Exponential increase over the first 5 epochs
λ_r	0.1	Exponential increase over the first 5 epochs
$\lambda_{rb,l}$	0.2	Exponential increase over the first 2 epochs

η is the learning rate, λ_c the classification weight, λ_s the stability weight, λ_r the final reconstruction weight, $\lambda_{rb,l}$ the intermediate reconstructions weights.
Exponential decrease follows the function $\exp(-5t^2)$ with $t \in [0, 1]$ from the start to the end of the decreasing interval. When increasing, t goes from 1 to 0.
Cosine decrease follows the function $\cos(\pi t) + 1$ with $t \in [0, 1]$ from the start to the end of the decreasing interval.

Table B.8. – Evolution of weights for HybridNet ResNet architecture for STL-10

	Value	Scheduling
η	0.01	Exponential decrease during the last 8 epochs
β_1	0.9	Exponential increase during the last 80 epochs down to 0.5
λ_c	0.1	Constant
λ_s	0.1	Exponential increase over the first 150 epochs
λ_r	0.01	Exponential decrease during the last 17 epochs
$\lambda_{rb,l}$	0.01	Exponential decrease during the last 17 epochs

η is the learning rate, β_1 the first momentum of Adam, λ_c the classification weight, λ_s the stability weight, λ_r the final reconstruction weight, $\lambda_{rb,l}$ the intermediate reconstructions weights.
Exponential decrease follows the function $\exp(-5t^2)$ with $t \in [0, 1]$ from the start to the end of the decreasing interval. When increasing, t goes from 1 to 0.
Cosine decrease follows the function $\cos(\pi t) + 1$ with $t \in [0, 1]$ from the start to the end of the decreasing interval.

APPENDIX C

EXPERIMENTAL DETAILS FOR DUALDIS

Contents

C.1 Data pre-processing

C.1.1 CelebA

The official CelebA dataset [1] contains ∼200K images for 10,177 identities. As is common, we used the cropped and aligned version. However, the number of images per identity varies and some have very few images. Since our purpose is to work on datasets with two classification tasks, we chose to reduce the number of identities to 2,000, keeping those with the highest number of images. Because of this, we obtain a dataset with ∼60K images.

The identities are our label y and the attributes provided with the dataset are were not preprocessed and are used as z.

1. http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

C.1.2 Yale-B

The Extended Yale-B dataset[2] is available for download in two variants: the "full" version contains 16128 images of 28 human subjects under 9 poses and 64 illumination conditions, each image has a large part of background; and the "cropped" version contains ∼2400 images with 38 subjects and 64 illumination conditions with no background. We chose to work with the cropped version.

The 38 identities constitute our identity label $\mathbf{y}$. The lighting source information is provided as 2 real values indicating the angles (elevation and azimuth) of the light source. We propose to convert this information in 14 "clusters", we show the id of each cluster between 0 and 13 in this table:

Elevation \ Azimuth	-95	-35	-20	20	35	95
80		1	2	3		
20	4	5	6	7	8	
-20	9	10	11	12	13	
-80						

Outside: 0 (back light)

Each image is attributes to one of the clusters, and the label $\mathbf{z}$ is a one-hot vector indicating the lighting source.

C.1.3 NORB

The NORB dataset[3] "contains images of 50 toys belonging to 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. The objects were imaged by two cameras under 6 lighting conditions, 9 elevations (30 to 70 degrees every 5 degrees), and 18 azimuths (0 to 340 every 20 degrees)." We use the base dataset without jitter. The 5 categories are used as our classes $\mathbf{y}$, and a process similar to Yale-B is used to create the labels $\mathbf{z}$, but with soft assignment:

- The 6 lighting classes are converted into a single unit with values between 0 (dark) and 1 (very light) with mapping as follows: [0.6, 0.3, 0, 0.7, 0.4, 1]
- The elevation is represented by 3 clusters e_i of centers [35, 50, 65] with an assignment to each defined as $\mathbf{z}_i = 1 - \min(1, |elevation - e_i|/15)$

2. http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html
3. https://cs.nyu.edu/~ylclab/data/norb-v1.0/

- The azimuth is represented by 4 clusters a_j of center [0, 90, 180, 270] with an assignment to each defined as $\mathbf{z}_j = 1 - \min(1, |azimuth - a_j|/9)$

This gives us a complete vector $\mathbf{z}$ of size 8.

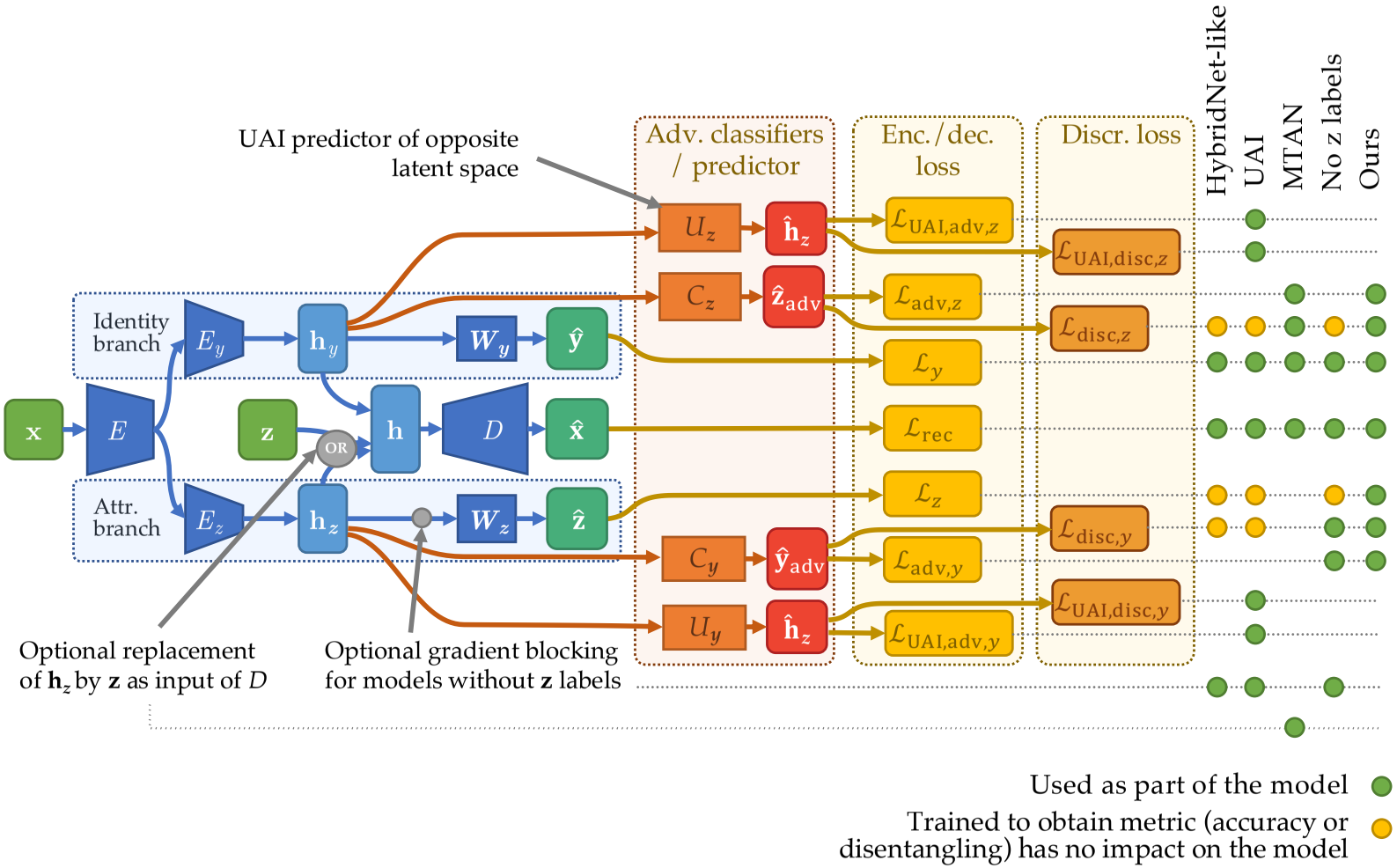
C.2 Architectures & Hyperparameters

C.2.1 Complete architecture overview

To obtain the different models that we report, we use start from a complete architecture with all possible options, and then choose which parts of the model we activate. This complete architecture is represented in Figure C.1.

In particular, we can see that we have:

- U_y and U_z which are the predictors that replace C_y and C_z for UAI model, and that predict $\mathbf{h}_y$ and $\mathbf{h}_z$. Their corresponding loss term is a MSE for both $\mathcal{L}_{\text{UAI,adv}}$ and $\mathcal{L}_{\text{UAI,disc}}$ that is maximized for $\mathcal{L}_{\text{UAI,adv}}$ and minimized for $\mathcal{L}_{\text{UAI,disc}}$.
- The possibility to replace $\mathbf{h}_z$ by $\mathbf{z}$ as the second input of the decoder D, which allows to produce MTAN.
- The possibility to block gradients between $\mathbf{h}_z$ and $\mathbf{W}_z$, which means that when activated we can train $\mathbf{W}_z$ to measure the quality of the representation $\mathbf{h}_z$ regarding the attributes, while not backpropagating this signal to E_z, therefore reproducing models that do not use $\mathbf{z}$ labels while keeping the metric.
- It is possible to train adversarial classifiers C_y and C_z with $\mathcal{L}_{\text{disc}}$ even when not proposed by the models, which allows to measure the quality of the disentangling and will have no impact of the actual model as long as $\mathcal{L}_{\text{adv}}$ are disabled since $\mathcal{L}_{\text{disc}}$ is only backpropagated in C.



For "HybridNet-like + attr" and "UAI + attr", simply remove the gradient blocking before $\mathbf{W}_z$

Figure C.1. – **Complete architecture with all possible options and variants**. We also show which loss terms are activated to reproduce the different models that we report.

C.2.2 Architecture details

In Table C.1, we provide the exact details about the architecture of the components in Figure C.1 depending on the experiments and the dataset, along with those general information:

- In the **encoder**, every layer is followed by batch normalization and ReLU.
- In the **decoder**, every layer is followed by a batch normalization and Leaky-ReLU(0.2), except last layer which has no activation or BN.
- In the **classifiers**, every intermediate layer is followed by a ReLU.
- **Layers are described using the following syntax:**
  - **Conv:** `128[k5][p0][s2]` is a convolutional layer with 128 output channels, a kernel of 5 (default is 3 if not written), padding of 0 (default is to keep same output size), stride 2 (default is 1)
  - **Deconv:** `dec128[k4][p0][s1]` is a transpose convolutional layer with 128 output channels, a kernel of 4 (default), padding of 0 (default is 1), stride 1 (default is 2) **Linear:** ℓ`128` is a linear layer with 128 output neurons
  - **Upsample:** `upsample` means an upsampling of a factor 2 using the nearest value
  - **MaxPool:** `maxpool2k3` is a max-pooling of stride 2 and kernel 3

C.2.3 Training and hyperparameters values

As a reminder, we have two losses composed of different loss terms, each weighted by a parameter λ that controls its importance. Here is the global loss:

$$\mathcal{L}_{\text{main}} = \lambda_{rec}\mathcal{L}_{\text{rec}} + \lambda_y\mathcal{L}_y + \lambda_z\mathcal{L}_z + \lambda_{adv,y}\mathcal{L}_{\text{adv},y} + \lambda_{adv,z}\mathcal{L}_{\text{adv},z} + \lambda_o\mathcal{L}_{\text{orth}}\,. \tag{C.1}$$

$$\mathcal{L}_{\text{disc}} = \lambda_{disc,y}\mathcal{L}_{\text{disc},y} + \lambda_{disc,z}\mathcal{L}_{\text{disc},z}\,. \tag{C.2}$$

This loss is optimized using Adam with the recommended hyperparamters: learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$. The number of epochs and batch sizes depends on the datasets and are given below.

For all the experiments, we used $\lambda = 1$ for all the classification losses: $\lambda_y = \lambda_z = \lambda_{disc,z} = \lambda_{disc,z} = \lambda_{UAI,disc,z} = \lambda_{UAI,disc,z} = 1$, and we set $\lambda_o = 1 \times 10^{-6}$

	Architecture for UAI	Architecture for all other models
	CelebA (image size 256×256)	
E	32pos2, 32pos1, 64p0, maxpool2k3, 80k1, maxpool2k3, 96p0, maxpool2k3, 128p0, 160pos2, 196p0	32pos2, 32pos1, 64p0, maxpool2k3, 80k1, maxpool2k3, 96p0, maxpool2k3
E_y/E_z	196p0	96p0, 128pos2, 196p0, 196p0
D	dec392pos1, 392, upsample, 392, upsample, 256, upsample, 196, upsample, 128, 128, upsample, 96, 96, upsample, 64, 64, 32, 3k1	
$\mathbf{W}_y$	ℓ2000	ℓ2000
$\mathbf{W}_z$	ℓ40	ℓ40
C_y	ℓ256, ℓ2000	ℓ256, ℓ256, ℓ2000
C_z	ℓ256, ℓ40	ℓ256, ℓ256, ℓ40
U_y	ℓ196	N/A
U_z	ℓ196	N/A
	Yale-B (image size 64×64)	
E	32k4s2, 40k4s2, 48k4s2, 76k4s2, 100k3p0	32k4s2, 40k4s2, 48k4s2
E_y/E_z	80k2p0	64k4s2, 72k3p0, 80k2p0
D	160k2p1, dec80, dec64, dec48, dec32, dec32, 32, 3none	
$\mathbf{W}_y$	ℓ38	ℓ38
$\mathbf{W}_z$	ℓ14	ℓ14
C_y	ℓ80, ℓ80, ℓ38	ℓ80, ℓ80, ℓ38
C_z	ℓ80, ℓ80, ℓ14	ℓ80, ℓ80, ℓ14
U_y	ℓ80	N/A
U_z	ℓ80	N/A
	NORB (image size 64×64)	
E	64k4s2, 64k4s2, 96k4s2, 164k4s2, 192k4s2	64k4s2, 64k4s2, 96k4s2
E_y/E_z	128k2p0	96k4s2, 128k3p0, 128k2p0
D	256k2p1, dec192, 128, dec128, 128, dec96, 96, dec64, 64, dec64, 64, 32, 1	
$\mathbf{W}_y$	ℓ5	ℓ5
$\mathbf{W}_z$	ℓ8	ℓ8
C_y	ℓ128, ℓ128, ℓ5	ℓ128, ℓ128, ℓ5
C_z	ℓ128, ℓ128, ℓ8	ℓ128, ℓ128, ℓ8
U_y	ℓ128	N/A
U_z	ℓ128	N/A

Table C.1. – **Architectures used for our experiments**.

Values of hyperparameters that depends on the dataset are provided in Table C.2.

	λ_{rec}	$\lambda_{adv,y}$	$\lambda_{adv,z}$	$\lambda_{UAI,adv,y}$	$\lambda_{UAI,adv,z}$	Batch size	Epochs
CelebA	0.3	0.1	0.1	0.3	0.3	32	330
Yale-B	1	0.08	0.08	0.3	0.3	64	400
NORB	10	0.25	0.25	0.3	0.3	128	250

Table C.2. – **Hyperparameters for the various experiements**.

C.3 Experiments details

C.3.1 Image editing

For image editing, we use the model trained for the ablation study of the datasets, and start by obtaining the representations $\mathbf{h}_y$ and $\mathbf{h}_z$ for some test images. For attribute modification, we move $\mathbf{h}_z$ in the direction i of each vector $\mathbf{w}_{z_i}$ to obtain $\mathbf{h}'_z = \mathbf{h}_z \pm \varepsilon \mathbf{w}_{z_i}$ of the model and produce images using the decoder: $\hat{\mathbf{x}} = D(\mathbf{h}_y, \mathbf{h}'_z)$. The amplitude of ε for the visualization was fixed after a quick visual check of what values looked. For identity / attributes mixing between images, we simply use $\mathbf{h}_y$ and $\mathbf{h}_z$ from different images and feed them to the decoder.

C.3.2 Semi-supervised learning

For semi-supervised learning, we use batches with a pre-defined number of supervised images in each batch. We iterate over the set of labeled and unlabeled images independently and consider an epoch as the loop over the unlabeled image set, during which we usually see images of the supervised set more than once depending on the size of the sets. This is a common setting, e.g. (Sajjadi et al. 2016; Tarvainen and Valpola 2017; Robert et al. 2018). The hyperparameters that differ from the model in the ablation study are provided in Table C.3.

	λ_{rec}	λ_z	$\lambda_{adv,y}$	$\lambda_{adv,z}$	Sup. batch size
4000	0.3	0.4	0.2	0.1	10
2000	0.5	0.4	0.2	0.1	10
1000	0.5	0.4	0.2	0.1	8
400	0.5	0.4	0.2	0.1	8

Table C.3. – **Hyperparameters for the SSL experiements**. "Sup. batch size" indicates the number of labeled images in each batch.

C.3.3 Data Augmentation on Yale-B

For the Data Augmentation (DA) experiments, we start by training a new model with different sizes of train datasets. Once trained, we produce 150 new images for each identity using the image editing technique we described in order to produce new images of the different attribute categories. For this, we iterate over the train images of each identity (after excluding train images with attributes that correspond to very bad lighting, *i.e.* attributes 0, 4, 8, 9, 13) and then randomly choose an attribute for which we do not already have enough images for this identity. This is done so that after DA, each identity has images that follows this distribution $\mathcal{D}$ over attributes 0 to 13:

$$\mathcal{D} = [1, 3, 3, 2, 5, 3, 10, 3, 5, 2, 2, 2, 2, 2]/45 \tag{C.3}$$

Then, a classifier with the architecture $\mathbf{W}_y \circ E_y \circ E$ is trained with Adam for 400 epochs on the original train set used for to train the generator + the generated images. It is then evaluated on all the remaining images of the original dataset.