Stochastic Weighted Function Norm Regularization

Amal Rannen Triki* Maxim Berman Matthew B. Blaschko
KU Leuven
KU Leuven, ESAT-PSI, Belgium
firstname.lastname@esat.kuleuven.be

Abstract

Deep neural networks (DNNs) have become increasingly important due to their excellent empirical performance on a wide range of problems. However, regularization is generally achieved by indirect means, largely due to the complex set of functions defined by a network and the difficulty in measuring function complexity. There exists no method in the literature for additive regularization based on a norm of the function, as is classically considered in statistical learning theory. In this work, we propose sampling-based approximations to weighted function norms as regularizers for deep neural networks. We provide, to the best of our knowledge, the first proof in the literature of the NP-hardness of computing function norms of DNNs, motivating the necessity of a stochastic optimization strategy. Based on our proposed regularization scheme, stability-based bounds yield a $\mathcal{O}(N^{-\frac{1}{2}})$ generalization error for our proposed regularizer when applied to convex function sets. We demonstrate broad conditions for the convergence of stochastic gradient descent on our objective, including for non-convex function sets such as those defined by DNNs. Finally, we empirically validate the improved performance of the proposed regularization strategy for both convex function sets as well as DNNs on real-world classification and segmentation tasks.

1 Introduction

Deep neural networks (DNNs) define the state of the art for an increasing number of problem domains. However, understanding their generalization behavior is still an open problem. Furthermore, due to the complex set of functions defined by DNNs, regularization is performed only indirectly and generally penalizes some property of the network parametrization rather than a direct measure of the function complexity.

Different approaches have been applied to explain the capacity of DNNs to generalize well, even though they can use a number of parameters several orders of magnitude larger than the number of training samples. [13] analyze stochastic gradient descent (SGD) applied to DNNs using the uniform stability concept introduced in [5]. However, the stability parameter they show depends on the number of training epochs, which makes the related bound on generalization rather pessimistic and tends to confirm the importance of early stopping for training DNNs [10]. More recently, [33] have suggested that classical learning theory is incapable of explaining the generalization behavior of deep neural networks. Indeed, by showing that DNNs are capable of fitting arbitrary sets of random labels, the authors make the point that the expressivity of DNNs is partially data-driven, while the classical analysis of generalization does not take the data into account, but only the function class and the algorithm. Nevertheless, learning algorithms, and in particular SGD, seem to have an important role in the generalization ability of DNNs. [17] show that using smaller batches results in better generalization. Other works (e.g. [15]) relate the implicit regularization applied by SGD to the flatness of the minimum to which it converges, but [8] have shown recently that even sharp minima can generalize well.

Previous work concerning the generalization of DNNs present several contradictory results. Taking a step back, it appears that our better understanding of classical learning models – such as kernel methods – with respect to DNNs comes from the well-defined hypothesis set on which the optimization is performed, and clear measures of the function complexity, such as the RKHS norm [30]. Indeed, while there clearly exists a mapping between the parameters of a DNN and the function it expresses, this mapping is non-trivial because of the architecture of the network and the non-linearities. Therefore, the function set represented by a given architecture is not understood well. Moreover, the structural risk minimization principle [32] dictates that the regularizer controls the learned mapping complexity. However, this is done only indirectly either via weight decay [22] which controls the norm of the network parameters but is not itself a norm of the function (Lemma 1), or other learning tricks such as dropout [14] or batch normalization [16].

In this work, we make a step towards bridging the described gap by introducing a new family of regularizers that approximates a proper function norm. We demonstrate that this approximation is necessary by, to the

*Amal Rannen Triki is also affiliated with Yonsei University.

best of our knowledge, the first proof in the literature that computing a function norm of DNNs is NP-hard (Proposition 1). We show that these regularizers, under reasonable assumptions, have good stability properties inducing bounds on generalization errors. Next, we identify conditions in which the SGD algorithm applied to the new objective converges. Our experiments reinforce these conclusions by showing that the use of these regularizers lowers the generalization error and that we achieve better performance than other regularization strategies in the small sample regime.

2 Weighted function norm of a DNN

A DNN is represented by its non-linearities (nodes) and its architecture (connections between the nodes). For every parameterization W, the DNN operates a function f_W between an input space $\mathcal{X}$ and a target space $\mathcal{Y}$. Training the network aims to minimize the empirical risk

$$R_N(W) = \frac{1}{N}\sum_{i=1}^{N} \ell(f_W(x_i), y_i), \tag{1}$$

where $\{(x_i, y_i)\} \in (\mathcal{X} \times \mathcal{Y})^N$ is an i.i.d. sample drawn from some unknown probability measure $P(x, y)$.

For a better control over the generalization error, especially in the small sample regime, we usually introduce a regulation term to the objective. In the statistical learning theory literature, this is most typically achieved through an additive penalty [23, 32]

$$\arg\min_W R_N(W) + \lambda\Omega(f_W), \tag{2}$$

where Ω is a measure of function complexity. Our main contribution is to use $\Omega(f_W) \approx \|f_W\|_*$ where $\|.\|_*$ is a proper function norm.

Lemma 1. *The weight norm used in weight decay is not a proper function norm.*

It is easy to see that a given f_W can map to different weights, and thus the weight norm is not even a function of f_W.

More particularly, we will consider an L_2-type function norm. The motivation behind this specific type of norm is that such a penalty tends to "convexify" the objective in the function space, and may result in a better minimum than the one obtained by controlling the parameters (cf. Section 4).

The canonical L_2 function form can be defined for a vector valued function f as

$$\|f\|_2 = \left[\int_{\mathcal{X}} \|f(x)\|_2^2 dx\right]^{\frac{1}{2}}. \tag{3}$$

As we mainly care about the function complexity over the data manifold, we suggest to use the function norm with respect to a probability measure $\hat{P}(x)$, whose restriction to the support of the data should have bounded Kullback-Leibler divergence from the true marginal distribution $P(x)$ (cf. Section 4.2). We will suppose that all the probability densities in this paper are absolutely continuous with respect to the Lebesgue measure. Therefore, this function norm can be defined as

$$\|f\|_{2,\hat{P}} = \left[\int \|f(x)\|_2^2 \, \hat{P}(x)\, dx\right]^{\frac{1}{2}}. \tag{4}$$

Note that this norm is a proper function norm. As the marginal distribution $P(x)$ is generally unknown, and when we do not have access to samples from this distribution that are independent from the training set, we suggest to learn a generative model on the available training data such as variational autoencoders or GANs [11, 19]. Such models can be used to sample from the approximate distribution $\hat{P}$.

Even if $\|f\|_{2,\hat{P}}^2$ is still inaccessible, it can be very easily approximated by

$$\frac{1}{M}\sum_{i=1}^{M} \|f(\hat{x}_i)\|_2^2 \tag{5}$$

for some set $\{\hat{x}_i\}$ drawn i.i.d. from $\hat{P}$. For samples outside the training set, this estimate is a U-statistic of order 1, and has an asymptotic Gaussian distribution to which finite sample estimates converge quickly [21].

To summarize, the objective that we propose to minimize can be written as follows

$$\arg\min_W R_N(W) + \frac{\lambda}{M}\sum_{i=1}^{M} \|f_W(\hat{x}_i)\|_2^2. \tag{6}$$

We next motivate this objective by showing that non-stochastic norm computation is NP-hard for deep networks (Section 3), and we subsequently demonstrate that this objective function leads to stability-based generalization bounds that converge as $\mathcal{O}(N^{-\frac{1}{2}})$ (for convex function sets) in Section 4.

3 NP-hardness of function norm computation

Let us go back to the exact L_2 function norm (Eq. (3)).

Proposition 1. *The computation of the canonical Lebesgue L_2-norm $\|f\|_2 \in \mathbb{R}^+$ from the weights of a network is NP-hard.*

We will prove this statement by a linear time reduction of the classic NP-complete problem of Boolean 3-satisfiability [7] to the computation of the norm of a particular network with rectified linear unit (ReLU) activation functions [12]. Furthermore, we can construct this network such that it always has finite L_2-norm. For this purpose, we define the following functions:

Definition 1. *For a fixed $\varepsilon < 0.5$, we define $f_0 : \mathbb{R} \to [0, 1]$ as:*

$$f_0(x) = \varepsilon^{-1}[\max(0, x + \varepsilon) - 2\max(0, x) + \max(0, x - \varepsilon)], \tag{7}$$

and

$$f_1(x) = f_0(x - 1). \tag{8}$$

These functions place some non-zero values in the ε neighborhood of $x = 0$ and $x = 1$, respectively, and zero elsewhere. Furthermore, $f_0(0) = 1$ and $f_1(1) = 1$ (Figure 1).

A sentence in 3-conjunctive normal form (3-CNF) consists of the conjunction of c clauses. Each clause is a disjunction over 3 literals, a literal being either a logical variable or its negation. For the construction of our network, each variable will be identified with a dimension of our input $x \in \mathbb{R}^p$, we denote each of c clauses in the conjunction b_j for $1 \leq j \leq c$, and each literal l_{j_k} for $1 \leq k \leq 3$ will be associated with $f_0(x_i)$ if the literal is a negation of the ith variable or $f_1(x_i)$ if the literal is not a negation. Note that each variable can appear in multiple clauses with or without negation, and therefore the indexing of literals is distinct from the indexing of variables.

Definition 2. *We define the function $f_\wedge : [0, 1]^c \to [0, 1]$ as follows:*

$$f_\wedge(z) = f_0\left(\sum_{i=1}^{c} z_i - c\right), \tag{9}$$

where $f_\wedge(z) = 1$ for $z = \mathbf{1}$, $\mathbf{1}$ being a vector of ones.

Definition 3. *We define the function $f_\vee : [0, 1]^3 \to [0, 1]$ as follows:*

$$f_\vee(z) = \sum_{j=1}^{3} f_0\left(\sum_{i=1}^{3} z_i - j\right). \tag{10}$$

A proof of $f_\vee$ having values in $[0, 1]$ is provided in the appendix A.

In order to ensure our network defines a function with finite Lebesgue measure, we may use the following function to truncate values outside the unit cube.

Definition 4. $f_T(x) = \|x\|_1 \cdot (1 + \dim(x))^{-1}$.

This function is constructed such that it has a value greater than one only strictly outside the unit cube, and is straightforward to implement in a shallow network with ReLU activations. For bounded f, $\max(f(x) - f_T(x), 0)$ will truncate all values of f outside a region of finite support guaranteeing finite Lebesgue measure.

Proving Proposition 1 is equivalent to proving the following:

Lemma 2. *Given a Boolean expression $\mathcal{B}$ with p variables in conjunctive normal form in which each clause is composed of three literals, we can construct, in time polynomial in the size of the predicate, a network of depth 4 and realizing a function $f : \mathbb{R}^p \to \mathbb{R}$ whose L_2 norm is strictly positive if the predicate is satisfiable, and zero otherwise.*

Proof. We construct the three hidden layers of the network as follows:

1. In the first layer, we compute $f_0(x_i)$ for the literals containing negations and $f_1(x_i)$ for the literals without negation. These operators introduce one hidden layer of at most $6p$ nodes.

2. The second layer computes the clauses of three literals using the function $f_\vee$. This operator introduces one hidden layer with a number of nodes linear in the number of clauses in $\mathcal{B}$.

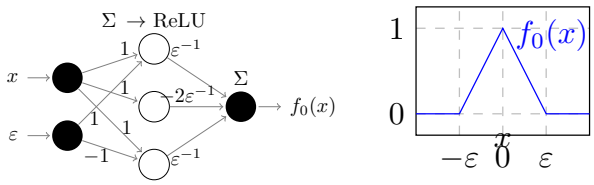


Figure 1: Neural network computing f_0 defined in Definition 1, and plot of this function.

3. Finally, each of the outputs of $f_\vee$ are concatenated into a vector and passed to the function $f_\wedge$. This operator requires one application f_0 and thus introduces one hidden layer with a constant number of nodes.

Let $f_{\mathcal{B}}$ be the function coded by this network. By optionally adding an additional layer implementing the truncation in Definition 4 we can guarantee that the resulting function has finite norm. It remains to show that the norm of $f_{\mathcal{B}}$ is strictly positive if and only if $\mathcal{B}$ is satisfiable.

If $\mathcal{B}$ is satisfiable, let $x \in \{0,1\}^p$ be a satisfying assignment of $\mathcal{B}$; by construction $f_{\mathcal{B}}(x)$ is 1, as all the clauses evaluate exactly to 1. $f_{\mathcal{B}}$ being continuous by composition of continuous functions, we conclude that $\|f_{\mathcal{B}}\|_2 > 0$.

Now suppose $\mathcal{B}$ not satisfiable. For a given clause b_j, consider the dimensions associated with the variables contained within this clause and label them x_{j_1}, x_{j_2}, and x_{j_3}. Now, for all 2^3 possible assignments of the variables, consider the 2^3 polytopes defined by restricting each x_{j_k} to be greater than or less than 0.5. Exactly one of those variable assignments will have $l_{j_1} \vee l_{j_2} \vee l_{j_3} = \text{false}$. The function value over the corresponding polytope must be zero. This is because the output of the jth $f_\vee$ must be zero over this region by construction, and therefore the output of the $f_\wedge$ will also be zero as the summation of all the $f_\vee$ outputs will be at most $c-1$. For each of the 2^p assignments of the Boolean variables at least one clause will guarantee that $f_{\mathcal{B}}(x) = 0$ for all x in the corresponding polytope, as the sentence is assumed to be unsatisfiable. The union of all such polytopes is the entire space $\mathbb{R}^p$. As $f_{\mathcal{B}}(x) = 0$ everywhere, $\|f_{\mathcal{B}}\|_2 = 0$. □

Corollary 1. $\|\max(f_{\mathcal{B}} - f_T, 0)\|_2 > 0 \iff \mathcal{B}$ *is satisfiable, and* $\max(f_{\mathcal{B}} - f_T, 0)$ *has finite Lebesgue measure for all* $\mathcal{B}$.

Corollary 2. *Although not all norms are equivalent in the space of continuous functions, Lemma* 2 *implies that any function norm for a network of depth* ≥ 4 *must be NP-hard since for all norms* $\|f\| = 0 \iff f = \mathbf{0}$.

4 Stability and generalization bounds

In this section, we show how our proposed regularization induces control of the generalization error with some conditions on the data. For this proof, we consider that the hypothesis function set is a convex subset of a linear space, as a general regularization scheme should exhibit good generalization bounds when applied to well behaved function sets. Note that this condition is not satisfied for DNNs, and satisfactory generalization bounds for DNNs remains an active area of research [8, 13, 17, 33]. See also [25] for a discussion of convexity of neural network function sets and generalization bounds.

If we consider the training set $S = \{(x_i, y_i), i = 1 \dots N\}$, a training algorithm A, f the function resulting from A applied to S and $f^{\setminus i}$ the function resulting from A applied to $S \setminus \{(x_i, y_i)\}$, uniform stability is defined as follows:

Definition 5 (Uniform stability)**.** *A has hypothesis stability* β *with respect to* ℓ *if*

$$\forall S, \forall i \in \{1, \dots, N\}, \ \|\ell(f(\cdot), \cdot) - \ell(f^{\setminus i}(\cdot), \cdot)\|_\infty \leq \beta. \tag{11}$$

If we consider the *leave-one-out* error defined as $R_{loo}(A, S) = \frac{1}{N}\sum_{i=1}^{N} \ell(f^{\setminus i}(x_i), y_i)$, the empirical error defined as $R_{emp}(A, S) = \frac{1}{N}\sum_{i=1}^{N} \ell(f(x_i), y_i)$, and the generalization error as $R(A, S) = \mathbb{E}_{(x,y)}[\ell(f(x), y)]$, the following theorem applies:

Theorem 1 ([5])**.** *If an algorithm A is* β*-uniform stable, and the loss function is bounded by* C $(0 \leq \ell(y, y') \leq C)$*, then for any* $N \geq 1$*, with probability* $1 - \delta$*,*

$$R(A, S) \leq R_{loo}(A, S) + \beta + (4N\beta + C)\sqrt{\frac{ln(1/\delta)}{2N}} \tag{12}$$

and

$$R(A, S) \leq R_{emp}(A, S) + 2\beta + (4N\beta + C)\sqrt{\frac{ln(1/\delta)}{2N}}. \tag{13}$$

In addition to the previous results, [5] provide tools to derive bounds on the stability of a regularized algorithm. In the following, we use these tools to prove that our regularization guarantees the uniform stability of the algorithm. Their study of regularized objectives is based on two conditions:

1. The hypothesis function set $\mathcal{F}$ should be a convex subset of a linear space. For example, we can consider a *Reproducing Kernel Hilbert Space* (RKHS). Even though the proposed regularizer is not non-decreasing with respect to the Hilbert norm related to the kernel and thus the representer theorem is violated, we can consider the span of the kernel on the training samples as our hypothesis set. Our first experiments (Sec. 5) are related to this setting.

2. The loss function ℓ should be σ-admissible with respect to $\mathcal{F}$, which means that ℓ is convex with respect to its first argument, and that

$$\begin{aligned} &\forall x_1, x_2 \in \mathcal{X}, \forall y' \in \mathcal{Y}, \forall f_1, f_2 \in \mathcal{F}, \\ &|\ell(f_1(x_1), y') - \ell(f_2(x_2), y')| \leq \sigma |f_1(x_1) - f_2(x_2)|. \end{aligned} \tag{14}$$

4.1 Stability of exact optimizer

Here, we suppose that we have an oracle that minimizes the objective

$$\arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i) + \frac{\lambda}{M} \sum_{i=1}^{M} \|f(\hat{x}_i)\|_2^2. \tag{15}$$

Note that *empirical* approximation of the function norm $f \mapsto \frac{1}{M} \sum_{i=1}^{M} \|f(\hat{x}_i)\|_2^2$ is still a proper closed convex function.

To derive a stability-based generalization bound for our algorithm, we make use of [5, Lemma 21]. In the following, for any operator O, let $d_O(f, f')$ be its Bregman divergence.

Lemma 3 ([5]). *If ℓ is σ-admissible with respect to $\mathcal{F}$, where $\mathcal{F}$ is a vector space and Ω is a proper closed convex function, then*

$$d_\Omega(f, f^{\setminus i}) + d_\Omega(f^{\setminus i}, f) \leq \frac{\sigma}{\lambda N} |f^{\setminus i}(x_i) - f(x_i)|, \tag{16}$$

suppose for simplicity of writing that f has a single output ($f \in \mathbb{R} \mapsto \mathbb{R}$).

Proposition 2. *If ℓ is σ-admissible, if the training data has a finite radius R, and if $\hat{P}(x) \geq m > 0, \forall x \in \mathcal{X}$ then the oracle defined above is β-uniform stable with*

$$\beta = \frac{R\sigma^2}{2\lambda m N}, \tag{17}$$

Proof. For $\Omega(f) = \frac{1}{M} \sum_{i=1}^{M} \|f(\hat{x}_i)\|_2^2$, we have $d_\Omega(g, g') = \frac{\lambda}{M} \sum_{i=1}^{M} \|g(\hat{x}_i) - g'(\hat{x}_i)\|_2^2$. Indeed, if $\langle g, g' \rangle$ is the canonical $L_{2,\hat{P}}$ inner product, we have

$$\begin{aligned} d_\Omega(g, g') = &\frac{1}{M} \sum_{i=1}^{M} \|g(\hat{x}_i)\|_2^2 \\ &- \frac{1}{M} \sum_{i=1}^{M} \|g'(\hat{x}_i)\|_2^2 - \langle g - g', \nabla\Omega(g') \rangle. \end{aligned} \tag{18}$$

As the Fréchet derivative of $f \mapsto f(\hat{x}_i)$ is $\delta_{\hat{x}_i}$,

$$\begin{aligned} d_\Omega(g, g') = &\frac{1}{M} \sum_{i=1}^{M} \|g(\hat{x}_i)\|_2^2 - \frac{1}{M} \sum_{i=1}^{M} \|g'(\hat{x}_i)\|_2^2 \\ &- \langle g - g', \frac{2}{M} \sum_{i=1}^{M} g'(\hat{x}_i) \delta_{\hat{x}_i} \rangle \qquad (19) \\ = &\frac{1}{M} \sum_{i=1}^{M} \|g(\hat{x}_i)\|_2^2 + \|g'(\hat{x}_i)\|_2^2 - 2\langle g'(\hat{x}_i), g(\hat{x}_i) \rangle \qquad (20) \\ = &\frac{1}{M} \sum_{i=1}^{M} \|g(\hat{x}_i) - g'(\hat{x}_i)\|_2^2. \qquad (21) \end{aligned}$$

Using Lemma 3, we have $\forall S, k$

$$\frac{2}{M}\sum_{i=1}^{M}\|f^{\setminus k}(\hat{x}_i)-f(\hat{x}_i)\|_2^2 \leq \frac{\sigma}{\lambda N}|f^{\setminus k}(x_k)-f(x_k)| \tag{22}$$

$$\leq \frac{\sigma|\mathcal{X}|}{\lambda N}\|f^{\setminus k}-f\|_{1,\mathcal{U}(\mathcal{X})}, \tag{23}$$

given that:

$$\forall x, \forall g, |g(x)| \leq |\mathcal{X}|.\|g\|_{1,\mathcal{U}(\mathcal{X})} \tag{24}$$

where $|\mathcal{X}|$ is the volume of the domain $\mathcal{X}$. Thus, as (23) is valid for any M, and as $\|f^{\setminus k}-f\|_{2,\hat{P}}^2 = \lim_{M\to\infty}\frac{1}{M}\sum_{i=1}^{M}\|f^{\setminus k}(\hat{x}_i)-f(\hat{x}_i)\|_2^2$, we can write:

$$\|f^{\setminus k}-f\|_{2,\hat{P}}^2 \leq \frac{\sigma|\mathcal{X}|}{2\lambda N}\|f^{\setminus k}-f\|_{1,\mathcal{U}(\mathcal{X})}. \tag{25}$$

As we have, for any function g

$$\|g\|_{1,\mathcal{U}(\mathcal{X})} = \frac{1}{|\mathcal{X}|}\int |g(x)|\mathbf{1}_{\mathcal{X}}(x)dx \tag{26}$$

$$= \frac{1}{|\mathcal{X}|}\int |g(x)|\frac{\mathbf{1}_{\mathcal{X}}(x)}{\hat{P}(x)}\hat{P}(x)dx \tag{27}$$

$$\leq \|g\|_{2,\hat{P}}\left\|\frac{\mathbf{1}_{\mathcal{X}}}{|\mathcal{X}|\hat{P}}\right\|_{2,\hat{P}} \text{ (by Cauchy-Schwartz)} \tag{28}$$

$$= \|g\|_{2,\hat{P}}\left[\frac{1}{|\mathcal{X}|}\mathbb{E}_{\mathcal{U}(\mathcal{X})}\left(\frac{1}{\hat{P}}\right)\right]^{\frac{1}{2}} \tag{29}$$

combined with (25), we obtain

$$\|f^{\setminus k}-f\|_{2,\hat{P}} \leq \frac{\sigma|\mathcal{X}|^{\frac{1}{2}}}{2\lambda N}\left[\mathbb{E}_{\mathcal{U}(\mathcal{X})}\left(\frac{1}{\hat{P}}\right)\right]^{\frac{1}{2}} \tag{30}$$

and

$$\|f^{\setminus k}-f\|_{1,\mathcal{U}(\mathcal{X})} \leq \frac{\sigma}{2\lambda N}\mathbb{E}_{\mathcal{U}(\mathcal{X})}\left(\frac{1}{\hat{P}}\right). \tag{31}$$

Thus, using the σ-admissibility of ℓ and (24)

$$|\ell(f(x),y)-\ell(f^{\setminus k}(x),y)| \leq \sigma|f(x)-f^{\setminus k}(x)| \tag{32}$$

$$\leq \sigma|\mathcal{X}|.\|f^{\setminus k}-f\|_{1,\mathcal{U}(\mathcal{X})} \tag{33}$$

$$\leq \frac{\sigma^2|\mathcal{X}|}{2\lambda N}\mathbb{E}_{\mathcal{U}(\mathcal{X})}\left(\frac{1}{\hat{P}}\right) \tag{34}$$

$$= \frac{\sigma^2}{2\lambda N}\int_{\mathcal{X}}\frac{1}{\hat{P}(x)}dx. \tag{35}$$

As this inequality holds for any (x,y) and any S, if $|\mathcal{X}| \leq R$ and $\hat{P}(x) \geq m > 0$, the considered algorithm is β-uniform stable with

$$\beta = \frac{\sigma^2 R}{2\lambda m N}. \tag{36}$$

Note that the condition about the data is satisfied in most practical cases, and the condition on $\hat{P}$ is satisfied as soon as the support of this distribution is large enough so that the tail falls outside of the data domain. □

Proposition 3. *If the loss function satisfies the condition in Theorem* 1*, then both* $\mathbb{E}_S(|R(A,S)-R_{emp}(A,S)|)$ *and* $\mathbb{E}_S(|R(A,S)-R_{loo}(A,S)|)$ *converge as* $\mathcal{O}(N^{-\frac{1}{2}})$.

The proof of this proposition is obtained by simply plugging the obtained stability parameter in Theorem 1.

4.2 Convergence of stochastic gradient descent

Exploiting the convexity of the regularizer, a full proof of the convergence of SGD is provided in appendix C.

Proposition 4 (Sufficient condition for SGD convergence)**.** *For all λ and $P_{XY}(x,y)$ s.t. $\int P_{XY}(x,y)dy = P(x)$, there exists a constant C such that SGD minimization of*

$$\mathbb{E}_{(x,y)\sim P_{XY}}\left[\ell(f(x),y)\right] + \lambda \mathbb{E}_{x\sim \hat{P}}\left[\|f(x)\|_2^2\right] \tag{37}$$

converges if $\frac{\hat{P}(x)}{P(x)} \leq C, \forall x \in \text{supp}(P)$.

For $\mathcal{S} \subseteq \mathcal{X}$, define $\hat{P}|_{\mathcal{S}}(x) = \begin{cases} \frac{\hat{P}(x)}{\int_{\mathcal{S}} \hat{P}(x)dx} \text{ if } x \in \mathcal{S}, \\ 0 \text{ otherwise.} \end{cases}$

Denote $\hat{C} = \frac{C}{\int_{\text{supp}(P)} \hat{P}(x)dx}$, we have for $x \in \text{supp}(P)$:

$$\frac{\hat{P}(x)}{P(x)} \leq C \iff \frac{\hat{P}|_{\text{supp}(P)}(x)}{P(x)} \leq \hat{C} \tag{38}$$

$$\iff \log\left(\frac{\hat{P}|_{\text{supp}(P)}(x)}{P(x)}\right) \leq \log\left(\hat{C}\right) \tag{39}$$

$$\implies D_{KL}(\hat{P}|_{\text{supp}(P)}||P) \leq \log(\hat{C}), \tag{40}$$

where D_{KL} denotes the KL divergence.

Note that $\hat{P}|_{\text{supp}(P)}$ is well defined thanks to the condition on $\hat{P}$ for stability, $\hat{P}(x) \geq m > 0, \forall x \in \mathcal{X}$.

5 Experiments and results

To test the proposed regularizer, we consider three different settings: (i) A classification task with kernelized logistic regression, where the hypothesis set is convex, and our derived stability and generalization bound holds without any additional assumptions on a network topology or degree of regularization; (ii) A classification task with DNNs; (iii) A segmentation task with DNNs.

5.1 Oxford Flowers classification with kernelized logistic regression

The objective in this experiment is to empirically validate the theory in Sec. 4.1. According to the theory, the generalization bound induced by our weighted function norm is of the same order as the generalization bound induced by the RKHS norm. The following experiment shows that both norms have similar behavior on the test data.

Data and Kernel For this experiment we consider the 17 classes Oxford Flower Dataset, composed of 80 images per class, and precomputed kernels that have been shown to give good performance on a classification task [26, 27]. We have taken the mean of Gaussian kernels as described in [9].

Settings To test the effect of the regularization, we train the logistic regression on a subset of 10% of the data, and test on 20% of the samples. The remaining 70% are used as potential samples for regularization. For both regularizers, the regularization parameter is selected by a 3-fold cross validation. For the weighted norm regularization, we used a 4 different sample sizes ranging from 20% to 70% of the data. This procedure is repeated on 10 different splits of the data for a better estimate. The optimization is operated by quasi-Newton gradient descent that is guaranteed to converge thanks to the convexity of the objective.

Results Figure 2 shows the means and standard deviations of the accuracy on the test set obtained without regularization, and with Tikhonov regularization using the Hilbert norm, along with the histogram of accuracies obtained with the weighted norm regularization with the different sample sizes and across the ten trials. This figure demonstrates the equivalent effect of both regularizer, as expected with the stability properties induced by both norms.

The use of the weighted function norm is more useful for DNNs, where no other direct function complexity control is known to be polynomial. The next two experiments show the efficiency of our regularizer when compared to other regularization strategies: Weight decay [22], dropout [14] and batch normalization [16].

For these experiments, we generate the regularization samples as suggested by our theoretical results, i.e. from an approximation $\hat{P}$ of the data distribution P, such that $D_{KL}(\hat{P}|_{\text{supp}(P)}\|P)$ is bounded, and the support

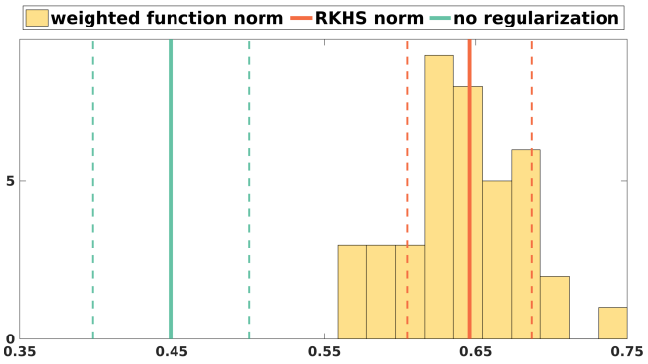
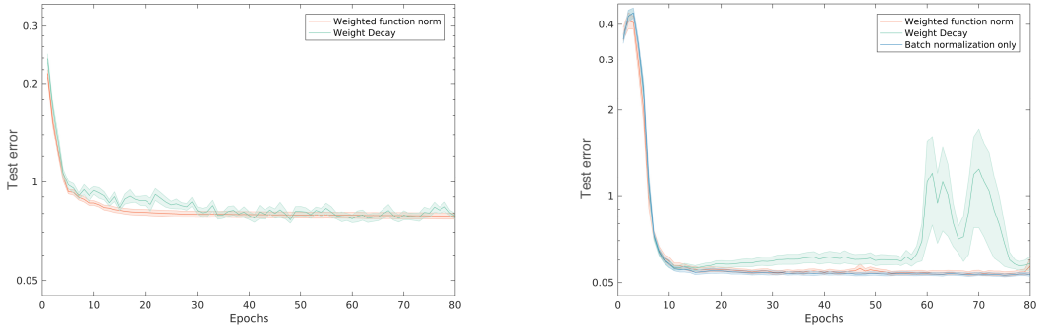


Figure 2: Histogram of accuracies with weighted function norm on the Oxford Flowers dataset over 10 trials with 4 different regularization sample sizes, compared to the mean and standard deviation of RKHS norm performance, and the mean and standard deviation of the accuracy obtained without regularization.



(a) Without batch normalization

(b) With batch normalization

Figure 3: Performance of weighted function norm regularization on MNIST in a low sample regime, compared to weight decay (a) without and (b) with batch normalization. The performance is averaged over 10 trials, training on different subsets of 600 samples, with model selection with 3-fold cross validation for regularization parameters. The regularization samples are sampled using a variational autoencoder.

of $\hat{P}$ is broader than the support of P. Variational autoencoders (VAE) have shown properties that make them a suitable choice for this approximation. For both experiments, we will use VAEs with 2 hidden layers as in [19]. More details about the training and sampling are given in appendix E.

5.2 MNIST classification

Data and Model In order to test the performance of the tested regularization strategies, we consider only a small subset of 600 samples of the MNIST dataset for training. The tests are conducted on 10,000 samples. We consider the LeNet architecture [20], with various combinations of weight decay, weighted function norm, and batch normalization (cf. Figure 3).

Settings We train the model on 10 different random subsets of 600 samples. A 3-fold cross validation is performed to select the regularization parameter for weight decay and weighted function norm. An 11th subset of samples independent of the 10 training sets is used to train the VAE. For each batch, a new sample is generated for the function norm estimation. SGD is performed using ADAM [18]. The selected models are applied to the test set, and classification error curves are averaged over the 10 trials.

Results Figure 3 displays the averaged curves and error bars. Figure 3a shows the curves obtained without batch normalization. Figure 3b shows the results with batch normalization. In both figures, we observe unpredictable variance caused by weight decay. This explains the need of additional implicit regularization, such as early stopping, when weight decay is used. For MNIST, however, batch normalization alone is sufficient to obtain a smooth and stable convergence towards a good performance. Our regularizer, in contrast to weight decay, also obtains good performance and stable convergence. In the next experiment, we show that our regularizer can improve performance over batch normalization on a real-world medical image segmentation task.

Class	BN	BN + WFN
LEFT SIDE		
Thalamus	0.75±0.02	0.77±0.01
Caudate	0.44±0.04	**0.54±0.03**
Putamen	0.57±0.03	0.60±0.04
Pallidum	0.47±0.04	0.52±0.4
RIGHT SIDE		
Thalamus	0.75±0.02	0.76±0.02
Caudate	0.47±0.04	0.46±0.04
Putamen	0.62±0.03	0.62±0.04
Pallidum	0.47±0.03	**0.53±0.03**
MEAN	0.57±0.02	**0.60±0.01**

Table 1: Performance of the networks on the 18 patients in the ISBR dataset averaged across folds, with batch normalization (BN), and the weighted function norm (WFN).

5.3 Brain image segmentation

Data and Model We consider the task of segmenting sub-cortical structures of the human brain in Magnetic Resonance (MR) image data, on the publicly available dataset provided by the Internet Brain Segmentation Repository (IBSR) [29] consisting of 18 densely annotated segmentation volumes. We used the deep segmentation network proposed by Shakeri et al. [31], which is a variation of the Deeplab architecture [6] adapted to this brain segmentation task. We replicate their evaluation setting on IBSR: a 3-fold cross-validated evaluation, with for each fold, respectively 11 – 1 – 6 train, validation, and test MR segmentation volumes, and selecting the subset of 8 primarily subcortical structures: left and right thalamus, caudate, putamen, and pallidum.

Settings The training disregards the 3d structure of the MR segmentation volumes, and treats each of the 256 axial brain slices of the dataset as independent 256×128 grayscale images. This results 2556 non-empty grayscale images for the whole dataset, with an over-representation of the background class – a low-data regime in the context of deep neural networks, where we expect regularization to play an essential role.

As we are interested in quantifying the direct impact of our proposed regularization method on the feed-forward neural network, we do not replicate the extra step of Gaussian CRF inference and directly compare the performance of the base network under various regularization regimes. For the variant with batch normalization we follow the standard procedure of inserting batch normalization layers between the convolutional layers and their ReLU activations of the network (see details in appendix D).

Following the experimental procedure of [31] we train the network with stochastic gradient descent and set the learning rate schedule to exponentially decrease from 10^{-2} to 10^{-4} over 35 epochs. For each fold, the λ regularization parameter of the function norm is selected among $(10^{-8}, 10^{-6}, 10^{-4})$, based on the performance of the network on the validation set averaged over the last 5 epochs of the training. The VAEs used for each fold were trained only on the training and validation sets. The obtained models are used during training to generate a regularization sample twice the size of the training sample. Figure 5 shows examples of generated images.

Results The mean per-class accuracy for each of the eight classes of interest along with the mean accuracy across all classes are given in Table 1, showing a significant improvement associated to the combined used of the weight function norm regularized and batch regularization with respect to batch normalization alone. Qualitative results are shown in Figure 4, where we can see that weighted function norm regularization results in better segmentation accuracy on fine subcortical structures.

6 Discussion

This paper proposes a new alternative for neural network regularization: a sampling-based approximation to a weighted L_2 function norm. The necessity of the sampling-based approximation comes from the NP-hardness of the function integration (Proposition 1). Sampling from a data-related distribution is proposed in order to avoid the curse of dimensionality. Even in the case that the proposed regularizer is undersampled and therefore only an approximation to a proper function norm, it still benefits from interesting properties that are lacking in other regularizers. Indeed, it controls directly the function complexity, and is a proper closed convex operator of the network function. This second condition is particularly important for uniform stability as shown in [5]. This property is exploited in Section 4 in order to show the uniform stability induced by the use of our regularizer

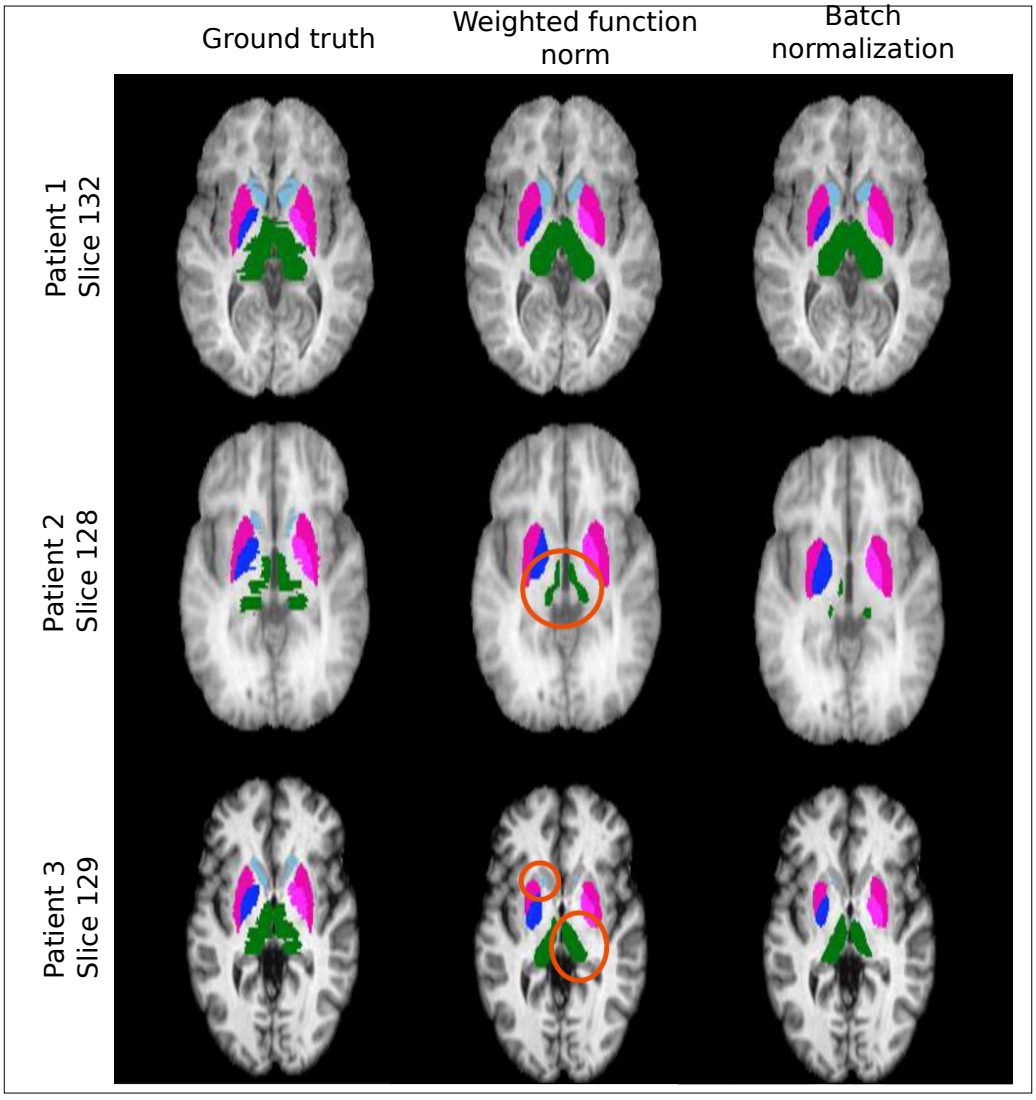


Figure 4: Some examples of segmentation on the ISBR dataset. In some of the slices, both regularization strategies behave similarly (first row). In many slices, the weighted function norm results in an improvement over batch normalization for detection of fine structures (red circles). See appendix F for better visualization.

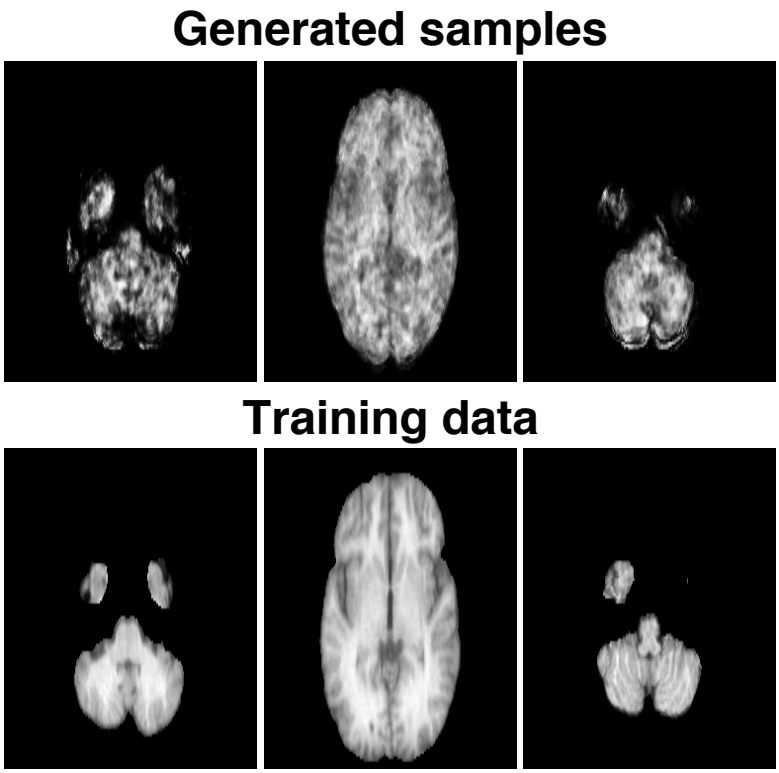


Figure 5: Samples from the VAE generator trained on 12 patients (training + validation set), and similar slices hand-picked from the training data.

and derive interesting $\mathcal{O}(N^{-\frac{1}{2}})$ generalization bounds. This proof can be extended in the future by a better understanding of the structure of the function set, and by the consideration of stochastic algorithms as in e.g. [13].

The proposed method adds a stochastic term to the classical empirical risk. Nevertheless, the SGD convergence is still guaranteed (as proved in appendix C), as soon as the sampling distribution used for the estimation of the function norm is sufficiently similar to the data distribution (Equation (40)).

In order to have an empirical proof-of-concept, we applied the function norm regularizer to a range of real-world prediction problems. Section 5 shows some results using regularization samples from the same distribution as the training data. The use of this distribution satisfies the sufficient condition of a bounded ratio of distributions, and therefore ensures the SGD convergence. The obtained results confirm the effect of the proposed regularizer on the generalization error estimated by the error on test samples. Indeed, for large samples, it shows a behavior comparable to classical DNN regularizers, and for small samples it shows a significantly better behavior consisting of (i) a faster convergence to the optimum, (ii) and a better optimum than the other methods.

The experiments demonstrate that the proposed regularization in combination with batch normalization has in all circumstances performance comparable to or better than the best existing alternatives proposed in the literature.

An important novelty of the proposed regularizer is that it induces a behavior that is well explained by notions of classical learning theory, making a step towards closing the gap between theory and practice in DNNs.

7 Conclusion

This paper proposes sampling-based approximations to weighted L_2 function norms as regularizers for deep neural networks, after observing and proving the NP-hardness of canonical function norm computation. Based on the convexity of this approximate norm over the function set, we prove $\mathcal{O}(N^{-\frac{1}{2}})$ generalization bounds for the obtained objective. Moreover, we identify sufficient conditions for the convergence of SGD on our objective, including that the distribution used for the norm estimation should have bounded KL divergence from the marginal distribution over input samples in the training data (Equation (40)). Finally, we implemented an empirical validation of the expected effect of the employed regularizer on generalization with experiments on the Oxford Flowers dataset, the MNIST image classification problem, and the ISBR brain segmentation task. As a conclusion, the proposed regularizer is not only able to considerably improve the performance of neural networks in the small sample regime, but also has a behavior that is well explained by classical learning theory.

Acknowledgments

This work is funded by Internal Funds KU Leuven, FP7-MC-CIG 334380, the Research Foundation - Flanders (FWO) through project number G0A2716N, and an Amazon Research Award.

References

[1] L. Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8), 1991.

[2] L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

[3] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[4] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[5] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2(Mar):499–526, 2002.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[7] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[8] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, 2017.

[9] P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *International Conference on Computer Vision*, pages 221–228, 2009.

[10] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

[11] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[12] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947, 2000.

[13] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, 2016.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

[15] S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

[17] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[19] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

[20] Y. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Säckinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *International Conference on Artificial Neural Networks*, 1995.

[21] A. J. Lee. *U-Statistics: Theory and Practice*. CRC Press, 1990.

[22] J. Moody, S. Hanson, A. Krogh, and J. A. Hertz. A simple weight decay can improve generalization. *Advances in Neural Information Processing Systems*, 4:950–957, 1995.

[23] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[24] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[25] B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In P. Grünwald, E. Hazan, and S. Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 1376–1401, 2015.

[26] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454, 2006.

[27] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

[28] H. Robbins and D. Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Herbert Robbins Selected Papers*, pages 111–135. Springer, 1985.

[29] T. Rohlfing. Image similarity and tissue overlaps as surrogates for image registration accuracy: Widely used but unreliable. *IEEE Transactions on Medical Imaging*, 31(2):153–163, 2012.

[30] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2001.

[31] M. Shakeri, S. Tsogkas, E. Ferrante, S. Lippe, S. Kadoury, N. Paragios, and I. Kokkinos. Sub-cortical brain structure segmentation using F-CNN's. In *International Symposium on Biomedical Imaging*, pages 269–272. IEEE, 2016.

[32] V. Vapnik. *Statistical learning theory*. Wiley, 1998.

[33] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

A NP-hardness of DNN L_2 function norm

A.1 Output of OR blocks

Following the steps of Proposition 3, this function is defined for $X \in \mathbb{R}^3$ and:

$$F(X) = f_0(\sum_i f_1(X_i) - 1) + f_0(\sum_i f_1(X_i) - 2) + f_0(\sum_i f_1(X_i) - 3) \tag{41}$$

To compute the values of F over $\mathbb{R}^3$, we consider two cases for every X_i: $X_i \in (1-\varepsilon, 1+\varepsilon)$ and $X_i \notin (1-\varepsilon, 1+\varepsilon)$.

Case 1: all $X_i \notin (1-\varepsilon, 1+\varepsilon)$: In this case, we have $\sum_i f_1(X_i) = 0$. Therefore,$|\sum_i f_1(X_i) - k| > \varepsilon, \forall k \in \{1, 2, 3\}$, and $F(X) = 0$.

Case 2: only one $X_i \in (1-\varepsilon, 1+\varepsilon)$: Without loss of generality, we suppose that $X_1 \in (1-\varepsilon, 1+\varepsilon)$ and $X_{2,3} \notin (1-\varepsilon, 1+\varepsilon)$. Thus:

$$\sum_i f_1(X_i) = 1 - \frac{1}{\varepsilon}|X_1 - 1|. \tag{42}$$

Thus, we have $\sum_i f_1(X_i) - 2 < -1$, $\sum_i f_1(X_i) - 3 < -2$, and $\sum_i f_1(X_i) - 1 < -\varepsilon \iff |X_1 - 1| > \varepsilon^2$. Therefore:

$$F(X) = \begin{cases} 1 - \frac{1}{\varepsilon^2}|X_1 - 1|, \text{ for } 0 \leq |X_1 - 1| \leq \varepsilon^2 \\ 0, \text{ otherwise.} \end{cases} \tag{43}$$

Case 3: two $X_i \in (1-\varepsilon, 1+\varepsilon)$: Suppose $X_{1,2} \in (1-\varepsilon, 1+\varepsilon)$. We have then:

$$\sum_i f_1(X_i) = 2 - \frac{1}{\varepsilon}|X_1 - 1| - \frac{1}{\varepsilon}|X_2 - 1|. \tag{44}$$

Therefore:

1. $\sum_i f_1(X_i) - 3 < -1$,
2. $$|\sum_i f_1(X_i) - 2| < \varepsilon \iff |X_1 - 1| + |X_2 - 1| < \varepsilon^2 \tag{45}$$
3. $$|\sum_i f_1(X_i) - 1| < \varepsilon \iff \varepsilon - \varepsilon^2 < |X_1 - 1| + |X_2 - 1| < \varepsilon + \varepsilon^2 \tag{46}$$

The resulting function values are then:

$$F(X) = \begin{cases} 1 - \frac{1}{\varepsilon^2}|X_1 - 1| - \frac{1}{\varepsilon^2}|X_2 - 1|, \text{ for } X_{1,2} \in (45) \\ 1 - \frac{1}{\varepsilon}|1 - \frac{1}{\varepsilon}|X_1 - 1| - \frac{1}{\varepsilon}|X_2 - 1||, \text{ for } X_{1,2} \in (46) \\ 0, \text{ otherwise.} \end{cases} \tag{47}$$

As $\varepsilon < \frac{1}{2}$, the regions (45) and (46) do not overlap.

Case 4: all $X_i \in (1-\varepsilon, 1+\varepsilon)$: We have then:

$$\sum_i f_1(X_i) = 3 - \frac{1}{\varepsilon}|X_1 - 1| - \frac{1}{\varepsilon}|X_2 - 1| - \frac{1}{\varepsilon}|X_3 - 1|. \tag{48}$$

Therefore:

1. $$\begin{aligned} &|\sum_i f_1(X_i) - 2| < \varepsilon \\ &\iff |X_1 - 1| + |X_2 - 1| + |X_2 - 1| < \varepsilon^2 \end{aligned} \tag{49}$$
2. $$\begin{aligned} &|\sum_i f_1(X_i) - 2| < \varepsilon \\ &\iff \varepsilon - \varepsilon^2 < |X_1 - 1| + |X_2 - 1| + |X_3 - 1| < \varepsilon + \varepsilon^2 \end{aligned} \tag{50}$$
3. $$\begin{aligned} &|\sum_i f_1(X_i) - 1| < \varepsilon \\ &\iff 2\varepsilon - \varepsilon^2 < |X_1 - 1| + |X_2 - 1| + |X_3 - 1| < 2\varepsilon + \varepsilon^2 \end{aligned} \tag{51}$$

The resulting function values are then:

$$F(X) = \begin{cases} 1 - \frac{1}{\varepsilon^2}\sum_i |X_i - 1|, \text{ for } X_{1,2,3} \in (49) \\ 1 - \frac{1}{\varepsilon}|1 - \frac{1}{\varepsilon}\sum_i |X_i - 1||, \text{ for } X_{1,2,3} \in (50) \\ 1 - \frac{1}{\varepsilon}|2 - \frac{1}{\varepsilon}\sum_i |X_i - 1||, \text{ for } X_{1,2,3} \in (51) \\ 0, \text{ otherwise.} \end{cases} \tag{52}$$

Again, as $\varepsilon < \frac{1}{2}$, the regions (49), (50) and (51) do not overlap. Finally,

$$\forall X \in \mathbf{R}^3, 0 \leq F(X) \leq 1. \tag{53}$$

A.2 Examples of an unsatisfiable predicate

For a predicate $\mathcal{B}$, and a corresponding network function N, to illustrate the proof of

$$\mathcal{B} \text{ is unsatisfiable } \Rightarrow \|N\|_2 = 0, \tag{54}$$

we consider an example of an unsatisfiable predicate of three variables x, y and z. A known and easy example of such a predicate is:

$$\begin{aligned}\mathcal{B} = &(x \vee y \vee z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \\ &\wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \\ &\wedge (\neg x \vee \neg y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)\end{aligned} \tag{55}$$

Figure 6 displays the values of the different corresponding OR blocks to each of the clauses over the unit cube. We see that at each of the eight cuboids, there is one OR block that evaluates to 0, while the seven others give values lower than 1.

B Weight decay does not define a function norm

It is straightforward to see that weight decay, i.e. the norm of the weights of a network, does not define a norm of the function determined by the network. Consider a layered network

$$f(x) = W_d\sigma(W_{d-1}\sigma(\dots\sigma(W_1x)\dots)). \tag{56}$$

where the non-linear activation function can be e.g. a ReLU. The L_2 weight decay complexity measure is

$$\sum_{i=1}^{d} \|W_i\|_{\text{Fro}}^2, \tag{57}$$

where $\|\cdot\|_{\text{Fro}}$ is the Frobenius norm. A simple counter-example to show this cannot define a function norm is to set any of the matrices $W_j = \mathbf{0}$ and $f(x) = 0$ for all x. However $\sum_{i=1}^{d} \|W_i\|_{\text{Fro}}^2$ can be set to an arbitrary value by changing the W_i for $i \neq j$ although this does not change the underlying function.

C Stochastic gradient descent convergence

In practice, the minimization of

$$\arg\min_W R_N(W) + \frac{\lambda}{M}\sum_{i=1}^{M} \|f_W(\hat{x}_i)\|_2^2\,. \tag{58}$$

We next demonstrate is done using a variant of SGD. In general, SGD minimizes an objective of the form [2]

$$C(W) = \mathbb{E}_{(x,y)\sim P(x,y)}(Q(x,y,W)) \tag{59}$$

with an update of the form

$$W_{t+1} = W_t - \gamma_t H(x_t, y_t, W_t) \tag{60}$$

where

$$\mathbb{E}_{(x,y)\sim P(x,y)}[H(x,y,W)] = \nabla_W C(W). \tag{61}$$

In our case, our regularized objective (58) approximates

$$\mathbb{E}_{(x,y)\sim P(x,y)}[\ell(f_W(x),y)] + \lambda\mathbb{E}_{x\sim\hat{P}(x)}[\|f_W(x)\|_2^2] \tag{62}$$

$$= \int\int \ell(f_W(x),y)P(x,y)dxdy + \lambda\int \|f_W(x)\|_2^2\hat{P}(x)dx \tag{63}$$

$$= \int\int \ell(f_W(x),y) + \lambda\|f_W(x)\|_2^2\frac{\hat{P}(x)}{P(x)}P(x,y)dxdy \tag{64}$$

$$= \underbrace{\mathbb{E}_{(x,y)\sim P(x,y)}[\ell(f_W(x),y) + \lambda\frac{\hat{P}(x)}{P(x)}\|f_W(x)\|_2^2]}_{=:C(W)} \tag{65}$$

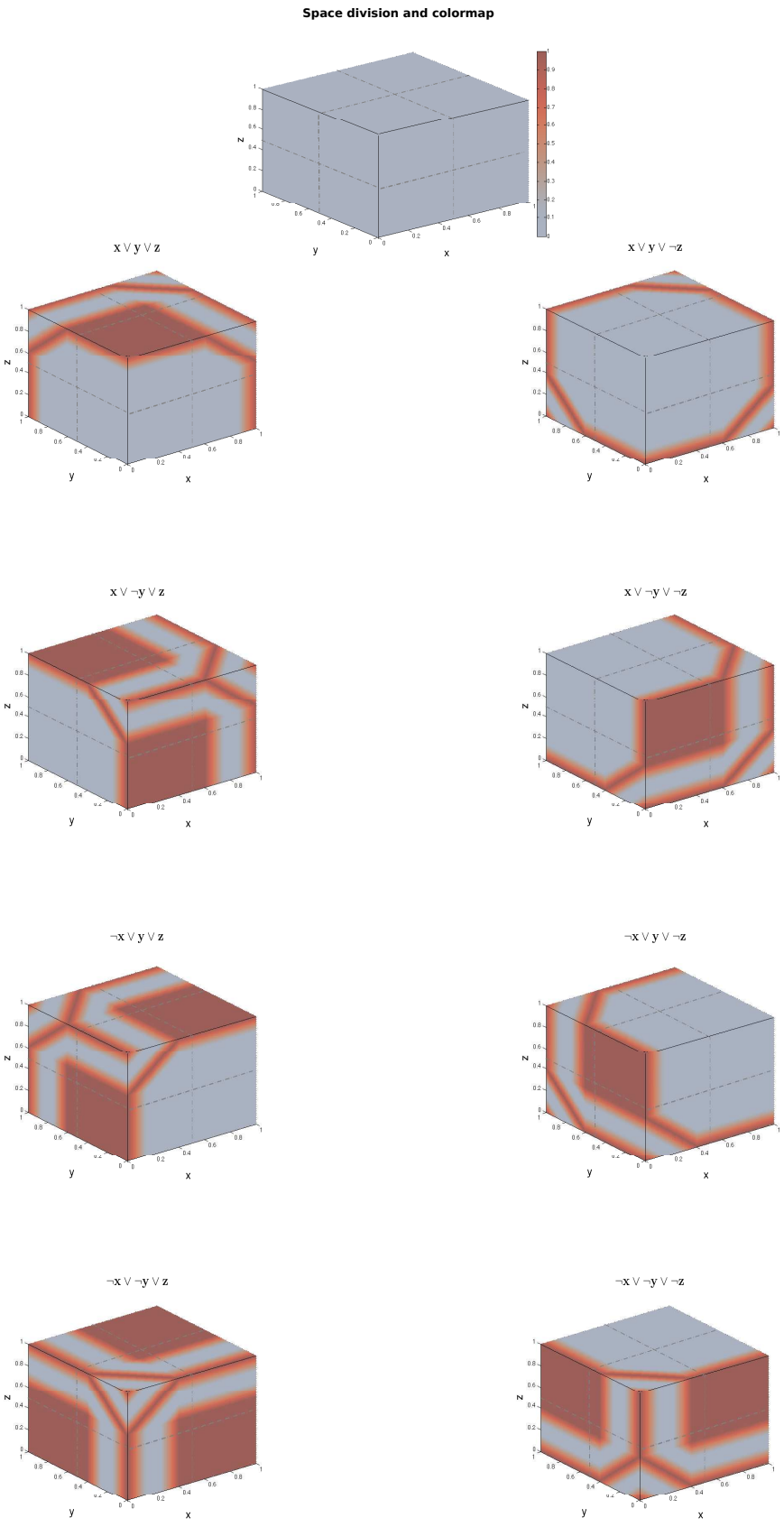


Figure 6: $f_\vee$ values on the unit cube

Note that the objective (65) is of the form (59). Moreover, if we define

$$H(x_t, y_t, W_t) = \nabla_W [\ell(f_W(x_t), y_t) + \lambda \frac{\hat{P}(x_t)}{P(x_t)} \|f_W(x_t)\|_2^2], \tag{66}$$

condition (61) holds. The quantity $\frac{\hat{P}(x_t)}{P(x_t)}$ appearing in the analysis does not have to be computed in practice given that the samples satisfy in expectation

$$\mathbb{E}_{\hat{x} \sim \hat{P}(\hat{x})}[\|f_W(\hat{x})\|_2^2] = \mathbb{E}_{x \sim P(x)}[\frac{\hat{P}(x)}{P(x)} \|f_W(x)\|_2^2]. \tag{67}$$

The question of the convergence of this procedure has been considered by a considerable amount of previous work. Many of these are based on the results of [28]. [24] showed that if $C(W)$ is convex, and if its gradient is Lipschitz continuous and bounded, then the optimal convergence rate is $\mathcal{O}(\frac{1}{t})$ and is obtained for $\gamma_t = \mathcal{O}(\frac{1}{t})$. In this case, the speed of decrease of γ_t has the best compromise between the speed of convergence and the control of the variance introduced by using one sample rather than all samples to approximate the gradient.

Bottou [2–4] has shown that this property holds for less constraining conditions on C. In [1], he has studied in particular the type of objective that one wants to minimize in neural networks. These objectives are not convex and may admit several minima. He shows that as long as: (i) $\exists C_{min}, \forall W, C(W) > C_{min}$, (ii) $\mathbb{E}_{(x,y)\sim P(x,y)}[H(x, y, W)] = \nabla_W C(W)$, (iii) $\mathbb{E}_{(x,y)\sim P(x,y)}[H(x, y, W)^T H(x, y, W)] \leq A + BC(W), A, B \geq 0$, (iv) $\sum_t \gamma_t = \infty$ and $\sum_t \gamma_t^2 < \infty$, $C(W_t)$ converges and $\nabla_W C(W_t)$ converges to 0, using SGD. We know that the second condition holds for both of the cases we study. Thus, for our methods, knowing that the two other conditions on C and H hold for $C_0(W) = \mathbb{E}_{(x,y)\sim P(x,y)}[\ell(f_W(x), y)]$ and $H_0(x_t, y_t, W_t) = \nabla_W \ell(f_{W_t}(x), y_t)$ (which are the classical settings for neural networks), we need to prove that they still hold after adding the regularization term. Let $C_{min,0}, A_0$ and B_0 be the parameters that ensures the first and third conditions for C_0 and H_0.

The objective (65) can be rewritten as

$$C(W) = C_0(W) + \lambda \mathbb{E}_{(x,y)\sim P(x,y)}[\frac{\hat{P}(x)}{P(x)} \|f_W(x)\|_2^2] \tag{68}$$

$$H(x_t, y_t, W_t) = H_0(x_t, y_t, W_t) + \lambda \frac{\hat{P}(x)}{P(x)} \nabla_W \|f_{W_t}(x_t)\|_2^2, \tag{69}$$

hence

$$\begin{aligned} H(x_t, y_t, W_t) = {} & H_0(x_t, y_t, W_t) \\ & + 2\lambda \frac{\hat{P}(x)}{P(x)} J_W f_{W_t}(x_t)^T f_{W_t}(x_t). \end{aligned} \tag{70}$$

As $\lambda \geq 0$ and $\mathbb{E}_{(x,y)\sim P(x,y)}[\frac{\hat{P}(x)}{P(x)} \|f_W(x)\|_2^2] \geq 0$, it follows that $C(W) > C_{min,0}$. In addition, we have

$$\mathbb{E}[H(x, y, W)^T H(x, y, W)] = \mathbb{E}[H_0(x, y, W)^T H_0(x, y, W)] \tag{71}$$

$$+ 2\lambda \mathbb{E}[\frac{\hat{P}(x)}{P(x)} H_0(x, y, W)^T J_W f_W(x)^T f_W(x)$$

$$+ \frac{\hat{P}(x)}{P(x)} f_W(x)^T J_W f_W(x) H_0(x, y, W)] \tag{72}$$

$$+ 4\lambda^2 \mathbb{E}[\frac{\hat{P}(x)^2}{P(x)^2} f_W(x)^T J_W f_W(x) J_W f_W(x)^T f_W(x)]. \tag{73}$$

We know that (71) is bounded above by $A_0 + B_0 C_0(W)$. Let us suppose that f_W and $J_W f_W$ are bounded for all W during the gradient descent. As we suppose that f_W has a finite L_2 norm, this condition is verified. Moreover, when $J_W f_W$ is bounded, H_0 is also bounded for many popular loss functions such as the cross-entropy loss. All our experiments are conducted with this loss. Let's suppose also that $\frac{\hat{P}(x)}{P(x)} \leq m, \forall x \in \text{supp}(P)$. Thus, we can find A_1 that bounds (72), and A_2 such that the third term is bounded by $4\lambda^2 A_2 \mathbb{E}[f_W(x)^T f_W(x)]$. Finally, it suffices to find λ such that $4\lambda^2 A_2 - B_0$ is positive, which will finish the proof of the third condition for C and H.

D Brain image segmentation: neural architectures

We detail in 2 the convolutional architecture used in our experiments on the IBSR dataset. The architecture is the one proposed by Shakeri et al. [31]. Dilated or *atrous* convolutions are used to increase the receptive field as proposed in [6]. We augment the network with batch normalization operators between the convolutions and their ReLU activations. During the training this architecture is augmented with a softmax-loss, commonly used in image segmentation networks.

Block	conv kernel	# filters	dilation	pool kernel	pool stride	batch norm.
1	7×7	64	1	3×3	2	yes
2	5×5	128	1	3×3	2	yes
3	3×3	256	2	3×3	1	yes
4	3×3	512	2	3×3	1	yes
5	3×3	512	2	3×3	1	yes
6	4×4	1024	4	no pooling		yes
7	1×1	39	1	no pooling		no

Table 2: Layers used for the brain image segmentation.

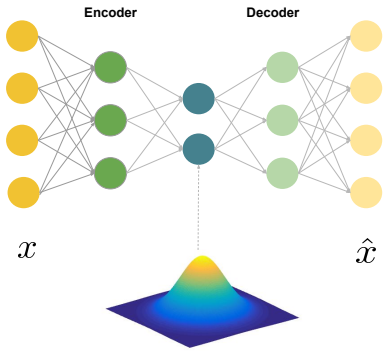


Figure 7: Variational autoencoder architecture.

E Variational autoencoders

To generate samples for DNNs regularization, we choose to train VAEs on the training data. The chosen architecture is composed of two hidden layers for encoding and decoding. Figure 7 displays such an architecture. For each of the datasets, the size of the hidden layers is set empirically to ensure convergence. The training is done with ADAM [18]. The VAE of IBSR data has 512 and 10 nodes in the first and second hidden layer respectively, and is trained during 1000 epochs. In order to have samples that are close to the data distribution but have a slightly broader support, we record the statistics (μ, σ) of the latent variable on the training samples, and generate samples for regularization by decoding a sample of $\mathcal{N}(\mu, 3\sigma)$.

F IBSR segmentation results

Figure 8 shows some examples of segmentation where we see the impact of using weighted function norm on top of batch normalization.

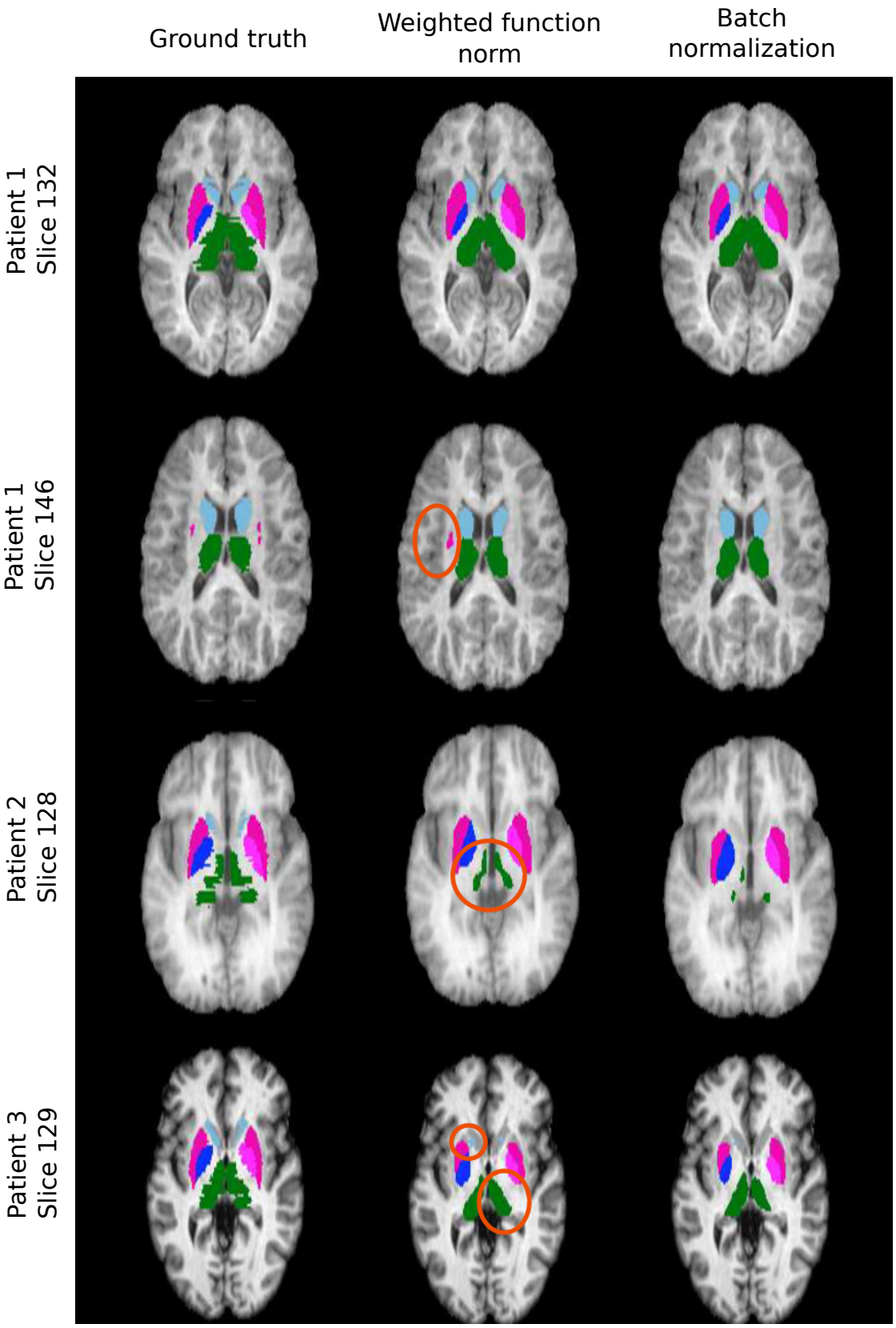


Figure 8: Some examples of segmentation on the ISBR dataset. In some of the slices, both regularization strategies behave similarly (first row). In many slices, the weighted function norm results in an improvement over batch normalization for detection of fine structures (red circles).