Auto-Annotation Quality Prediction for Semi-Supervised Learning with Ensembles

Dror Simon   Miriam Farber   Roman Goldenberg

Amazon.com

`drorsimo/mffarber/romang@amazon.com`

Abstract

Auto-annotation by ensemble of models is an efficient method of learning on unlabeled data. Wrong or inaccurate annotations generated by the ensemble may lead to performance degradation of the trained model. To deal with this problem we propose filtering the auto-labeled data using a trained model that predicts the quality of the annotation from the degree of consensus between ensemble models. Using semantic segmentation as an example, we show the advantage of the proposed auto-annotation filtering over training on data contaminated with inaccurate labels.

Moreover, our experimental results show that in the case of semantic segmentation, the performance of a state-of-the-art model can be achieved by training it with only a fraction (30%) of the original manually labeled data set, and replacing the rest with the auto-annotated, quality filtered labels.

1. Introduction

Semi-supervised learning, using the combination of a smaller set of labeled data and a larger set of unlabeled data, is becoming increasingly important with the growing capacity of trained models and their tasks complexity. Higher capacity models require more training data, and complex tasks make manual annotation more labor intensive. Ensembles of models have been successfully used for automatic annotation of unlabeled data [5, 43]. In self-training scenarios, multiple instances of the target model trained on the labeled data form an ensemble. The ensemble, which is said to be more accurate than a single model, then labels the unlabeled data. In model-compression scenarios, ensembles extract the knowledge from a set of more powerful pre-trained models.

Using ensembles for unlabeled data annotation, however, raises one issue that is often not properly addressed: the annotation quality. In general, the ensemble cannot guarantee that the annotations it generates meet the quality bar. Using wrong or inaccurate annotations for training may negatively affect the target model.

In this work we propose a method for predicting the quality of the annotations generated by an ensemble. The approach uses a model trained to assess the quality of the generated annotation from the degree of consensus between ensemble models.

We propose to refine the auto-labeled data set by discarding samples with low predicted annotation quality. We show that training on a refined reduced set is advantageous over using a lager set, contaminated with inaccurate data.

The main contributions of the paper are:

- We introduce an automatic filtering of auto-annotations generated by ensembles, using a trained model that predicts annotation quality from ensemble models degree of consensus.

- We propose an auto-annotation quality control scheme for semantic segmentation, which filters bad labels at pixel level, yielding a refined partial image labeling.

- We demonstrate that training semantic segmentation model with quality filtering is advantageous over training with no filtering.

- We show how the proposed method achieves the same accuracy as a state-of-the-art model, while manually annotating of only a fraction (30%) of the training set.

2. Related work

Ever since is was demonstrated that ensembles of models can boost the accuracy [18], and the reasons ensembles outperform any single model they are composed of were identified [13], they have been used extensively to achieve state-of-the-art performance. A large body of work exists on the subject, reviewed in [37, 32, 21].

According to the taxonomy developed in [37, 25], the ensemble approaches are divided into non-generative - using a fixed set of pre-designed models, and generative - generating models by acting on the base learning algorithm or

on the structure of the training set. While the approach we propose can be generally applied to both types, in the experiments presented here we used ensemble generation by bagging [3] the same model trained with different permutations and augmentations of the training set. Other techniques include bagging by varying parameter settings [6], a low computational cost snapshot ensembles [26, 20, 7] and many others.

Here we combine bagging with a complimentary approach to accuracy boosting - ensembling by input transformations. Variants of it were used in various computer vision tasks including object detection [40, 14], image classification [23], recognition [36] and keypoints detection [31].

Using ensembles of models for self-training was proposed in [5], where a knowledge distillation is performed on unlabeled data by an ensemble trained on a smaller set of labeled data. The idea of self-labeling goes back to 1965 [35], and since then was a subject of research in semi-supervised learning [45]. The benefits of using ensembles for semi-supervised learning are advocated in [44].

Here we propose a model distillation regime where the quality of the labeling generated by an ensemble is estimated by an additional, second-level model. This is closely related to the stacked generalization [41] meta-learning technique [15]. Unlike the method in [19] that uses soft labels for knowledge distillation, we train a network to filter out bad and unreliable labels.

Decontaminating noisy labels from training data was explored in [2, 4]. Those approaches assume no prior knowledge about the training set labels, whereas in our case the labels are generated for initially unlabeled data by an ensemble. In this work we leverage the degree of consensus between ensemble models as additional information to assess labels quality.

We demonstrate the effectiveness of the proposed technique on the example of semantic segmentation [17]. Prior work on not fully supervised semantic segmentation include weakly and partially supervised techniques [22, 30, 39], self-supervision [42] and ensemble knowledge transfer [29]. To the best of our knowledge, this is the first work that performs semi-supervised training of semantic segmentation model on auto-annotated unlabeled data, generated by ensembles with quality filtering.

3. Annotation quality prediction

Let $f : \mathbb{X} \to \mathbb{Y}$ be the target model to be trained. We use an ensemble of models $e = (f_1, \ldots, f_k), f_j : \mathbb{X} \to \mathbb{Y}$, to automatically label the unlabeled data for training the target model f. Models in e are trained on a labeled data set $\mathbb{S} = \{(x^{(i)}, \bar{y}^{(i)})\}, x^{(i)} \in \mathbb{X}, \bar{y}^{(i)} \in \mathbb{Y}$.

We use the ensemble to generate labels for the large unlabeled set $\mathbb{U} = \{x^{(i)}\}$ in the following way:

- Run $e : \mathbb{X} \to \mathbb{Y}^k$ on $\mathbb{U}$ to generate vectors of labels $\mathbb{L} = \{(y_1^{(i)}, \ldots, y_k^{(i)})\}$.

- Apply a fusing function $g : \mathbb{Y}^k \to \mathbb{Y}$ to combine the ensemble labels into a single label: $\widehat{\mathbb{L}} = \{\hat{y}^{(i)}\} = g(\mathbb{L}) = (g \circ e)(\mathbb{U})$

The g function can be implemented by a plethora of ensemble fusion methods [32]. It can either generate a novel labeling $\hat{y}$ from ensemble outputs $(y_1, \ldots, y_k)$, or select one of them, i.e. $\hat{y} = g(y_1, \ldots, y_k) = y_j, j \in \{1, 2, \ldots, k\}$.

We can then train the target model on the generated labeled set $\mathbb{T} = \{\mathbb{U}, \widehat{\mathbb{L}}\} = \{(x^{(i)}, \hat{y}^{(i)})\}$. In a general case, some of the automatically generated labels in $\widehat{\mathbb{L}}$ are expected to be wrong. Therefore, a supposedly better approach would be to remove the corresponding data samples from $\mathbb{T}$.

To filter out bad labels, one can use an ad-hoc method. For instance, if ensemble models reports a confidence level associated with the output, it can be used to filter out the low quality annotations. Such filtering is quite simple in some scenarios, e.g. for object/image classification tasks, while in others, e.g. object detection, it is becoming less trivial. The confidence associated with each detected object does not provide any information on the likelihood of missing an object. For example, to deal with this issue in keypoint and object detections, [31] filter out false positives by thresholding the confidence score to get the average number of annotated instances per unlabeled image roughly equal to the average number of instances per labeled image.

In a more principled way, we propose to train a function q, that predicts the quality of the labeling generated by an ensemble, based on the degree of consensus between ensemble models. Function $q : \mathbb{Y}^k \to \{0, 1\}$ receives the ensemble output $(y_1, \ldots, y_k)$ and generates a quality score in $\{0, 1\}$. This is similar to Wolpert's ensemble stacking approach [41], but instead of merging ensemble outputs to generate a fused labeling, the q function predicts the labeling quality for a fixed fusor g. The function q is trained using a labeled data set $\mathbb{Q} = \{(x^{(i)}, \bar{y}^{(i)})\}$. For a data sample $(x, \bar{y}) \in \mathbb{Q}$, the input for the q model is the ensemble output $e(x)$ and the ground truth is the indicator function $\mathbb{1}_{(g \circ e)(x) = \bar{y}}$.

We then use q to filter the auto-annotated set $\mathbb{T}$ by discarding data samples with low predicted annotation quality to yield a refined labeled data set $\mathbb{T}^* = \{(x, \hat{y}) \in \mathbb{T} \mid q(e(x)) = 1\}$

4. Training with auto-annotations and the EM algorithm

In this section, we step back to perform a more principled analysis of the auto-annotation procedure, providing a mathematical justification to the benefit of using auto-

annotated data from an optimization process point of view. We extend our approach by viewing the auto-annotation procedure as an iterative process, in which the following steps are repeated several times.

1. Start from a manually labeled data set $\mathbb{S}$.

2. Train an ensemble of models on the available data set.

3. Using the ensemble, obtain auto-annotations on an unlabeled data set.

4. Apply quality filtering to remove poorly annotated data, thus improving the quality of the auto-annotations.

5. Add the auto-annotations to the available data set.

6. Repeat steps 2-5 several times to improve model's performance, until performance is stabilized.

In this paper, we consider a special case of this process, analyzing the model's improvement after one iteration of the steps above. To make the discussion in this section more concrete, we concentrate on a classification problem.

We perform our analysis under a semi-supervised framework. This is similar, in concept, to the analysis in [1]. In this scheme, using the notations of the previous section, we are given a set of labeled samples $\mathbb{S} = \{(x^{(i)}, \bar{y}^{(i)})\}$ and a set of unlabeled samples $\mathbb{U} = \{x^{(t)}\}$. Our goal is to fit a model $p_{\boldsymbol{\theta}}(x, y)$ to the data, characterized by a set of parameters $\boldsymbol{\theta}$. In order to do that, we would like to optimize the log likelihood function $\ell(\boldsymbol{\theta})$, defined via

$$\sum_{(x^{(i)}, \bar{y}^{(i)}) \in \mathbb{S}} \log p\left(x^{(i)}, \bar{y}^{(i)}; \boldsymbol{\theta}\right) + \sum_{x^{(t)} \in \mathbb{U}} \log p\left(x^{(t)}; \boldsymbol{\theta}\right).$$
(1)

If all our data was labeled, we could have approached this optimization problem using methods such as gradient descent. However, part of our data is unlabeled, resulting in two types of parameters we need to fit: The parameters $\boldsymbol{\theta}$ and the missing labels for the data in $\mathbb{U}$. Since in this discussion we focus on a classification problem, we can view these missing labels as class probability vectors, associated with each $x \in \mathbb{U}$. Thus, we would like to find both $\boldsymbol{\theta}$ and the mentioned probability vectors.

In order to do that, we can utilize a well-known likelihood maximization method: Expectation Maximization (EM) algorithm [12], which is an iterative approach that converges to a local maximum. We first obtain an initial estimation for $\boldsymbol{\theta}$, by fitting the model (in our case, training the network) using only the labeled data. Afterwards, each iteration of EM consists of two steps:

• In the E-step, the posterior probabilities of each class $1 \leq c \leq C$ (where C is the total number of classes) are estimated for each sample $x^{(t)} \in \mathbb{U}$, using the model and its parameters. The posterior probability vector is of the form

$$\left(p\left(\hat{y}^{(t)} = 1 \middle| x^{(t)}; \boldsymbol{\theta}\right), \ldots, p\left(\hat{y}^{(t)} = C \middle| x^{(t)}; \boldsymbol{\theta}\right) \right).$$
(2)

This step is performed by activating the model on a sample and inferring the probabilities of each class.

• In the M-step, we optimize a lower bound of the likelihood function w.r.t. the parameters of the model (this is a well known lower bound. See, for example, section 2 in [27]). Thus, $\boldsymbol{\theta}$ equals to argmax (over $\boldsymbol{\theta}$) of

$$\sum_{x^{(m)}} \sum_{c=1}^{C} p\left(y = c \middle| x^{(m)}; \boldsymbol{\theta}\right) \log p\left(x^{(m)}, y = c; \boldsymbol{\theta}\right).$$
(3)

In the summation above, $x^{(m)}$ is summed over all the elements in $\mathbb{U}$ and the first coordinate of the elements in $\mathbb{S}$. For the latter case, y is chosen to be $\bar{y}^{(m)}$. For the former case, y enumerates over all possible classes.

This step corresponds to training the model on the inferred posterior probabilities. In the case where the model is a neural network, applying the maximization step is equivalent to training the network, using the cross-entropy loss.

In our method, we add an additional intermediate classification phase (C-step) to the EM algorithm, which we refer to as *auto-annotation*. After the expectation phase, a per-class probability exists for each sample. We leverage these probabilities by modifying the posteriors via taking the most probable class, i.e. $p\left(y = c \middle| x^{(m)}; \boldsymbol{\theta}\right) = 1$ if c is the class with the largest posterior probability, and otherwise it equals 0.

To conclude, we first train a model on the labeled samples, leading to an initial set of parameters $\boldsymbol{\theta}$. Then, we apply the first two steps of our revised EM algorithm. We apply the trained model on unlabeled samples (E-step), and then we classify them to produce auto-annotations for the unlabeled data (C-step). Finally, the M-step consists of retraining the model, this time using labeled samples along with newly auto-annotated ones. Note that, while we use the auto-annotated labels for the unlabeled samples, we keep the original annotations for the labeled ones.

Clearly, a satisfying convergence of the described approach relies on the accuracy of the auto-annotations themselves. When the auto-annotations are too noisy, the model is not expected to improve its generalization [16].

To reduce the noise caused by false auto-annotations, and improve the convergence, we propose to use two means. The first is to strengthen the model itself by averaging an

ensemble of models, as suggested in [34, 28]. Furthermore, as described in the previous section, we train an additional classifier q aimed to predict false auto-annotations and eliminate them from the retraining process.

5. Semi-supervised semantic segmentation with auto-annotation quality prediction

To demonstrate the proposed approach, we implement a semi-supervised training of a semantic segmentation model using auto-annotation with ensemble of models and quality filtering. We chose the task of semantic segmentation since creating manual annotations for this task is extremely labor intensive, resulting in a relatively limited amount of such annotations. Therefore, highly accurate auto-annotations could be especially useful for this task.

To populate the ensemble we use both multiple models and data augmentation, following the data distillation method presented in [31]. First, we train the same model multiple (three) times using different parameters initialization and training samples reshuffling. In addition, each model is fed six augmented versions of the input image: two horizontal flips × three scales (x0.5, x1.0, and x1.5) . This effectively corresponds to an ensemble of size 18.

In the experiments described in the next section we merge ensemble results into a single label using a simple softmax averaging [34, 38, 24, 28]. We refer to a collection of such labels as unfiltered auto annotations. The fusing function g is defined as

$$g(\sigma_1, \ldots, \sigma_k) = \operatorname*{argmax}_{i \in [1, \ldots, C]} \sum_{j=1}^{k} \sigma_j, \qquad (4)$$

where C is the number of classes in the semantic segmentation model, and $\sigma_j \in \mathbb{R}^C$ is a softmax class probability vector generated by the j-th model of an ensemble of size k.

Interestingly, since semantic segmentation models can be trained on a partially labeled image, we do not necessarily need to accept or discard the image labeling as a whole. Instead, we can do it selectively, by making a decision per pixel. Pixels with unreliable labeling are marked as a special "ignore" class in the labels mask and do not contribute to the gradient back-propagation during the training.

In addition, if the segmentation model generates not only the class labels, but also class probabilities, we can use those as inputs to the quality filter q. Here we implement q using a convolutional neural network (CNN) that receives $k \times C$ input channels - $(\sigma_1, \ldots, \sigma_k)$ and outputs a quality mask in $\{0, 1\}$.

The network has four hidden conv-ReLU layers. The first layer has 40 output channels while the rest of them have 20. This is followed by the output conv-sigmoid layer with one output channel.

6. Experiments

In this section we demonstrate the strength of the proposed quality prediction approach. We show that a model trained on a combination of manually labeled data and quality filtered auto-annotations achieves better performance than a model trained on a combination of manually labeled data and unfiltered auto annotations. In fact, we show that quality filtering allows replacing significant amount of manually-annotated images by auto-annotated images, without any degradation in accuracy.

6.1. Training procedure

The experiments are performed with DeepLab network [1] [9], a state of the art network for semantic segmentation. The training protocols involve 90000 iterations, batch size of 12, learning rate of 0.01, output stride of 16, and decoder output stride of 4. We follow the same cropping convention as recommended in [8], and use crop size of 769 during training. Finally, we use Xception65 [10] pretrained on ImageNet [33] as a backbone. The evaluation results, both on the validation and the test sets are obtained using single scale and output stride of 16.

6.2. Data set

We evaluate our approach on the pixel-level semantic labeling task of Cityscapes data set [11]. This data set has 19 semantic labels (and additional void labels that are not used for evaluation), and consists of 5000 images, which are split into training, validation and test sets of sizes 2975, 500, and 1525 respectively. We report *mIoU* (mean *IOU*) scores and also *IoU* scores of each one of the 19 classes following the Cityscapes definitions. Results are reported on the validation and the test sets.

Our experiments involve several splits of data. In all the splits below, all the 19 classes are represented.

- **Cityscapes full training set** - 2975 labeled images that form the Cityscapes training data.

- **Cityscapes extra set** - 3600 images sampled from the 20000 coarsely annotated Cityscapes data set. We do not use any available labels from this data set, only the images themselves.

- **Cityscapes small training set** - 30% of the labeled images in the Cityscapes training set.

- **Cityscapes tiny training set** - 15% of the labeled images in the Cityscapes training set.

As the 19 classes are unbalanced by nature, when choosing subsets of data sets (training or unlabeled), it is important to ensure that smaller classes are well represented. To this

---

[1] https://github.com/tensorflow/models/tree/master/research/deeplab

Figure 1. Rows - top to bottom: (1) Original image, (2) unfiltered auto-annotations, (3) quality filtered auto-annotations, (4) ground truth annotation. Columns: three different examples. Black pixels in the fourth row represent void classes that are not counted as part of the 19 classes and do not contribute to the *mIoU* score. Black pixels in the third row represent pixels that are masked out by the quality filter.

end we utilize the following automatic approach: For the labeled data set, we choose subsets (30% and 15%, in our experiments) based on automatic filtering of the manual labels (choosing only images in which the number of labeled pixels of certain classes is high enough). For the unlabeled set, we first generate auto-annotations via ensemble of models, and then apply the same selection logic as for the labeled set, using auto-annotations instead of manual labels.

6.3. Training with and without quality prediction

In this experiment we show that a model trained on manually labeled data + quality filtered auto-annotations outperforms a model trained on manually labeled data + unfiltered auto annotations. The former model surpasses the latter by 0.8% in *mIoU*.

The experiment, whose results are summarized in table 1, consists of the following steps: We first train the network on the Cityscapes tiny training set 3 times, and evaluate the network's performance (the best accuracy among the 3 is indicated in the first column in the table). We then produce unfiltered auto-annotations for the Cityscapes extra

set by fusing ensemble results following equation (4). The ensemble consists of the 3 trained models together with 6 augmentations associated with each model, effectively corresponds to an ensemble of size 18 as described in the previous section.

We proceed by performing two additional trainings: first, we train the network on a data set which consists of the Cityscapes tiny training set and the unfiltered auto-annotations described above. The network's performance is indicated in the second column of the table. Next, we refine the auto-annotations by applying the quality filter, described in section 3. The quality filter is trained on the Cityscapes tiny training set. Finally, we train the network on a data set which consists of the Cityscapes tiny training set and the quality filtered auto-annotations. The network's performance is indicated in the third column of the table.

As shown in the table, adding unfiltered auto-annotated data improves the trained model accuracy by 1.5%. Adding quality filtering improves the performance by additional 0.8%, which leads to overall improvement of 2.3%.

We chose the tiny training set for this experiment, as in

Manual annotations only	Manual + unfiltered auto-annotations	Manual + quality filtered auto-annotations
74.0%	75.5%	76.3%

Table 1. *mIoU* on the validation set of the experiment depicted in section 6.3.

this scenario the auto-annotated data forms large portion of the training data. In fact, the auto-annotated data forms roughly 90% of the training data used for the training procedures depicted in the second and third columns of table 1. This mimics the real world scenario in which available unlabeled data sets are much larger than their manually-labeled counterparts. Our approach shows that such data sets can be utilized effectively to improve network's performance.

6.4. Training with and without quality prediction – per class examination

The experiment from the previous section shows a clear improvement in performance when filtered auto-annotated data is added to the training data. In this section, we perform a class-wise IoU examination, obtained via the experiment above. Based on the results presented in table 2, we arrive to the following conclusions:

1. A model trained on manually labeled data + quality filtered auto-annotations outperforms a model trained on manually labeled data + unfiltered auto annotations on each one of the 19 classes.

2. A model trained on manually labeled data + quality filtered auto-annotations performs significantly better on underrepresented classes, compared to a model trained on manually labeled data only, adding up to ~20% to the IoU in such classes.

For the first conclusion, one can see that for each class in table 2, the values in the fifth column surpass the values in the fourth column. This is a strong demonstration of the benefit of the quality filter. The largest gain is demonstrated for the rarest classes (lower rows), with 4% gain to the most rare class (train).

The second conclusion is achieved by comparing the third and the fifth columns in table 2. Classes in which the performance of the model trained on manual annotations + quality filtered auto-annotations surpasses the performance of the model trained on manual annotations only are highlighted in green. Those that perform the same are highlighted in yellow. One can see that in 14 out of 19 classes, the performance of the former model was at least as good as the latter. In 11 out of these classes, the performance improved. The most significant improvement is obtained in under-represented classes like train (19.7% improvement), bus (7.8% improvement), truck (9.6% improvement) and

fence (4.9% improvement). One can also see an improvement in the rarest 7 classes.

6.5. Training with quality prediction on additional data splits

Following the observed accuracy gain depicted in section 6.3, we experiment with manually-labeled training sets of various sizes. We show that adding quality filtered auto-annotated data improves the model performance in all cases, with the largest improvement achieved for the smallest manually-labeled training set.

Following the same procedure as in section 6.3, we train the model first on manually labeled data only, and then on manually labeled data + quality filtered auto-annotated data. We repeat the experiment 3 times by reducing the manually labeled training set from 100%, to 30% and, finally, to 15%.

Our results on the validation set are summarized in table 3. In table 4 we also present the accuracy measured on the test set for the models trained on the full manually-labeled training set. As can be seen in the table, we obtain an improvement of 0.6% when adding auto-annotated data to the Cityscapes full training set. The improvement increases even further, when the auto-annotated data is added to the small training set and the tiny training set, in which cases we get improvements of 2% and 2.3% respectively.

Moreover, the accuracy of the network trained on the Cityscapes small training set+auto-annotated data is 77%, which is identical to the accuracy of the network trained on Cityscapes full training set. This demonstrates that using our approach, with only 30% of the available manual annotations, we achieve the same accuracy levels as training with the entire manually annotated training set (saving 70% of manual annotations).

6.6. Performance of the quality filtering

In this section, we shed some light on the performance of the quality filter itself. In the beginning of the section, we examine retention rate (percentage of pixels that were not masked out by the filter) and auto-annotation precision (both filtered and unfiltered). In the end of the section, we visualize quality filtered auto-annotations.

Consider the results in table 5. The unfiltered and quality filtered auto-annotations of the Cityscapes validation set are used to measure the unfiltered and quality filtered auto-annotation precision respectively. Precision rates increase for all classes when quality filter is applied, showing that our filter indeed learns to identify wrong annotations and to mask them out. The improvement in performance, measured in IoU, of a model trained on manually labeled data + quality filtered auto-annotations versus a model trained on manually labeled data + unfiltered auto annotations is reported in the fifth column. The improvement in IoU is correlated with the improvement in precision (correlation

Class	Num of ground truth occurrences	Manual annotations only	Manual + unfiltered auto-annotations	Manual + quality filtered auto-annotations
building	500	91.7%	91.6%	91.7%
sidewalk	499	81.8%	82.4%	83.1%
pole	499	63.3%	61.2%	62.2%
road	498	97.7%	97.7%	97.8%
vegetation	493	92.1%	92.0%	92.1%
traffic sign	487	76.5%	76.0%	76.5%
car	486	94.7%	93.7%	94.0%
sky	473	94.5%	94.9%	94.9%
person	453	81.0%	79.3%	79.8%
fence	394	54.1%	57.8%	59.0%
bicycle	392	75.0%	74.0%	74.8%
traffic light	385	67.6%	65.5%	65.9%
terrain	351	61.4%	63.2%	63.3%
wall	339	49.1%	49.6%	49.8%
rider	303	59.8%	59.7%	60.3%
truck	187	70.6%	78.3%	80.2%
motorcycle	178	61.4%	60.8%	63.1%
bus	161	79.2%	85.8%	87.0%
train	95	55.2%	70.9%	74.9%

Table 2. IoU per class on the validation set, as measured in the experiments reported in table 1. The first column lists the 19 classes in the Cityscapes data set. The second column indicates the number of images in the validation set (out of 500) that contain the specified class. For example, train is the rarest class, available only in 95 of the images, while building is the most common class, available in all the 500 images. Classes are ordered from the most common (first row) to the most rare (last row). Classes in which the performance of the model trained on manual annotations + quality filtered auto-annotations surpasses the performance of the model trained on manual annotations only are highlighted in green. Those that perform the same are highlighted in yellow.

Size of the manually labeled training set	Manually labeled data only	Manual + quality filtered auto-annotations
Full set (100%)	77.0%	77.6%
Small set (30%)	75.0%	77.0%
Tiny set (15%)	74.0%	76.3%

Table 3. mIoUs on the validation set. The data sets in the left column are described in section 6.2. In the middle column, the models were trained on the corresponding manually labeled training set only. In the right column, the models were trained on the corresponding manually labeled set+ quality filtered auto-annotated images from the Cityscapes extra set.

Manually labeled training set	Manually labeled data only	Manual + quality filtered auto-annotations
Full set (100%)	76.83%	77.29%

Table 4. mIoUs on the test set.

coefficient of 0.48).

Retention rates are indicated in the right-most column in the table. These are the percentage of pixels in non-void locations (based on ground truth) that are retained after applying the quality filter. Overall, 96.2% of the pixels are retained. These results, in combination with the results in table 1, show that by applying our quality filter we can improve model's performance while retaining most of the pixels (only 3.8% of the pixels are masked out).

Let us visually inspect unfiltered ensemble annotations and the auto-annotations obtained by applying our quality filter on them. The quality filter has been trained on Cityscapes tiny training set using the models that correspond to the bottom row in table 3.

Consider the examples in figure 1. The top row depicts three images from the Cityscapes validation set. The corresponding manual labeling (the ground truth) appear in the bottom row. The second row depicts the unfiltered ensemble labeling. Row number three contains the same figures, after undergoing quality filtering. Pixels that are filtered out by the quality filter appear in black.

One can see that the quality filter identified correctly a substantial amount of errors. For example, a major part of the car's hood is masked out by the filter. The filter also correctly masked out the misclassified parts of the wall in the left and middle examples – see the purple area (second row), that is correctly filtered out (third row), as can be verified by the ground truth (last row). In the right column example, the misclassified area on the left car is correctly masked out by

Class	Annotation precision			Diff between IoU of filtered and unfiltered	retention
	filtered auto-annotations	unfiltered auto-annotations	diff		
building	97.0%	95.7%	1.3%	0.1%	97.5%
sidewalk	94.5%	89.9%	4.7%	0.7%	86.5%
pole	90.8%	81.3%	9.6%	1.0%	70.6%
road	99.7%	99.1%	0.5%	0.1%	97.9%
vegetation	96.5%	95.4%	1.1%	0.1%	97.5%
traffic sign	95.3%	92.4%	2.8%	0.5%	89.5%
car	98.0%	97.4%	0.6%	0.3%	97.7%
sky	98.1%	97.3%	0.8%	0.0%	98.1%
person	93.2%	90.2%	3.0%	0.5%	92.3%
fence	87.6%	81.9%	5.7%	1.2%	82.9%
bicycle	91.2%	85.4%	5.7%	0.8%	84.7%
traffic light	90.5%	86.7%	3.8%	0.4%	86.6%
terrain	88.0%	83.8%	4.2%	0.1%	89.0%
wall	85.8%	80.4%	5.4%	0.2%	90.4%
rider	84.3%	78.2%	6.0%	0.6%	83.7%
truck	96.3%	90.2%	6.1%	1.9%	90.8%
motorcycle	93.1%	86.3%	6.8%	2.3%	82.2%
bus	92.4%	90.4%	2.1%	1.2%	95.8%
train	96.2%	90.9%	5.3%	4.0%	86.9%

Table 5. Auto-annotation precision, improvement in IoU, and retention rates on the validation set. Columns: (1) 19 classes in the Cityscapes data set. (2) Annotation precision of unfiltered auto-annotations on the cityscapes validation set (precision is relative to the ground truth). (3) Annotation precision of quality filtered (according to section 6.3) auto-annotations on the validation set. (4) Differences in precision values (filtered minus unfiltered). (5) Model IoU improvement - trained on manually labeled data + quality filtered auto-annotations vs. a model trained on manually labeled data + unfiltered auto annotations - this is the difference between the last two columns from table 2). (6) Filter retention rates - the percentage of pixels in non-void locations (based on ground truth) that are retained after applying the quality filter. Overall, 96.2% of the pixels are retained.

the quality filter, as well as an area on the right wall. In all three examples, additional more subtle masked out areas can be found in the vicinity of the trees and other small objects. Overall, the filter managed to mask out considerable amount of misclassified pixels, creating more accurate auto-annotated data, which in turn led to an improvement of 0.8% in the trained model accuracy, as shown in table 1.

7. Discussion

The accuracy gain due to the proposed quality filtering is higher for "less experienced" teacher ensembles: the more mistakes the teacher makes, the more errors the quality filter can fix. There is a trade-off between the requested auto-annotation quality and the amount of generated training data. While in the conducted experiments we used a fixed quality threshold, we would like to explore the influence of the quality-quantity trade-off on the trained model accuracy.

We can further enhance the training process by iterating over the "train the teachers ensemble", "auto-annotate the unlabeled data", and "train the target model" steps, as discussed in section 4.

Another interesting research direction is using ensemble stacking (instead of softmax averaging) for the fusing function g and building a joint multi-task model for g and q together.

8. Conclusions

We propose a generic method for quality prediction of automatic annotations generated by an ensemble of models. We adapt the proposed approach to semantic segmentation by doing label quality filtering at pixel level. We show that refining the auto-annotated training set by discarding data samples with low predicted label quality improves the trained model accuracy. We demonstrate that the performance of the state-of-the-art model can be achieved by training it with only a fraction (30%) of the original manually labeled data set, and replacing the rest with the auto-annotated, quality filtered labels.

References

[1] M.-R. Amini and P. Gallinari. Semi-supervised logistic regression. *ECAI*, pages 390–394, 2002. 3

[2] R. Barandela and E. Gasca. Decontamination of training samples for supervised pattern recognition methods. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 621–630. Springer, 2000. 2

[3] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 2

[4] C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167, 1999. 2

[5] C. Bucila, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 535–541, 2006. 1, 2

[6] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 137–144, New York, NY, USA, 2004. ACM. 2

[7] H. Chen, S. Lundberg, and S.-I. Lee. Checkpoint ensembles: Ensemble methods from a single training process. *arXiv preprint arXiv:1710.03282*, 2017. 2

[8] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv: 1706.05587*, 2017. 4

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *ECCV*, 2018. 4

[10] F. Chollet. Deep learning with depthwise separable convolutions. *CVPR*, 2017. 4

[11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CVPR*, 2016. 4

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood for incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–38, 1977. 3

[13] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000. 1

[14] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, 2014. 2

[15] R. P. Duin and D. M. Tax. Experiments with classifier combining rules. In *International Workshop on Multiple Classifier Systems*, pages 16–29. Springer, 2000. 2

[16] D. Flatow and D. Penner. On the robustness of convnets to training on noisy labels. http://cs231n.stanford.edu/reports/flatow_penner_report, 2017. 3

[17] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017. 2

[18] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, Oct 1990. 1

[19] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*, 2015. 2

[20] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get M for free. *ICLR*, 2017. 2

[21] A. Jurek, Y. Bi, S. Wu, and C. Nugent. A survey of commonly used ensemble-based classification techniques. *The Knowledge Engineering Review*, 29(5):551–581, 2014. 1

[22] A. Khoreva, R. Benenson, J. H. Hosang, M. Hein, and B. Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *CVPR*, volume 1, page 3, 2017. 2

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[24] L. I. Kuncheva. An application of owa operators to the aggregation of multiple classification decisions. In *The ordered weighted averaging operators*, pages 330–343. Springer, 1997. 4

[25] L. I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004. 1

[26] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016. 2

[27] T. Minka. Expectation-maximization as lower bound maximization. *Tutorial available at http://research.microsoft.com/ minka/papers/em.html*, 1999. 3

[28] U. Naftaly, N. Intrator, and D. Horn. Optimal ensemble averaging of neural networks. *Network: Computation in Neural Systems*, 8(3):283–296, 1997. 4

[29] I. Nigam, C. Huang, and D. Ramanan. Ensemble knowledge transfer for semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1499–1508. IEEE, 2018. 2

[30] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1742–1750, 2015. 2

[31] I. Radosavovic, P. Dollár, R. B. Girshick, G. Gkioxari, and K. He. Data distillation: Towards omni-supervised learning. *CVPR*, 2018. 2, 4

[32] M. Re and G. Valentini. Ensemble methods: a review. 2011. 1, 2

[33] 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*, 2015. 4

[34] R. Schapire. The strenght of weak learnability,(1990) machine learning 5 197-227; y. freud. *Boosting a weak learning algorithm by majority*, pages 256–285, 1995. 4

[35] H. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965. 2

[36] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 2

[37] G. Valentini and F. Masulli. Ensembles of learning machines. In *Italian Workshop on Neural Nets*, pages 3–20. Springer, 2002. 1

[38] M. Van Breukelen, R. Duin, D. Tax, and J. Den Hartog. Combining classifiers for the recognition of handwritten digits. In *1st IAPR TC1 Workshop on Statistical Techniques in Pattern Recognition, Prague, Czech Republic*, pages 13–18, 1997. 4

[39] A. Vezhnevets and J. M. Buhmann. Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning. 2010. 2

[40] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001. 2

[41] D. H. Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992. 2

[42] X. Zhan, Z. Liu, P. Luo, X. Tang, and C. C. Loy. Mix-and-match tuning for self-supervised semantic segmentation. *arXiv preprint arXiv:1712.00661*, 2017. 2

[43] Y. Zhou and S. Goldman. Democratic co-learning. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 594–602, Nov 2004. 1

[44] Z.-H. Zhou. When semi-supervised learning meets ensemble learning. In *International Workshop on Multiple Classifier Systems*, pages 529–538. Springer, 2009. 2

[45] X. Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4, 2006. 2