Model Composition: Can Multiple Neural Networks Be Combined into a Single Network Using Only Unlabeled Data?

Amin Banitalebi-Dehkordi[1]
amin.banitalebi@huawei.com

Xinyu Kang[2]
xinyu.kang@alumni.ubc.ca

Yong Zhang[1]
yong.zhang3@huawei.com

[1] Huawei Technologies Canada Co., Ltd.

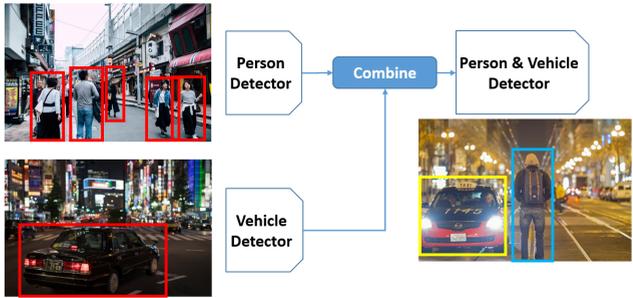[2] University of British Columbia, Vancouver, Canada

Abstract

The diversity of deep learning applications, datasets, and neural network architectures necessitates a careful selection of the architecture and data that match best to a target application. As an attempt to mitigate this dilemma, this paper investigates the idea of combining multiple trained neural networks using unlabeled data. In addition, combining multiple models into one can speed up the inference, result in stronger, more capable models, and allows us to select efficient device-friendly target network architectures. To this end, the proposed method makes use of generation, filtering, and aggregation of reliable pseudo-labels collected from unlabeled data. Our method supports using an arbitrary number of input models with arbitrary architectures and categories. Extensive performance evaluations demonstrated that our method is very effective. For example, for the task of object detection and without using any ground-truth labels, an EfficientDet-D0 trained on Pascal-VOC and an EfficientDet-D1 trained on COCO, can be combined to a RetinaNet-ResNet50 model, with a similar mAP as the supervised training. If fine-tuned in a semi-supervised setting, the combined model achieves +18.6%, +12.6%, and +8.1% mAP improvements over supervised training with 1%, 5%, and 10% of labels. Code is released as supplementary [2].

1 Introduction

Deep learning has enabled achieving outstanding results on a wide range of applications in computer vision and image processing [5, 27]. However, the diversity of datasets and neural network architectures necessitates a careful selection of model architecture and training data that match best to the target application. Often times, for a same task, many models are available. These models might be trained on different datasets, or might come in different capacities, architectures, or even bit precisions.

Motivation: A natural question that arises in this case, is whether we can combine the neural networks so that one combined network can perform the same task as several input networks. Fig. 1 shows an example, where two input object detection models to detect 'person' and 'vehicle' are combined in one model. The benefits of combining models include: a) possible latency improvements due to running one inference as opposed to many, b) in case input models cover partially overlapping or non-overlapping classes/categories, one can build a stronger model with the union of the classes/categories through model composition

Figure 1: An example of the proposed model composition approach for object detection. Two input models for detecting 'person' and 'vehicle' categories are combined to create one model that does both.

(i.e. merging models' skills as in Fig. 1), and c) for applications involving model deployment, e.g. for cloud services providers, it can reduce the deployment frequency/load.

Challenges: Creating a combined model from several input models is a challenging task. First, depending on the target task, the output model may need to have a specific architecture, and not necessarily one that is dictated by the input models. The input models themselves also might have different architectures. Second, in case input models are provided by users of a cloud system, or by different model creators/clients, the individual model owners would likely prefer not to share their training data, labels, not even weights or code. A privacy preserving model composition approach should rely on only a minimum amount of information from the model creators. Third, input models may have only partially overlapping or disjoint class categories. This imposes a major challenge when combining the individual models.

Existing methods: The existing solutions are mostly based on techniques such as knowledge distillation [6, 15, 34] or ensembling [35], which may be useful when classes/categories are identical and labeled data are available, but not for the case of arbitrary classes/categories with only unlabeled data. More details regarding the existing approaches are provided in section 2. In summary, to the best of our knowledge, the existing methods do not fully address the three challenges mentioned above.

Our contributions: In this paper, we propose a simple yet effective method to address the model composition of neural networks. Our method supports combination of an arbitrary number of networks with arbitrary architectures. To train a combined model, we leverage the abundance of unlabeled data and having labels or original training data of the input models is not a requirement. However, if any labeled data are available, the algorithm uses them to further boost the performance of the output model. Furthermore, we put no restrictions on the type and number of object categories of the input models. We demonstrate the effectiveness of our method through an extensive set of experiments for the task of object detection.

2   Related works

Related to our work are the following approaches:
Network Ensembling: Ensembling is a common way of aggregating the predictions of more than one models. Ensembling strategies are well explored in the literature [9, 28, 35]. Simplest ways could be naive averaging of predictions.
Architectural Combination: These methods create new architectures from the input models. Adaptive Feeding (AF) [53] proposes to use simultaneously two small and large networks that are trained to perform a same task. A linear classifier decides which examples to go to the small or large model. Their goal was to improve the inference speed. In another work, [10] Unifying&Merging (U&M) proposes to design a new architecture based on existing input architectures, to support learning multiple tasks.
MultiTask Networks: MultiTask networks learn multiple tasks in one model [18, 25, 51]. Tasks train simultaneously, not by combining already trained individual networks.

Incremental Learning (IL): Gradually adding new categories while trying to limit the catastrophic forgetting [22].

Dataset Merging: [23] Dataset merging is closest work to our study. It proposes to combine datasets by filling the missing annotations of non-overlapping categories.

While existing works have some correlation to the problem we study, none of them directly addresses this problem. Specifically, our proposed method combines neural networks of same tasks (e.g. classification, detection, etc.) using unlabeled data. If any labels are available, it will use them to further boost the performance. On the other hand, most existing methods mentioned above require labels to be available. In addition, our method supports having overlapping or disjoint target label categories while existing methods such as multi-task networks, U&M, AF, ensembling, or Dataset Merging only support homogeneous categories. Moreover, our method is architecture agnostic. In contrast, AF, MultiTask, IL, and Dataset Merging, don't support arbitrary input model architectures.

It is also worth noting that most existing methods require access to input model weights or code, to construct a combined model. Methods such as multi-tasking, IL, Dataset Merging or U&M need full access to input models, in order to design a new combined architecture. Our method only requires an inference API, and thus treats the input models as black boxes, which in turn leads to a better privacy protection for the clients.

3 Model composition strategy

This section provides details regarding the model composition method we use in this paper. Fig. 2 shows the inputs and outputs of this process. In addition, Fig. 3 shows a flow-diagram of different steps within this method. As observed from Fig. 3, a number of models are provided as inputs. We then collect the predictions of these models over an unlabeled set of images. These predictions are filtered and aggregated to form a set of generated pseudo-labels, which is later used to train the output model M. If any labeled data are available, M will be fine-tuned on them. Algorithm 1 shows a break-down of these different steps.

An embodiment of how our method would be implemented for usage in a cloud services provider platform is demonstrated in Fig. 4. As shown in this figure, in the context of a cloud services provider, model composition can be leveraged for: less frequent model/data deployment/transfer, building stronger models, faster overall inference, empowering the model markets, and encouraging users to share their models in an incentive sharing strategy.

3.1 Filtering pseudo-labels

Since the input model predictions are not always perfectly accurate, the generated pseudo-labels will be noisy, and therefore less reliable. These training examples could have an adverse impact on the training of the output model. We filter out such kind of examples, by employing an entropy-based thresholding mechanism.
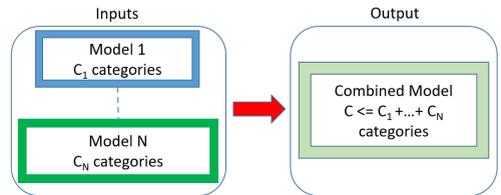

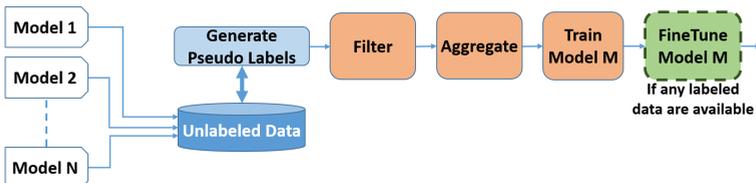
Figure 2: Inputs and output of model composition.

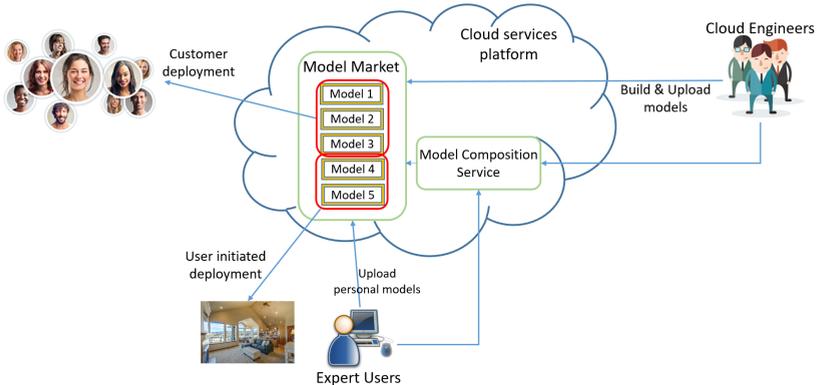Figure 3: Flow-diagram of the model composition method used in our study.

Figure 4: An embodiment of how our model composition strategy would be implemented in a cloud services platform. Models shown as groups are combined with each other.

For a given input x and a network function f such that $p = f(x)$, the entropy is given by $H(f,x) = -\sum_i p_{f_i} \log p_{f_i}$. An unreliable pseudo-label may be discarded if $H(f,x) > \tau$, for some threshold τ. Although entropy thresholding does not guarantee a perfect filtering of bad pseudo-labels, but in practice it works well and has been used as a confidence indicator for similar purposes [24, 26, 30]. Note that for some tasks such as object detection, models output a confidence score that can also be used for filtering bad pseudo-labels.

3.2 Aggregation of pseudo-labels

Next, in the pseudo-label aggregation phase, we employ a consensus based strategy, where the majority of the input models need to agree on a pseudo-label in order for it to qualify as a candidate and pass to the next step. Pseudo-label aggregation can be done in various ways such as unanimous (all models agree), affirmative (union of all predictions), consensus (majority voting), etc. [9]. Our experiments showed all these methods can be used with minor performance variations. We chose the consensus approach for the experiments since for combining a higher number of models, intuitively it makes more sense (See section 4 for a 10-model example). Note that for some tasks such as image classification, the aggregation will be a simple majority voting mechanism. For some other tasks such as object detection, it becomes more complicated due to the nature of the task. Here, we review our method of pseudo-label aggregation for object detection, which can also be extended to other similar tasks such as instance segmentation, tracking, etc.

Details of the pseudo-label aggregation strategy: Let $\overline{\mathcal{D}}$ denote the unlabeled dataset used. The input to the pseudo-label aggregation procedure is a list $\mathcal{S} = [\overline{\mathbf{y}}_1, ..., \overline{\mathbf{y}}_N]$, where each $\overline{\mathbf{y}}_i$ itself is a list of detections from an input model over all unlabeled training images in $\overline{\mathcal{D}}$. We then create a new list $\mathcal{S}_{im} = [p_1, ..., p_{|\overline{\mathcal{D}}|}]$ so that each p_i contains predictions of all models on one single image, and length of $\mathcal{S}_{im}$ is equal to number of all images in $\overline{\mathcal{D}}$.

Next, for each element p_i, we unite the predictions by their category names and the overlapping of their bounding boxes. If the overlapped area of any two elements in p_i is higher than a certain threshold, and meanwhile if these two elements are of the same category, then they are treated as detections of the same object, which are further grouped together into a sub-list: $\overline{p}_i = \{\overline{p}_{ij}\}$. Subsequently, we decide whether to keep each element $\overline{p}_{ij}$ depending on the number of unique models with predictions included in $\overline{p}_{ij}$, denoted by K_{ij}. In the most strict case, $\overline{p}_{ij}$ is kept in the list only when $K_{ij} = N_{ij}$, where $N_{ij} \leq N$ is the maximum number of models that may predict the object category corresponding to $\overline{p}_{ij}$; If we want a majority voting, then $\overline{p}_{ij}$ is kept when $K_{ij} \geq N_{ij}/2$; If a simple stacking strategy is used,

---

Algorithm 1 Our Model Composition Approach

Inputs: Input models $\mathbf{M}_1, ..., \mathbf{M}_N$; unlabeled data $\overline{\mathcal{D}}$
Optional: Output model architecture; labeled data $\mathcal{D}$
Output: Combined model $\mathbf{M}$

  procedure MODELCOMPOSITION$(\mathbf{M}_1, ..., \mathbf{M}_N, \overline{\mathcal{D}})$
    $\mathcal{S} \leftarrow \emptyset$
    for $i \leftarrow 1, 2, ..., N$ **do**
      $\{\overline{\mathbf{y}}_i\} \leftarrow GeneratePseudoLabels(\mathbf{M}_i, \overline{\mathcal{D}})$
      $\{\overline{\mathbf{y}}_i\} \leftarrow FilterPseudoLabels(\{\overline{\mathbf{y}}_i\})$
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{\overline{\mathbf{y}}_i\}$
    $\overline{\mathcal{S}} \leftarrow AggregatePseudoLabels(\mathcal{S})$
    Initialize $\mathbf{M}$
    for $j \leftarrow 1, 2, ..., n_{epochs}$ **do**
      for $B \leftarrow GetBatch(\overline{\mathcal{D}}, \overline{\mathcal{S}})$ **do**
        $\mathbf{M} \leftarrow ApplyGradients(B, \mathbf{M})$

Algorithm 2 Our Pseudo-Label Aggregation

Inputs: Pseudo-labels $\mathcal{S}$ from N input models;
  unlabeled data $\overline{\mathcal{D}}$; aggregation strategy A
Output: Aggregated pseudo-labels list $\overline{\mathcal{S}}$

  procedure AGGREGATE$(\mathcal{S}, N, \overline{\mathcal{D}}, A)$
    $\mathcal{S}_{im} \leftarrow GroupByImages(\mathcal{S}, \overline{\mathcal{D}}); \quad \overline{\mathcal{S}} \leftarrow \emptyset$
    Let $\{p_1, p_2, ..., p_{|\overline{\mathcal{D}}|}\} \leftarrow \mathcal{S}_{im}$
    for p_i in $\{p_1, p_2, ..., p_{|\overline{\mathcal{D}}|}\}$ **do**
      $\overline{p}_{i_{new}} \leftarrow \emptyset$
      $\{\overline{p}_{ij}\} \leftarrow GroupByObject(p_i)$
      for $\overline{p}_{ij}$ in $\{\overline{p}_{ij}\}$ **do**
        Let $K_{ij} = |UniqueModels \in \overline{p}_{ij}|$
        $N_{ij} = |$Models supporting $\overline{p}_{ij}$ class$|$
        if A is 'unanimous' **then**
          delete $\overline{p}_{ij}$ if $K_{ij} < N_{ij}$
        else if A is 'consensus' **then**
          delete $\overline{p}_{ij}$ if $K_{ij} < N_{ij}/2$
        $\overline{p}_{ij_{soft}} \leftarrow$ SoftNMS$(\overline{p}_{ij})$
        $\overline{p}_{i_{new}} \leftarrow \overline{p}_{i_{new}} \cup \overline{p}_{ij_{soft}}$
      $\overline{\mathcal{S}} \leftarrow \overline{\mathcal{S}} \cup \overline{p}_{i_{new}}$

---

then $\overline{p}_{ij}$ is kept regardless of K_{ij}. At this point, each $\overline{p}_{ij}$ could still have several candidate detections for the same region. Processing all of them through the detection network is not only cumbersome but could also decrease the overall performance. Therefore, we applied soft non-maximum suppression (Soft-NMS) [8] to each $\overline{p}_{ij}$ to filter the predictions a second time. Algorithm 2 formally captures these steps and Figure 5 demonstrates an example.

Remark 1: $\overline{p}_{ij}$ for image i, represents a list of bounding boxes predicted on a particular object j, i.e. detections of a same object by different models. K_{ij} is the number of unique models in $\overline{p}_{ij}$. N_{ij} is the number of models that have the category of $\overline{p}_{ij}$ in their label set (i.e. number of models that actually have the capability of detecting that object category). As such, in general $K_{ij} \leq N_{ij} \leq N$. In an ideal case where all eligible models can detect an object ij, we will have $K_{ij} = N_{ij}$. If all input models have the same category label set, $N_{ij} = N$. In the case input models have different but overlapping categories (i.e. there is at least one category that is not supported by all models), for at least some i and j, $N_{ij} < N$. If all models have strictly different categories (no overlap), $K_{ij} = N_{ij} = 1$. And finally if some particular categories only belong to one model, for those categories $K_{ij} = N_{ij} = 1$.

Remark 2: The experiments in Section 4 contain various practical scenarios in which different aspects of our method are evaluated. Moreover, Figure 9 in supplementary materials [7] shows a scenario where 10 input models with a diverse count and type of object categories are combined. For example in this figure, 'teddy bear' is only in M_1, 'bicycle' is in M_1, M_3, 'potted plant' is in M_2, M_3, M_4, etc. In addition, we also explore in Section 4 the task of combining a face detection model with a mask detection one.
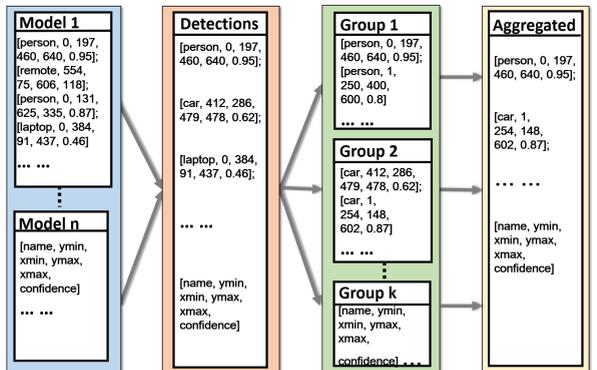

Figure 5: Pseudo-label aggregation scheme, an example.

3.3 Training pipeline

Once pseudo-labels are filtered and combined, they will be used to train the output model architecture. Any available labeled data will be used in a final fine-tuning stage to improve the performance. Note that pseudo-labels are generated from unlabeled data. This is because input model owners may only share an interface to their model API for inference, not necessarily the weights, code, architecture, training data, or labels, to protect their privacy. We treat the input models as black boxes. In other words, we only pass a set of arbitrary unlabeled images through them, and collect their predictions to use as pseudo-labels. This further allows us to choose an arbitrary architecture and size for the output model that combines the class categories of the input models. Consequently, our model composition method is agnostic to the training hyper-parameters of the input models such as various optimizers, learning rate schedules, batch sizes, etc.

It is also worth noting that this way of creating composite models can help with light domain shifts. As we see in section 4, input models trained on different datasets (for the same task) can still be effectively combined (even with different sets of categories). To what extent exactly our method can robustly support domain shifts remains out of the scope of this work, and we leave that as a future direction.

4 Experiment results and discussions

4.1 Experiments setup

Selected model architectures: We have selected the task of object detection as the main experiments task due to its importance and wide-spread usage in practical applications. That being said, we will also provide results on the task of image classification, as it is often used as a baseline experiment task. For object detection, we utilized the following architectures: EfficientDet-D0 [29], EfficientDet-D1 [29], and RetinaNet-ResNet-50 [21]. For the classification task, we used: ResNet-18 [14], ResNet-152[14], and DenseNet-121[16].

Datasets: We used three sets of benchmarking datasets for object detection: COCO [20], Pascal-VOC [11], and Open-Images-V5 [19] (referred by OID hereafter). For classification, we use Caltech-256 [13] and OID datasets.

Evaluation metrics: We follow the common practice by using the mean Average Precision, mAP @IoU=0.50:0.95, as the main metric to evaluate the performance of object detection models. We report top-1 accuracy for classification.

Training protocols and settings: We adopt code from [1] for the object detection experiments, and use the same training hyper-parameters with ImageNet [7] pre-trained backbones. We trained all the models using SGD with a momentum of 0.9. We increased the learning rate from $8e^{-3}$ to $8e^{-2}$ for the first epoch and then trained the remaining 300 epochs using a cosine decay rule with a moving average decay set at 0.9998. Soft NMS was utilized to filter the pseudo-label detections in our method. We used an IoU threshold of 0.5 and a confidence threshold of 0.001. For the classification experiments, models were trained for 200 epochs, using an in-house code-base. An SGD optimizer with momentum 0.9 was used, and learning rate was exponentially increased from 0 to 0.01 for the first 8 epochs and then annealed down exponentially to 0.0001 in the remaining epochs.

4.2 Object detection results

Our experiments are categorized in various scenarios, which are explained in this subsection. These scenarios cover various possible cases of input models' architectures, training data, and what kind of unlabeled data were used in our algorithm. Table 1 provides a summary of these scenarios. In Table 1, training data in each case is constructed from the training set

Experiment	Architecture	Model/Method	Train Set	Size
Scenario 1	EffDet-D0	input (supervised)	COCO subset 1	72K
	EffDet-D0	input (supervised)	COCO subset 2	66K
	EffDet-D0	input (supervised)	COCO subset 3	81K
	EffDet-D0	Upper-bound	COCO subsets union	89K
	EffDet-D0	ModelComp (Ours)	unlabeled COCO	118K
	EffDet-D0	ModelComp (Ours)	unlabeled OID*	1.9M
Scenario 2	EffDet-D0	input (supervised)	COCO	118K
	EffDet-D0	input (supervised)	VOC	17K
	EffDet-D0	Upper-bound	COCO+VOC	135K
	EffDet-D0	ModelComp (Ours)	unlabeled COCO+VOC	135K
	EffDet-D0	ModelComp (Ours)	unlabeled OID*	1.9M
Scenario 3	EffDet-D1	input (supervised)	COCO	118K
	EffDet-D0	input (supervised)	VOC	17K
	RetinaNet-R50	Upper-bound	COCO+VOC	135K
	RetinaNet-R50	ModelComp (Ours)	unlabeled COCO+VOC	135K
	RetinaNet-R50	ModelComp (Ours)	unlabeled OID*	1.9M
Scenario 4	EffDet-D0	10 inputs (supervised)	10 COCO partitions	≈12K each
	EffDet-D0	Upper-bound	COCO	118K
	EffDet-D0	ModelComp (Ours)	unlabeled COCO	118K
	EffDet-D0	ModelComp (Ours)	unlabeled OID	100K

Table 1: Data-splits and models for object detection. *We also evaluate on a 118K subset of OID.

Model skill	Categories supported
Transportation	person, bicycle, car, motorcycle, bus, truck, traffic light, fire hydrant, stop sign, parking meter
Sports	person, bicycle, frisbee, skis, snowboard, sports ball, skateboard, baseball bat, baseball glove, motorcycle
Home	person, bicycle, chair, couch, bed, dining table, skateboard, refrigerator, toilet, tv

Table 2: Object categories for input models of scenario 1.

of VOC, COCO, OID, or a subset of them (unlabeled). Validation sets are also built from the validation sets of VOC and COCO: a subset of COCO (union of input categories) for scenario 1, union of the val set of COCO & VOC for scenario 2 & 3, and val set of COCO for scenario 4. As such, validation sets may be different across different scenarios, but are the same within one scenario. Moreover, the class distributions of data for scenario 1 & 4 are shown in Table 2 and Figure 9 (supplementary), and for scenario 2 & 3 it follows the distributions of COCO & VOC. Next, we will go over the details of each experiment.

Scenario 1: Combining detectors with different expertise: We took 3 models, each trained on a subset of the COCO dataset but designed for a different purpose, one for detection of transportation related objects, one for sports related, and the other for home objects. These categories have some partial overlap. Table 2 shows the object categories used for each model. The combined model achieved by our model composition procedure combines the skills of the input models, and builds a stronger model with all object categories. We tried our method in two ways: one using unlabeled COCO images (similar data distribution to training data, but without using the labels), and the other using unlabeled OID (open images) dataset (entirely different dataset with different distribution). The upper-bound of the performance would be to train a model with all labels of all object categories (supervised). This model achieved 35.11% mAP on validation set of COCO (considering only object categories corresponding to the ones it was trained on). On the same validation set, our method achieved 32.61% when using unlabeled COCO, and 30.97% when using unlabeled OID. This shows that our method can effectively combine the models with different expertise, and achieve a performance close to that of the supervised upper-bound model. We further investigated the performance of our method if partial labels are available for fine-tuning, in a semi-supervised manner. Table 3 shows the results for this experiment. As observed, with fine-tuning, our method could even surpass the supervised model with 100% of labels.

Scenario 2: Combining input models that are trained on entirely different datasets. In scenario 1, input models had different expertise, by getting trained on different subsets of COCO (examples roughly came from a similar distribution). Scenario 2 investigates a more challenging case, where input models were trained with data from entirely different datasets, hence different distributions. To this end, we trained input models on COCO and

Experiment	Method	Proportion of labeled data used (%)						
		0	1	5	10	30	50	100
Scenario 1	Supervised: COCO$_{subsets}$	-	3.9	16.4	20.3	28	30.8	35.1
	Ours: COCO^U+FT	**32.6**	32.6	33.5	33.7	34	34	35.7
	Ours: OID^{U-118k}+FT	28.5	29.1	29.6	31.8	32.8	33.9	35.3
	Ours: OID^U+FT	31	31	31.3	32.3	33.9	34.6	**36**
Scenario 2	Supervised: COCO+VOC	-	5.3	16.3	19.7	25	27.5	32.9
	Ours: [COCO+VOC]^U+FT	**29**	29.2	29.3	30	30.3	30.4	**33.1**
	Ours: OID^{U-118k}+FT	26	26.5	27	27.9	29.2	30.3	32.5
	Ours: OID^U+FT	27.4	27.4	27.8	29	30	30.5	32.7
Scenario 3	Supervised: COCO+VOC	-	4	12.6	18.5	27.5	30.6	35
	Ours: [COCO+VOC]^U+FT	**34**	34.2	34.6	35	35.4	35.9	**38**
	Ours: OID^{U-118k}+FT	16	22.5	24.9	26	28.1	30.2	33.9
	Ours: OID^U+FT	16	22.6	25.2	26.6	29	30.4	34.1
Scenario 4	Supervised: COCO	-	1.2	15.6	19.2	24.4	27.9	**33.6**
	Ours: COCO^U+FT	**24.4**	24.5	26.7	27.7	28.6	29.1	33.1
	Ours: OID^U+FT	16.6	16.8	19.9	21.6	25.2	27.1	32.4

Table 3: Object detection results: FT, Ours, and U, refer to fine-tuning, Model Composition, and unlabeled, respectively. Combined models, even without any labels, show a competitive performance.

Pascal-VOC datasets respectively. Similar to the previous scenario, we studied two choices of unlabeled data for our Model Composition method: a) unlabeled data from the same distribution as training data (in this case COCO+VOC images without using labels), and b) unlabeled data from a different dataset all together, e.g. the OID dataset. Note that the input models were trained on a different number of object categories (with overlap), and the output combined model was trained to support the union of object categories of the input models.

Table 3 shows the results of this experiment. It is observed from Table 3 that in the unsupervised case (i.e. no labeled data was used), our method achievs 29% and 27.4% mAP, close to the fully supervised performance of the upper-bound model. We also see from Table 3 that when partially labeled data are used for further fine-tuning, our method shows significant improvements over supervised training. In particular, when using 1%, 5%, and 10% of labels, our method shows +22.1%, +13%, and +10.3% gaps over supervised training.

Scenario 3: Combining input models with different architectures, that are trained on entirely different datasets. In this scenario, we studied the most generic case, in which input models have different architectures, are trained on different datasets, and with a different number of object categories. The output model also was chosen to have a different architecture than the input models (See Table 1). This scenario evaluated whether our method can combine the knowledge of models trained on different circumstances, data, and architecture, to a desired new and different architecture.

Table 3 shows the results of this experiment. It is observed from Table 3 that our method is very effective, and in some cases performs even better than supervised training with 100% of labels. When partial labels are available for fine-tuning, our method shows a strong performance, with large gaps compared to supervised training, especially in the low label range. Moreover, Table 3 shows that unsupervised training with our method achieved an mAP of 34%, only 1% below supervised training with all labels. After fine-tuning, we were able to meet the same performance as supervised training with only 10% of the original data.

Scenario 4: Having a large number of input models. This scenario investigates the case when a larger number of input models are provided. This would increase the diversity among the models since they can be trained on different data, or object categories, and thus results in a more challenging situation. To this end, we assumed 10 input models. Each model was trained on a randomly selected subset of the COCO dataset, so that training data for each model had no overlap to the other models. However, object categories could have overlap, as their type and count were chosen randomly. Supplementary materials [**7**] provides a visualization of the type and count of the object categories used for these 10 models. Since

each model was trained with roughly 10% of the COCO training set, different number of object categories for different models resulted in a different per-class size of training data. The imbalance here made model composition harder, but mimicked realistic situations where training data can in fact be imbalanced for input models. As mentioned, for these 10 models, categories were randomly selected and the number of categories was selected from 5, 10, 20, 30, and 40. Note that generating $10\times$ pseudo-labels on unlabeled data can be time-consuming (although it can be parallelized in production). Therefore, we only used 100K randomly selected examples from the OID dataset for this experiment.

We observe from Table 3 that our model composition method can effectively combine the 10 input models into a single new model with the union of their object categories.

Remark 3: A note on the unsupervised performance of OID: As observed in Table 3, in the challenging scenarios of 3 and 4, the unsupervised (0% labels) performance of model composition with OID^U is considerably lower that that of $COCO^U$ or $[COCO+VOC]^U$. In this regard, there are a few points worth a mention:

- In general, using unrelated arbitrary data is expected to result in a lower performance compared to using data from the same distribution as the input models' train set, since pseudo-labels will be less reliable. This is exacerbated in challenging tasks such as scenario 3 where input models are trained on different data and have different architectures with respect to each other and the output model; or in scenario 4 where there are a large number of input models trained on different small-scale data.

- It is worth reminding that the case of purely unsupervised model composition means combining an arbitrary number of black-box models (trained on arbitrary data with arbitrary architecture or categories), all without using any labels. In that sense, the real baseline to compare against is the supervised training, which performs much worse than model composition in low data regimes, even in the case of unrelated OID^U data.

- Moreover, the main goal of the paper is to explore whether or not neural networks can be combined using only unlabeled data, and if yes, to what extent (hence the title). We observe from the results that the answer is for the most part yes; however, in case unlabeled data from the original distributions was not available, for some challenging scenarios a small percentage of labels may be needed to achieve a decent performance.

- In a completely unsupervised setting, model composition can still effectively combine input models. The performance will be improved if the unlabeled set size is larger.

Remark 4: A note on practical applications: As mentioned in Section 3, a fundamental motivation for our work is a cloud services application, as shown in Figure 4, in which engineers and expert users can leverage a model composition service to build stronger models with combined skills, especially in the presence of a large variety of trained models and datasets on the cloud. Different scenarios in the experiments were also inspired by such a philosophy, but designed at various levels of difficulty. Here, we add a new practical use-case. In this new experiment, we combine separate models of face and mask detection, to build one that is suitable for both face & mask detection. Results are shown in Table 4.

Model/Expertise	Train set	Validation set	AP(%)
input: Face (D0)	face data 1 (20007)	face data 1 (4079)	52.29
input: Face (D0)	MAFA-faces (30870)	MAFA-faces (5338)	44.86
input: Mask (D1)	MAFA-masks (30870)	MAFA-masks (5338)	29.63
ModelComp (R50): w/o filtering & aggregation	face+mask (50877)	face+mask (9417)	30.72
ModelComp (R50): w/o aggregation	face+mask (50877)	face+mask (9417)	34.48
Ours, ModelComp: Face & Mask (R50)	face+mask (50877)	face+mask (9417)	**38.90**

Table 4: Combining face and mask detectors with different architectures trained on different datasets. Face data 1: Face images from WIDERFACE [52] and medical masks datasets [2, 3] (including both faces with-masks and without-masks). MAFA faces and masks are obtained from the MAFA dataset [12]. We also provide an ablation on the filtering and aggregation modules.

Ablation on the number of categories: Next, we study the impact of the number of class categories in the performance. To this end, we take two input models from scenario 4, and combine them with a varying number of classes. M_1 is trained on 5 and M_2 is trained on 10 object categories (see Figure 9 in supplementary). Each time we add a number of random categories of M_2 to M_1, so the combined model can have 5,6,...,15 classes. Table 5 shows the results. Note that in each case the validation/training set will be different as it includes images of a particular set of categories. M_1 and M_2 have mAP of 27.1% & 25.4%, respectively (each has roughly 12K training, and 1K validation examples). In general, higher number of classes results in slightly lower mAP, but we should also note that unlabeled set size becomes also larger (i.e. more pseudo-labels).

#Categoreis	5	6	7	10	15
ModelComp	27.1	26.8	26.6	24.7	23.9

Table 5: Ablation on classes. Details in 4.2.

4.3   Image classification results

In addition to our main results on the task of object detection, we also provide a highlight of our results on the task of image classification. Similar to object detection, we designed the classification experiments in the form of different scenarios.

Scenario 1: 3 input models, ResNet-18, each trained on 1/3 of the Caltech-256 dataset.
Scenario 2: 3 models, ResNet-18, ResNet-152, and DenseNet-121, trained on Caltech-256.

In both scenarios, we tried model composition with unlabeled data from the Caltech-256 dataset (i.e. similar data distribution but without labels), and a 160K subset unlabeled data from OID (i.e. a different dataset altogether). Table 6 shows the results for these scenarios.

In Table 7, we provide a comparison between our method and two additional baselines: i) a simple model ensemble by aggregating directly the prediction of the input models; ii) knowledge distillation when using the input models as teachers, such as [4, 15]. For the second baseline, we consider the vanilla distillation [15] but with soft labels.

It is observed from the results that the proposed method is effective in combining image classification models. In both the unsupervised and semi-supervised cases, our method performs competitively compared to supervised models, even when 100% of labels are used.

Experiment	Method	Proportion of labeled data used (%)					
		0	1	5	10	50	100
	Supervised: Caltech	-	16.2	44	61	79.4	82.4
Scenario 1	Ours: Caltech^U+FT	**83**	82.6	82.8	82.9	83	**83.5**
	Ours: OID^U+FT	69	70	71.2	72.9	79	81.6
	Supervised: Caltech	-	16.2	43.9	60.9	79.4	82.4
Scenario 2	Ours: Caltech^U+FT	**83.2**	82.1	81.8	81.8	83.1	**83.3**
	Ours: OID^U+FT	71.6	68.5	71.5	72.4	78.8	81.6

Table 6: Image classification results: FT, Ours, and U, refer to fine-tuning, Model Composition, and unlabeled, respectively. Combined models, even without any labels, show a competitive performance.

5   Conclusion

This paper proposed a method for combining multiple trained neural networks into a single model, using unlabeled data. To this end, first the input models' predictions (pseudo-labels) were collected. The pseudo-labels were then filtered based on confidence scores of the predictions. Next, a consensus aggregation strategy was incorporated to combine these pseudo-labels. The remaining pseudo-labels were used to train the output model. The proposed method supported the use of an arbitrary number of input models with arbitrary architectures and categories. Performance evaluations on various datasets, tasks, and network architectures demonstrated the effectiveness of the proposed method.

	Scenario 1	Scenario 2
Ensembling	53.7	59.8
Vanilla distillation	64.2	68.2
Ours: OID^U	69	71.6

Table 7: More classification results on OID^U.

References

[1] Efficientdet repository. https://github.com/google/automl/tree/master/efficientdet. Accessed: 2020-08.

[2] Medical masks dataset. https://www.kaggle.com/ivandanilovich/medical-masks-dataset-images-tfrecords, . Accessed: 2021-08.

[3] Humans in the loop medical mask dataset. https://humansintheloop.org/resources/datasets/medical-mask-dataset/, . Accessed: 2021-08.

[4] S. Ahn, She. X. Hu, A. Damianou, N. Lawrence, and Zh. Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9163–9171, 2019.

[5] M.Z. Alom, T.M. Taha, Ch. Yakopcic, S. Westberg, P. Sidike, M.Sh. Nasrin, M. Hasan, B.C. Van Essen, A.A. Awwal, and V.K. Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292, 2019.

[6] A. Banitalebi-Dehkordi. Knowledge distillation for low-power object detection: A simple technique and its extensions for training compact models using unlabeled data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021.

[7] A. Banitalebi-Dehkordi, X. Kang, and Y. Zhang. Model composition: Can multiple neural networks be combined into a single network using only unlabeled data? In *32nd British Machine Vision Conference, BMVC*, 2021. Supplementary materials. 0508supp.pdf.

[8] N. Bodla, B. Singh, R. Chellappa, and L. Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017.

[9] A. Casado-García and J. Heras. Ensemble methods for object detection. In *European conference on artificial intelligence, ECAI*, 2020.

[10] Y.-M. Chou, Y.-M. Chan, J.-H. Lee, Ch.-Y. Chiu, and Ch.-S. Chen. Unifying and merging well-trained deep neural networks for inference stage. *arXiv preprint arXiv:1805.04980*, 2018.

[11] M. Everingham, L. Van Gool, Ch.K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[12] Shiming Ge, Jia Li, Qiting Ye, and Zhao Luo. Detecting masked faces in the wild with lle-cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2682–2690, 2017.

[13] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[14] K. He, X. Zhang, Sh. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[16] G. Huang, Zh. Liu, L. Van Der Maaten, and K.Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[17] R. Socher L.-J. Li K. Li J. Deng, W. Dong and L. FeiFei. Imagenet: A large-scale hierarchical image database. *IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

[18] A. Jha, A. Kumar, B. Banerjee, and V. Namboodiri. SD-MTCNN: self-distilled multi-task CNN. In *31st British Machine Vision Conference 2020, BMVC 2020, UK, September 7-10, 2020*. BMVA Press, 2020.

[19] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, Sh. Kamali, S. Popov, M. Malloci, A. Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.

[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[22] C. Peng, K. Zhao, and B.C. Lovell. Faster ilod: Incremental learning for object detectors based on faster rcnn. *arXiv preprint arXiv:2003.03901*, 2020.

[23] A. Rame, E. Garreau, H. Ben-Younes, and Ch. Ollion. Omnia faster r-cnn: Detection in the wild through dataset merging and soft distillation. *arXiv preprint arXiv:1812.02611*, 2018.

[24] M. Rottmann, K. Kahl, and H. Gottschalk. Deep bayesian active semi-supervised learning. *arXiv preprint arXiv:1803.01216*, 2018.

[25] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[26] A. Saporta, T.-H. Vu, M. Cord, and P. Pérez. Esl: Entropy-guided self-supervised learning for domain adaptation in semantic segmentation. *arXiv preprint arXiv:2006.08658*, 2020.

[27] A. Shrestha and A. Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, 7:53040–53065, 2019.

[28] R. Solovyev, W. Wang, and T. Gabruseva. Weighted boxes fusion: ensembling boxes for object detection models. *arXiv preprint arXiv:1910.13302*, 2019.

[29] M. Tan, R. Pang, and Q.V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10790, 2020.

[30] S. Teerapittayanon, B. McDanel, and H.-T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[31] S. Vandenhende, S. Georgoulis, L. Van Gool, and B. De Brabandere. Branched multi-task networks: Deciding what layers to share. In *31st British Machine Vision Conference 2020, BMVC 2020, UK, September 7-10, 2020*. BMVA Press, 2020.

[32] Sh. Yang, P. Luo, Ch. Ch. Loy, and X. Tang. Wider face: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[33] H. Zhou, B. Gao, and J. Wu. Adaptive feeding: Achieving fast and accurate detections by adaptively combining object detectors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3505–3513, 2017.

[34] P. Zhou, L. Mai, J. Zhang, N. Xu, Z. Wu, and L. Davis. M2KD: incremental learning via multi-model and multi-level knowledge distillation. In *31st British Machine Vision Conference 2020, BMVC 2020, UK, September 7-10, 2020*. BMVA Press, 2020.

[35] Zh. Zhou, J. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.