Improving Dataset Volumes and Model Accuracy with Semi-Supervised Iterative Self-Learning

Robert Dupre, Jiri Fajtl, Vasileios Argyriou, *Member, IEEE,* and Paolo Remagnino, *Senior Member, IEEE*

***Abstract*—Within this work a novel semi-supervised learning technique is introduced based on a simple iterative learning cycle together with learned thresholding techniques and an ensemble decision support system. State-of-the-art model performance and increased training data volume are demonstrated, through the use of unlabelled data when training deeply learned classification models. The methods presented work independently from the model architectures or loss functions, making this approach applicable to a wide range of machine learning and classification tasks. Evaluation of the proposed approach is performed on commonly used datasets when evaluating semi-supervised learning techniques as well as a number of more challenging image classification datasets (CIFAR-100 and a 200 class subset of ImageNet).**

***Index Terms*—semi-supervised, image classification, deep learning, machine learning.**

I. Introduction

SEMI-SUPERVISED learning has become one of the most prevalent topics within image processing and computer vision research in recent years. With the ever increasing availability of high powered GPU hardware and the success of deep learning on applications such as computer vision [1], [2], speech recognition [3], [4], and natural language processing [5], [6], the need for large scale datasets to support these methods becomes a higher priority, as well as a bottleneck to performance improvement. Issues of cost and time still remain prohibitive to the creation of these large datasets, particularly in specific domains with high demand on labelling such as memorability estimation [7], [8], video summarization [9] and others. ImageNet [10] was one of the most successful and pioneering large scale datasets and still stands as a benchmark when it comes to dataset volume, with no cleanly labelled dataset out-doing the size of ImageNet by any meaningful amount. This critical mass of data size is the biggest target of semi-supervised learning techniques, developing ways in which unlabelled or noisy data can be utilised without the need for expensive and time consuming processes which can 'clean' the data.

In the standard semi-supervised learning framework, using a number of training samples with strong annotations, the goal is to infer the appropriate labels for the rest of the data.

This work was supported in part by the European Union's Horizon 2020 Programme for Research and Innovation Actions within IoT (2016): Large Scale Pilots: Wearables for smart ecosystem. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

R.Dupre, J.Fajtl, V.Argyriou and P. Remagnino are members of the Robot Vision Team (RoViT) at Kingston University.

Manuscript received June 25, 2018.

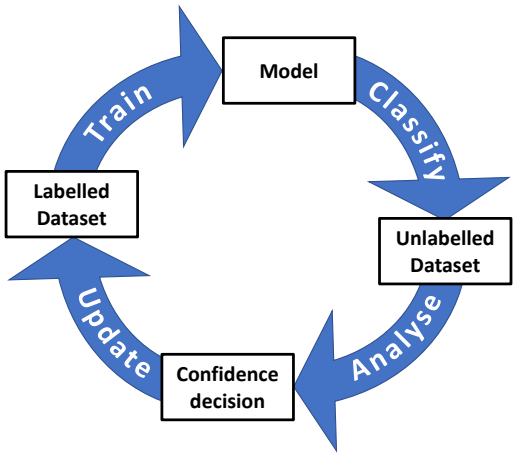


Fig. 1. An overview of the Iterative Learning (IL) cycle, in which a model is trained on labelled data and used to classify new unlabelled samples. The class decisions for the new samples are then evaluated based on the confidence and then added to the labelled training dataset and the process repeated.

Semi-supervised processes have been applied successfully in many areas such as image classification and segmentation [11], natural language processing [12] and artificial intelligence [13].

These processes highlight two distinct areas where semi-supervised learning could be useful: First, is the obvious addition of training samples to improve the accuracy of the trained models; these techniques can be utilised to make use of noisy data in a more superficial way, Joulin et al. [14] propose training the networks with very large (100M images), but weakly labelled data to develop robust and diverse visual features, and evaluating their generalisation on a number of transfer learning tasks. Veit et al. [15] extend this concept by adding a fine tuning stage using clean data to further improve accuracy. The second of these areas is the development of strategies to automatically label this data, thus increasing the size of these datasets. This often draws criticism due to the introduction of incorrectly labelled samples which might skew or even hinder learning. However this idea is being challenged, with the suggestion that many learning frameworks are resistant to the presence of noise. Natarajan et al. [16], look at the presence of noisy labels in binary classification data and how a model and its loss function can be manipulated to become more robust to these issues. In their experiments even in data where over 30 percent of binary labels were inverted, good accuracy was still obtained within their tasks.

In much of the current work in this area the training methods, architectures and loss functions utilised have been

developed specifically for that semi-supervised application making them difficult to transfer amongst tasks and almost impossible to transfer to a new area. The following work looks to approach semi-supervised learning from a different direction; rather than re-imagine the model architectures, which are being trained, or the loss functions that train them, the goal of this work is to iteratively reclassify the dataset such that the model being trained is only ever exposed to what it considers fully labelled data. The focus is on incremental, unsupervised improvements to a model and the data, increasing a training dataset volume for use elsewhere as well as the classification performance. As such the proposed methodology has a number of clear benefits over existing techniques:

- A simple and easily implemented semi-supervised learning framework.
- Independent from model architecture or loss functions.
- Novel learned thresholding techniques and metrics to supervise the dataset growth
- Applicable to a wide range of classification tasks.

This paper will proceed as follows: a overview of related work is given in Section II, followed by the Methodology where an outline is given of how the iterative learning process is performed. Results follow, demonstrating the performance of this technique on both a typical semi-supervised benchmark and some more challenging datasets. Finally conclusions are drawn and future work discussed.

II. Related Work

Typically, semi-supervised deep learning is tackled through the use of novel model architectures, regularization methods or loss functions combining outputs from known labels with unknown to provide more accurate outputs. For example, Laine [17] utilises an architecture based on ensemble predictions, acquired during training of a network at different epochs or under different regularization and input conditions. These ensemble predictions often provide a more accurate label, and therefore can be utilised to increase the accuracy of the network being trained. As such, two approaches are suggested: the $\prod$-model and temporal ensembling. The $\prod$-model works by evaluating the network input twice, due to the use of dropout regularization and augmentations, two evaluations of the same input (under the same network weights) will produce different results. Using these two evaluations, a loss function is created that minimizes the error on the labelled data but also the difference between the two evaluations of the input. Temporal ensembling simplifies this process by using prior evaluations of an input to create an ensemble, rather than evaluating an input twice. This yields faster training times and through the use of multiple prior evaluations the ensemble prediction can be considered less noisy.

Miyato et al. [18], [19] consider a novel regularization method to semi-supervised learning. Through the use of Virtual Adversarial Training (VAT) in which the training goals are extended from a model's likelihood to include it's local distributional smoothness (LDS) on the posterior distribution for a given sample. The goal of VAT is to find the maximum adversarial perturbation of a real data point based on the model's output and the original point. However, efficiently computing the optimal adversarial perturbation is a complex task requiring multiple forward and backpropagations.

Luo et al. [20] propose an extension to the above perturbation-based methods, in the form of the Smooth Neighbour on Teacher Graphs (SNTG). By constructing a low-dimensional graph for 'similar' neighbouring points based on the predictions of a teacher model, predictions of the student model can be refined for not only the data point under consideration, but also those neighbouring points as defined by the graph.

Rasmus et al. [21] develop the Γ-model which evaluates an input image with an additional generated noisy sample. The loss function focuses on a consistency cost between the two resultant outputs, the intuition being that with the generated targets a manifold representation of a class will be modelled thus improving generalisation. However, the generated examples will not always represent the original target well and thus the produced manifolds may have inherent error. Tarvainen et al. [22] take the concept of temporal ensembling and extend the concept from averaging label predictions to averaging model weights. This has a number of advantages; most notably that the process has the effect of updating and improving all the layer outputs, as well as creating a faster feedback loop between the teacher and student models which allows for better results from few samples.

Sajjadi et al. [23] take a similar approach, but rather than using a generative approach to create new samples, existing labelled samples are run through a training model multiple times with extensive transformations applied and loss minimised across all the resultant outputs.

One of the most difficult problems in semi-supervised learning is the estimation of the posterior probability of unknown labels. Cicek et al. [24] use learning speed during the backpropagation to estimate the labelling error. This idea is based on an observation that the convergence of the Stochastic Gradient Decent (SGD) optimization is directly related to the number of correct labels in the training dataset. Training loss decreases rapidly with correctly labelled training samples while becoming very slow with corrupted labels. Similarly to our approach Cicek at al. [24] use an update loop which updates the distribution of unknown labels, and an inner loop which simulates the optimization procedure over a small number of epochs.

Another interesting approach suggested by Haeusser et al. [25] in which they propose a "learning by association" technique. This postulates that good embedding of labelled and unlabelled samples have a high similarity if they belong to the same class. The association is established by first "walking" from the labelled samples to the nearest unlabelled ones and back. If the class of the starting and ending samples is the same, this class is associated with the unlabelled sample. These associations are formulated as a fully differentiable function and included in the training cost.

Szummer and Jaakkola [26] developed a label propagation method based on a Markov random walk that uses a limited number of labelled samples to classify a much larger set of unlabelled data points. This classification model assumes

that each sample has a label or a distribution $P(y|i)$ over the class label. Any data point, labelled or unlabelled, is interpreted as a sample at the t step of a Markov random walk. Classes for the unknown labels are chosen such that they maximize the posterior probability of that label given the starting and new, already labelled samples. Maximum likelihood with Expectation Maximization (EM) or maximum margin techniques are used to estimate the model parameters. In general this method is based on the assumption that the nearest neighbours on a low dimensional structure are likely to have the same label.

The early success of deep learning motivated Weston et al. [27] to explore it for semi-supervised learning in a conceptually simple method. Weston et al. selects an unsupervised or a semi-supervised learning algorithm, such as the Label Propagation by Zhu and Ghahramani [28] and LapSVM by Belkin et al. [29], and add it as a regularizer to an existing deep model architecture at a single or multiple layers. This model is then jointly trained with labelled and unlabelled samples. Authors showed that training a semi-supervised embedding with a supervised deep multi-layer architecture on any (or all) layers of the network can bring real benefits in complex tasks.

Very recently French et al. [30] presented a technique derived from the mean teacher variant [22] which achieves state of the art results in a variety of benchmarks. The main contribution is in the area of domain adaptation demonstrating excellent results in the MNIST to SVHN knowledge transfer. The improvements mainly revolve around a class balancing and confidence thresholding, where the Gaussian based scaling of the unsupervised loss was replaced with an experimentally derived confidence threshold.

III. Methodology

The core assumption in this work is that generalization error always decreases with more training samples as shown by [31] and recently [32]. This can be formally expressed as:

$$GE \leq \mathcal{O}\left(\frac{C_f}{N}\right) + \mathcal{O}\left(\frac{Nd}{L}\log(L)\right) \quad (1)$$

Where GE is generalization error, N the number of neurons in the network, C_f is smoothness of the approximated function, L the number of training samples and d the input dimension. To address this problem the Iterative Learning (IL) approach is presented. An overview of IL is given in Figure 1, where the iterative nature of the method is given by the train, classify, analyse and finally update cycle. This process looks to increase the size of a training dataset rather than make use of unknown labelled samples in the training phase. The algorithm is designed to be general and as such the complexity of the various aspects is dictated by application. Figure 2 outlines the benefits of the methodology in both the increase in the dataset (top) and the increase in validation accuracy (bottom).

A. Iterative Learning (IL)

The pseudo code for Iterative Learning (IL) is detailed in Algorithm 1 and outlines the nature of the process and the two main operations: Firstly, a model (θ) is trained on a

Input : $\mathbb{D}_l$, $\mathbb{D}_u$, θ
Output: $\mathbb{D}_{l\_updated}$, $\mathbb{D}_{u\_updated}$

while $num\_candidates > 0$ **do**
  for t in $[1, num\_epochs]$ **do**
    train_model(θ, $\mathbb{D}_l$)
  end
  validate($\mathbb{D}_v$)
  build_average_class_distributions($\mathbb{D}_l$)
  for x_m^{ul} in $\mathbb{D}_u$ **do**
    y_m = evaluate(θ, x_m^{ul})
    if $\sum(cf_a, cf_c, cf_d) > thresh$ **then**
      $num\_candidates++$
      $\mathbb{D}_l \mathrel{+}= x_m^{ul}$
      $\mathbb{D}_u \mathrel{-}= x_m^{ul}$
  end
end

Algorithm 1: Iterative Learning

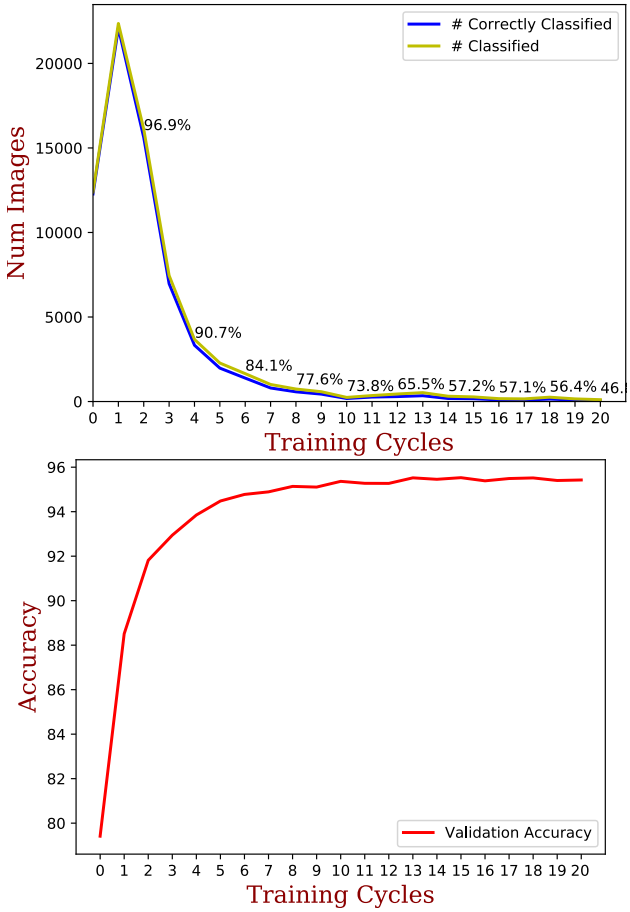


Fig. 2. Example iterative learning experiment run for 20 cycles using the ResNet-18 Architecture on the SVHN Dataset. (top) Number of samples added to the dataset and the accuracy of those additions per iteration (percentage labels added intermittently highlighting accuracy of those added samples). (bottom) The validation accuracy per iteration.

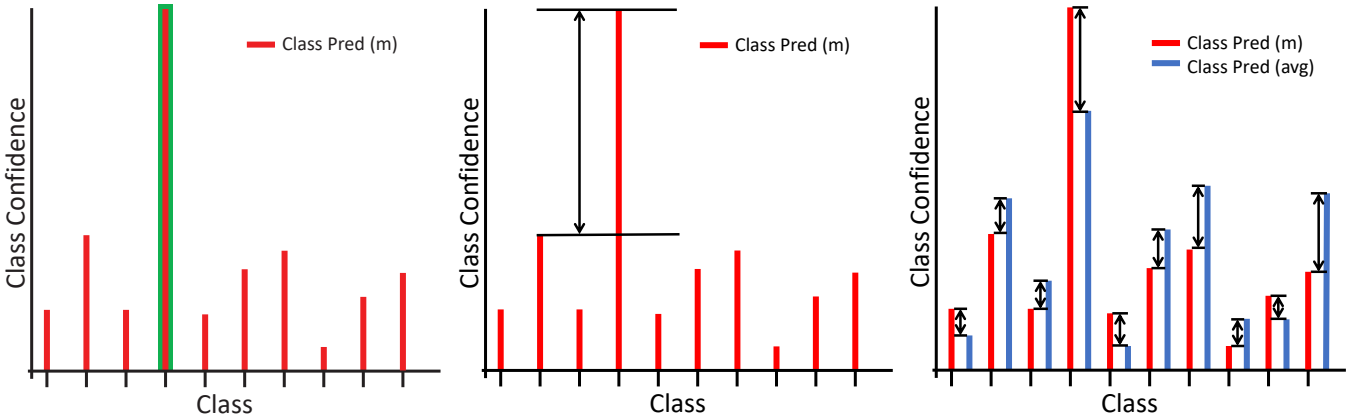


Fig. 3. Confidence metrics, illustrated using a synthetic class distribution for sample x_m^u in $\mathbb{D}_u$, evaluated using model θ: (left) The highest activation of all the classes. (middle) The difference between the highest and second highest activations of all the classes. (right) The sum of Euclidean difference between the activation distribution and the mean distribution for that class from train data.

cleanly labelled dataset, $\mathbb{D}_l$, and validated on the $\mathbb{D}_v$ dataset. The training of the model is performed in a relevant way to the application and task, neither the architecture nor the loss functions are changed in any way. Secondly, the unlabelled samples are classified and the process of updating the training set is run. Unlabelled data is evaluated using the now trained model θ and classifications features and labels are recorded, based on this data candidates are selected to be incorporated into the training set. These candidates are selected according to a voting system based on a number of confidence metrics (outlined further below). The training set is then updated to include these confidently labelled samples and the process repeats with the model being retrained from scratch on the newly updated training set.

Let $x \in \mathbf{R}^d$ represent an input variable in d dimensions and $y \in \mathbf{L}^C$ represent the label associated with that sample, where C represents the number of possible class labels. In this work x_i represents a cropped image and y_i^c the label from C-classes. From the pool of cleanly labelled and unlabelled data, three datasets are constructed: Labelled ($\mathbb{D}_l = x_n^l, y_n^l | n = 1, \ldots, N^l$), derived from a portion of the cleanly labelled data. Unlabelled ($\mathbb{D}_u = x_m^u, y_m^u | m = 1, \ldots, M^u$), indexed from only unlabelled data and finally validation ($\mathbb{D}_v = x_o^v, y_o^v | o = 1, \ldots, O^v$), derived from the remaining subset of the cleanly labelled data.

The primary issue when adding newly labelled samples to the training dataset is ensuring the model is confident that the additions are labelled correctly. To aid this process of labelling unknown samples, a number of metrics have been devised based solely on the posterior probabilities produced from model θ. Importantly, there are no additional clustering or preprocessing steps applied to the unlabelled data of any kind, the only assumption made within this work is that the data is of a similar quality, context and application as that within the cleanly labelled.

These confidence metrics cover three distinct areas computed from the output of the model after evaluation of the unlabelled data $\mathbb{D}_u$: Firstly (Figure 3 left), the common single highest class activation obtained from the posterior distribution c_a (higher is better). Formally, consider an unlabelled sample $\mathbf{x}^{(\mathbf{j})} \in \mathbb{D}_u$

$$y_1, y_2 = \underset{y}{\operatorname{argmax}}\, P(\mathbf{y}|\mathbf{x}^{(\mathbf{j})}; \theta) \tag{2}$$

Where y_1 and y_2 are labels corresponding to the first and second highest posterior probabilities. The c_a is then

$$c_a = P(y_1|\mathbf{x}^{(\mathbf{j})}; \theta) \tag{3}$$

Second (Figure 3 middle), the difference between the highest and second highest activation c_b (larger difference is better) is computed according to Eq. 4.

$$c_b = P(y_1|\mathbf{x}^{(\mathbf{j})}; \theta) - P(y_2|\mathbf{x}^{(\mathbf{j})}; \theta) \tag{4}$$

Lastly (Figure 3 right), c_c is calculated as the Euclidean distance between the posterior distribution for the unlabelled sample $\mathbf{x}^{(j)}$ and the average distribution $p_t(y_1)$ for the predicted class y_1 (lower score is better). $p_t(y_1)$ is computed over all training samples of class y_1. These average distributions per class are computed at the end of each model training iteration and are recorded for use in these confidence computations.

$$p_t(c) = \frac{1}{N} \sum_{i}^{N} P(\mathbf{y}|\mathbf{x}_c^{(i)}; \theta) \tag{5}$$

Where $\mathbf{y}$ are labels of all classes, $\mathbf{x}_c^{(i)}$ is a sample from the current training, labelled dataset $\mathbb{D}_l$ belonging to class c and is N number of training samples. c_c is then calculated as:

$$c_c = \|P(\mathbf{y}|\mathbf{x}^{(j)}; \theta) - p_t(y_1)\| \tag{6}$$

For each of these three metrics a value is returned, in the cases of c_a and c_b the value returned by the model should be high and for c_c the distance between the two posterior probability distributions should be low, however the c_c scores are inverted so as to have a uniform, higher is better policy. The weighted sum of these metrics scores is then used to provide a final confidence score for a specific unlabelled sample $\mathbf{x}^{(j)}$. As some metrics are more informative that others

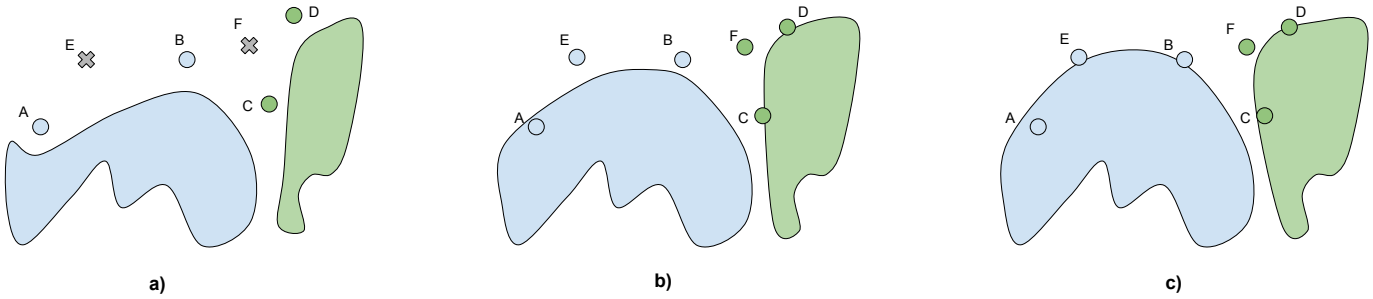


Fig. 4. a) new, unlabelled samples A,B,C and D are classified as belonging to the blue or green class due to their proximity to the respective manifolds. Samples E and F cannot be identified yet given the current model's knowledge. b) after retraining the model with new samples A,B,C and D the model can now recognize samples E and F. These samples are then added to the new training set and the update cycle repeats. Image c) shows manifolds updated with the samples E and F.

their contribution to the final confidence c should reflects this. The weighting itself is found experimentally by evaluating the accuracy of each metric in turn on a set of unlabelled samples and basing the weight on how accurate that metric is. Importantly these values may change based on application as certain metrics may be more informative in different problems.

$$c = c_a w_a + c_b w_b + \frac{1}{c_c} w_c \tag{7}$$

Using a defined threshold T_c, samples can now be approved for inclusion in the labelled dataset $\mathbb{D}_l$ updated for use in the next training iteration.

$$\mathbb{D}_l \leftarrow (x^{(j)}, y_1) \quad \text{if} \quad c > T_c \tag{8}$$

The model is then re-trained over the updated training dataset $(\mathbf{y}, \mathbf{X}) \in \mathbb{D}_l$.

$$\theta = \underset{\theta}{\operatorname{argmax}}\, P(\mathbf{y}|\mathbf{X}; \theta) \tag{9}$$

If the ground truth labels $\mathbf{y}$ are known for the evaluated, unlabelled samples $\mathbf{X_u}$, the unsupervised, labelling accuracy q given the threshold T_c and model parameters θ can be calculated.

$$q = acc(\mathbf{X_u}, \mathbf{y}, T_c, \theta) \tag{10}$$

Where the function $acc()$ calculates a percentage of correctly labelled samples given the threshold T_c.

The definition of the threshold T_c could be defined manually, allowing for policies where only very confidently analysed samples are added or, through the use of a lower threshold, a more "quantity over quality" policy can be adopted. In this work the threshold value T_c is learned after the initial train of the model in which only cleanly labelled data is used. A process is run to find a threshold whereby if the labelled training data was considered "unlabelled" and was evaluated by the model, how higher a threshold T_c would be required to only let correct classifications with a certain accuracy T_a to be included in $\mathbb{D}_l$ updated. Given a model θ trained on the clean, labelled dataset $\mathbb{D}_l$, and desired minimal accuracy T_a, we can calculate T_c as:

$$T_c = \max t_c \quad \text{subject to} \quad acc(\mathbf{X}, \mathbf{y}, t_c, \theta) > T_a \tag{11}$$

Where $(\mathbf{X}, \mathbf{y}) \in \mathbb{D}_l$ and t_c is the current threshold being evaluated. In the case of this work the accuracy T_a was set to higher that 99%. This process is run on training data as the model will be most confident on samples it has already seen and, as a result of this, impose a higher threshold than one defined using the validation set.

B. Iterative Learning-Ensemble (IL-E)

In addition to the threshold techniques already discussed, an Iterative Learning - Ensemble (IL-E) method, was also devised using the well established ensembling technique to improve the confidence scores of the unlabelled samples [17], [21], [22]. This process adds an additional step to the evaluation process whereby before an unlabelled sample is evaluated using the trained model (θ), a number of augmentations are applied such that $\mathbb{D}_{aug} = \mathbf{x}_a^{(j)}, y^{(j)} | a = 1, \ldots, A^j$ represents a single sample $\mathbf{x}^{(j)}$, augmented in A different ways, each with the same label $y^{(j)}$. The augmented samples, including the original, are now passed to the model for inference and the posterior probability vectors, or class distributions, for all the augmented samples $\tilde{z}$ returned.

$$\tilde{z} = P(y|\mathbb{D}_{aug}; \theta) \tag{12}$$

The goal of the ensembling process is to define a class distribution for use in the confidence metrics, based on one of the sample augmentations. The chosen distribution is then scaled by the similarity of the class distributions returned as a result of Eq. 12. Scaling is based on the standard deviation on the posterior probabilities between class labels across the augmented samples, which is then subtracted from $\tilde{z}$. Finally, the augmented sample a with the highest posterior probability for any class label, is selected and its class distribution used as input $\mathbf{x}^{(j)}$ for the confidence metrics highlighted in Eq. 3, 4 and 6.

$$a = \underset{a}{\operatorname{argmax}}(\tilde{z}_a - \sigma) \tag{13}$$

Augmented ensembles which when evaluated differ in their class distributions would result in a larger standard deviation, which in turn would penalise the final confidence score $\mathbf{x}_a^{(j)}$ more so than when the model produces more similar distributions across the ensemble. Use of a single distribution from the ensemble to the aforementioned confidence metrics, which the model has selected with the highest confidence in an individual class, allows for the model to use the best fit sample to the established manifolds. This process is applied both when new samples are added but also during the learning of the confidence threshold T_c

By leveraging these incremental updates the model can be utilised to identify only those samples that it is most confident belong to a respective class. As a result the model develops its knowledge of specific classes and is therefore better able to identify additional samples in latter iterations. This process is symbolically shown in Figure 4, whereby a subset of new, unlabelled, samples get projected closer to the existing manifolds due to already learned characteristics of respective classes. These samples are then labelled and added to the training dataset and the model is re-trained. The manifolds are now updated, reflecting the information brought in by the added training samples.

By extension, better knowledge in one class has a knock on effect to the other classes as the model is better able to eliminate certain postulations from less well understood classes. The result is an overall improvement over time to validation accuracy and more specifically improved definition of some class identities. Additionally as the model is re-initialized at the beginning of each iteration, this method can leverage randomly initialised weights to help with the classification of unlabelled samples.

The most important aspect of this method is the need to ensure that the model is suitably confident in the labelling of a new sample before it is added to the training set. To achieve this a number of stringent metrics are utilised to ensure only the most confident of classifications are used. The iterative process is repeated until the set confidence metric thresholds return no candidate samples that meet the required expectations to be included in a new training set or when validation accuracy no longer deviates. Additionally, during the process, the confidence metrics for each newly labelled sample is recorded, as is each iteration of the dataset, providing the ability to utilise dataset snapshots from each iteration.

IV. Results

A. Experiment Environment

Much of the work within the area of semi-supervised learning is benchmarked against the SVHN [33] dataset. In addition to this and to better validate the performance of this iterative approach on a more challenging task, experiments are also conducted on the CIFAR-100 dataset [34], and a 200-class subset of ImageNet known as Tiny ImageNet. CIFAR-100 is a 100-class dataset with a total of 60,000 32-by-32 pixel colour images, of which there are 600 images per class, 500 training images and 100 testing images. Tiny ImageNet contains a total of 100,000 64-by-64 pixel colour images, divided into a 10,000 image validation set and a 90,000 image training set, each of the 200-classes has 450 train images and 50 test. The SVHN dataset contains 630,420 32-by-32 pixel colour images of cropped digits, 73,257 digits for training, 26,032 digits for validation and 531,131 for testing. Some examples from these three datasets are given in Figure 5.

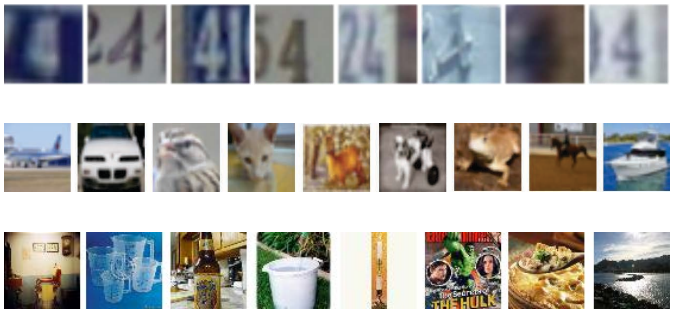


Fig. 5. Example Images from: (top) SVHN, (middle) CIFAR-100 and (bottom) TinyImageNet.

To evaluate the concept of iterative learning we conduct evaluations of the method focusing on two areas: Classification improvement and the increase in the dataset volume to achieve that improvement. The first, evaluates the algorithm's ability to improve the classification results for a given model. As such, an iterative loop is set up as described in Algorithm 1 and is run for the three models across the three datasets. This loop is run for a maximum of 75 iterations or until either accuracy reaches a plateau or no new samples are added to training set. As mentioned in the Methodology Section, the architecture of the model is not under review and in fact the only constraint for which model is chosen is that it fits the task well. To this end a number of well known image classification architectures are evaluated including the ResNet-18 architecture [1], LeNet-5 [35] and JFNet (see Appendix A). These models were chosen as they represent a range of architecture types, spanning both many and few layers and likewise relatively low numbers of parameters as well as high (Appendix B). To evaluate the proposed methodology in both full training as well as fine tuning scenarios, untrained versions of LeNet-5 and JFNet architectures and a pre-trained ResNet-18 model were utilised. Lastly, this range of model architectures were deliberately selected for their varied performance across the three datasets used in these evaluations.

For all experiments the same cross entropy loss function was used. Stochastic gradient decent is also utilised for all experiments with a starting learning rate of 0.01, with a scheduled step to 0.001 after 100 epochs. Weights for the metrics c_a, c_b, and c_c, were set to 1, 0.5 and 0.25 respectively. Additionally dataset images are normalised for standard deviation and mean pixel values and a random crop augmentation ($\pm$ 2 pixels) was applied during training. The use of ambiguous tuning parameters here is deliberate; the proposed method is designed to be general and although very high results are obtained on the SVHN dataset this work is not designed as a parametric tuning exercise, rather an indication that this method is beneficial even in suboptimal conditions.

TABLE I
ITERATIVE LEARNING RESULTS TABLE ACROSS THE THREE DATASETS (SVHN, CIFAR-100, TINYIMAGENET), INCLUDING RESULTS FROM THE IL AND IL-E METHODOLOGIES, WITH COMPARATIVE RESULTS WHERE POSSIBLE, AND FULL AND SUBSET BENCHMARK TRAINING RESULTS.

Model	Error Rate% (σ 1k Benchmark	Error Rate % (Improvement) 1k Samples	Error Rate% (σ) Full Benchmark	Added Samples (Accuracy %)
		SVHN		
GAN [36]	N/A	5.88%	N/A	-
VAT+EntMin [19]	N/A	3.86%	N/A	-
$\prod$ + SNTG [20]	14.46% (±0.71)	3.82% (-10.64)	2.81% (±0.07)	-
$\prod$ model [17]	19.30% (±3.89)	4.82% (-14.48)	2.54% (±0.04)	-
Temporal Ensembling [17]	19.30% (±3.89)	4.42% (-14.88)	2.74% (±0.06)	-
ResNet-18 (IL)	19.74% (±0.32)	5.12% (-14.62)	2.98% (±0.04)	72,077 (94.89%)
LeNet-5 (IL)	25.24% (±1.55)	9.74% (-15.5)	7.16% (±0.09)	71,703 (90.82%)
JFNet (IL)	20.18% (±0.55)	5.52% (-14.66)	3.84% (±0.05)	70,893 (94.45%)
ResNet-18 (IL-E)	19.74% (±0.32)	**4.29% (-15.45)**	2.98% (±0.04)	71,068 (94.89%)
LeNet-5 (IL-E)	25.24% (±1.55)	11.11% (-14.13)	7.16% (±0.09)	42,999 (96.86%)
JFNet (IL-E)	20.18% (±0.50)	5.64% (-14.54)	3.84% (±0.05)	66,421 (96.13%)
		CIFAR-100		
	5k Benchmark	5k Samples	Full Benchmark	
$\prod$ + SNTG [20]	N/A	37.97% (10k Samples)	26.32% (±0.04)	-
$\prod$ model [17]	N/A	39.19% (10k Samples)	26.32% (±0.04)	-
Temporal Ensembling [17]	N/A	38.65% (10k Samples)	26.30% (±0.15)	-
VAT+EntMin (ResNet-18)	32.49% (±0.45)	31.54% (-0.95)	17.53% (±0.09)	-
ResNet-18 (IL)	32.49% (±0.45)	29.68% (-2.8)	17.53% (±0.09)	9,592 (96.19%)
LeNet-5 (IL)	89.21% (±0.22)	87.52% (-1.69)	65.55% (±0.38)	411 (74.45%)
JFNet (IL)	67.85% (±0.39)	66.44% (-1.41)	39.66% (±0.22)	3,638 (77.24%)
ResNet-18 (IL-E)	32.49% (±0.45)	**28.09% (-4.4)**	17.53% (±0.09)	42,526 (75.1%)
LeNet-5 (IL-E)	89.21% (±0.22)	87.47% (-1.74)	65.55% (±0.38)	375 (72.53%)
JFNet (IL-E)	67.85% (±0.22)	66.49% (-1.36)	39.66% (±0.22)	4,786 (73.21%)
		Tiny ImageNet		
	10k Benchmark	10k Samples	Full Benchmark	
$\prod$ model (ResNet-18)	37.47% (±0.46)	36.34% (-1.13)	27.38% (±0.15)	-
VAT+EntMin (ResNet-18)	37.47% (±0.46)	42.1%(+4.63)	27.38% (±0.15)	-
ResNet-18 (IL)	37.47% (±0.46)	**33.35% (-4.12)**	27.38% (±0.15)	53,522 (82.85%)
LeNet-5 (IL)	95.48% (±0.43)	94.01% (-1.47)	81.58% (±0.27)	40 (35%)
JFNet (IL)	83.40% (±0.12)	81.61% (-1.79)	60.98% (±0.25)	1,919 (73.89%)
ResNet-18 (IL-E)	37.47% (±0.46)	33.68% (-3.79)	27.38% (±0.15)	56,619 (81.37%)
LeNet-5 (IL-E)	95.48% (±0.43)	94.43% (-1.05)	81.58% (±0.27)	69 (43.49%)
JFNet (IL-E)	83.40% (±0.12)	81.61% (-1.79)	60.98% (±0.25)	684 (83.19%)

B. Model Accuracy

Initially benchmarks are run for each of the three models on the three datasets. Results Table I (columns 1 & 3) outline the benchmark error rates for each of these model architectures on both a subset of the training data and the full. The subset size is based on 50 samples per class of the training data for each dataset, CIFAR-100 uses 5,000 samples and Tiny ImageNet uses 10,000 samples. As the SVHN dataset is one of the most commonly used datasets when comparing semi-supervised learning techniques, the standard 1,000 samples is used (100 samples from each of the 10 classes). Each training subset is made up of an even distribution of classes with images from each class chosen at random. Each experiment was conducted 4 times with the average results presented along with the standard deviation given in brackets. The inclusion of these benchmarks is vital, especially for any result that uses a customised loss function or architecture, as without, it is difficult to ascertain if improvement gains can be attributed to the model architecture used or the semi-supervised method.

The Iterative learning (IL) and Iterative Learning-Ensemble (IL-E) methods were applied to the three datasets, initialised with the same subsets as seen in the benchmarks, importantly there are no pre processing or clustering steps taken on any of the datasets used during experimentation. The SVHN dataset provides a relatively simple classification task from which to analyse the iterative learning approach. Although this task could be considered non-representational of the complexity of modern day image classification tasks and their respective datasets, it does allow for a comparison with many existing semi-supervised learning techniques. As such Table I outlines the error rates for this and the remaining two datasets. Column 2 of the table shows the error rates when using a semi-

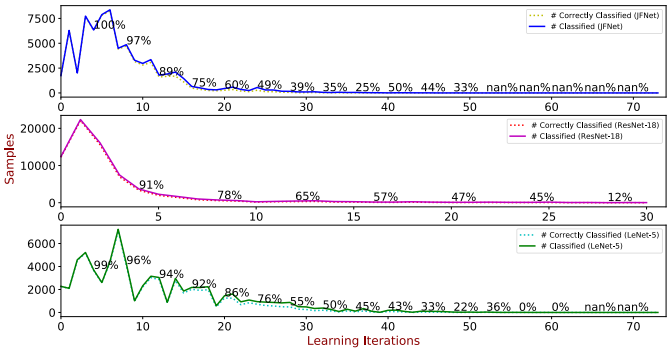


Fig. 6. Addition of newly labelled samples to the training dataset for each model architecture during the Iterative Learning (IL) cycles on the SVHN dataset. The solid coloured line representing the number of samples added during that iteration and the dashed line the number of those samples that were correctly labelled (with intermittent numeric labels).

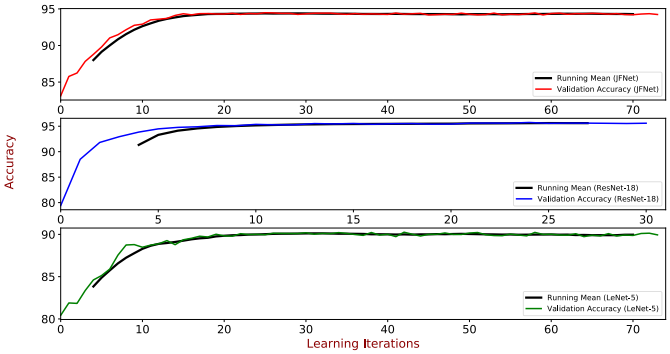


Fig. 7. The classification accuracy on the SVHN dataset during the Iterative Learning (IL) cycles, for each of the three model architectures, (coloured lines) per iteration accuracy, (black lines) the moving average (over 8 samples).

supervised learning technique initialised with a subset of training samples. The value in brackets within this column represents the improvement over the respective original benchmark. As can be seen the proposed IL-E method demonstrates the greatest improvement over the original benchmark (-15.45%). In the case of the proposed work this result is based on the lowest error rate from the iterative learning cycles and is based on validation results, as in a real world application this control set would need to be utilised to make an assessment of if the learning process is still informative or not. As can be seen, the proposed method achieves a close to state of the art result on the SVHN dataset with no change to the ResNet-18 architecture or the loss functions used and represents a considerable improvement on the original benchmark. The Smooth Neighbour on Teacher Graphs method [20] has a slightly lower overall error rate for SVHN dataset than the other methods tested, however they also return the lowest improvement over their original benchmark, highlighting that their semi-supervised method was arguably the least effective and in fact less effective at reducing error that the original method they extend.

Additionally it is worth noting that in the SVHN examples the training dataset is grown from the original 1000 samples to upwards of 70,000 samples (of a total 72,257 unlabelled possibilities) with an accuracy of over 90%. Figure 6 shows the number of samples added to the training set after each cycle of the iterative learning process for the SVHN dataset, highlighted values are the percentage accuracy of those additions (nan values represent cycles where no new samples were above the required threshold). After an initial burst, a gradual decline in the volume of additions to the dataset is seen. This follows the intuition that the most confident and easily classifiable samples would be added first with that increase in training set size further helping to develop the model to classify further unlabelled samples. This effect tallies with the sharp rise in validation accuracy that can be seen in Figure 7, which depicts the progression of the accuracy over the course of the iterative learning process.

The tests conducted with the CIFAR-100 and TinyImageNet datasets are designed to give an overview of the iterative learning process in less ideal scenarios. In all iterative learning examples an improvement in model error rates is seen, however the increased complexity of the CIFAR-100 and TinyImageNet datasets demonstrates how the process works when the initial learning iterations do not return as lower error rates as seen with the SVHN examples. The main contributing factor to this is the rate with which additional samples are added to the training set, this is in turn directly effected by the initial accuracy of the model after the first self training loop. Figure 8 & 9 shows the same analysis as with SVHN in terms of the samples added to the training set and the classification accuracy over the course of the iterative learning process for the CIFAR-100 and TinyImageNet datasets. The evaluation on the CIFAR-100 dataset is especially interesting as the three models used have widely varying performance on this dataset, ResNet-18 has initially high accuracy with numerous added samples which, for the most part, are well above 90% accurate. Comparatively LeNet-5 has very low performance on this dataset, this is not surprising given the low number of parameters within this network architecture and the last fully connected layers which originally contained less neurons that the number of classes in these datasets, however even with these issues the Iterative Learning process is still able to make new additions to the dataset and improve the error rate throughout the cycles.

Results on the TinyImageNet dataset follow a similar trend. The ResNet-18 model provides good initial accuracy even though there are twice as many classes, this is likely due to the increased image size, as a result of this the familiar downward slope of the dataset additions and accuracy is seen with the correlated increase in classification accuracy. In this example the classification peaks and begins to tail off after iteration twelve. This is soon after the dataset addition accuracy drops below 75%, this change in validation accuracy is a good indication that the iterative learning process has run its course and the dataset additions past this point are not adding to the model learning and therefore should be disregarded.

C. Dataset Volume and Accuracy

Evaluation of model improvement is based on error rate, however as the model develops with additional data being added to the training set, it is useful to see if the error rate is being reduced universally across all classes or if improvement in some classes has an impact on others. In the case of SVHN dataset and the JFNet model, it is the former. Figure 10 shows the per class classification difference, on the validation set, for a model trained on the benchmark samples and the model after running the Iterative Learning (IL) process. In this case the effect is only positive, with the iteratively learnt model classifying more samples for each class correctly than the original model did.

This improvement is not as universal when it comes to datasets with more classes, Figure 11 shows the same evaluation on the CIFAR-100 dataset using the JFNet model. In this case it can be seen that the effect on the validation dataset is not as universal, with some class's accuracy improving at the cost of others. The CIFAR-100 dataset has 100 classes grouped evenly into 20 subclasses, providing an interesting

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2019.2913986, IEEE Transactions on Image Processing

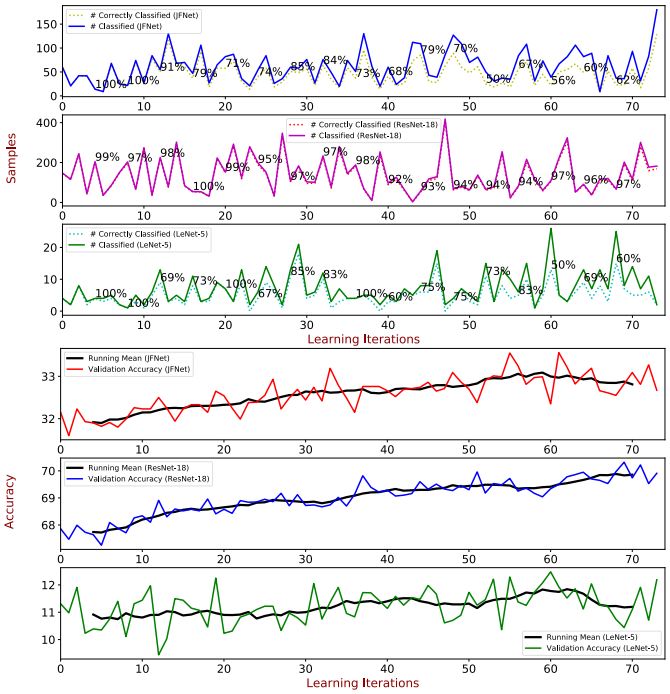


Fig. 8. (Top) Addition of newly labelled samples to the training dataset and (Bottom) classification accuracy over the course of the self learning process for the CIFAR-100 dataset, for each of the three model architectures, (coloured lines) per iteration accuracy, (black lines) the moving average (over 8 samples).

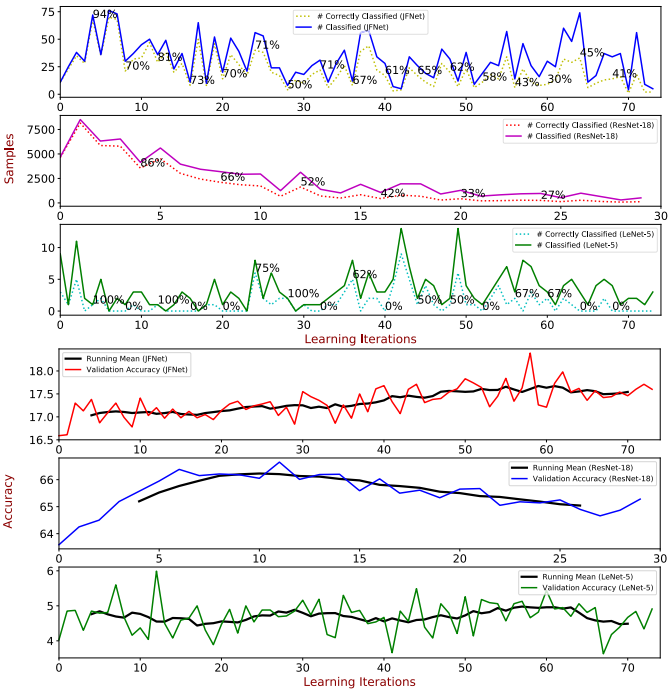


Fig. 9. (Top) Addition of newly labelled samples to the training dataset and (Bottom) classification accuracy over the course of the self learning process for the TinyImageNet dataset, for each of the three model architectures, (coloured lines) per iteration accuracy, (black lines) the moving average (over 8 samples).

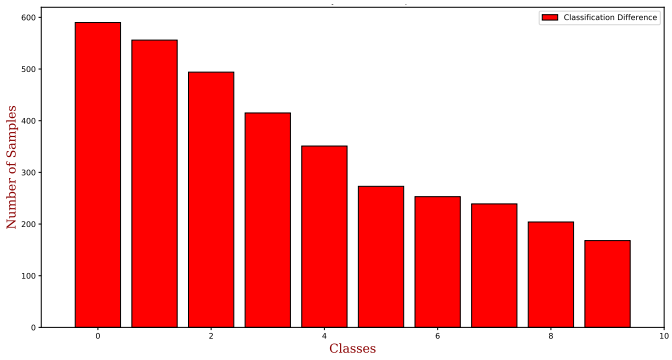


Fig. 10. The effect Iterative Learning (IL) has had on the validation dataset accuracy on the SVHN dataset, using the JFNet Model. Given as the increase or decrease to each individual class accuracy between the benchmark model versus the same model architecture after self learning (results presented from highest to lowest improvement).

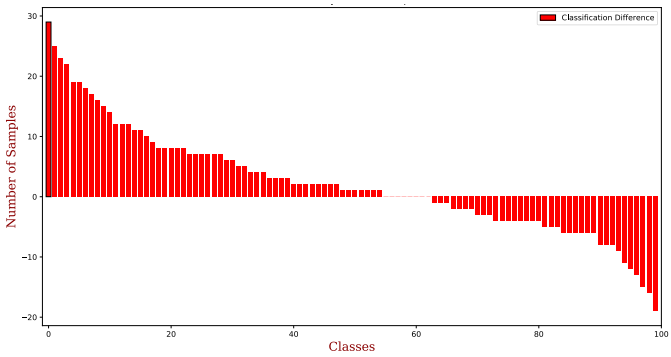


Fig. 11. The effect Iterative Learning (IL) has had on the validation dataset accuracy on the CIFAR-100 dataset, using the JFNet Model. Given as the increase or decrease to each individual class accuracy between the benchmark model versus the same model architecture after self learning (results presented from highest to lowest improvement).

look at the effect improvement in classification accuracy in one class has on similar classes. Four of the ten most improved classes in Figure 11 share a subclass with one or more of the classes in the ten least improved. Suggesting that classification accuracy in one class can have an effect on similar objects within the subclass. In addition, seven of the top ten most improved classes, share three subclasses, demonstrating that the method might exhibit cumulative gains across similar objects, for example the subclass *insect* has three entries in the top ten improved classes and one in the least improved.

Outlined in the methodology are two process by which each unlabelled sample is added to the training dataset, firstly using the learnt thresholds (the standard Iterative Learning (IL) method) and secondly with an added ensemble augmentation stage (IL-E). The Iterative Learning process looks to both improve model accuracy but also increase the dataset volume in a meaningful way and as such it is important to review how accurate the additional newly labelled samples are for both methods. Results Table I (column 4) outlines the number of added samples in the training dataset with the accuracy of those additions in brackets, based on the dataset iteration at

the time of highest validation accuracy. These results correlate to error rates that a specific model achieves for a given dataset.

The SVHN results are universally above 90% accuracy and add a large number of samples to the training set, this is as a result of error rates being initially low and as a product of the models improvement on the problem (all reduce their error rate by more than 14%). The use of the IL-E process seems to make the adding of new samples to the dataset more stringent with a higher level of accuracy at the expense of the number of added samples. This increase in sample accuracy is likely due to the effect a considerably smaller class pool has on the confidence metric c_c, as the smaller the number of classes the smaller the cumulative distance is, and in turn the more this metric contributes to the final confidence threshold that is defined at the beginning of the process. The opposite of this is seen in the results for the CIFAR-100 dataset. In this case the largest improvement in error rate comes from the ResNet-18 architecture using the IL-E method. This is as a result of a large increase in training data but with a lower accuracy (over 40,000 new samples over IL's 9,592). This drop in accuracy in the added data is due to a lower learned confidence threshold at the beginning of the process, impacted by the larger number of classes and the effect on confidence metric c_c. The closest comparative work using the CIFAR-100 dataset uses temporal ensembling [17] and achieves an error rate of $38.65\% \pm 0.51$ using 10,000 labelled samples to initialise their semi-supervised learning technique, over 10% higher than the IL-E method which uses just 5,000 samples. Demonstrating that even in more complex tasks the iterative learning concept performs at a state of the art level.

In general the Iterative Learning-Ensemble technique tends to improve the classification accuracy of a model, however the improvement is reliant on the model have a reasonable level of classification accuracy in the first place. This problem is exacerbated by problems with a large number of classes as the ensemble results will tend to be more confused and thus the standard deviation between the augmented samples will be higher. The other consideration when using the IL-E process is the by adding an additional step to the labelling stage additional computation power is required slowing the labelling process down.

In order to compare all methods on a fair ground, the VAT [19] and $\prod$-model [17] methods were implemented using the ResNet-18 architecture and evaluated on the challenging CIFAR-100 and Tiny ImageNet datasets. The Temporal Ensembling method was not implemented since its architecture and performance are very similar to the $\prod$-model.

VAT on the CIFAR-100 dataset produces the second best performance with an error rate of 31.54%, a marginal performance gain of 0.95% over the benchmark. This was, however, possible only when initiating the training with a pre-trained ResNet on the 5k CIFAR-100 dataset. At the begging of the training the error rate significantly grew over first few epochs and then slowly decreased. The VAT method employs a compositional loss function with cross entropy and adversarial loss components. When training the VAT method with ResNet-18 initialised on the ImageNet, as in all other experiments, the VAT method achieves only very high error rates, significantly above the benchmark values. The VAT training appears to disrupt already learned parameters despite starting with a model trained with cross entropy loss on identical, labelled samples. This suggests that the adversarial loss component of the method has a negative effect on the convergence of the ResNet network.

Tiny ImageNet proves to be an even more challenging dataset for the VAT method when using the ResNet-18 architecture. In this case the ResNet-18 model was again bootstrapped using the 10k dataset with cross entropy loss. The model was trained over 500 epochs and encountered similar behaviour to when training on CIFAR-100 dataset. In this case the error rate never decreased below the benchmark even after extensive hyper-parameters tuning, namely the epsilon, SGD and Adam learning rate, weights decay, different labelled/unlabelled batch size ratios as well as various data augmentations. The $\prod$-model performs better with the ResNet-18 model, although a slight drop in accuracy was encountered at the beginning of the training when starting with a model pre-trained on the 10k labelled dataset.

Results of these experiments suggest that the investigated semi-supervised methods with regularisation techniques have difficulties when applied to deep architectures such as ResNet-18 on complex datasets and require careful parameters tuning. In contrast, the presented IL and IL-E methods do not enforce any specific inside-loop training procedure, making it suitable for applications in a majority of image classification methods.

V. Conclusion

As demonstrated in the IL and IL-E methods, the presented simple iterative approach to semi-supervised learning has a number of benefits. Most notable being state of the art error rates on the CIFAR-100 dataset and near state of the art on SVHN dataset, with no required changes to the train methods or model architectures used. The presented methods demonstrate the ability for a model to leverage its own confidence scores to improve itself in the presence of stringent enough metrics and thresholds to govern the process. The simplicity and general nature of the IL methods lend themselves well to image classification tasks and it is believed the process could be applicable in other classification tasks. An additional benefit to the process is the non trivial ability to significantly improve the volume of datasets with only a marginal effect on accuracy, allowing for the use of large volumes of unlabelled data to further improve deeply learned models. Finally an interesting by product of this process is the increase in classification accuracy of a model in specific classes, this property of the algorithm could be further leveraged and focused to target generalisation of specific classes, for example in human detection in object detection tasks.

Appendix A
The JFNet Architecture

The JFNet architecture was developed as a simple, generic CNN for classification of low resolution images. The architecture was inspired by the work of G. Huang et al. [37] on densely connected convolutional networks. However, in

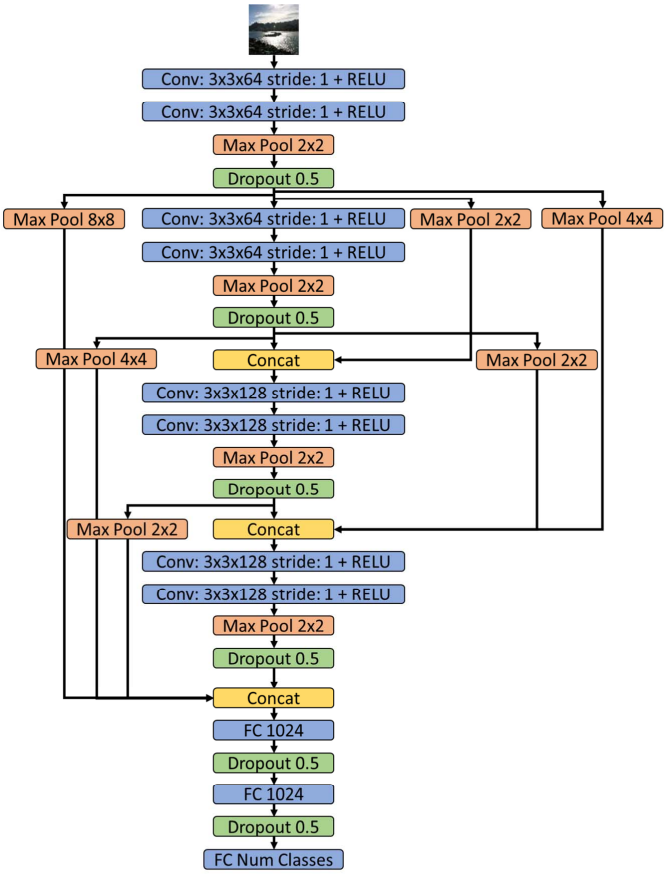


Fig. 12. JFNet architecture.

contrast to Huang's work, JFNet reduces spatial resolution and increases depth of feature maps within a single dense block and is terminated by three fully connected layers for classification as shown in Figure 12.

APPENDIX B
MODEL OVERVIEW

TABLE II
OVERVIEW OF MODEL ARCHITECTURES

Model	Parameters	Layers
JFNet (Appendix A)	11,037,386	11
LeNet-5 [35]	61,706	6
ResNet-18 [1]	11,181,642	18

APPENDIX C
USED HARDWARE

All our experiments were conducted on a Dell Precision Tower 7910 XCTO server with two GPUs TITAN Xp, Intel Xeon E5-2623 v3, 128GB DDR4 RAM and 512GB SSD.

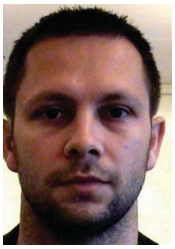
REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proceedings of the Advances In Neural Information Processing Systems, 2012, pp. 1097–1105.
[3] A. Graves and N. Jaitly, "Towards End-To-End Speech Recognition with Recurrent Neural Networks," JMLR Workshop and Conference Proceedings, vol. 32, no. 1, pp. 1764–1772, 2014.
[4] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82–97, 2012.
[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Proceedings of the Advances in Neural Information Processing systems, 2014, pp. 3104–3112.
[6] Y. Goldberg, "Neural network methods for natural language processing," Synthesis Lectures on Human Language Technologies, vol. 10, no. 1, pp. 1–309, 2017.
[7] A. Khosla, A. S. Raju, A. Torralba, and A. Oliva, "Understanding and predicting image memorability at a large scale," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2390–2398.
[8] J. Fajtl, V. Argyriou, D. Monekosso, and P. Remagnino, "Amnet: Memorability estimation with attention," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6363–6372.
[9] J. Fajtl, H. S. Sokeh, V. Argyriou, D. Monekosso, and P. Remagnino, "Summarizing videos with attention," in Proceeding of the ACCV AIU workshop, 2018.
[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: A large-scale hierarchical image database." in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.
[11] S. Hong, H. Noh, and B. Han, "Decoupled Deep Neural Network for Semi-supervised Semantic Segmentation," in Proceedings of the Advances in Neural Information Processing Systems, vol. 28, 2015, pp. 1495—-1503.
[12] J. Weston, N. E. C. L. America, and I. Way, "A Unified Architecture for Natural Language Processing : Deep Neural Networks with Multitask Learning," in Proceedings of the 25th International Conference on Machine Learning, vol. 20. ACM, 2008, pp. 160–167.
[13] A. Carlson, J. Betteridge, and B. Kisiel, "Toward an Architecture for Never-Ending Language Learning." in Proceedings of the AAAI Conference on Artificial Intelligence, 2010, pp. 1306–1313.
[14] A. Joulin, L. van Der Maaten, A. Jabri, and N. Vasilache, "Learning visual features from large weakly supervised data," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9911 LNCS, pp. 67–84, 2016.
[15] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie, "Learning from noisy large-scale datasets with minimal supervision," in Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, vol. 2017-Janua, 2017, pp. 6575–6583.
[16] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari, "Learning with Noisy Labels," in Proceedings of the Advances in Neural Information Processing Systems, 2013, pp. 1196–1204.
[17] S. Laine and T. Aila, "Temporal Ensembling for Semi-Supervised Learning," in Proceedings of the International Conference on Learning Representation, 2017, pp. 1–13.
[18] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial Training Methods for Semi-Supervised Text Classification," in Proceedings of the International Conference on Learning Representation, 2017, pp. 1–11.
[19] T. Miyato, S. ichi 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, 2018.
[20] Y. Luo, J. Zhu, M. Li, Y. Ren, and B. Zhang, "Smooth Neighbors on Teacher Graphs for Semi-supervised Learning," CoRR, vol. abs/1711.0, 2017.
[21] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, "Semi-Supervised Learning with Ladder Networks," in Proceedings of

the Advances in Neural Information Processing Systems, 2015, pp. 3546–3554.
[22] A. Tarvainen and H. Valpola, "Mean teachers are better role models : Weight-averaged consistency targets improve semi-supervised deep learning results," in Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 1195–1204.
[23] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization With Stochastic Transformations and Perturbations for Deep Semi-Supervised Learning," in Advances in Neural Information Processing Systems, 2016, pp. 1163–1171.
[24] S. Cicek, A. Fawzi, and S. Soatto, "SaaS: Speed as a Supervisor for Semi-supervised Learning," arXiv preprint arXiv:1805.00980, 2018.
[25] P. Haeusser, A. Mordvintsev, and D. Cremers, "Learning by association– a versatile semi-supervised training method for neural networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 89–98.
[26] M. Szummer and T. Jaakkola, "Partially labeled classification with Markov random walks," in Proceedings of the Advances in Neural Information Processing Systems 14, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 945–952.
[27] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep Learning via Semi-Supervised Embedding," in Neural Networks: Tricks of the Trade. Springer, 2009, pp. 639–655.
[28] X. Zhu and Z. Ghahramani, "Learning from Labeled and Unlabeled Data with Label Propagation," Carnegie Mellon University Technical Report, no. CMU-CALD-02-107, 2002.
[29] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," Journal of machine learning research, vol. 7, no. Nov, pp. 2399–2434, 2006.
[30] G. French, M. Mackiewicz, and M. Fisher, "Self-ensembling for visual domain adaptation," in Proceedings of the International Conference on Learning Representations, 2018.
[31] A. R. Barron, "Approximation and estimation bounds for artificial neural networks," Machine Learning, vol. 14, no. 1, pp. 115–133, 1994.
[32] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," in Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 6241–6250.
[33] Y. Netzer and T. Wang, "Reading digits in natural images with unsupervised feature learning," in Proceedings of the Advances in Neural Information Processing Systems, 2011, p. 5.
[34] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Ph.D. dissertation, 2009.
[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2323, 1998.
[36] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," in Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 1–9.
[37] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, no. 2, 2017, pp. 2261–2269.

Rob Dupre is a post doctorate researcher working under the Robot Vision Team (RoViT) at Kingston university focusing in the fields of crowd analysis and simulation. Rob completed a BSc degree in Games Technology in 2014, developing Human Computer Interfaces (HCI) for use in game environments, he later undertook a PhD in automated risk assessment for domestic robotics in 2017, with a focus of the identification and reduction of risky objects and their positions in indoor environments. Rob is currently working on the large scale IoT project MONICA, under the Horizon 2020 EU funding scheme. Other research interests include semi-supervised learning and crowd density analysis.

Jiri Fajtl With an engineering background in electronic and radio communication, Jiri spent more than 20 years in industry involved in research and development in domains ranging from DSP, peer to peer networking, dynamic systems control, computer vision and machine learning, mainly for robotics. In 2015 he received MSc from Kingston University, London in embedded systems and computer vision, and currently pursuing his PhD in machine learning for computer vision, also at Kingston University.



Vasileios Argyriou received his BSc degree in computer science from Aristotle University of Thessaloniki, Greece, in 2001 and his MSc and PhD degrees from the University of Surrey, in 2003 and 2006, respectively, both in electrical engineering working on registration. From 2001 to 2002, he held a research position at the AIIA Lab, Aristotle University, working on image and video watermarking. He joined the Communications and Signal Processing (CSP) Department, Imperial College, London, in 2007, where he was a Research Fellow working on 3D object reconstruction. Now, he is a Professor at Kingston University, working on computer vision and AI for crowd and human behavior analysis, computer games, entertainment, and medical applications. Also, research is conducted on educational games and on HCI for augmented and virtual reality (AR/VR) systems.

Paolo Remagnino Prof. Remagnino leads the Robot Vision Team (ROVIT), in the Department of Computer Science at Kingston University. Prof. Remagnino research is mainly concerned with the development of innovative methods for image and video interpretation, making wide use of pattern recognition, machine and deep learning and distributed intelligence techniques. Prof. Remagnino has published over 120 scientific articles in international conferences and high impact journals. Currently, Prof. Remagnino is the principal investigator of four projects on video analytics for security with fixed and mobile cameras.