Iterative Pseudo-Labeling with Deep Feature Annotation and Confidence-Based Sampling

Bárbara C. Benato*, Alexandru C. Telea†, Alexandre X. Falcão*
*Laboratory of Image Data Science,
Institute of Computing, University of Campinas,
Campinas, Brazil
{barbara.benato, afalcao}@ic.unicamp.br
† Department of Information and Computing Sciences,
Faculty of Science, Utrecht University,
Utrecht, The Netherlands
a.c.telea@uu.nl

***Abstract*—Training deep neural networks is challenging when large and annotated datasets are unavailable. Extensive manual annotation of data samples is time-consuming, expensive, and error-prone, notably when it needs to be done by experts. To address this issue, increased attention has been devoted to techniques that propagate uncertain labels (also called pseudo labels) to large amounts of unsupervised samples and use them for training the model. However, these techniques still need hundreds of supervised samples per class in the training set and a validation set with extra supervised samples to tune the model. We improve a recent iterative pseudo-labeling technique, *Deep Feature Annotation* (DeepFA), by selecting the most confident unsupervised samples to iteratively train a deep neural network. Our confidence-based sampling strategy relies on only *dozens* of annotated training samples per class with no validation set, considerably reducing user effort in data annotation. We first ascertain the best configuration for the baseline – a self-trained deep neural network – and then evaluate our *confidence* DeepFA for different confidence thresholds. Experiments on six datasets show that DeepFA already outperforms the self-trained baseline, but confidence DeepFA can considerably outperform the original DeepFA and the baseline.**

I. Introduction

The success of supervised deep neural networks is evident in many applications. However, the need for large annotated training sets is a well-known problem [1], [2]. Data augmentation and transfer learning aim to address this problem, with semi-supervised learning [3]–[5], its variants pseudo-labeling [6], [7] and meta pseudo-labeling [8], and few-shot learning [9], [10], all receiving increased attention.

Semi-supervised learning methods [3]–[5] propagate labels from a small set of supervised samples to a large set of unsupervised ones by exploiting their distribution in a given latent feature space. Pseudo-labeling approaches [6], [7] (a particular case of self-training) essentially adopt the semi-supervised strategy with the apprentice model assigning uncertain (pseudo) labels to unsupervised samples. Meta pseudo-labeling [8] uses an auxiliary model (teacher) to generate pseudo labels to train the primary model (student). In few-shot learning [9], [10], the model is designed from a handful of supervised samples with or without unlabeled data. Whenever many unsupervised samples are available, semi-supervised learning techniques should be preferred to increase the number of labeled training samples and, consequently, improve feature learning and classification performances.

In semi-supervised learning, pseudo-labeling was first proposed for more effectively fine-tuning a pretrained model [6]. The model can be retrained with a large annotated dataset by assuming that pseudo labels are actual labels. Yet, label propagation errors can negatively affect the performance of classifiers trained from them [11], [12]. To mitigate the problem, the confidence of the apprentice model has been included in the loss function [5], [13]. Yet, pseudo-labeling methods still require a training set with hundreds of supervised samples per class and a validation set with extra supervised samples (at least other 1000 labeled samples) to guarantee reasonable label-propagation accuracy [6]–[8], [14].

In this work, we improve an iterative meta pseudo-labeling strategy, named *Deep Feature Annotation* (*DeepFA*) [15], using the confidence of the auxiliary model used for label propagation to select the unsupervised samples for training the primary model. In *DeepFA*, the auxiliary model is a combination of the t-SNE projection technique [16] applied to a latent feature space (the last convolutional layer) of the primary model, and a semi-supervised optimum-path forest (OPFSemi) classifier [17]. The primary model is trained with a small supervised set, generating the first latent feature set for label propagation by the auxiliary model. In DeepFA, the auxiliary model propagates labels to all unsupervised samples, similar to [18]. This may increase label propagation errors since unsupervised samples, which are far away from supervised ones, may receive incorrect labels. As OPFSemi has no parameters to optimize, DeepFA does not need an extra validation set with supervised samples.

In contrast, we propose to retrain the primary model by using only the most confident unsupervised samples, with confidence given by the label propagation method. We call this variant *confidence DeepFA* (*conf-DeepFA*). Retraining the primary model, latent feature projection, and label propagation repeat for a few iterations to improve the primary model. At each iteration, the primary model is expected to improve its latent feature space reducing the label propagation errors of the auxiliary model. Given that OPFSemi is sensitive to the curse of high dimensionality, we use t-SNE to reduce the feature space to two dimensions. This combination produces a labeling function statistically independent to the labeling function of the primary model, satisfying the main requirement for meta-learning.

We first assess the best training scheme of the primary model and use it as a baseline. We selected VGG-16 [19] pretrained with ImageNet and adapted it for six different datasets with and without iterative pseudo-labeling. We then compared the best scheme against the original *DeepFA* [15] and our *conf-DeepFA* using both fixed and adaptive thresholds to select samples with the most confident labels. To emphasize the potential of *conf-DeepFA*, we used only 1% of supervised samples (i.e., just dozens of samples per class) and no validation set. Our results indicate that our method can significantly outperform the original DeepFA and the baseline methods.

II. Deep Feature Annotation

As a semi-supervised method, *DeepFA* [20] iteratively repeats three steps – deep feature learning, feature space projection, and pseudo-labeling – as described next (see also Fig. 1).

A. Deep Feature Learning

One may conceptually divide a classification deep neural network into (a) layers for feature extraction, (b) fully connected layers for feature space reduction and (c) a decision layer for prediction, being (b-c) an MLP classifier. We are interested in the features of the last convolutional layer of VGG-16 (after max-pooling) that result from (a), where the feature space is still high and sparse.

To minimize user effort for annotation, *DeepFA* uses the ability of pre-trained CNNs to transfer knowledge between scenarios – e.g., from natural to medical images – using few supervised samples and few training epochs. To do this, in our work, we fine-tune VGG-16 with ImageNet's [21] pre-trained weights using the few available supervised images. Finally, the true-labeled images and pseudo-labeled ones by *DeepFA* are used to retrain VGG-16 in the next iterations of our loop.

B. Feature Space Projection

The features of VGG-16's last convolutional layer are projected by t-SNE [16] on a 2D embedded space. Rauber et al. [22] showed that high classification accuracy relates to a good separation of classes in a 2D projection of the samples' feature vectors. More exactly, they showed that, if a 2D projection (in particular, t-SNE) presents good class separation, then a good class separation can also be found in the data space. Benato et al. [11], [15] showed that label propagation (using two semi-supervised classifiers) in a 2D t-SNE projection space leads to better classification results than label propagation in the higher-dimensional latent feature space of an autoencoder. Extending the above, Benato et al. [20] used label propagation in a 2D projected space to create large training sets for deep learning.

C. Pseudo-labeling

OPFSemi was used for pseudo-labeling on a 2D t-SNE projection [11], [15], [20] and in the original high-dimensional feature space [15], [18]. Although OPFSemi's confidence values have been used to improve pseudo-labeling for data annotation, this was not analyzed within the iterative pseudo-labeling loop in [20] to create large training sets for deep learning.

OPFSemi maps supervised and unsupervised samples to nodes of a graph and computes an optimum-path forest rooted at supervised samples. Two types of cost values are calculated to each unsupervised sample. Let $L(u) \in 1, 2, ..., c$ be the (pseudo) label assigned by OPFSemi to an unsupervised sample u. The label $L(u)$ is equal to the class $\lambda(s) \in 1, 2, ..., c$ of the supervised sample s, which has offered the optimum path to u among paths offered from all supervised samples. Let $C(u)$ be the cost of that optimum path from s and $C' > C(u)$ be the second least path cost offered to u by a supervised sample p, whose class $\lambda(p)$ is different to the class $L(u) = \lambda(s)$. Then the confidence value $V(u) = C'/(C(u) + C') \in [0, 1]$ is assigned to the unsupervised sample u. Higher is C' more confident OPFSemi is that $L(u)$ is $\lambda(s)$.

All labels assigned by OPFSemi with a confidence V above a threshold τ are used for VGG-16's training. In this work, we explore different τ values and propose an adaptive approach in which τ is increased along the *conf-DeepFA* iterations. In contrast to [15], we do not ask the user to choose the τ value, since we wish to validate a confidence-based sampling based on OPFSemi during the *DeepFA*'s looping. While this simplifies our approach by releasing the user from the effort of choosing a (good) τ threshold, this also potentially removes insights that users could use to select better τ values. Exploring how our automatic method compares to the user-in-the-loop approach is subject for future work.

III. Experiments

A. Datasets

We choose six datasets to validate our proposed method. The first one is the public MNIST [23] dataset. MNIST has 0 to 9 handwritten digits grayscale images of 28×28 pixels. We use 5000 random samples from the original training dataset.

The subsequent five datasets come from a Parasite image collection [24]. This collection has three main dataset types: (i) *Helminth larvae*, (ii) *Helminth eggs*, and (iii) *Protozoan cysts*. These datasets contain color microscopy images of 200×200 pixels of the most common species of human intestinal parasites in Brazil, responsible for public health problems in most tropical countries [24]. The datasets are challenging since they are unbalanced and contain an impurity class as the majority class, having samples very similar to parasites, making classification hard (see Fig. 2). Table I shows the number, type, and amount of samples per class for each of the three datasets (i-iii) listed above. To these three datasets,

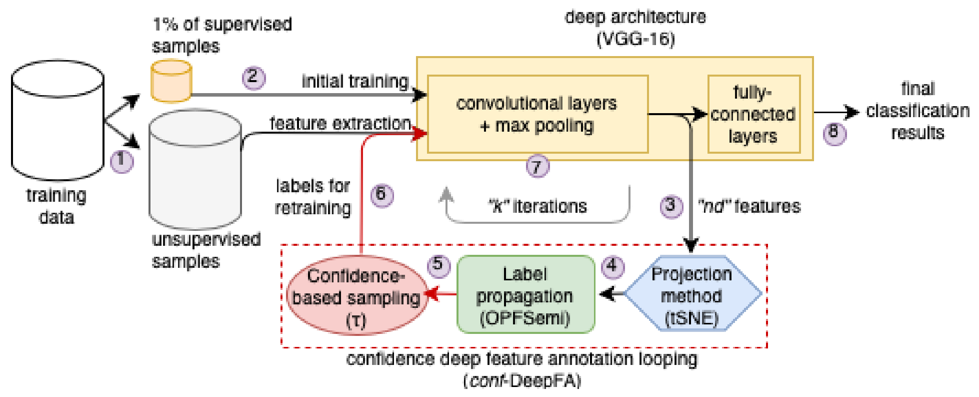
1% of supervised samples
deep architecture (VGG-16)
2
initial training
convolutional layers + max pooling
fully-connected layers
final classification results
8
feature extraction
1
training data
unsupervised samples
labels for retraining
6
7
"k" iterations
3
"nd" features
Confidence-based sampling (τ)
5
Label propagation (OPFSemi)
4
Projection method (tSNE)
confidence deep feature annotation looping (conf-DeepFA)

Fig. 1. Proposed *conf-DeepFA* method. A training dataset is split into (1) supervised and unsupervised sets. We consider only 1% of supervised samples (images) to train a deep neural network (2). Using the network, we extract features from both supervised and unsupervised data (3) and project them in a 2D embedded space (4). In this space, we propagate labels from the supervised samples (5). However, we select only unsupervised samples with the most confident labels, as assigned by the label propagation method (6), to retrain the deep neural network (7). The loop 3-4-5-6-7 repeats for a few iterations. Finally, the classification results (8) are obtained from the fully-connected layers of the trained model. Our contribution extends *DeepFA* [20] by adding the confidence-based sampling (red round shape in the red-dashed box).

we also add the *Helminth eggs* and *Protozoan cysts* datasets without the impurity class, leading to a total of 5 datasets.

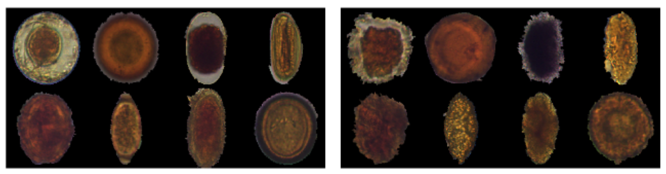

Fig. 2. Examples of species of H.Eggs (left) and similar images of impurities (right).

TABLE I
THREE PARASITES DATASETS: NUMBER OF CLASSES, CLASS NAMES, AND NUMBER OF SAMPLES PER CLASS.

dataset	class	# samples
(i) *Helminth larvae* (2 classes)	*S.stercoralis*	446
	impurities	3068
	total	3,514
(ii) *Helminth eggs* (9 classes)	*H.nana*	348
	H.diminuta	80
	Ancilostomideo	148
	E.vermicularis	122
	A.lumbricoides	337
	T.trichiura	375
	S.mansoni	122
	Taenia	236
	impurities	3,444
	total	5,112
(iii) *Protozoan cysts* (7 classes)	*E.coli*	719
	E.histolytica	78
	E.nana	724
	Giardia	641
	I.butschlii	1,501
	B.hominis	189
	impurities	5,716
	total	9,568

B. Experimental Setup

To reproduce the scenario of few supervised samples, we define a supervised training set S with only 1% of supervised samples of an entire dataset D, assuming the unsupervised U and test T sets with 69% and 30% of samples, respectively ($D = S \cup U \cup T$). A very small S simulates the real-world scenario when one has a large D but manual effort is needed to label samples to create S. We randomly divide each dataset D into S, U, and T in a stratified manner and also generate *three* distinct splits for each experiment for further statistical analysis of our classification results. Table II shows the number of supervised samples in S for each of the six datasets introduced in Sec. III-A.

TABLE II
NUMBER OF SUPERVISED SAMPLES IN S FOR EACH CHOSEN DATASET.

	MNIST	H.eggs (w/o imp)	P. cysts (w/o imp)	H. larvae	H. eggs	P. cysts
S	50	17	38	35	51	95
U	3450	1220	2658	2424	3527	6602

To evaluate our method, we get the probability of VGG-16's last fully-connected layer and compute accuracy and Cohen's κ, since we have unbalanced datasets. $\kappa \in [-1, 1]$ gives the agreement level between two distinct predictions, where $\kappa \leq 0$ means no possibility and $\kappa = 1$ means the full possibility of agreement occurring by chance, respectively. We also compute the number of correct labels assigned in U for each proposed experiment to evaluate the label propagation accuracy.

C. Implementation details

As stated before, our aim is using *conf-DeepFA* without an additional validation set, whose creation, as explained, would ask for more user supervision (i.e., effort in data annotation). For this, we fix all pipeline's parameters without any parameter optimization step. Specifically: OPFSemi has no parameters; for t-SNE, we used the default parameters in *scikit-learn*.

The VGG-16 architecture was implemented in Python using Keras [25]. The original fully-connected layers were replaced by two fully connected layers with 4096 neurons and rectified linear activation, followed by a decision layer with c neurons, where c equals the number of classes for each dataset (see

Tab. I), and softmax activation. The model is trained by error backpropagation for a categorical cross-entropy function and using stochastic gradient descent with a linearly decaying learning rate initialized at 0.1 and momentum of 0.9, respectively. We loaded ImageNet pre-trained weights and used a linear decay of 1×10^{-6} over 15 epochs. The pre-trained weights for convolutional layers were fixed for the feature extraction experiments and unfrozen for fine-tuning, respectively.

D. Proposed experiments

First, we evaluate the impact of VGG-16's training with and without fine-tuning the convolutional layers when loading ImageNet pre-trained weights (Sec. III-C). We also evaluate VGG-16 for label propagation, using pseudolabels produced by VGG-16 to feed its training in the next iteration of the data annotation looping (see Fig. 1). We executed four experiments, as follows; in the next items, *ft* stands for fine-tuning and *fe* stands for feature extraction, respectively:

- *VGG-16$_{ft}$*: VGG-16 with pre-trained weights and convolutional layers *unfrozen* trained on S and tested on T;
- *self-VGG-16$_{ft}$*: VGG-16 with pre-trained weights and convolutional layers *unfrozen* trained on S. Pseudolabels are obtained for *all* samples in U. S and the pseudolabeled U are used to train VGG-16, and the network is tested on T. Each one of the 5 iterations repeats this process;
- *VGG-16$_{fe}$*: VGG-16 with pre-trained weights and convolutional layers *frozen* trained on S and tested on T;
- *self-VGG-16$_{fe}$*: VGG-16 with pre-trained weights and convolutional layers *frozen* trained on S. Pseudolabels are obtained for *all* samples in U. S and U are used to train VGG-16, and the network is tested on T (each one of the 5 iterations repeats this process).

We found out that self-VGG-16$_{fe}$ achieves better results (Sec. III-E), so we defined this training procedure for the subsequent experiments described below. We evaluate the impact of OPFSemi's confidence sample in the *DeepFA* looping by the following experiments:

- *DeepFA*: VGG-16 is trained on S. Deep features for $S \cup U$ from the last convolutional layer are projected in 2D with t-SNE, and used next for OPFSemi pseudo labeling from S to *all* samples in U. OPFSemi's pseudolabels are used to retrain VGG-16, and the network is tested on T (one iteration of *DeepFA* looping out of five);
- *conf-DeepFA$_{\tau=x}$*: VGG-16 is trained on S. Deep features for $S \cup U$ from the last convolutional layer are projected in 2D with t-SNE, and used for OPFSemi pseudo labeling from S to U_τ, for samples with confidence above $\tau = x$. We choose $x = \{0.7, 0.8, 0.9\}$. OPFSemi's pseudolabels are used to retrain VGG-16, and the network is tested on T (one iteration of *conf-DeepFA* looping out of five);
- *conf-DeepFA$_{\tau=\alpha}$*: VGG-16 is trained on S. Deep features for $S \cup U$ from the last convolutional layer are projected in 2D with t-SNE, and used for OPFSemi pseudo labeling from S to U_τ for samples with confidence above τ. τ is increased from 0.8 to 0.96 by 0.4 in each *conf-DeepFA* looping iteration. OPFSemi's pseudolabels are used to retrain VGG-16, and the network is tested on T. The looping has five iterations.

E. Experimental results

Table III shows the results of the experiments in Sec. III-D that investigate the impact of the parameters' fine-tuning in a pre-trained VGG-16 deep architecture when using a small amount of supervised samples to generate pseudo labels. We show mean propagation accuracy, classification accuracy, κ, and standard deviation of *three* different splits for VGG-16 trained only with S and tested on T with feature extraction (*VGG-16$_{fe}$*) and fine-tuning (*VGG-16$_{ft}$*). Also, we present the results for VGG-16 trained with its labeled samples on U over five iterations (*self-VGG-16$_{fe}$* and *self-VGG-16$_{ft}$*). We see first that both experiments – feature extraction and fine-tuning – do not show relevant gain in propagation accuracy or κ along with the iterations. The results are even worse from *VGG-16$_{ft}$* to *self-VGG-16$_{ft}$*. In general, VGG's feature extraction results show an increase of almost 20% in accuracy and κ for most datasets. This shows that the results in [20] can get better when using feature extraction instead of fine-tuning, even though we use fewer training epochs in our work.

A separate interesting question raised in [20] is: How to improve the OPFSemi's pseudo labeling over the iterations? To answer this, we propose to use the confidence-based sampling as stated before. Table IV shows the mean propagation accuracy, classification accuracy, κ, and standard deviation of the three splits for the proposed experiments, for the last of the five executed iterations. For all datasets (except *P.cysts*), we see that selecting the most confident samples by OPFSemi during the *DeepFA* looping *improves* the pseudo-labeling results over the iterations. When using $\tau = 0.8$, MNIST, *H.larvae*, and *P.cysts* obtained the best results. For *H.eggs* without impurities, $\tau = 0.7$ shows the best results. For *P.cysts* without impurities, the results of $\tau = 0.9$ and $\tau = \alpha$ (adaptive) show the best (and similar) results. The proposed confidence-based looping annotation did not improve *P.cysts* with impurities. However, this dataset is also the most challenging one: seven classes, more samples, and almost 60% total samples are impurities. We conclude that confidence-based sampling shows clear added value in nearly all situations. However, we also note that selecting τ may depend on the dataset and its difficulty.

IV. Discussion

A. Does the confidence-based sampling improve the DeepFA looping?

Figure 3 shows the average κ and propagation accuracy for the *DeepFA* looping with fully-pseudo-labeling of all samples (*DeepFA*), our proposed *conf-DeepFA* using OPFSemi's confidence sampling for pseudo labeling, with different ways of selecting the confidence threshold τ, and the best result for the VGG-16 experiments (*self-VGG$_{fe}$*, see Sec. III-E), for all six studied datasets. For datasets yielding higher κ values, we

TABLE III

RESULTS FOR VGG-16 CONSIDERING FEATURE EXTRACTION AND FINE-TUNING. BEST VALUES PER METRIC AND DATASET IN BOLD.

dataset	metric	*VGG-16*$_{ft}$	*self-VGG-16*$_{ft}$	*VGG-16*$_{fe}$	*self-VGG-16*$_{fe}$
MNIST	prop. acc	-	0.447238 ± 0.146	-	**0.586000 ± 0.007**
	acc	**0.629555 ± 0.037**	0.441334 ± 0.149	0.614444 ± 0.015	0.592222 ± 0.020
	kappa	**0.588195 ± 0.041**	0.378648 ± 0.166	0.571176 ± 0.017	0.546162 ± 0.023
H.eggs (w/o imp)	prop. acc	-	**0.758825 ± 0.088**	-	0.744004 ± 0.114
	acc	**0.790961 ± 0.050**	0.779033 ± 0.095	0.738858 ± 0.054	0.774011 ± 0.131
	kappa	**0.752807 ± 0.060**	0.735591 ± 0.113	0.693278 ± 0.060	0.734030 ± 0.153
P.cysts (w/o imp)	prop. acc	-	0.399481 ± 0.010	-	**0.648739 ± 0.111**
	acc	0.561130 ± 0.093	0.400519 ± 0.011	**0.736159 ± 0.027**	0.650230 ± 0.101
	kappa	0.324051 ± 0.175	0.020734 ± 0.021	**0.626632 ± 0.039**	0.483706 ± 0.170
H.larvae	prop. acc	-	0.897384 ± 0.031	-	**0.912837 ± 0.038**
	acc	0.874566 ± 0.001	0.886572 ± 0.017	0.893523 ± 0.017	**0.908689 ± 0.040**
	kappa	0.021406 ± 0.019	0.174158 ± 0.208	0.256836 ± 0.203	**0.385892 ± 0.402**
H.eggs	prop. acc	-	0.773803 ± 0.034	-	**0.847308 ± 0.018**
	acc	**0.858323 ± 0.013**	0.775750 ± 0.034	0.848327 ± 0.017	**0.850934 ± 0.014**
	kappa	**0.734333 ± 0.019**	0.519971 ± 0.114	0.713649 ± 0.030	0.714227 ± 0.038
P.cysts	prop. acc	-	0.730327 ± 0.022	-	**0.817978 ± 0.004**
	acc	0.758853 ± 0.077	0.734239 ± 0.028	0.818182 ± 0.004	**0.824800 ± 0.011**
	kappa	0.542967 ± 0.218	0.492070 ± 0.107	0.697633 ± 0.009	**0.705397 ± 0.022**

TABLE IV

RESULTS FROM THE LAST ITERATION FOR PROPOSED EXPERIMENTS WITH FULLY LABEL PROPAGATION (*DeepFA*), AND CONFIDENCE-BASED LABEL PROPAGATION (*conf-DeepFA*) WITH CONFIDENCE HIGHER THAN $\tau = 0.7$, CONFIDENCE HIGHER THAN $\tau = 0.8$, CONFIDENCE HIGHER THAN $\tau = 0.9$, AND ADAPTATIVE CONFIDENCE (FROM 0.80 TO 0.96 OVER 5 ITERATIONS). BEST VALUES PER DATASET IN BOLD.

dataset	metric	*DeepFA*	*conf-DeepFA*$_{\tau=0.7}$	*conf-DeepFA* $_{\tau=0.8}$	*conf-DeepFA* $_{\tau=0.9}$	*conf-DeepFA* $_{\tau=\alpha}$
MNIST	prop. acc	0.790000 ± 0.047	0.782286 ± 0.029	**0.821714 ± 0.018**	0.750000 ± 0.028	0.795429 ± 0.007
	acc	0.797778 ± 0.049	0.788000 ± 0.030	**0.822666 ± 0.022**	0.740222 ± 0.032	0.651778 ± 0.062
	kappa	0.775103 ± 0.054	0.764348 ± 0.034	**0.802863 ± 0.024**	0.710961 ± 0.036	0.612766 ± 0.069
H.eggs (w/o imp)	prop. acc	**0.983293 ± 0.004**	0.983832 ± 0.002	0.974401 ± 0.020	0.981945 ± 0.003	0.983832 ± 0.004
	acc	0.790961 ± 0.050	**0.973007 ± 0.006**	0.971123 ± 0.013	0.938481 ± 0.056	0.806654 ± 0.126
	kappa	0.752807 ± 0.060	**0.968042 ± 0.007**	0.965848 ± 0.015	0.927708 ± 0.066	0.771216 ± 0.148
P.cysts (w/o imp)	prop. acc	0.800569 ± 0.035	0.805143 ± 0.049	0.793274 ± 0.069	0.824060 ± 0.019	**0.828141 ± 0.012**
	acc	0.819493 ± 0.041	0.826413 ± 0.039	0.814590 ± 0.060	**0.842561 ± 0.004**	0.824394 ± 0.033
	kappa	0.756949 ± 0.054	0.764035 ± 0.052	0.747127 ± 0.086	**0.785441 ± 0.006**	0.762919 ± 0.041
H.larvae	prop. acc	0.954182 ± 0.008	0.964213 ± 0.017	**0.964349 ± 0.012**	0.941846 ± 0.039	0.951471 ± 0.014
	acc	0.955450 ± 0.002	0.959558 ± 0.015	**0.965561 ± 0.004**	0.958926 ± 0.014	0.943128 ± 0.010
	kappa	0.789743 ± 0.010	0.800052 ± 0.099	**0.837948 ± 0.029**	0.804689 ± 0.082	0.705475 ± 0.069
H.eggs	prop. acc	0.936743 ± 0.011	0.936091 ± 0.005	**0.937209 ± 0.008**	0.931806 ± 0.007	0.930967 ± 0.006
	acc	**0.942634 ± 0.016**	0.943938 ± 0.003	0.942634 ± 0.009	0.908518 ± 0.022	0.853107 ± 0.025
	kappa	0.899307 ± 0.027	**0.901604 ± 0.006**	0.898922 ± 0.015	0.831488 ± 0.043	0.719695 ± 0.054
P.cysts	prop. acc	0.732716 ± 0.056	0.769748 ± 0.026	**0.780300 ± 0.018**	0.748843 ± 0.048	0.744811 ± 0.068
	acc	0.740973 ± 0.056	0.792755 ± 0.027	**0.816905 ± 0.027**	0.818066 ± 0.022	0.731104 ± 0.082
	kappa	0.580626 ± 0.092	0.652254 ± 0.051	**0.699603 ± 0.054**	0.689325 ± 0.039	0.450283 ± 0.243

note that *DeepFA* obtained similar results compared with our proposed *conf-DeepFA* modifications. However, we see a gain of almost 5% in κ and propagation accuracy for the most challenging datasets. For *P.cysts* with impurities, the gain is actually higher than 10% in κ and 17% in propagation accuracy – for which *DeepFA* obtained worse results than VGG-16. In short, our proposal of using *DeepFA* with OPFSemi's confidence sampling (*conf-DeepFA*) entries in Fig. 3) obtained the best results for most tested datasets.

B. Does the confidence-based sampling improve DeepFA along the iterations?

Figure 4 shows κ and propagation accuracy for one split of MNIST along five iterations of the proposed experiments. First, we see that all compared approaches yielded an increase from the first to the second iteration, except *self-VGG-16*$_{fe}$. Also, we see that both κ and the propagation accuracy slightly decrease after the third iteration. This may suggest that the proposed method saturates, mainly by the higher decrease in κ despite of propagation accuracy. The learned pseudo-labels and the original images can be used as input for a better (known) deep architecture. Figure 5 shows the plot for train and validation loss and accuracy considering 20% (from the S set) as validation set during one split of MNIST training. The initial learning curve and the learning curves for each iteration are also shown. The learning curves show that the labeled samples can improve the network convergence along the iterations. A different deep network can be tested at the final stage. Also, some unsupervised quality measure can be proposed to define the best feature space found at certain iterations and, consequently, the best iteration of the method.

C. Choosing OPFSemi's confidence threshold

The proposed adaptive selection of the confidence threshold (*conf-DeepFA*$_{\tau=\alpha}$) seems to be promising only for one of the tested datasets. It shows a higher decreasing in κ, when compared with the experiments without changing the confidence threshold τ along the iterations. As outlined in Sec. III-E, choosing OPFSemi's confidence value may depend on the dataset, its difficulty, number of samples, number of classes, and class imbalance. This can also be seen in Fig. 3 where it is not possible to define a *single* confidence threshold τ for *all* chosen datasets. Although this fact has been already noted in [15], it was not considered within a looping of data annotation as we proposed in this work. Rather, in [15], the authors proposed user interaction to define the best confidence value based on the user analysis of the 2D space projection guided by the data distribution and OPFSemi's confidence values (mapped to colors). We intend to follow the same strategy to find the best confidence value for *conf-DeepFA* looping.

D. Limitations

As discussed before, we intend to explore other deep learning architectures to understand why and when the proposed *conf-DeepFA* looping stagnates during network training. Also, we validated our method for only six datasets, one semi-supervised classifier, and one projection method. More experiments involving additional datasets, classifiers, and 2D projection techniques are needed to generalize our findings.

V. Conclusion

We proposed an approach for increasing the quality of image classification and extracted feature spaces when using very few supervised samples during the training process. We evaluate the feature space generated by VGG-16 by feature extraction and fine-tuning strategies when using the small number of supervised images available. We use the best resulting feature space to let the OPFSemi propagation technique label unsupervised samples on a 2D t-SNE projection of the feature space in an iterative fashion. To improve label propagation accuracies and classification results, we include a confidence sampling strategy to OPFSemi's pseudo labeling to define the most confident samples for training VGG-16.

Our results show that the VGG-16 without fine-tuning, i.e., only being used for feature extraction, can improve accuracy and κ with few supervised samples. Also, when considering OPFSemi with the confidence sampling strategy, our results show an improvement in propagation accuracy and κ for

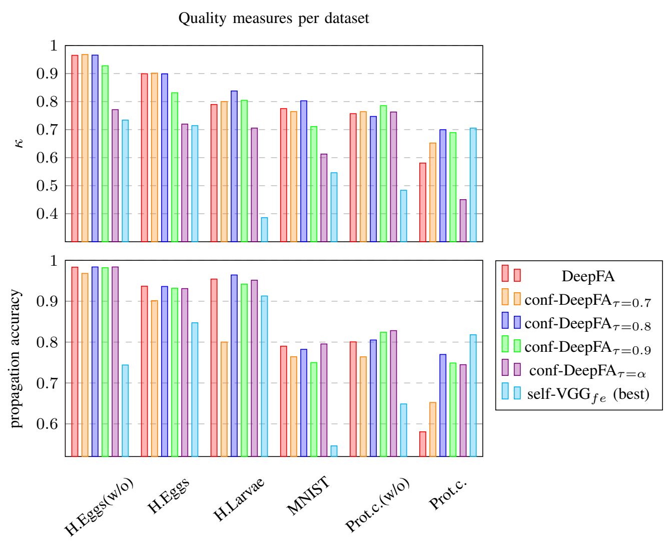

Fig. 3. Results of κ (top) and propagation accuracies (bottom) for the studied datasets, considering self-VGG-16$_{fe}$ (best result) and DeepFA experiments. Our confidence-based DeepFA variations proposed in this paper are marked as *DeepFA*$_\tau$. The datasets are ordered by higher κ values in x axis (from left to right).

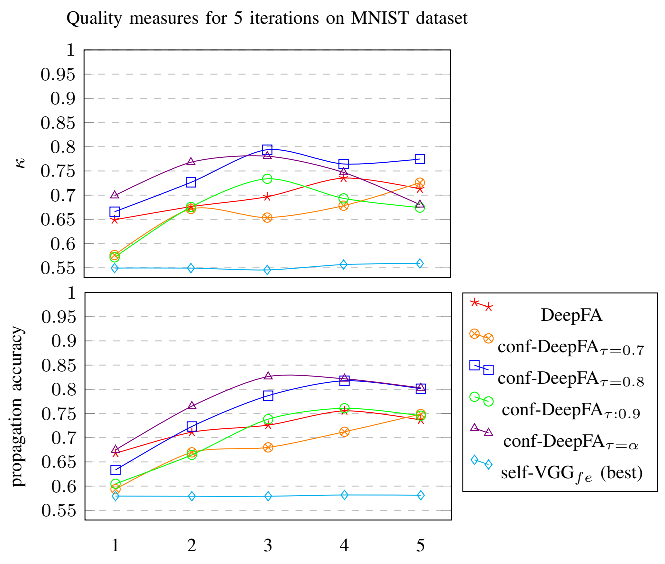

Fig. 4. Results of κ (top) and propagation accuracies (bottom) for the MNIST dataset in one split over 5 iterations, considering self-VGG-16$_{fe}$ (best result), *DeepFA*, and *conf-DeepFA* experiments.

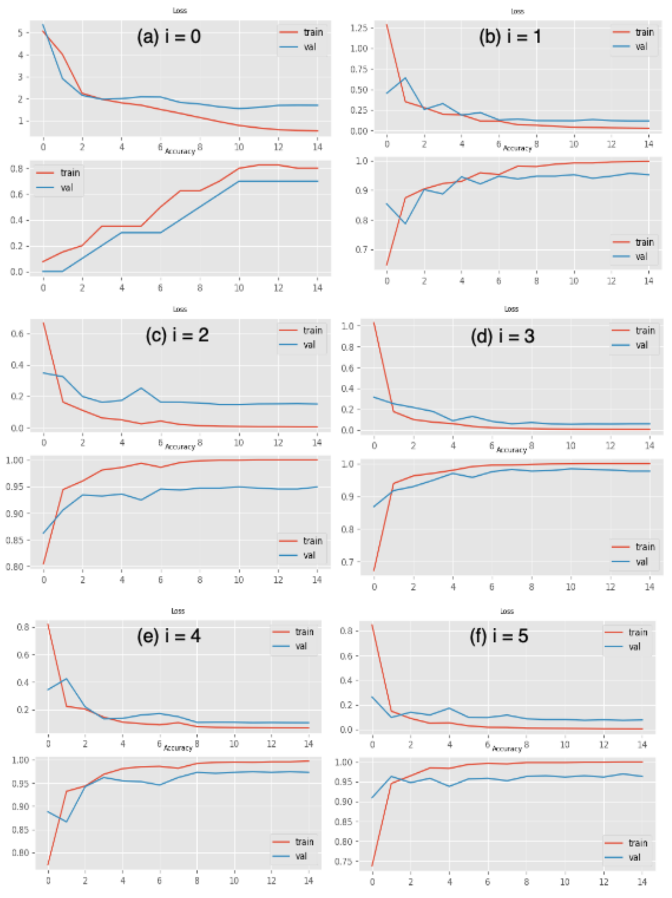

Fig. 5. Plots for loss and accuracy for one split of MNIST dataset. The (a) initial learning curves and the for each iteration (b, c, d, e, f) is presented.

most of the evaluated datasets. The small gain for some chosen confidence thresholds τ and some datasets shows that this choice may depend on the dataset. To solve this dataset dependency, we plan next to include user knowledge to provide a semi-automatic pseudo-labeling along the lines in [15] but considering the proposed looping of the deep feature annotation method. Additionally, we aim to explore more datasets, as well as compare our proposed method with recent semi-supervised strategies for creating pseudolabels.

Acknowledgments

The authors acknowledge FAPESP grants $\#2014/12236-1$, $\#2019/10705-8$, CAPES grants with Finance Code 001, and CNPq grants $\#303808/2018-7$.

References

[1] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. ECCV*, 2014, pp. 740–755.

[2] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era." in *Proc. ICCV*, 2017, pp. 843–852.

[3] Z. Li, B. Ko, and H.-J. Choi, "Naive semi-supervised deep learning using pseudo-label," *P2P Netw Appl*, vol. 12, pp. 1–11, 2018.

[4] H. Wu and S. Prasad, "Semi-supervised deep learning using pseudo labels for hyperspectral image classification," *IEEE TIP*, vol. 27, no. 3, pp. 1259–1270, 2018.

[5] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Label propagation for deep semi-supervised learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5070–5079.

[6] D. H. Lee, "Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. ICML-WREPL*, 2013.

[7] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[8] H. Pham, Z. Dai, Q. Xie, and Q. V. Le, "Meta pseudo labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 11 557–11 568.

[9] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[10] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] B. C. Benato, A. C. Telea, and A. X. Falcão, "Semi-supervised learning with interactive label propagation guided by feature space projections," in *Proc. SIBGRAPI*, 2018, pp. 392–399.

[12] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Pseudo-labeling and confirmation bias in deep semi-supervised learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[13] W. Shi, Y. Gong, C. Ding, Z. M. Tao, and N. Zheng, "Transductive semi-supervised deep learning using min-max features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 299–315.

[14] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1979–1993, 2018.

[15] B. C. Benato, J. F. Gomes, A. C. Telea, and A. X. Falcão, "Semi-automatic data annotation guided by feature space projection," *Pattern Recognition*, vol. 109, p. 107612, 2021.

[16] L. van der Maaten, "Accelerating t-SNE using tree-based algorithms," *JMLR*, vol. 15, no. 1, pp. 3221–3245, 2014.

[17] W. Amorim, A. Falcão, J. Papa, and M. Carvalho, "Improving semi-supervised learning through optimum connectivity," *Patt Recogn*, vol. 60, pp. 72–85, 2016.

[18] W. Amorim, G. Rosa, Rogério, J. Castanho, F. Dotto, O. Rodrigues, A. Marana, and J. Papa, "Semi-supervised learning with connectivity-driven convolutional neural networks," *Pattern Recognition Letters*, vol. 128, pp. 16 – 22, 2019.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: arxiv.org/abs/1409.1556

[20] B. C. Benato, J. F. Gomes, A. C. Telea, and A. X. Falcão, "Semi-supervised deep learning based on label propagation in a 2d embedded space," *arXiv preprint arXiv:2008.00558*, 2020.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, p. 211–252, Dec. 2015.

[22] P. Rauber, A. Falcão, and A. Telea, "Projections as visual aids for classification system design," *Information Visualization*, 2017.

[23] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: yann.lecun.com/exdb/mnist

[24] C. Suzuki, J. Gomes, A. Falcão, S. Shimizu, and J.Papa, "Automated diagnosis of human intestinal parasites using optical microscopy images," in *Proc. Symp. Biomedical Imaging*, April 2013, pp. 460–463.

[25] F. Chollet *et al.*, "Keras," https://keras.io, 2015.