Semantic Augmentation with Self-Supervised Content Mixing for Semi-Supervised Learning

Rémy Sun
LIP6, Sorbonne Université
Thales Land and Air Systems
remy.sun@lip6.fr

Clément Masson
Thales Land and Air Systems
clement.masson@fr.thalesgroup.com

Gilles Hénaff
Thales Land and Air Systems
gilles.henaff@fr.thalesgroup.com

Nicolas Thome
CEDRIC, Conservatoire National des Arts et Métiers
nicolas.thome@lecnam.net

Matthieu Cord
LIP6, Sorbonne Université
matthieu.cord@lip6.fr

Abstract

Leveraging unlabeled samples is a crucial issue for improving performances in semi-supervised learning. We introduce the SAMOSA framework that adds novel self-supervised reconstruction modules and loss terms. This facilitates semantic augmentation mixing semantic components from labeled samples and non semantic characteristics from unlabeled ones. We demonstrate the effectiveness of SAMOSA on standard SSL procedures Mean Teacher [24] and MixMatch [2].

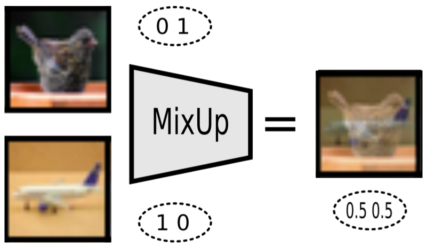
1 Introduction

Deep architectures have proven capable of solving a variety of tasks such as classification [8, 14] or object detection [22]. This is however contingent on using a large amount of labeled data to supervise model training. That is seldom the case in practice where labeling comes at a significant cost.

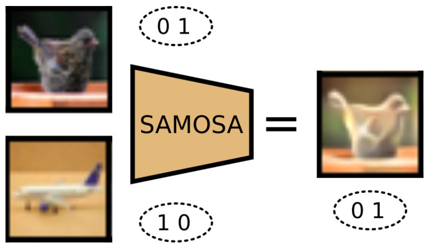
A more realistic setting is Semi-Supervised Learning (SSL), where some data is labeled but most of the available data is unlabeled and serves to avoid overfitting on the labeled data. This has been tackled in a number of ways, ranging from generative modeling [16, 27] to graph based methods [11, 3] and, most notably, consistency based methods [24, 18, 15, 17, 20]. Recently, mixing augmentations like Mixup - which mixes (Fig. 1a) two samples/label pairs by interpolating samples and labels - have been used to great success [2, 1, 6, 26, 4] in SSL by combining unlabeled and labeled samples.

We suggest creating artificial labeled samples that only inherit the label of one parent through mixing. Instead of interpolating pixels, this would require intervening on semantic and non-semantic content. To this end, we propose a novel neural architecture that separates input information into semantic information useful to a classifier, and auxiliary information necessary for reconstruction. Furthermore, our SAMOSA framework can mix semantic and non-semantic content thanks to a novel decoder inspired by work in generative modeling and edition [9, 12]. Fig. 1b shows how SAMOSA combines a bird picture with color tones from a plane picture.

We develop three main contributions in this paper 1) A novel self-supervised learning scheme and architecture - SAMOSA - that can be added to a classifier to learn to separate semantic components from non-semantic ones 2) A new data augmentation mixing semantic and non-semantic contents

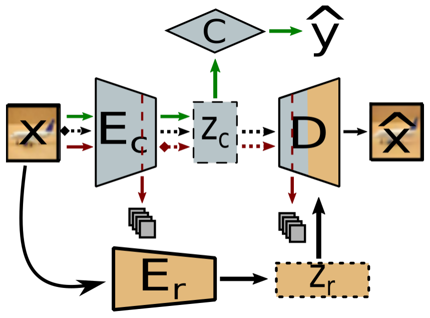


(a) Messy MixUp hybridization with soft labels

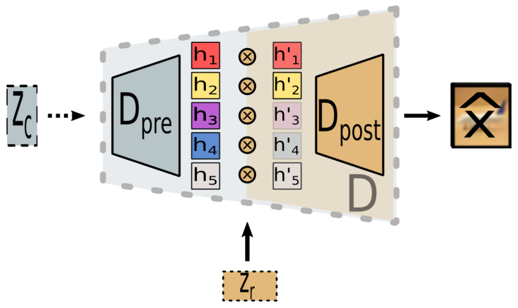


(b) SAMOSA hybridization with hard labels

Figure 1: While MixUp linearly interpolates samples (and labels) to poor effect, SAMOSA clearly mixes semantic and non semantic contents (allowing direct label transfer).



(a) SAMOSA architecture



(b) Structure of D: Asymmetrical decoder

Figure 2: Overview of the proposed framework. Dashed lines indicate forward without backpropagation. (a) z_c and z_r are extracted from input x. x is classified solely from z_c **(green arrows)**, while z_c and z_r are used to reconstruct the input **(black arrows)**. Intermediate maps are tied to regularize E_c **(red arrows)** (b) D reconstructs x from z_c, with z_r modulating which parts of D are expressed.

from different samples while preserving semantic classes 3) we verify experimentally that SAMOSA can improve two standard SSL algorithms: Mean Teacher [24] and MixMatch [2].

2 SAMOSA

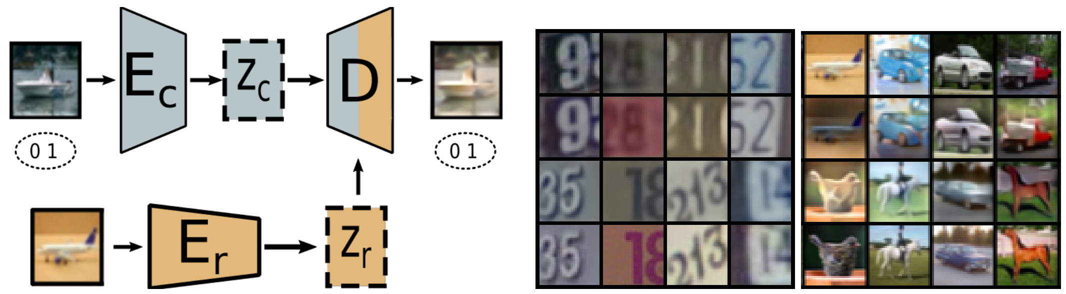
Our framework optimizes a base semantic encoder/feature extractor and classifier. We add an auxiliary non-semantic encoder and reconstruction module used for semantic mixing augmentation.

First, SAMOSA introduces a novel architecture presented in Fig. 2a. It uses two encoders E_c and E_r (one semantic - with regards to the classification process - and one non semantic), a simple classifier C and a bi-modal decoder D. An input x is classified by mapping it to a feature representation $z_c = E_c(x)$, then feeding it to a classifier for predictions $\hat{y} = C(z_c) = C(E_c(x))$. Our additional reconstruction modules E_r and D are discussed in Sec. 2.1.

We train the classifier modules E_c and C to minimize a two term loss $\mathcal{L}_{SAMOSA} = \mathcal{L}_{0,X} + \Omega_{SAMOSA}$. The base loss $\mathcal{L}_{0,X}$ is a proxy to the base method we seek to improve "X"'s training process (through computations figured by green arrows on Fig. 2a). The training of E_r and D is tied to SAMOSA's specific self-supervised regularizer Ω_{SAMOSA} (discussed in Sec. 2.2). We now discuss the asymmetrical architectures E_r and D we add to allow construction of hybridized samples.

2.1 Adding a Non-Supervised Reconstruction Module

Our goal is to mix the semantic content of one sample with the non-semantic content of another. This requires both separating the two contents from samples and reconstructing from those contents in a modular fashion. To some extent, this has been achieved in style transfer [9, 10] (defining "non-semantic" targets) and generative modeling [12] (with no direct reconstruction) by - for instance -



(a) hybridization process. (b) SVHN (100 labels) (c) CIFAR10 (1000 labels)

Figure 3: Mixing contents for data augmentation. (a) Mixing a boat's semantic content with a plane's non-semantic content. (b-c) Examples of generated hybrids (format: $\left[x^{(1)}; D(z_c^{(1)}, z_r^{(2)}); D(z_c^{(2)}, z_r^{(1)}); x^{(2)}\right]$).

manipulating intermediary feature map statistics. We develop an architecture along similar principles that reconstructs inputs without pre-conceptions of what constitutes non-semantic information.

We retain E_c as our semantic encoder, and add a separate encoder E_r for the remaining non semantic information. A novel asymmetrical bi-modal decoder D then reconstructs the input images from the outputs of the encoders: an input x is mapped to a non-semantic feature representation $z_r = E_r(x)$, leading to a reconstructed image $\hat{x} = D(z_c, z_r) = D(E_c(x), E_r(x))$. This last reconstruction process is facilitated by the very peculiar structure of D.

Asymmetrical decoder D Crucially, we design a novel decoder module (Fig. 2b) to combine semantic and non semantic feature spaces. An immediate concern is that an unconstrained non-semantic feature space might store all the necessary information to reconstruct the input. This would make the semantic feature space z_c [23] useless for reconstruction.

To prevent this, we shift z_r's role from affecting *what* is reconstructed to *how* z_c translates to a reconstruction. As shown in Fig. 2b, D can be broken down into D_{pre} and D_{post} such that $h = D_{pre}(z_c) \in \mathbb{R}^{S \times H \times W}$ is a stack of S intermediate reconstruction maps. z_r serves as a set of S gating weights $\in [0, 1]$ (s.t. $\sum_i z_r^i = 1$) so that only some maps $h' = z_r \odot h$ influence the final reconstruction $D_{post}(h')$ (e.g. only the red, yellow and white maps in Fig. 2b). While this can be seen as a rescaling of feature maps (like in style transfer) [9, 12, 10], the absence of style targets might lead to z_r selecting all maps. As this would make z_r useless, we activate it with a softmax function to ensure only a few maps are kept by z_r. Those modules are trained with a novel self-supervised learning scheme to help proper separation of contents.

2.2 Learning Scheme and Hybridization

Model optimization SAMOSA relies on a self-supervised regularizer term Ω_{SAMOSA} to leverage its peculiar architecture (Eq. 1). $\mathcal{L}_{rec}$ optimizes for reconstruction of inputs. To prevent E_c from contributing all the reconstruction information, only E_r and D are trained for reconstruction. E_c is regularized to match D's reconstructions through an auxiliary regularizer term Ω_{rec}.

$$\Omega_{SAMOSA} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{recons,int}\Omega_{rec}. \tag{1}$$

$\mathcal{L}_{rec} = \frac{1}{\#\mathcal{D}} \sum_{x \in \mathcal{D}} \|D(E_c(x), E_r(x)) - x\|_2^2$ (black arrows on Fig. 2a) matches inputs x to model reconstructions $D(E_c(x), E_r(x))$ through the L2 distance between the two. E_c is deliberately not optimized here as skip connections [8] already let a lot of input information trickle down. We found optimizing E_c for reconstruction led D to rely entirely on E_c and ignore E_r.

$\Omega_{rec} = \frac{1}{\#\mathcal{D}} \sum_{x \in \mathcal{D}} \|E_{c,int}(x) - D_{int}(E_c(x), E_r(x)))\|_2^2$ leverages our decoder's asymmetrical structure to regularize E_c. Ω_{rec} ties the last intermediate features $E_{c,int}(x)$ extracted by E_c to the first intermediate reconstructions $D_{int}(E_c(x), E_r(x))$ generated by D (red arrows on Fig. 2a). Importantly, the first intermediate reconstructions are purely semantic as they are prior to re-modulation by z_r. Hybrids can then be incorporated either during training, or after (retraining on augmented data).

Table 1: Comparative accuracies (%) in different experiments.

(a) Gains with SAMOSA on CIFAR10.

Method	CIFAR10	
	250	500
Purely supervised)	27.8 ± 0.9	35.4 ± 1.9
Mean Teacher[1], [24]	61.3 ± 3.3	75.3 ± 2.5
+ SAMOSA (ours)	$\mathbf{68.1 \pm 3.3}$	$\mathbf{80.7 \pm 1.0}$
MixMatch[12], [2]	$\mathbf{82.4 \pm 0.4}$	86.8 ± 0.2
+ SAMOSA (ours)	$\mathbf{84.1 \pm 2.2}$	$\mathbf{89.4 \pm 0.8}$

(b) Ablation Study on the components of SAMOSA.

$\mathcal{L}_0$ / Ω_{rec} / Aug	Mean Teacher	MixMatch
	250	500
✓/ ✗/ ✗	61.3 ± 3.3	86.8 ± 0.2
✓/ ✓/ ✗	67.2 ± 3.3	88.5 ± 1.2
✓/ ✗/ ✓	61.3 ± 2.6	87.8 ± 0.4
✓/ ✓/ ✓	$\mathbf{68.1 \pm 3.2}$	$\mathbf{89.4 \pm 0.8}$

Semantic augmentation We now leverage our novel framework to generate semantically consistent hybrids by extracting and then combining semantic/auxiliary contents from two different samples (Fig. 3). Given samples $x^{(1)}$ (label $y^{(1)}$) and $x^{(2)}$, we extract the relevant features $z_c^{(1)} = E_c(x^{(1)})$, $z_r^{(1)} = E_r(x^{(1)})$, $z_c^{(2)} = E_c(x^{(2)})$ and $z_r^{(2)} = E_r(x^{(2)})$. $\tilde{x} = D(z_c^{(1)}, z_r^{(2)})$ is now a sample with class $y^{(1)}$, which we only keep if $C(E_c(\tilde{x})) = y^{(1)}$ to avoid disturbing decision boundaries too much. Note that with this, we generate a strong augmentation of x_1 and teach the classifier to group x_1 with its strongly augmented version in a similar line to work in contrastive representation learning [7, 25].

3 Experiments

We demonstrate here how SAMOSA can improve upon two SSL staples: a pure consistency based method (Mean Teacher [24]) and a non label-preserving mixing based method (MixMatch [2]). We conduct experiments on the CIFAR10 dataset [13] with 250, 500 and 1000 labels. We also push the model by only keeping 100 labels on the SVHN dataset [19]. We operate on a standard WideResNet-28-2 [28] which is widely used in the SSL literature as a base model ($C \circ E_c$). We evaluate performance through classification accuracy, presented as $\mu \pm \sigma$, with μ the average value and σ the standard deviation across three seeded runs (random initializations). We bold results shown to be better with $p < 0.1$ by a paired t-test in the following. Refer to the Supplementary for details.

Gains: Mean Teacher (MT) We train the model with $\mathcal{L}_{SAMOSA}$ for 300 epochs (with the reconstruction module), then hybridize every labeled sample with 10 unlabeled samples. The model is then retrained with this additional labeled data. This is repeated again for a total three training cycles.

Tab. 1a shows improvements from using the SAMOSA framework on top of Mean Teacher. Gains were also noticed for 1000 labels (from 87.6 ± 0.3 to 88.7 ± 0.3), but the more significant gains are observed with only few labels (e.g. 250 labels). To explore using even less labels, we also trained with 100 labels on SVHN. An important accuracy gain from 62.5 ± 3.7 to 66.2 ± 2.2 is observed which is significant given such very low label settings are especially interesting for application purposes.

Gains: MixMatch (Mix) We train the model according to MixMatch's procedure with the addition of Ω_{SAMOSA} (and the reconstruction modules). Every batch, with probability $p = 0.2$, we replace the MixUp step with our semantic mixing step.

As can be seen from Tab. 1a, sizable gains are achieved on both the 500 and 250 labels CIFAR10 settings, and can also be observed on 1000 labels (from 88.2 ± 0.5 to 89.9 ± 0.1). Results on the SVHN setting varied too much to assess whether the 93.8 ± 2.8 score obtained by our method was an improvement on the MixMatch baseline.

Ablation study Results from Tab. 1b show both the self-supervised reconstruction loss and the augmented hybrids provide significant gains in accuracy. Interestingly, the regularizer Ω_{rec} is the greatest influence for both experiments. Nevertheless, improvements can consistently be observed from adding augmented samples to the regularized model, especially when the model is regularized (probably due to hybrids of better quality).

[1]Reproduced by adapting available code (see Supplementary)
[2]Different setting from [2] for fast training. Results reported from weight averaged model following [2].

Discussion In this paper we introduced SAMOSA, a framework that improves existing SSL algorithms by refining classifier features through self-supervised reconstruction. Thanks to its separation of semantic and non semantic components, SAMOSA generates hybrids mixing the semantic content and non semantic characteristics of different samples for a data augmentation that teaches the model to cluster samples that share semantic content. We observed experimentally that SAMOSA improves both the Mean Teacher and MixMatch algorithms, with noticeable gains given little labeled data. We also demonstrated the usefulness both of our reconstruction module for classifier regularization, and of the semantically consistent hybrid augmentation.

References

[1] David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *ICLR*, 2020.

[2] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: a holistic approach to semi-supervised learning. *NeurIPS*, 2019.

[3] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[4] Geoff French, Avital Oliver, and Tim Salimans. Milking cowmask for semi-supervised image classification. *arXiv preprint*, 2020.

[5] Raphael Gontijo-Lopes, Sylvia J. Smullin, Ekin D. Cubuk, and Ethan Dyer. Affinity and diversity: Quantifying mechanisms of data augmentation. *ArXiv preprint*, 2020.

[6] Ryuichiro Hataya and Hideki Nakayama. Unifying semi-supervised and robust learning by mixup. *Limited Labeled Data (LLD) Workshop at ICLR*, 2019.

[7] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *CVPR*, 2020.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *ECCV*, 2016.

[9] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *ICCV*, 2017.

[10] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *ECCV*, 2018.

[11] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. *CVPR*, 2019.

[12] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CVPR*, 2019.

[13] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012.

[15] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *ICLR*, 2017.

[16] Chongxuan Li, Kun Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. *NeurIPS*, 2017.

[17] Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. *CVPR*, 2018.

[18] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[19] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NeurIPS*, 2011.

[20] Sungrae Park, Jun-Keon Park, Su-Jin Shin, and Il-Chul Moon. Adversarial dropout for supervised and semi-supervised learning. *AAAI*, 2018.

[21] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *ArXiv preprint*, 2017.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015.

[23] Thomas Robert, Nicolas Thome, and Matthieu Cord. Hybridnet: Classification and reconstruction cooperation for semi-supervised learning. *ECCV*, 2018.

[24] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *NeurIPS*, 2017.

[25] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. *ICLR*, 2020.

[26] Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. *IJCAI*, 2019.

[27] Si Wu, Guangchang Deng, Jichang Li, Rui Li, Zhiwen Yu, and Hau-San Wong. Enhancing triplegan for semi-supervised conditional instance synthesis and classification. *CVPR*, 2019.

[28] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016.

Appendix

A Detailed Experimental Setting

We provide here details of the experimental setting used to run experiments presented in Sec. 3 of the main paper. Secs. A.1,A.2 and A.3 discuss the precise parameters and scheduling while Sec. B gives a more detailed account of the architectures used.

Implementation was coded in python3 using the pytorch framework, parameters follow their default pytorch value unless specified otherwise. The code base was largely adapted from `https://github.com/ThomasRobertFr/hybridnet` for the Mean Teacher implementation and general code backbone, MixMatch implementation was adapted almost directly from `https://github.com/YU1ut/MixMatch-pytorch` (with some modifications to follow [2] more closely).

Most experiments were run on a *nationwide* computation cluster with 261 Nodes of 2 Intel Cascade Lake 6248 CPUs and 4 Nvidia V100 SXM2 32 Go GPUs operating RedHat version 8.1 with conda 4.8.0. The SVHN Mean Teacher based experiments were run on another workstation with an Intel Xeon Silver 4108 CPU and 4 Nvidia Titan Xp 12 Go GPUs operating Debian Bullseye.

A.1 General Setting

We operate on a standard WideResNet-28-2 (1.5M parameters) which is widely used in the Semi-Supervised Learning literature as a base model ($C \circ E_c$). E_r follows the same architecture as E_c with an additional final linear layer and softmax activation to obtain activation gates. The skeleton of D follows an inverted 13-layer 4-4-4 CNN architecture, with D_{pre} being a 4-4 block and D_{post} being made up of the last 4 block and final convolution. Downsampling/Upsampling convolutions are stabilized with an additional Batch Normalization layer in MixMatch experiments. Weights were initialized according to default pytorch layer initializations (Kaiming initializations). Batch Normalization layers followed standard pytorch configuration (momentum = 0.1). Gradients are clipped to unit norm to stabilize training, optimizers details vary from setting to setting (detailed below).

Samples are randomly flipped horizontally (only for CIFAR10) and shifted by up to 4 pixels both horizontally and vertically with reflect padding. The resulting augmented samples are then standardized channel-wise according to train set statistics. No holdout validation set is kept for either dataset but hyper-parameters are mostly directly adapted from [23, 2].

Reduced settings parameters for the Case Study were identical to normal settings outlined below with the sole differences being that they were run on only one gpu with halved learning rates and batch sizes.

A.2 Mean Teacher Based SAMOSA on CIFAR10/SVHN

The model is trained over 2 gpus using a SGD optimizer with weight decay $5e-4$, Nesterov momentum 0.9 and cosine learning rate (base 0.2 learning rate) for 300/150 epochs over the unlabeled samples (the rate is set to fully decrease over 350/200 epochs), with the reconstructive modules E_r and D optimized by a secondary SGD optimizer following the same parameters.

After each training and subsequent augmentation step, the model learning rate is reset and training resumes (leading to training for an overall 900 epochs). $\lambda_{recons} = 0.25, \lambda_{recons,int} = 0.5/0.1$ during training, consistency targets are optimized with weight $\lambda_{cons} = 300$, with batches of 124/30 labeled samples and 388/500/500 unlabeled samples (similarly to [23]). In the second and third training passes, the model is only optimized over the augmented dataset from epoch 150/50 to 250/100 of each cycle, and is optimized over the true dataset for the remaining epochs (following discussions in [21, 5]).

Consistency is actually optimized following the dual head trick used in [24] where the classifier outputs two different decision y_{preds} and y_{cons}. y_{cons} is optimized to reconstruct the EMA target's y_{preds}^{EMA} (weight $\lambda_{cons} = 300$), and y_{preds} is tied to reconstruct y_{cons} (weight $\lambda_{logitdist} = 0.01$). This limits the need for precise scheduling of consistency weights by having a "buffer" prediction head. The EMA model is updated with decay 0.97 (following a linear start for early iterations).

A.3 MixMatch Based SAMOSA on CIFAR10/SVHN

The classifier $C \circ E_C$ is trained over 2 gpus using a AdamW optimizer with weight decay 0.02, $\beta = (0.9, 0.999)$ and constant learning rate 0.002 for 900/600 epochs over the unlabeled samples, with the reconstructive modules E_r and D optimized with a SGD optimizer with weight decay $5e-4$, Nesterov momentum 0.9 and cosine learning rate (base learning rate 0.2) for 900/600 epochs (set to fully decrease over 950/650 epochs).

$\lambda_{recons} = 0.25, \lambda_{recons,int} = 0.5$, pseudolabel targets are optimized with weight $\lambda_u = 37.5/250$ (linear ramp over 16384 iterations first), with batches of 256/100 labeled samples and 256/100 unlabeled samples (following the proportions in [2], in the CIFAR10 250 label case the batch sizes are instead 250 and 250). Similarly to the Mean Teacher setting, we only incorporate hybrid samples from epoch $100/50$ to epoch $800/350$. Outside of this window, the model is optimized according to the standard MixMatch procedure (only on Mixed Up examples).

A.4 Hyperparameter Ranges Investigated

Values for $\lambda_{cons}, \lambda_{logit_dist}, \lambda_u, \lambda_{recons,int}, \lambda, \beta$, as well as decay and learning rates were directly taken from [23, 2] with no parameter search. Batch sizes for the Mean Teacher experiments were also taken from [23]. Batch sizes for the MixMatch experiments were taken to balance out labeled and unlabeled samples while being as large as possible (for faster training). We tried turning hybrid augmentation on {0,50,100,150} epochs into training and turning it of {100,50,0} epochs before the end of training on the Mean Teacher CIFAR10 1000 labels setting, and adapted the training schedule for other settings by taking the optimal setting with regards to test set accuracy on the CIFAR10 1000 label setting.

B Network Outlines

The general outline and hyperparameters of the neural architectures used in our experiments are detailed in Tables 4-9.

Table 2: General structure of E_c.

Layer	Details	Output shape
Input	Preprocessed image x	$3 \times 32 \times 32$
Conv2d	16 filters, 3×3 kernels, pad 1	$16 \times 32 \times 32$
WideResBlock	32 filters, 3×3 kernels, pad 1	$32 \times 32 \times 32$
WideResBlock	32 filters, 3×3 kernels, pad 1	$32 \times 32 \times 32$
WideResBlock	32 filters, 3×3 kernels, pad 1	$32 \times 32 \times 32$
WideResBlock	32 filters, 3×3 kernels, pad 1	$32 \times 32 \times 32$
WideResBlock	64 filters, 3×3 kernels, pad 1, stride 2	$64 \times 16 \times 16$
WideResBlock	64 filters, 3×3 kernels, pad 1	$64 \times 16 \times 16$
WideResBlock	64 filters, 3×3 kernels, pad 1	$64 \times 16 \times 16$
WideResBlock	64 filters, 3×3 kernels, pad 1	$64 \times 16 \times 16$
WideResBlock	128 filters, 3×3 kernels, pad 1, stride 2	$128 \times 8 \times 8$
WideResBlock	128 filters, 3×3 kernels, pad 1	$128 \times 8 \times 8$
WideResBlock	128 filters, 3×3 kernels, pad 1	$128 \times 8 \times 8$
WideResBlock	128 filters, 3×3 kernels, pad 1	$128 \times 8 \times 8$

Table 3: General structure of E_r.

Layer	Details	Output shape
Input	Preprocessed image x	$3 \times 32 \times 32$
Conv2d	16 filters, 3×3 kernels, pad 1	$16 \times 32 \times 32$
WideResBlock	32 filters, 3×3 kernels, pad 1	$32 \times 32 \times 32$
WideResBlock	32 filters, 3×3 kernels, pad 1	$32 \times 32 \times 32$
WideResBlock	32 filters, 3×3 kernels, pad 1	$32 \times 32 \times 32$
WideResBlock	32 filters, 3×3 kernels, pad 1	$32 \times 32 \times 32$
WideResBlock	64 filters, 3×3 kernels, pad 1, stride 2	$64 \times 16 \times 16$
WideResBlock	64 filters, 3×3 kernels, pad 1	$64 \times 16 \times 16$
WideResBlock	64 filters, 3×3 kernels, pad 1	$64 \times 16 \times 16$
WideResBlock	64 filters, 3×3 kernels, pad 1	$64 \times 16 \times 16$
WideResBlock	128 filters, 3×3 kernels, pad 1, stride 2	$128 \times 8 \times 8$
WideResBlock	128 filters, 3×3 kernels, pad 1	$128 \times 8 \times 8$
WideResBlock	128 filters, 3×3 kernels, pad 1	$128 \times 8 \times 8$
WideResBlock	128 filters, 3×3 kernels, pad 1	$128 \times 8 \times 8$
Flatten	$C \times H \times W \rightarrow CHW$	8192
Linear	64 units	64
Softmax		64

Table 4: General structure of C.

Layer	Details	Output shape
Input	Semantic features z_c	$128 \times 8 \times 8$
BatchNormalization2d		$128 \times 8 \times 8$
LeakyReLU	Negative slope 0.1	$128 \times 8 \times 8$
Global Average Pooling **f**	128 units	128
Linear	10 units, primary output, takes f as input	10
Linear	10 units, stability output, takes f as input	10

Table 5: General structure of D.

Layer	Details	Output shape
Input	Semantic features z_c, (Style weights z_r)	$128 \times 8 \times 8, (64)$
DeconvBlock	128 filters, 3×3 kernels, pad 1	$128 \times 8 \times 8$
DeconvBlock	128 filters, 3×3 kernels, pad 1	$128 \times 8 \times 8$
DeconvBlock	128 filters, 3×3 kernels, pad 1	$128 \times 8 \times 8$
DeconvBlock	64 filters, 3×3 kernels, pad 1, stride 2	$64 \times 16 \times 16$
DeconvBlock	64 filters, 3×3 kernels, pad 1	$64 \times 16 \times 16$
DeconvBlock	64 filters, 3×3 kernels, pad 1	$64 \times 16 \times 16$
DeconvBlock	64 filters, 3×3 kernels, pad 1	$64 \times 16 \times 16$
DeconvBlock **h**	64 filters, 3×3 kernels, pad 1, stride 2	$64 \times 32 \times 32$
Auxiliary modulation **h'**	$\mathbf{z_r} \odot \mathbf{h}$	64×32
DeconvBlock	64 filters, 3×3 kernels, pad 1, takes h'	$64 \times 32 \times 32$
DeconvBlock	64 filters, 3×3 kernels, pad 1	$64 \times 32 \times 32$
DeconvBlock	64 filters, 3×3 kernels, pad 1	$64 \times 32 \times 32$
DeconvBlock **h**	16 filters, 3×3 kernels, pad 1	$16 \times 32 \times 32$
Deconv2d	3 filters, 3×3 kernels, pad 1	$3 \times 32 \times 32$

Table 6: WideResBlock of F filters ands stride s.

Layer	Details	Output shape
Input	Preprocessed image **x**	$C \times H \times W$
BatchNormalization2d		$C \times H \times W$
LeakyReLU	Negative slope 0.1	$C \times H \times W$
Conv2d	F filters, 3×3 kernels, pad 1, stride s	$F \times \frac{C}{s} \times \frac{H}{s}$
BatchNormalization2d		$F \times \frac{C}{s} \times \frac{H}{s}$
LeakyReLU	Negative slope 0.1	$F \times \frac{C}{s} \times \frac{H}{s}$
Conv2d r	F filters, 3×3 kernels, pad 1	$F \times \frac{C}{s} \times \frac{H}{s}$
Conv2d **x** (replaces)	iff $F \neq C$, F filters, 1×1 kernels, stride s, takes x	$F \times \frac{C}{s} \times \frac{H}{s}$
BatchNormalization2d **x** (replaces)	iff $F \neq C$, takes x, only for MixMatch	$F \times \frac{C}{s} \times \frac{H}{s}$
Fusion	**x + r**	$F \times \frac{C}{s} \times \frac{H}{s}$

Table 7: DeconvBlock of F filters and stride s.

Layer	Details	Output shape
Input	Preprocessed image **x**	$C \times H \times W$
BatchNormalization2d		$C \times H \times W$
LeakyReLU	Negative slope 0.1	$C \times H \times W$
Deconv2d	F filters, 3×3 kernels, pad 1, stride s	$F \times sC \times sH$

C Significance of Improvements (t-tests)

We report here the results of one-sided paired t-tests for the hypothesis that our methods indeed outperform studied baselines (such that $\mathcal{H}_0$="The two methods are equivalent", $\mathcal{H}_1$="Method A is better than Method B") as discussed in Sec. 3 of the main paper. Run results are paired with respect to their initialization seed (and are therefore indeed fully paired). Gaussianity of the distributions compared can be assumed given standard practices in the literature. The results of the paired t tests are written out as *pvalue (test statistic)*.

C.1 Improvements on Base Methods

Table 8: Significance tests with Mean Teacher as a base algorithm.

Hypothesis $\mathcal{H}_1$	CIFAR10		
	250	500	1000
SAMOSA-MT > Mean Teacher	$\mathbf{.013(6.00)}$	$\mathbf{.064(2.52)}$	$\mathbf{.004(11.72)}$

As can be seen from Tab. 8, the gains observed in Tab. 1a of the main paper when applying SAMOSA to the Mean Teacher Framework are fairly significant.

Table 9: Significance tests with MixMatch as a base algorithm.

Hypothesis $\mathcal{H}_1$	CIFAR10		
	250	500	1000
SAMOSA-Mix > MixMatch	$.111(1.75)$	$\mathbf{.024(4.20)}$	$.136(1.50)$

As can be seen from Tab. 9, the gains observed in Tab. 1a when applying SAMOSA to the MixMatch Framework are fairly significant. While the CIFAR10 1000 labels experiment results in gains that would not allow rejecting $\mathcal{H}_0$ at significance p>0.9, this could be due to the small number of runs not being suited to detect a slight effect.

C.2 Component Ablation Study

Table 10: Significance tests of the Ablation study comparisons.

Hypothesis $\mathcal{H}_1$	Mean Teacher	MixMatch
	250	500
$\textcircled{2} > \textcircled{1} : \Omega_{rec} + \mathcal{L}_{Base} > \mathcal{L}_{Base}$	$\mathbf{.013(5.99)}$ $\mathbf{.007(8.66)}$	
$\textcircled{3} > \textcircled{1} : \mathcal{L}_{Base} + Aug > \mathcal{L}_{Base}$	✗(-0.08)	$\mathbf{.066(2.47)}$
$\textcircled{2} > \textcircled{3} : \Omega_{rec} + \mathcal{L}_{Base} > \mathcal{L}_{Base} + Aug$	$\mathbf{.035(3.53)}$	$.227(0.92)$
$\textcircled{4} > \textcircled{1} : \Omega_{rec} + \mathcal{L}_{Base} + Aug > \mathcal{L}_{Base}$	$\mathbf{.013(6.00)}$	$\mathbf{.024}(4.20)$
$\textcircled{4} > \textcircled{2} : \Omega_{rec} + \mathcal{L}_{Base} + Aug > \Omega_{rec} + \mathcal{L}_{Base}$	$\mathbf{.061(2.60)}$	$\mathbf{.045(3.12)}$
$\textcircled{4} > \textcircled{3} : \Omega_{rec} + \mathcal{L}_{Base} + Aug > \mathcal{L}_{Base} + Aug$	$\mathbf{.016(5.51)}$	$\mathbf{.075(2.29)}$

We present in Tab. 10 comparisons between the various ablations performed in Tab. 1b. Considering each line in Tab. 1b refers to one ablation setting, we associate each line of the table with its position: $\textcircled{n}$ refers to the nth line.

For the most part, results suggest (full SAMOSA) $\textcircled{4} > \textcircled{2} > \textcircled{3} > \textcircled{1}$ (Base method). While the full SAMOSA framework seems to outperform setting $\textcircled{2}$ on average, the gains do not allow for rejecting $\mathcal{H}_0$: "$\textcircled{4} = \textcircled{2}$" in favor of $\mathcal{H}_1$: "$\textcircled{4} > \textcircled{2}$" in general.

D Qualitative Evaluation: generated hybrids

We present here examples of hybridizations performed by SAMOSA for different settings (in addition to those shown in Fig. 3). We present hybrids between samples $x^{(1)}$ (first line) and $x^{(2)}$ (fourth line). The second line presents hybrids combining the semantic component of $x^{(1)}$ and the auxiliary component of $x^{(2)}$, with the third line presenting hybrids combining the semantic component of $x^{(2)}$ and the auxiliary component of $x^{(1)}$.

The hybrids presented are those generated at the end of training (in the case of the MixMatch based SAMOSA), or those generated at the end of the first training cycle (in the case of the Mean Teacher based SAMOSA). Overall, the results follow the format: $\begin{bmatrix} x^{(1)} \\ D(z_c^{(1)}, z_r^{(2)}) \\ D(z_c^{(2)}, z_r^{(1)}) \\ x^{(2)} \end{bmatrix}$

D.1 SVHN Hybrids

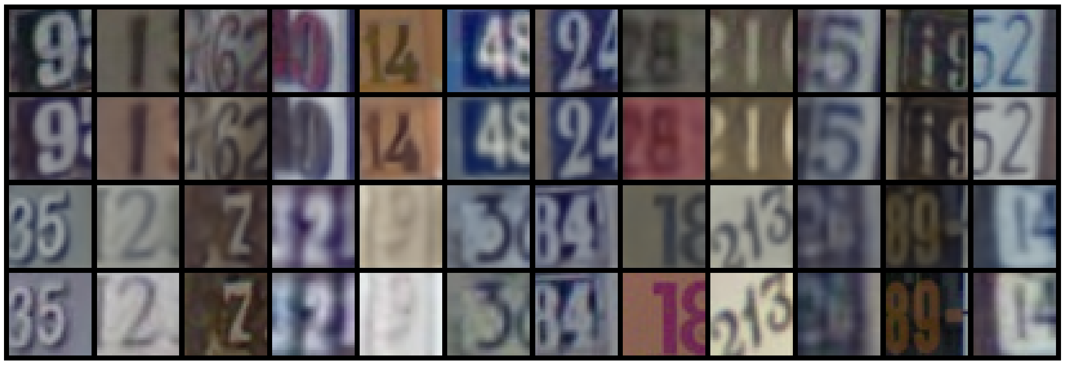


Figure 4: Hybrids generated by the MT model for SVHN (100 labels)

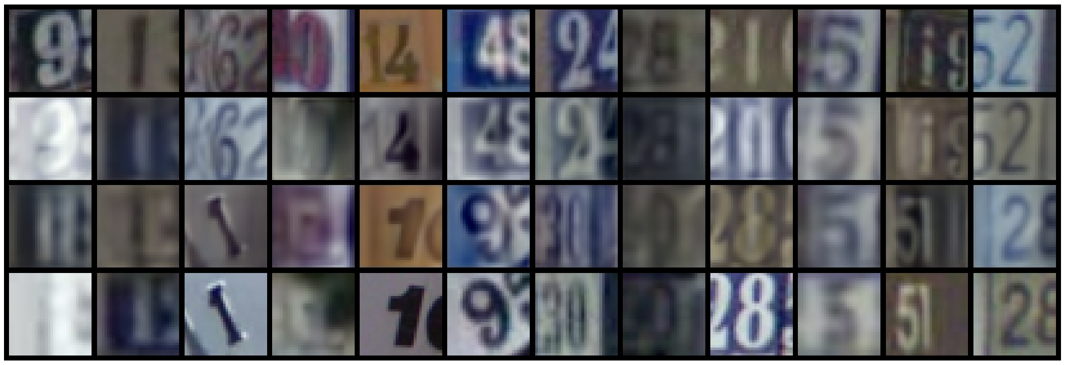


Figure 5: Hybrids generated by the Mix model for SVHN (100 labels)

D.2 CIFAR10 Hybrids

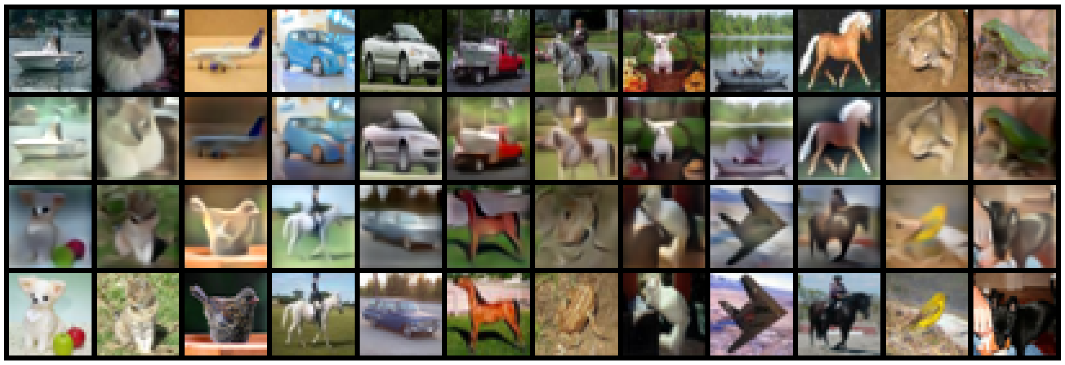
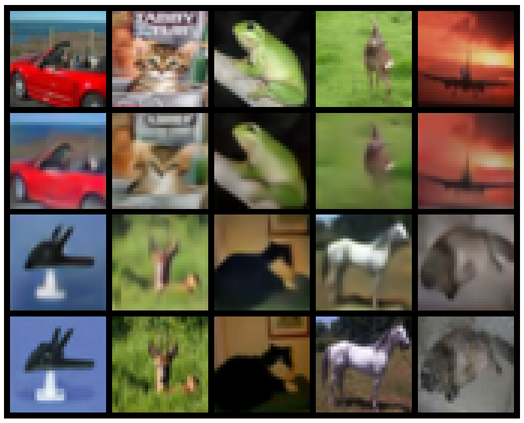
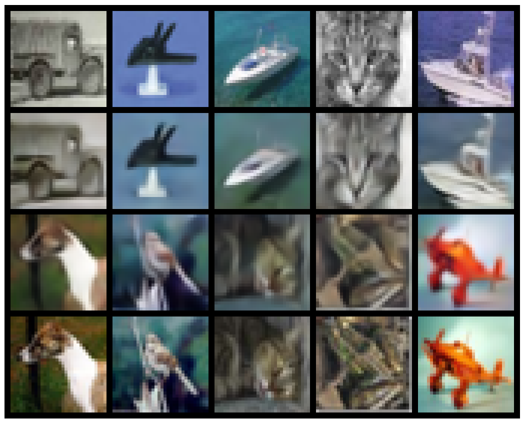


Figure 6: Hybrids generated by the MT model for CIFAR (1000 labels)



(a) 500 labels



(b) 250 labels

Figure 7: Hybrids generated by the MT model for CIFAR (500 and 250 labels)

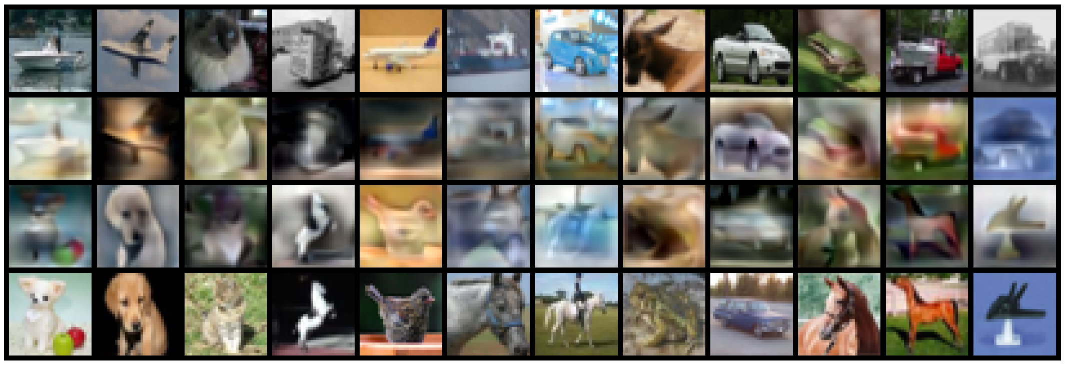


Figure 8: Hybrids generated by the Mix model for CIFAR (1000 labels)