

HybridNet: Classification and Reconstruction Cooperation for Semi-Supervised Learning

Thomas Robert¹, Nicolas Thome², and Matthieu Cord¹

¹ Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

{thomas.robert,matthieu.cord}@lip6.fr

² CEDRIC - Conservatoire National des Arts et Métiers, 75003 Paris, France
nicolas.thome@cnam.fr

Abstract. In this paper, we introduce a new model for leveraging unlabeled data to improve generalization performances of image classifiers: a two-branch encoder-decoder architecture called HybridNet. The first branch 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. HybridNet is able to outperform state-of-the-art results on CIFAR-10, SVHN and STL-10 in various semi-supervised settings. In addition, visualizations and ablation studies validate our contributions and the behavior of the model on both CIFAR-10 and STL-10 datasets.

Keywords: Deep learning, semi-supervised learning, regularization, reconstruction, invariance and stability, encoder-decoder

1 Introduction

Deep learning and Convolutional Neural Networks (ConvNets) have shown impressive state-of-the-art results in the last years on various visual recognition tasks, *e.g.* image classification [1,2,3], object localization [4,5,6], image segmentation [7] and even multimodal embedding [8,9,10]. Some key elements are the use of very deep models with a huge number of parameters and the availability of large-scale datasets such as ImageNet. When dealing with smaller datasets, however, the need for proper regularization methods becomes more crucial to control overfitting [11,12,13,14]. An appealing direction to tackle this issue is to take advantage of the huge number of unlabeled data by developing semi-supervised learning techniques.

Many approaches attempt at designing semi-supervised techniques where the unsupervised cost produces encoders that have high data-likelihood or small reconstruction error [15]. This strategy has been followed by historical deep learning approaches [16], but also in some promising recent results with modern

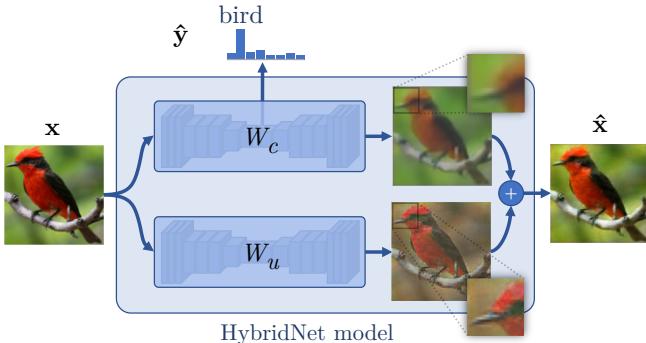


Fig. 1. Illustration of HybridNet behavior: the input image is processed by two network paths of weights W_c and W_u ; each path produces a partial reconstruction, and both are summed to produce the final reconstruction, while only one path is used to produce a classification prediction. Thanks to a joint training of both tasks, the weights W_c and W_u influence each other to cooperate

ConvNets [17,18]. However, the unsupervised term in reconstruction-based approaches arguably conflicts with the supervised loss, which requires intra-class invariant representations. This is the motivation for designing auto-encoders that are able to discard information, such as the Ladder Networks [19].

Another interesting regularization criterion relies on stability. Prediction functions which are stable under small input variations are likely to generalize well, especially when training with small amounts of data. Theoretical works have shown the stability properties of some deep models, *e.g.* by using harmonic analysis for scattering transforms [20,21] or for Convolution Kernel Machines [22]. In addition, recent semi-supervised models incorporate a stability-based regularizer on the prediction [23,24,25].

In this paper, we propose a new approach for regularizing ConvNets using unlabeled data. The behavior of our model, called HybridNet, is illustrated in Fig. 1. It consists in a “hybrid” auto-encoder with the feature extraction path decomposed into two branches.

The top branch encoder, of parameters W_c , is connected to a classification layer that produces class predictions while the decoder from this branch is used to partly reconstruct the input image from the discriminative features, leading to \hat{x}_c . Since those features are expected to extract invariant class-specific patterns, information is lost and exact reconstruction is not possible. To complement it, a second encoder-decoder branch of parameters W_u is added to produce a complementary reconstruction \hat{x}_u such that the sum $\hat{x} = \hat{x}_c + \hat{x}_u$ is the final complete reconstruction.

During training, the supervised classification cost impact W_c while an unsupervised reconstruction cost is applied to both W_c and W_u to properly reconstruct the input image. The main assumption behind HybridNet is that the two-path architecture helps in making classification and reconstruction cooper-

ate. To encourage this, we use additional costs and training techniques, namely a stability regularization in the discriminative branch and a branch complementarity training method.

2 Related Work

Training deep models with relatively small annotated datasets is a crucial issue nowadays. To this end, the design of proper regularization techniques plays a central role. In this paper, we address the problem of taking advantage of unlabeled data for improving generalization performances of deep ConvNets with semi-supervised learning [26].

One standard goal followed when training deep models with unlabeled data consists in designing models which fit input data well. Reconstruction error is the standard criterion used in (possibly denoising) Auto-Encoders [15,27,28,29], while maximum likelihood is used with generative models, *e.g.* Restricted Boltzmann Machines, Deep Belief Networks or Deep Generative Models [16,30,31,32]. This unsupervised training framework was generally used as a pre-training before supervised learning with back-propagation [33], potentially with an intermediate step [34]. The currently very popular Generative Adversarial Networks [35] also falls into this category. With modern ConvNets, regularization with unlabeled data is generally formulated as a multi-task learning problem, where reconstruction and classification objectives are combined during training [17,18,36]. In these architectures, the encoder used for classification is regularized by a decoder dedicated to reconstruction.

This strategy of classification and reconstruction with an Auto-Encoder is however questionable, since classification and reconstruction may play contradictory roles in terms of feature extraction. Classification arguably aims at extracting invariant class-specific features, improving sample complexity of the learned model [37], therefore inducing an information loss which prevents exact reconstruction. Ladder Networks [19] have historically been designed to overcome the previously mentioned conflict between reconstruction and classification, by designing Auto-Encoders capable of discarding information. Reconstruction is produced using higher-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 lose some information since it must be able to correct low-level errors that require details.

Another interesting regularization criterion relies on stability or smoothness of the prediction function, which is at the basis of interesting unsupervised training methods, *e.g.* Slow Feature Analysis [38]. Adding stability to the prediction function was studied in Adversarial Training [39] for supervised learning and further extended to semi-supervised learning in the Virtual Adversarial Training method [40]. Other recent semi-supervised models incorporate a stability-based regularizer on the prediction. The idea was first introduced by [23] and proposes to make the prediction vector stable toward data augmentation (translation, ro-

tation, shearing, noise, *etc.*) and model stochasticity (dropout) for a given input. Following work [24,25] slightly improves upon it by proposing variants on the way to compute stability targets to increase their consistency and better adapt to the model’s evolution over training.

When using large modern ConvNets, the problem of designing decoders able to invert the encoding still is an open question [41]. The usual solution is to mirror the architecture of the encoder by using transposed convolutions [42]. This problem is exacerbated with irreversible pooling operations such as max-pooling that must be reversed by an upsampling operation. In [17,18], 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 [43] and i-RevNet [44], being inspired by second generation of bi-orthogonal multi-resolution analysis and wavelets [45] from the signal processing literature.

To sum up, using reconstruction as a regularization cost added to classification is an appealing idea but the best way to efficiently use it as a regularizer is still an open question. As we have seen, when applied to an auto-encoding architecture [17,18], reconstruction and classification would compete. To overcome the aforementioned issues, we propose HybridNet, a new framework for semi-supervised learning. Presented on Fig. 2, this framework can be seen as an extension of the popular auto-encoding architecture. In HybridNet, the usual auto-encoder that does both classification and reconstruction is assisted by an additional auto-encoder so that the first one is allowed to discard information in order to produce intra-class invariant features while the second one retains the lost information. The combination of both branches then produces the reconstruction. This way, our architecture prevents the conflict between classification and reconstruction and allows the two branches to cooperate and accomplish both classification and reconstruction tasks.

Compared to Ladder Networks [19], our two-branch approach without direct skip connection allows for a finer and learned information separation and is thus expected to have a more favorable impact in terms of discriminative encoder regularization. Our HybridNet model also has conceptual connections with wavelet decomposition [46]: 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. HybridNet also differs from reversible models [43,44] by the explicit decomposition between supervised and unsupervised signals, enabling the discriminative encoder to have fewer parameters and better sample complexity.

In this paper, our contributions with the HybridNet framework are twofold: first, in Section 3.1, we propose an architecture designed to efficiently allow both reconstruction and classification losses to cooperate; second, in Section 3.2, we design a training loss adapted to it that includes reconstruction, stability in the discriminative branch and a branch complementarity technique. In Section 4, we perform experiments to show that HybridNet is able to outperform state-

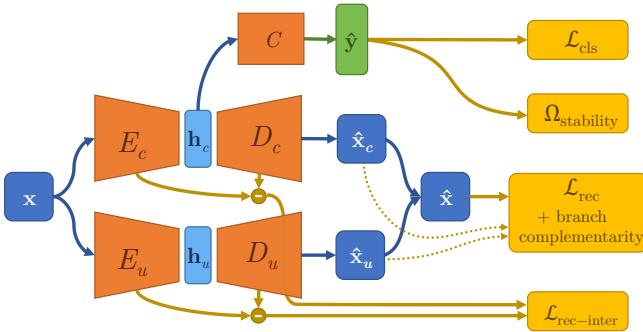


Fig. 2. General description of the HybridNet framework. E_c and C correspond to a classifier, E_c and D_c form an autoencoder that we call *discriminative path*, and E_u and D_u form a second autoencoder called *unsupervised path*. The various loss functions used to train HybridNet are also represented in yellow

of-the-art results in various semi-supervised settings on CIFAR-10, SVHN and STL-10. We also provide ablation studies validating the favorable impact of our contributions. Finally, we show several visualizations on CIFAR-10 and STL-10 datasets analogous to Fig. 1 to validate the behavior of both branches, with a discriminative branch that loses information that is restored by the second branch.

3 HybridNet: a semi-supervised learning framework

In this section, we detail the proposed HybridNet model: the chosen architecture to mix supervised and unsupervised information efficiently in Section 3.1, and the semi-supervised training method adapted to this particular architecture in Section 3.2.

3.1 Designing the HybridNet architecture

General architecture. As we have seen, classification requires intra-class invariant features while reconstruction needs to retain all the information. To circumvent this issue, HybridNet is composed of two auto-encoding paths, the *discriminative path* (E_c and D_c) and the *unsupervised path* (E_u and D_u). Both encoders E_c and E_u take an input image x and produce representations h_c and h_u , while decoders D_c and D_u take respectively h_c and h_u as input to produce two partial reconstructions \hat{x}_c and \hat{x}_u . Finally, a classifier C produces a class prediction using discriminative features only: $\hat{y} = C(h_c)$. Even if the two paths can have similar architectures, they should play different and complementary roles. The discriminative path must extract discriminative features h_c that should eventually be well crafted to perform a classification task effectively, and produce a purposely partial reconstruction \hat{x}_c that should not be perfect

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 . This way, it can produce a complementary reconstruction $\hat{\mathbf{x}}_u$ so that, when merging $\hat{\mathbf{x}}_u$ and $\hat{\mathbf{x}}_c$, the final reconstruction $\hat{\mathbf{x}}$ is close to \mathbf{x} . The HybridNet architecture, visible on Fig. 2, can be described by the following equations:

$$\begin{aligned} \mathbf{h}_c &= E_c(\mathbf{x}) & \hat{\mathbf{x}}_c &= D_c(\mathbf{h}_c) & \hat{\mathbf{y}} &= C(\mathbf{h}_c) \\ \mathbf{h}_u &= E_u(\mathbf{x}) & \hat{\mathbf{x}}_u &= D_u(\mathbf{h}_u) & \hat{\mathbf{x}} &= \hat{\mathbf{x}}_c + \hat{\mathbf{x}}_u \end{aligned} \quad (1)$$

Note that the end-role of reconstruction is just to act as a regularizer for the discriminative encoder. The main challenge and contribution of this paper is to find a way to ensure that the two paths 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 co-operate and contribute to the reconstruction. Indeed, with such an architecture, we could end up with two paths that work independently: a classification path $\hat{\mathbf{y}} = C(E_c(\mathbf{x}))$ and a reconstruction path $\hat{\mathbf{x}} = \hat{\mathbf{x}}_u = D_u(E_u(\mathbf{x}))$ and $\hat{\mathbf{x}}_c = 0$. We address both those issues through the design of the architecture of the encoders and decoders as well as an appropriate loss and training procedure.

Branches design. 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 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.* [17,19,47].

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, that can be reversed either by upsampling or unpooling. An upsampling will increase the spatial size of a feature map without any additional information while an unpooling, used in [17,18], will use spatial information (*pooling switches*) from the corresponding max-pooling layer to do the upsampling. 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. An example of HybridNet architecture is presented in Fig. 3.

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 in different ways by 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.

3.2 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 will address the question of training this architecture efficiently. The complete loss is composed of various terms as illustrated on Fig. 2. It comprises terms for classification with \mathcal{L}_{cls} ; final reconstruction with \mathcal{L}_{rec} ; intermediate reconstructions with $\mathcal{L}_{\text{rec-inter}b,l}$ (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 :

$$\mathcal{L} = \lambda_c \mathcal{L}_{\text{cls}} + \lambda_r \mathcal{L}_{\text{rec}} + \sum_{b \in \{c,u\}, l} \lambda_{rb,l} \mathcal{L}_{\text{rec-inter}b,l} + \lambda_s \Omega_{\text{stability}}. \quad (2)$$

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

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:

$$\ell_{\text{cls}} = \ell_{\text{CE}}(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_i \mathbf{y}_i \log \hat{\mathbf{y}}_i, \quad \mathcal{L}_{\text{cls}} = \frac{1}{n_s} \sum_k \ell_{\text{cls}}(\hat{\mathbf{y}}^{(k)}, \mathbf{y}^{(k)}). \quad (3)$$

Reconstruction losses. In HybridNet, we chose 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 merge by addition in pixel space is also analogous to wavelet decomposition where the signal is decomposed into low- and high-pass branches that are then decoded and summed in pixel space. The reconstruction loss that we use is a simple mean-squared error between the input

and the sum of the partial reconstructions:

$$\ell_{\text{rec}} = \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 = \|\hat{\mathbf{x}}_u + \hat{\mathbf{x}}_c - \mathbf{x}\|_2^2, \quad \mathcal{L}_{\text{rec}} = \frac{1}{n} \sum_k \ell_{\text{rec}}(\hat{\mathbf{x}}^{(k)}, \mathbf{x}^{(k)}) . \quad (4)$$

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 [17,18,19] 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 would contribute to the final reconstruction. This is applied in both branches ($b \in \{c, u\}$):

$$\mathcal{L}_{\text{rec-inter } b,l} = \frac{1}{n} \sum_k \|\hat{\mathbf{h}}_{b,l}^{(k)} - \mathbf{h}_{b,l}^{(k)}\|_2^2 . \quad (5)$$

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.

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:

$$\ell_{\text{rec-balanced}} = \begin{cases} \|\hat{\mathbf{x}}_u + \text{stopgrad}(\hat{\mathbf{x}}_c) - \mathbf{x}\|_2^2 & \text{if } \|\hat{\mathbf{x}}_u - \mathbf{x}\|_2^2 \geq \|\hat{\mathbf{x}}_c - \mathbf{x}\|_2^2 \\ \|\text{stopgrad}(\hat{\mathbf{x}}_u) + \hat{\mathbf{x}}_c - \mathbf{x}\|_2^2 & \text{otherwise} \end{cases} . \quad (6)$$

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.

In order to further encourage this behavior, we propose to use a *stability regularizer*. Such a regularizer is currently at the core of the models that

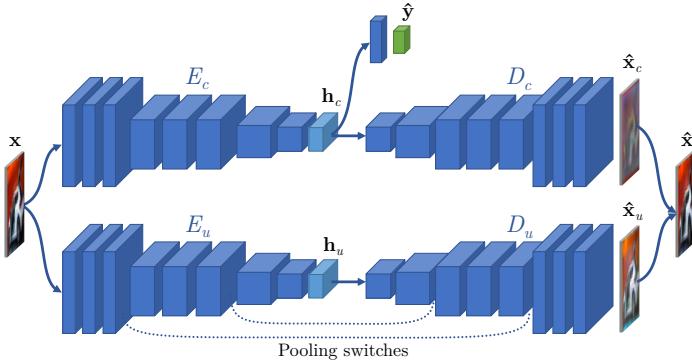


Fig. 3. Example of HybridNet architecture where an original classifier (ConvLarge) constitutes E_c and has been mirrored to create D_c and duplicated for E_u and D_u , with the addition of unpooling in the discriminative branch

give state-of-the-art results in semi-supervised setting on the most common datasets [23,24,25]. The principle is to encourage the classifier’s output prediction $\hat{y}^{(k)}$ 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 squared euclidean distance between the output $\hat{y}^{(k)}$ and a “stability” target $\mathbf{z}^{(k)}$. Multiple methods have been proposed to compute such a target [23,24,25], 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:

$$\Omega_{\text{stability}} = \frac{1}{n} \sum_k \|\hat{y}^{(k)} - \mathbf{z}^{(k)}\|_2^2. \quad (7)$$

By applying this loss on \hat{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.

In the experiments, we tried both Temporal Ensembling [24] and Mean Teacher [25] methods and did not see a major difference. In Temporal Ensembling, the target $\mathbf{z}^{(k)}$ is a moving average of the $\hat{y}^{(k)}$ over the previous pass of $\mathbf{x}^{(k)}$ in the network during training; while in Mean Teacher, $\mathbf{z}^{(k)}$ is the output of a secondary model where weights are a moving average of the weights of the model being trained.

4 Experiments

In this section, we will study and validate the behavior of our novel framework. We first perform 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.

In these experiments, we use three image datasets: SVHN [48], CIFAR-10 [49] and STL-10 [50]. Both SVHN and CIFAR-10 are 10-classes datasets of 32×32 pixels images. SVHN has 73,257 images for training, 26,032 for testing and 531,131 extra images used only as unlabeled data. CIFAR-10 has 50,000 training images and 10,000 testing images. For our semi-supervised experiments, we only keep N labeled training samples (with $N/10$ samples per class) while the rest of the data is kept unlabeled, as is commonly done. STL-10 have the same 10 classes as CIFAR-10 with 96×96 pixels images. It is designed for semi-supervised learning since it contains 10 folds of 1,000 labeled training images, 100,000 unlabeled training images and 8,000 test images with labels.

4.1 HybridNet framework validation

First, we 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.

This study was mainly performed using the ConvLarge architecture [19] on CIFAR-10 since it's a very common setup used in recent semi-supervised experiments [23,24,25]. The design of the HybridNet version of this architecture follows Section 3 (illustrated in Fig. 3) and uses Temporal Ensembling to produce stability targets \mathbf{z} . Additional results are provided using an adapted version of ConvLarge for STL-10 with added blocks of convolutions and pooling.

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 (Eq. (2)) could have been optimized on a validation set but for these experiments they were simply 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. All the details of the experiments – exact architecture, hyperparameters, optimization, *etc.* – are provided in the appendix.

Ablation study of the architecture. We start this analysis by validating our architecture with an ablation study on CIFAR-10 with various 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 1. 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 use E_c and C ; and the auto-encoder (similar to [17]) 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.

Table 1. Ablation study performed on CIFAR-10 with ConvLarge architecture

Model	Labeled samples		
	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

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 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.

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 3.2, both through classification results and visualizations.

First, in Table 2a we show 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 ~5 pts gains can be combined to work in concert and reach the final score line *i* of a ~10 pts gain.

Second, to interpret how the branches behave we propose to visualizing the different reconstructions \hat{x}_c , \hat{x}_u and \hat{x} for different combinations of loss terms in

Table 2. Detailed ablation studies when activating different terms and techniques of the HybridNet learning. These results are obtained with ConvLarge on CIFAR-10 with 2000 labeled samples and ConvLarge-like on STL-10 with 1000 labeled samples

(a) Test accuracy (%) (b) Visualization of partial and combined reconstructions

	$\mathcal{L}_{\text{classify}}$	$\mathcal{P}_{\text{stability}}$	$\mathcal{L}_{\text{rec}}(\text{hybrid})$	$\mathcal{L}_{\text{rec-inter}}$	$\mathcal{L}_{\text{rec-balanced}}$	CIFAR-10	STL-10	$\mathcal{L}_{\text{rec}}(\text{hybrid})$	$\mathcal{L}_{\text{rec-inter}}$	$\mathcal{L}_{\text{rec-balanced}}$	$\mathcal{P}_{\text{stability}}$	\mathbf{x}	$\hat{\mathbf{x}}_c$	$\hat{\mathbf{x}}_u$	$\hat{\mathbf{x}}$	\mathbf{x}	$\hat{\mathbf{x}}_c$	$\hat{\mathbf{x}}_u$	$\hat{\mathbf{x}}$	\mathbf{x}	$\hat{\mathbf{x}}_c$	$\hat{\mathbf{x}}_u$	$\hat{\mathbf{x}}$		
a	✓					71.5	65.6																		
b	✓	✓				74.6	69.8																		
c	✓		✓			72.4	67.8																		
d	✓		✓	✓		74.0	–																		
e	✓		✓	✓	✓	75.2	–																		
f	✓	✓	✓			77.7	71.5																		
g	✓	✓	✓			77.4	–																		
h	✓	✓	✓	✓		80.8	72.2																		
i	✓	✓	✓	✓	✓	81.6	74.1																		

Table 2b. 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 branch to produce a weak reconstruction (lines *d*) and is complemented by the branch balancing technique (lines *e*) to produce balanced reconstructions in both branch. 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.

Visualization of information separation on CIFAR-10 and STL-10. Overall, we can see in Table 2b 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 type 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 Fig. 4. These confirm the observations on CIFAR-10 with a very good final reconstruction composed of a rough reconstruction that lacks texture and color details from the discriminative branch, completed by low-level details of shape, texture, writings, color correction and background information from the unsupervised branch.

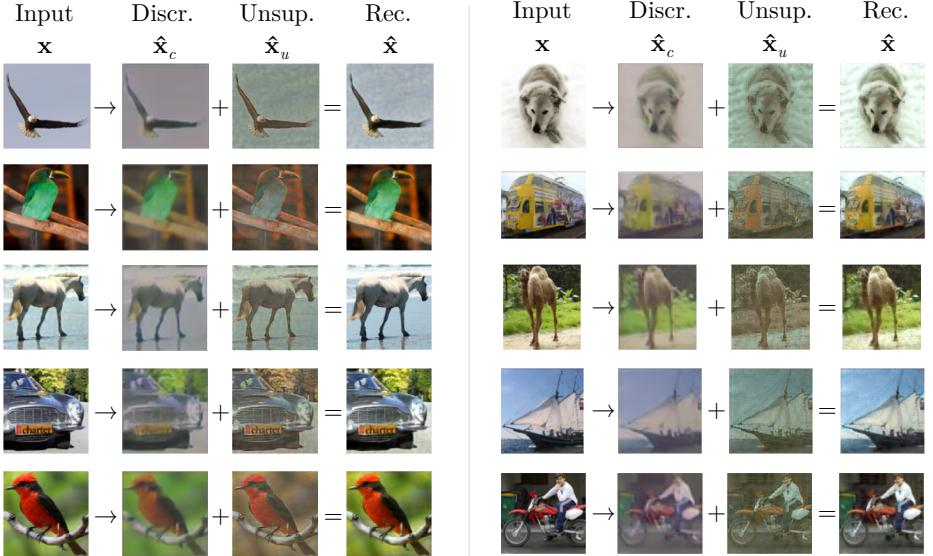


Fig. 4. Visualizations of input, partial and final reconstructions of STL-10 images using a HybridNet model derived from a ConvLarge-like architecture

4.2 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 semi-supervised learning 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 [51], which is used in Mean Teacher [25] 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 3.2, including Mean Teacher to produce stability targets $\mathbf{z}^{(k)}$. The training protocol follow exactly the protocol of Mean Teacher [25] for CIFAR-10 and a similar one for SVHN and STL-10 for which [25] 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 (we tried values 0.25, 0.5 and 1.0 for both of them). Details are in the appendix.

The results of these experiments are presented in Table 3. 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 [25], with an improvement of

Table 3. Results on CIFAR-10, STL-10 and SVHN using a ResNet-based HybridNet. “Mean Teacher ResNet” is our classification & stability baseline; results marked with * are not reported in the original paper and were obtained ourselves

Dataset	CIFAR-10			SVHN		STL-10	
	1000	2000	4000	500	1000	1000	1000
SWWAE [17]						23.56	25.67
Ladder Network [19]			20.40				
Improved GAN [53]	21.83	19.61	18.63	18.44	8.11		
CatGAN [52]			19.58				
Stability regularization [23]			11.29	6.03			
Temporal Ensembling [24]			12.16	5.12	4.42		
Mean Teacher ConvLarge [25]	21.55	15.73	12.31	4.18	3.95		
Mean Teacher ResNet [25]	10.10		6.28	*2.33	*2.05	*16.8	
ResNet baseline [51]	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	

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 [23,24], and over the Ladder Networks [19] and GAN-based techniques [52,53].

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

5 Conclusion

In this paper, we described a novel semi-supervised framework called HybridNet that proposes 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. This architecture is accompanied by a loss and training technique that allows the architecture to behave in the desired way. In the experiments, we validate the significant performance boost brought by HybridNet in comparison with several other common architectures that use reconstruction losses and stability. We also show that HybridNet is able to produce state-of-the-art results on multiple datasets.

With two latent representations that explicitly encode classification information on one side and the remaining information on the other side, our model may be seen as a competitor to the fully reversible RevNets models recently proposed, that implicitly encode both types of information. We plan to further explore the relationships between these approaches.

Acknowledgements. This work was funded by grant DeepVision (ANR-15-CE23-0029-02, STPGP-479356-15), a joint French/Canadian call by ANR & NSERC.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS). (2012)
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
3. Durand, T., Mordan, T., Thome, N., Cord, M.: WILDCAT: Weakly Supervised Learning of Deep ConvNets for Image Classification, Pointwise Localization and Segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
4. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems (NIPS). (2016)
5. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
6. Mordan, T., Thome, N., Cord, M., Henaff, G.: Deformable Part-based Fully Convolutional Network for Object Detection. In: British Machine Vision Conference (BMVC). (2017)
7. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2018)
8. Engilberge, M., Chevallier, L., Pérez, P., Cord, M.: Finding beans in burgers: Deep semantic-visual embedding with localization. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
9. Carvalho, M., Cadène, R., Picard, D., Soulier, L., Thome, N., Cord, M.: Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. Special Interest Group on Information Retrieval (SIGIR) (2018)
10. Ben-Younes, H., Cadène, R., Thome, N., Cord, M.: Mutan: Multimodal tucker fusion for visual question answering. IEEE International Conference on Computer Vision (ICCV) (2017)
11. Krogh, A., Hertz, J.A.: A simple weight decay can improve generalization. In: Advances in Neural Information Processing Systems (NIPS). (1992)
12. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. Journal of Machine Learning Research (JMLR) (2016)
13. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research (JMLR) (2014)
14. Blot, M., Robert, T., Thome, N., Cord, M.: Shade: Information-based regularization for deep learning. In: IEEE International Conference on Image Processing (ICIP). (2018)
15. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems (NIPS). (2007)
16. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science (2006)

17. Zhao, J., Mathieu, M., Goroshin, R., LeCun, Y.: Stacked What-Where Auto-encoders. In: International Conference on Learning Representations Workshop (ICLR-W). (2016)
18. Zhang, Y., Lee, K., Lee, H.: Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In: International Conference on Machine Learning (ICML). (2016)
19. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., Raiko, T.: Semi-supervised learning with ladder networks. In: Advances in Neural Information Processing Systems (NIPS). (2015)
20. Mallat, S.: Group invariant scattering. Communications on Pure and Applied Mathematics (CPAM) (2012)
21. Bruna, J., Mallat, S.: Invariant scattering convolution networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2013)
22. Bietti, A., Mairal, J.: Group Invariance, Stability to Deformations, and Complexity of Deep Convolutional Representations. In: Advances in Neural Information Processing Systems (NIPS). (2017)
23. Sajjadi, M., Javanmardi, M., Tasdizen, T.: Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In: Advances in Neural Information Processing Systems (NIPS). (2016)
24. Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. In: International Conference on Learning Representations (ICLR). (2017)
25. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: Advances in Neural Information Processing Systems (NIPS). (2017)
26. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison (2005)
27. Ranzato, M., Szummer, M.: Semi-supervised learning of compact document representations with deep networks. In: International Conference on Machine Learning (ICML). (2008)
28. Ranzato, M., Huang, F.J., Boureau, Y.L., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2007)
29. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: International Conference on Machine Learning (ICML). (2008)
30. Ranzato, M., Poultney, C., Chopra, S., Lecun, Y.: Efficient learning of sparse representations with an energy-based model. In: Advances in Neural Information Processing Systems (NIPS). (2007)
31. Larochelle, H., Bengio, Y.: Classification using discriminative restricted boltzmann machines. In: International Conference on Machine Learning (ICML). (2008)
32. Kingma, D.P., Mohamed, S., Rezende, D.J., Welling, M.: Semi-supervised learning with deep generative models. In: Advances in Neural Information Processing Systems (NIPS). (2014)
33. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? Journal of Machine Learning Research (JMLR) (2010)
34. Goh, H., Thome, N., Cord, M., Lim, J.H.: Top-down regularization of deep belief networks. In: Advances in Neural Information Processing Systems (NIPS). (2013)
35. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (NIPS). (2014)

36. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I.: Adversarial autoencoders. In: International Conference on Learning Representations (ICLR). (2016)
37. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer (2009)
38. Thériault, C., Thome, N., Cord, M.: Dynamic Scene Classification: Learning Motion Descriptors with Slow Features Analysis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2013)
39. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (ICLR). (2015)
40. Miyato, T., Maeda, S.i., Koyama, M., Nakae, K., Ishii, S.: Distributional smoothing with virtual adversarial training. In: International Conference on Learning Representations (ICLR). (2016)
41. Wojna, Z., Ferrari, V., Guadarrama, S., Silberman, N., Chen, L.C., Fathi, A., Uijlings, J.: The devil is in the decoder. In: British Machine Vision Conference (BMVC). (2017)
42. Dumoulin, V., Visin, F.: A guide to convolution arithmetic for deep learning. Technical Report (2016)
43. Gomez, A.N., Ren, M., Urtasun, R., Grosse, R.B.: The reversible residual network: Backpropagation without storing activations. In: Advances in Neural Information Processing Systems (NIPS). (2017)
44. Jacobsen, J.H., Smeulders, A., Oyallon, E.: i-RevNet: Deep Invertible Networks. In: International Conference on Learning Representations (ICLR). (2018)
45. Sweldens, W.: The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In: Wavelet Applications in Signal and Image Processing III. (1995)
46. Mallat, S.G., Peyré, G.: A wavelet tour of signal processing : the sparse way. Academic Press (2009)
47. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision (ECCV). (2014)
48. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning. (2011)
49. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical Report (2009)
50. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: International Conference on Artificial Intelligence and Statistics (AISTATS). (2011)
51. Gastaldi, X.: Shake-shake regularization of 3-branch residual networks. In: International Conference on Learning Representations Workshop (ICLR-W). (2017)
52. Springenberg, J.T.: Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. In: International Conference on Learning Representations (ICLR). (2016)
53. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., Chen, X.: Improved techniques for training gans. In Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., eds.: Advances in Neural Information Processing Systems (NIPS). Curran Associates, Inc. (2016)

A Additional visualizations of HybridNet

Additional visualizations of HybridNet behavior are presented on Fig. 5 for CIFAR-10 and Fig. 6 for STL-10.

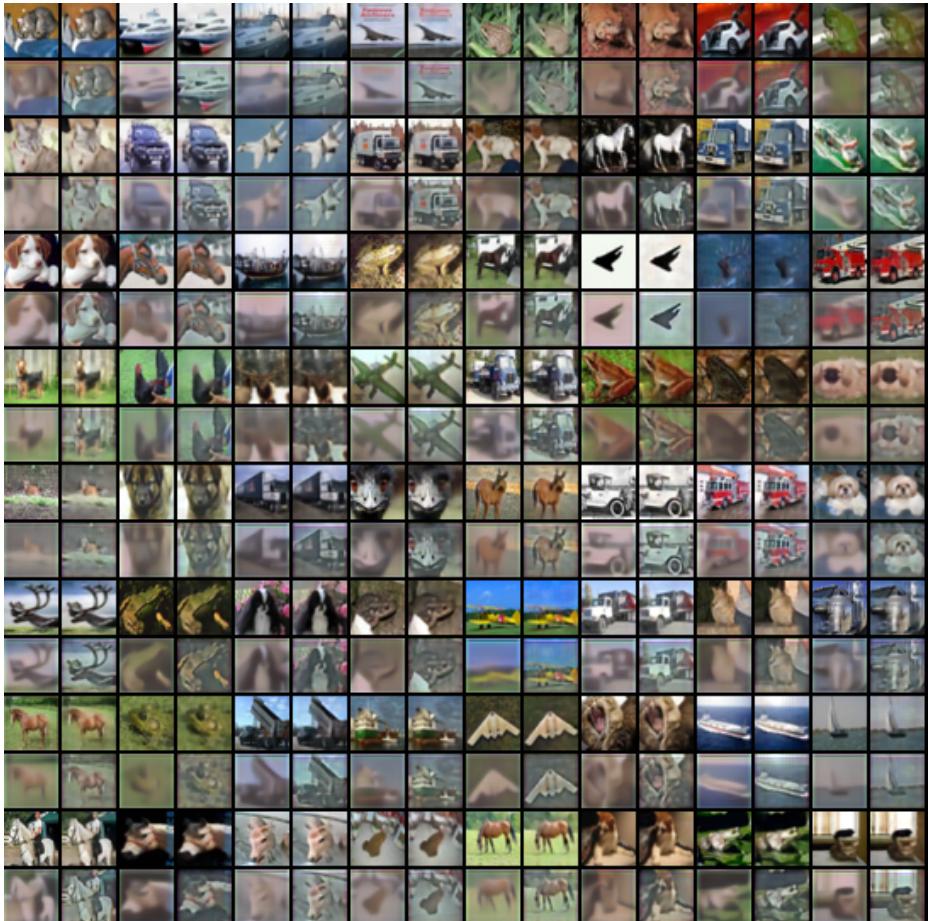


Fig. 5. Example of visualizations for a ConvLarge-based HybridNet (complete training loss) on CIFAR-10. For each input image, there is block of 4 images on the figure with the following organization: $\begin{bmatrix} \mathbf{x} & \hat{\mathbf{x}} \\ \hat{\mathbf{x}}_c & \hat{\mathbf{x}}_u \end{bmatrix}$

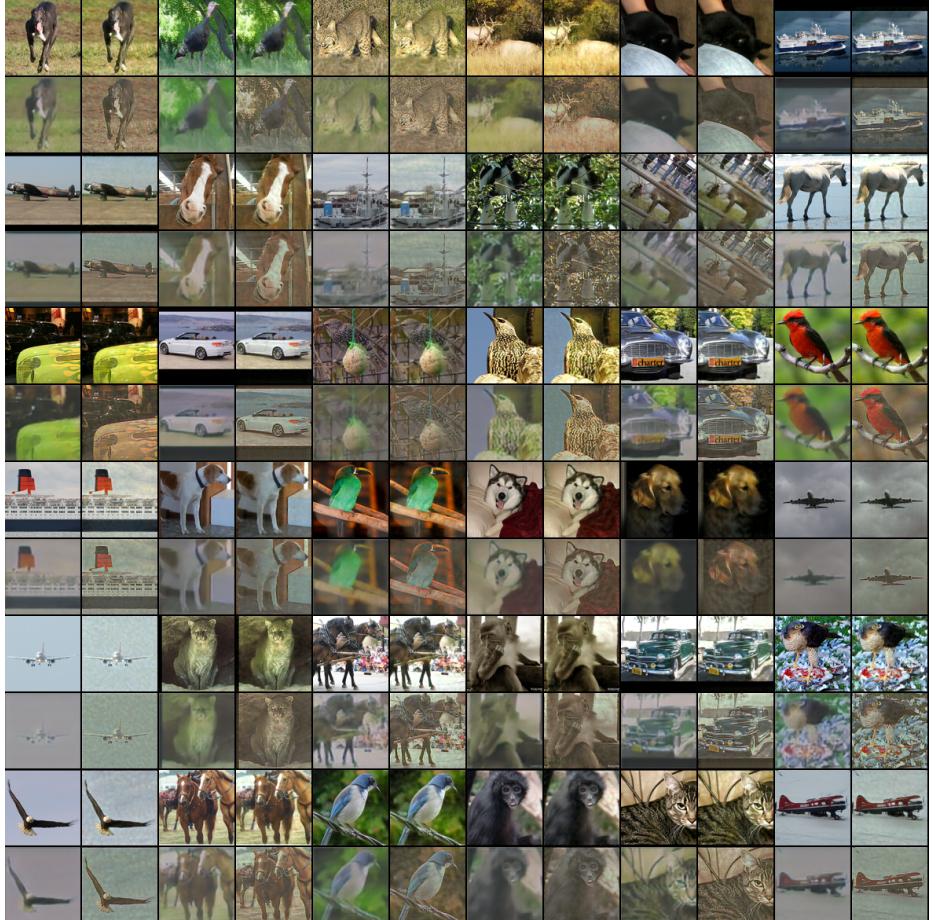


Fig. 6. Example of visualizations for a ConvLarge-like-based HybridNet (complete training loss) on STL-10. For each input image, there is block of 4 images on the figure

with the following organization: $\begin{bmatrix} \mathbf{x} & \hat{\mathbf{x}} \\ \hat{\mathbf{x}}_c & \hat{\mathbf{x}}_u \end{bmatrix}$

B Experimental setup for ConvLarge on CIFAR-10

B.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 on Table 4.

B.2 Training details

The training method is similar to the one presented in recent paper using ConvLarge [23,24,25].

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 5. For the ablation study, parts of the model are removed and/or some weights are set to 0.

C Experimental setup for ConvLarge-like on STL-10

C.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 6.

C.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 batches of 30 unlabeled samples and 2 labeled samples. Hyperparameters’ values and scheduling over training are detailed in Table 7.

D Experimental setup for ResNet on CIFAR-10 and SVHN

The experimental setup described below follows the one described in [25] for a fair comparison.

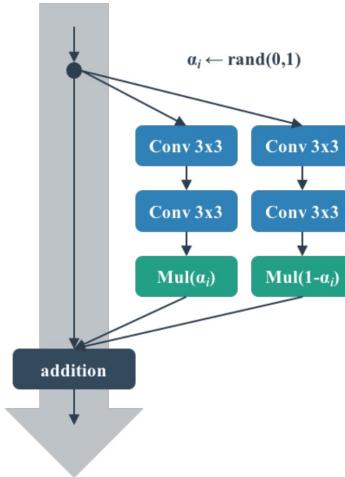


Fig. 7. Shake-Shake building block

D.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 in [51]. A Shake-Shake building block consists of 2 similar branches each containing 2 convolutions, with the first one possibly having a stride greater than 1. The two branches are averaged with a weight α (see Fig. 7 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 8.

D.2 Training details for CIFAR-10

The model’s training is based on the settings of [25]. 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 9.

D.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 10 for SVHN.

E Experimental setup for ResNet on STL-10

E.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.

E.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 11.

Table 4. Architecture of the HybridNet ConvLarge architecture for CIFAR-10

Encoders E_c and E_u		
Input	$\tilde{\mathbf{x}}$	$32 \times 32 \times 3$
Convolution	128 filters, 3×3 , <i>same</i> padding	$32 \times 32 \times 128$
Convolution	128 filters, 3×3 , <i>same</i> padding	$32 \times 32 \times 128$
Convolution	128 filters, 3×3 , <i>same</i> 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 , <i>same</i> padding	$16 \times 16 \times 256$
Convolution	256 filters, 3×3 , <i>same</i> padding	$16 \times 16 \times 256$
Convolution	256 filters, 3×3 , <i>same</i> 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 , <i>valid</i> padding	$6 \times 6 \times 512$
Convolution	256 filters, 1×1 , <i>same</i> padding	$6 \times 6 \times 256$
Convolution	128 filters, 1×1 , <i>same</i> 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 , <i>same</i> padding	$6 \times 6 \times 256$
TConvolution	512 filters, 1×1 , <i>same</i> padding	$6 \times 6 \times 512$
TConvolution	256 filters, 3×3 , <i>valid</i> padding	$8 \times 8 \times 256$
Upsampling	2×2 (unpooling in D_u)	$16 \times 16 \times 256$
TConvolution	256 filters, 3×3 , <i>same</i> padding	$16 \times 16 \times 256$
TConvolution	256 filters, 3×3 , <i>same</i> padding	$16 \times 16 \times 256$
TConvolution	128 filters, 3×3 , <i>same</i> padding	$16 \times 16 \times 128$
Upsampling	2×2 (unpooling in D_u)	$32 \times 32 \times 128$
TConvolution	128 filters, 3×3 , <i>same</i> padding	$32 \times 32 \times 128$
TConvolution	128 filters, 3×3 , <i>same</i> padding	$32 \times 32 \times 128$
TConvolution	3 filters, 3×3 , <i>same</i> 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 5. 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 6. 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 , <i>same</i> padding	$96 \times 96 \times 64$
Convolution	64 filters, 3×3 , <i>same</i> padding	$96 \times 96 \times 64$
Pooling	Maxpool 2×2	$48 \times 48 \times 64$
Convolution	128 filters, 3×3 , <i>same</i> padding	$48 \times 48 \times 128$
Convolution	128 filters, 3×3 , <i>same</i> padding	$48 \times 48 \times 128$
Pooling	Maxpool 2×2	$24 \times 24 \times 128$
Convolution	256 filters, 3×3 , <i>same</i> padding	$24 \times 24 \times 256$
Convolution	256 filters, 3×3 , <i>same</i> padding	$24 \times 24 \times 256$
Pooling	Maxpool 2×2	$12 \times 12 \times 256$
Convolution	256 filters, 3×3 , <i>same</i> 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 , <i>valid</i> padding	$3 \times 3 \times 512$
Dropout	$p = 0.5$	$3 \times 3 \times 512$
Convolution	512 filters, 1×1 , <i>same</i> padding	$3 \times 3 \times 512$
Dropout	$p = 0.5$	$3 \times 3 \times 512$
Convolution	10 filters, 1×1 , <i>same</i> 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 , <i>same</i> padding	$12 \times 12 \times 256$
Upsampling	2×2 (unpooling in D_u)	$24 \times 24 \times 256$
TConvolution	256 filters, 3×3 , <i>same</i> padding	$24 \times 24 \times 256$
TConvolution	128 filters, 3×3 , <i>same</i> padding	$24 \times 24 \times 128$
Upsampling	2×2 (unpooling in D_u)	$48 \times 48 \times 128$
TConvolution	128 filters, 3×3 , <i>same</i> padding	$48 \times 48 \times 128$
TConvolution	64 filters, 3×3 , <i>same</i> padding	$48 \times 48 \times 64$
Upsampling	2×2 (unpooling in D_u)	$96 \times 96 \times 64$
TConvolution	64 filters, 3×3 , <i>same</i> padding	$96 \times 96 \times 64$
TConvolution	3 filters, 3×3 , <i>same</i> 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 7. 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 8. 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 9. 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 10. 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 11. 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.