# Progressive Recurrent Learning for Visual Recognition

Xutong Ren[1], Lingxi Xie[2], Chen Wei[1], Siyuan Qiao[2], Chi Su[3], Jiaying Liu[1], Alan L. Yuille[2]
[1]Peking University, [2]The Johns Hopkins University [3]Kingsoft
tonghelen@pku.edu.cn 198808xc@gmail.com weichen582@pku.edu.cn joe.siyuan.qiao@gmail.com
suchi@kingsoft.com liujiaying@pku.edu.cn alan.yuille@jhu.edu

## Abstract

*Computer vision is difficult, partly because the mathematical function connecting input and output data is often complex, fuzzy and thus hard to learn. A currently popular solution is to design a deep neural network and optimize it on a large-scale dataset. However, as the number of parameters increases, the generalization ability is often not guaranteed, e.g., the model can over-fit due to the limited amount of training data, or fail to converge because the desired function is too difficult to learn.*

*This paper presents an effective framework named progressive recurrent learning (PRL). The core idea is similar to curriculum learning which gradually increases the difficulty of training data. We generalize it to a wide range of vision problems that were previously considered less proper to apply curriculum learning. PRL starts with inserting a recurrent prediction scheme, based on the motivation of feeding the prediction of a vision model to the same model iteratively, so that the auxiliary cues contained in it can be exploited to improve the quality of itself. In order to better optimize this framework, we start with providing perfect prediction, i.e., ground-truth, to the second stage, but gradually replace it with the prediction of the first stage. In the final status, the ground-truth information is not needed any more, so that the entire model works on the real data distribution as in the testing process. We apply PRL to two challenging visual recognition tasks, namely, object localization and semantic segmentation, and demonstrate consistent accuracy gain compared to the baseline training strategy, especially in the scenarios of more difficult vision tasks.*

## 1. Introduction

Image recognition is a fundamental task of computer vision, which aims at understanding semantic contents from raw pixels. This is often difficult, because the underlying connections (*e.g.*, a mathematical function) between low-level pixels and high-level semantics are often complicated and fuzzy, *e.g.*, there exist a lot of elements in the data space which are either meaningless or ambiguous [24]. In the deep learning era, researchers design deep neural networks as hierarchical and composite functions [16][11]. However, the difficulty of training a network increases as its complexity [9] does. Despite some technical improvements designed to alleviate instability of training [23][34][14], the learned model still suffers from over-fitting, which is arguably caused by the overhigh complexity of the designed model so that the limited amount of training data can be interpreted in some improper ways.

Despite designing more complex models, an alternative solution lies in optimizing an existing model better. This paper investigates an algorithm along this direction. It is named curriculum learning [2], which measures the difficulty of each training sample and adjusts data distribution during the training process, so that the model is optimized with gradually increasing complexity, and thus becomes more stable and less likely to over-fit data. However, this idea was only applied to a limited range of vision tasks, because it is often hard to determine the difficulty level of training data and further partition them into different groups. The major contribution of this paper is to provide a new framework named **progressive recurrent learning** (PRL), which uses entropy to define the difficulty of training data distribution, and gradually reduces the level of auxiliary cues to construct training data with varying difficulties.

PRL is built upon a straightforward idea named coarse-to-fine learning, in which we train two models $\mathbb{M}^{\mathrm{C}}$ and $\mathbb{M}^{\mathrm{F}}$ simultaneously, with the former taking input data $\mathbf{x}$ and outputting $\mathbf{y}^{\mathrm{C}} = \mathbf{f}^{\mathrm{C}}(\mathbf{x})$, and the latter taking input data $\mathbf{x}$ as well as an auxiliary cue $\mathbf{z}$ and outputting $\mathbf{y}^{\mathrm{F}} = \mathbf{f}^{\mathrm{F}}(\mathbf{x}, \mathbf{z})$. Here, $\mathbf{z} = \mathbf{g}(\tilde{\mathbf{y}})$ is a function of $\tilde{\mathbf{y}}$, and $\tilde{\mathbf{y}}$ is either the ground-truth $\mathbf{y}^{\star}$ or the coarse prediction $\mathbf{y}^{\mathrm{C}}$. This implies that the overall model is formulated in a recurrent form with the output being used as input repeatedly. In the training process, we control the difficulty by changing the probability of sampling $\mathbf{y}^{\star}$ and $\mathbf{y}^{\mathrm{C}}$ in computing $\mathbf{z}$, *e.g.*, the probability of sampling $\mathbf{y}^{\star}$ is $1-t$, where $t \in [0,1]$ indicates the fraction of elapsed training iterations. This is to say, the ground-truth annotation $\mathbf{y}^{\star}$ is used to provide a warm start of training $\mathbb{M}^{\mathrm{F}}$, but gradually replaced by the coarse prediction $\mathbf{y}^{\mathrm{C}}$ so that training difficulty increases. At the end of training, $t = 1$ so that $\mathbb{M}^{\mathrm{F}}$ does not rely on any extra information, and can be applied in the testing process as a refinement of $\mathbb{M}^{\mathrm{C}}$. In practice, PRL demonstrates two-fold benefits. First, the gradually increasing training difficulty makes the optimization process more stable, *i.e.*, converges better. Second, the coarse-to-fine iteration pushes the entire model towards higher accuracy.

PRL is a generalized framework which can be applied to a wide range of vision problems. In this paper, we study two examples, namely, object localization and semantic segmentation. In these tasks, the desired output is either an object bounding box or a segmentation mask, both of which can be easily represented as a vector or a matrix $\mathbf{y}$. We start with providing perfect supervision ($t = 0$) to the fine stage. To prevent it from directly taking this information and thus not learning any useful knowledge, we perform a transformation function on $\mathbf{y}$ before combining it with input data $\mathbf{x}$. Experiments reveal consistent accuracy gain brought by PRL to the baseline models. Empirical analysis by comparing training and testing losses verifies our motivation, *i.e.*, such improvement comes from alleviating the risk of over-fitting.

The remainder of this paper is organized as follows. Section 2 briefly reviews related work, and Section 3 presents the PRL approach as well as some analysis. After instantiating our approach on two visual recognition tasks in Section 4, we draw our conclusions in Section 5.

## 2. Related Work

Deep learning [17] in particular deep convolutional neural networks have been dominating the field of computer vision. The fundamental idea is to build a hierarchical structure to learn complicated visual patterns from a large-scale database [6]. As the number of network layers increases from tens [16][33][35] to hundreds [11][13], the network's representation ability becomes stronger, but training these networks becomes more and more challenging. Various techniques have been proposed to improve numerical stability [23][14] and over-fitting [34], but the transferability from training data to testing data is still below satisfactory. It was pointed out that this issue is mainly caused by the overhigh complexity of deep networks, so that the limited amount of training data can be interpreted in an unexpected way [24]. There exist two types of solutions, namely, curriculum learning and coarse-to-fine learning.

The basic idea of curriculum learning [2] is to gradually increase the difficulty of training data, so that the model can be optimized in a faster and/or more stable manner. This idea was first brought up by referring to how humans are taught to learn a concept and verified effective also for computer algorithms [15]. It was later widely applied to a wide range of learning tasks, including visual recognition [30][36] and generation [31], natural language processing [19][27] and reinforcement learning [44][40]. Curriculum learning was theoretically verified a good choice in transfer learning [39], multi-task learning [25] and sequential learning [1] scenarios, and there have been discussions on the principle of designing curriculum towards better performance [46]. A similar idea (gradually increasing training difficulty) was also adopted in online hard example mining [32][5], but the latter case often started with a regular data distribution which is gradually adjusted towards difficult training data. The major drawback of curriculum learning lies in the requirement of evaluating the difficulty of training data, which is not easy in general. This paper provides a framework to bypass this problem.

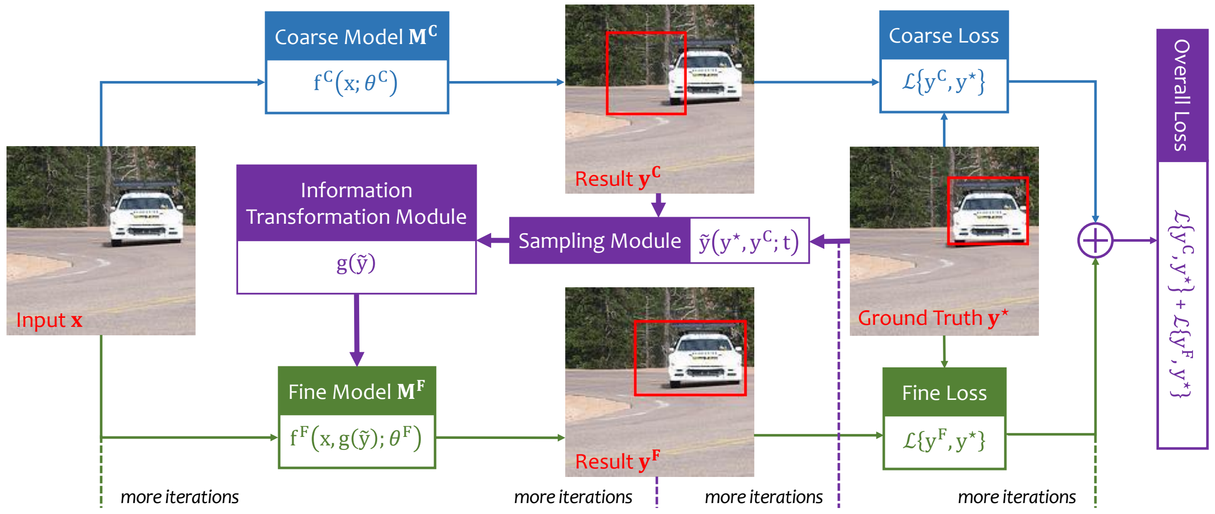
Another idea, named coarse-to-fine learning, was based on the observation that a vision model can rethink its prediction to amend errors [3]. Researchers designed several approaches for refining visual recognition in an iterative manner. These approaches can be explained using auto-context [37] or formulated into a fixed-point model [20]. Examples include the coarse-to-fine models for image classification [8], object detection [4], semantic segmentation [47], pose estimation [38], image captioning [10], *etc*. It was verified that joint optimization over coarse and fine stages boosts the performance [43], which raised an issue of the communication between coarse and fine stages in the training process – we desire feeding coarse-stage output to fine-stage input, but when the coarse model has not been well optimized, this can lead to unstable performance in optimization. The method proposed in this paper can largely alleviate this issue.

## 3. Our Approach

This section describes the progressive recurrent learning (PRL) algorithm. We first briefly review the curriculum learning algorithm, the precursor of PRL. Based on the limitation of curriculum learning, we propose PRL and provide some theoretical analysis. The applications of PRL on two vision problems are illustrated in the next section.

### 3.1. Background: Curriculum Learning

Machine learning often starts with constructing a model $\mathbb{M} : \mathbf{y} = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$, where $\mathbf{x}$ and $\mathbf{y}$ are input and output, and $\boldsymbol{\theta}$ denotes the parameters. In the context of deep learning, $\mathbf{f}(\cdot)$ describes the network architecture, and $\boldsymbol{\theta}$ the learnable weights. The goal is to find the optimal $\boldsymbol{\theta}$ that best fits a training dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$. In the modern era, $\boldsymbol{\theta}$ often contains millions of parameters while the amount of training data, $N$, can still be limited, *e.g.*, thousands. Therefore, it is possible that $\mathbb{M}$ is over-fitted, *i.e.*, $\boldsymbol{\theta}$ interprets the



Coarse Model $\mathbf{M}^\mathrm{C}$
$\mathrm{f}^\mathrm{C}(\mathrm{x};\theta^\mathrm{C})$
Result $\mathbf{y}^\mathrm{C}$
Coarse Loss
$\mathcal{L}\{\mathrm{y}^\mathrm{C},\mathrm{y}^\star\}$
Overall Loss
$\mathcal{L}\{\mathrm{y}^\mathrm{C},\mathrm{y}^\star\}+\mathcal{L}\{\mathrm{y}^\mathrm{F},\mathrm{y}^\star\}$
Input $\mathbf{x}$
Information Transformation Module
$\mathrm{g}(\tilde{\mathrm{y}})$
Sampling Module
$\tilde{\mathrm{y}}(\mathrm{y}^\star,\mathrm{y}^\mathrm{C};\mathrm{t})$
Ground Truth $\mathbf{y}^\star$
Fine Model $\mathbf{M}^\mathrm{F}$
$\mathrm{f}^\mathrm{F}(\mathrm{x},\mathrm{g}(\tilde{\mathrm{y}});\theta^\mathrm{F})$
Result $\mathbf{y}^\mathrm{F}$
Fine Loss
$\mathcal{L}\{\mathrm{y}^\mathrm{F},\mathrm{y}^\star\}$
more iterations
more iterations
more iterations
more iterations

Figure 1. Illustration of the progressive recurrent learning (PRL) algorithm (best viewed in color). The coarse input $\mathbf{x}$ is fed into a coarse model, and the output is recurrently updated towards higher accuracy. The sampling module $\tilde{\mathbf{y}}(\mathbf{y}^\star,\mathbf{y}^\mathrm{C};t)$ placed at the center denotes the key part of PRL, which starts with ground-truth but gradually biases towards prediction as the training process continues. Due to the space limit, we only show one iteration here though this framework can be trained and evaluated in a recurrent manner.

training data in $\mathcal{D}$ perfectly, but fails to generalize well to new, unseen testing data.

Instead of designing more complex models, there were efforts in optimizing an existing network in a safer way. One of them is named curriculum learning [2], which assumed that each training data $\mathbf{x}$ to be sampled from a distribution $\mathcal{P}$ with $P(\mathbf{x})$ being the density function at $\mathbf{x}$. Then, the training process is parameterized using time $t \in [0,1]$. Each training sample is assigned with a difficulty measure $d(\mathbf{x}) \in \mathbb{R}$, and a weighting term $w_t(\mathbf{x}) \in [0,1]$. The actual sampling distribution $\mathcal{Q}_t$ satisfies $Q_t(\mathbf{x}) \propto P(\mathbf{x})\,w_t(\mathbf{x})$. The exact coefficient for $Q_t(\mathbf{x})$ is determined by the normalization rule $\int Q_t(\mathbf{x})\,\mathrm{d}\mathbf{x} = 1$. This process is named curriculum learning if the following two constraints are satisfied [2]:

$$w_{t_1}(\mathbf{x}) \leqslant w_{t_2}(\mathbf{x}) \quad \forall \mathbf{x}, \forall 0 \leqslant t_1 < t_2 \leqslant 1, \tag{1}$$

which means that for each $\mathbf{x}$, $w_t(\mathbf{x})$ is monotonically increasing with respect to $t$, and

$$\mathbb{H}[Q_{t_1}] < \mathbb{H}[Q_{t_2}] \quad \forall 0 \leqslant t_1 < t_2 \leqslant 1, \tag{2}$$

where $\mathbb{H}[\cdot]$ denotes the entropy of the distribution. The general idea of curriculum learning is to measure the difficulty of training data. If $\mathbf{x}_1$ is equally or more difficult than $\mathbf{x}_2$, then $w_t(\mathbf{x}_1) \leqslant w_t(\mathbf{x}_2)$ for each step $t$. Eventually, when $t = 1$ there is $w_1(\mathbf{x}) = 1$ for all $\mathbf{x}$. This formulation leans towards sampling easy cases at first, and gradually converges to the real distribution, *i.e.*, $\mathcal{Q}_1 \equiv \mathcal{P}$.

### 3.2. Progressive Recurrent Learning

A major limitation of curriculum learning lies in an explicit way of defining the difficulty of each $\mathbf{x}$, which is often difficult as few datasets provided such information for each training sample. To deal with this issue, we borrow the definition from information theory and use the entropy of data distribution to define the difficulty of training data. Section 3.4 provides a theoretical analysis that entropy in our training process is indeed gradually growing with time.

To instantiate this general idea, we borrow a simple framework named coarse-to-fine learning and, by adding an auxiliary cue to the fine stage, we can easily sample training data with different difficulty levels by controlling how much this cue takes advantage from the ground-truth label.

Consider a training sample $(\mathbf{x},\mathbf{y}^\star)$ where $\mathbf{y}^\star$ denotes the ground-truth label. Due to the difficulty of directly learning the model $\mathbb{M}: \mathbf{y} = \mathbf{f}(\mathbf{x};\boldsymbol{\theta})$, we decompose it into two components, namely, a coarse model and a fine model. The *coarse* model $\mathbb{M}^\mathrm{C}: \mathbf{y}^\mathrm{C} = \mathbf{f}^\mathrm{C}\left(\mathbf{x};\boldsymbol{\theta}^\mathrm{C}\right)$ obtains a rough prediction $\mathbf{y}^\mathrm{C}$ and the *fine* model $\mathbb{M}^\mathrm{F}: \mathbf{y}^\mathrm{F} = \mathbf{f}^\mathrm{F}\left(\mathbf{x},\mathbf{z};\boldsymbol{\theta}^\mathrm{F}\right)$ polishes the prediction by feeding it to the network with input data again with an auxiliary cue $\mathbf{z}$. We assume that $\mathbf{z}$ is closely related to $\mathbf{y}^\star$, so that $\mathbb{M}^\mathrm{F}$ can take advantage of this additional information, and becomes easier to train and performs better in the testing process[1]. The overall goal of

[1]In the testing process, we can first apply $\mathbb{M}^\mathrm{C}$ to $\mathbf{x}$ to obtain $\mathbf{y}^{(0)}$ and then repeatedly execute $\mathbb{M}^\mathrm{F}$ to update $\mathbf{y}^{(l)}$ ($l$ is the iteration index) until convergence or some terminating condition is satisfied.

optimization is:

$$\arg\min_{\boldsymbol{\theta}^{\mathrm{C}},\boldsymbol{\theta}^{\mathrm{F}}} \left|\mathbf{f}^{\mathrm{F}}\left(\mathbf{x},\mathbf{g}\left(\mathbf{f}^{\mathrm{C}}\left(\mathbf{x};\boldsymbol{\theta}^{\mathrm{C}}\right)\right);\boldsymbol{\theta}^{\mathrm{F}}\right)-\mathbf{y}^{\star}\right|^{2} \quad (3)$$

It remains to determine how to optimize $\mathbb{M}^{\mathrm{F}}$ ($\mathbb{M}^{\mathrm{C}}$ simply follows a regular training strategy), in particular how to design the auxiliary cue $\mathbf{z}$. We expect $\mathbb{M}^{\mathrm{F}}$ to learn the knowledge that $\mathbf{z}$ is related to $\mathbf{y}^{\star}$, but do not hope it to deliver too much information so that $\mathbb{M}^{\mathrm{F}}$ largely relies on $\mathbf{z}$ and ignores $\mathbf{x}$. In addition, note that $\mathbf{y}^{\star}$ is not available in the testing process, *i.e.*, it needs to be replaced by $\mathbf{y}^{\mathrm{C}}$. Considering these factors, we design $\mathbf{z} = \mathbf{g}(\tilde{\mathbf{y}})$, where $\tilde{\mathbf{y}}$ takes the value of $\mathbf{y}^{\star}$ or $\mathbf{y}^{\mathrm{C}}$, and $\mathbf{g}(\cdot)$ is a transformation function which weakens the information delivered by $\mathbf{z}$. The probability of choosing $\mathbf{y}^{\star}$ instead of $\mathbf{y}^{\mathrm{C}}$ determines the fraction of *oracle information* provided to $\mathbb{M}^{\mathrm{F}}$, which serves as a practical way of controlling the difficulty of training data.

Based on this, we apply a progressive learning process[2] to gradually increase training difficulty. We set a variable $t$ which is positively related to the fraction of elapsed iterations. In each iteration, the probability that $\mathbf{y}^{\star}$ is selected and fed into $\mathbf{z}$ is $1-t$, and in the remaining case we choose $\mathbf{y}^{\mathrm{C}}$ instead:

$$\tilde{\mathbf{y}}\left(\mathbf{y}^{\star},\mathbf{y}^{\mathrm{C}};t\right)=\left\{\begin{array}{ll}\mathbf{y}^{\star} & \text{if } a\sim\mathcal{U}(0,1)>t \\ \mathbf{y}^{\mathrm{C}} & \text{otherwise}\end{array}\right., \quad (4)$$

where $\mathcal{U}$ denotes a uniform distribution. There are of course other options, *e.g.*, computing a weighted average of $\mathbf{y}^{\star}$ and $\mathbf{y}^{\mathrm{C}}$, *i.e.*, $\tilde{\mathbf{y}}\left(\mathbf{y}^{\star},\mathbf{y}^{\mathrm{C}};t\right)=(1-t)\cdot\mathbf{y}^{\star}+t\cdot\mathbf{y}^{\mathrm{C}}$. In this paper, we investigate Eqn (4) and show its effectiveness.

In summary, in each training iteration, we take the ground-truth label $\mathbf{y}^{\star}$ and the output of the coarse model $\mathbf{y}^{\mathrm{C}}$, compute $\mathbf{z}$ basing on Eqn (4) and feed $\mathbf{z}$ to the fine model. The overall loss function is:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}\left\{\mathbf{f}^{\mathrm{C}}\left(\mathbf{x};\boldsymbol{\theta}^{\mathrm{C}}\right),\mathbf{y}^{\star}\right\}+\mathcal{L}\left\{\mathbf{f}^{\mathrm{F}}\left(\mathbf{x},\mathbf{z};\boldsymbol{\theta}^{\mathrm{F}}\right),\mathbf{y}^{\star}\right\} \\ &= \left|\mathbf{f}^{\mathrm{C}}\left(\mathbf{x};\boldsymbol{\theta}^{\mathrm{C}}\right)-\mathbf{y}^{\star}\right|^{2}+\left|\mathbf{f}^{\mathrm{F}}\left(\mathbf{x}\oplus\mathbf{g}(\tilde{\mathbf{y}});\boldsymbol{\theta}^{\mathrm{F}}\right)-\mathbf{y}^{\star}\right|^{2}, \quad (5)\end{aligned}$$

where $\oplus$ denotes concatenation at the channel dimension. When $\mathbf{y}^{\mathrm{C}}$ is chosen as $\tilde{\mathbf{y}}$, the gradient of the second term involves both $\boldsymbol{\theta}^{\mathrm{C}}$ and $\boldsymbol{\theta}^{\mathrm{F}}$ and the coarse and fine models are optimized jointly. We add a coarse loss term so that $\boldsymbol{\theta}^{\mathrm{C}}$ is better optimized [35][18] and the entire model achieves a higher stability [43]. The complete training process is described in Algorithm 1.

In the testing stage, we use $\mathbb{M}^{\mathrm{C}}$ to compute the *first* output $\mathbf{y}^{(0)}$, and iteratively feed it to $\mathbb{M}^{\mathrm{F}}$, *i.e.*, the $l$-th iteration produces $\mathbf{y}^{(l)}$ from $\mathbf{y}^{(l-1)}$, and this process continues

[2]The term of "progressive learning" was used in a few prior approaches [42][29][21], but with different motivations. Our goal is to gradually increase the difficulty of training data.

**Algorithm 1:** Progressive Recurrent Learning: Training

```
Input : Training set $\mathcal{D}=\{(\mathbf{x}_n,\mathbf{y}_n^{\star})\}_{n=1}^{N}$, initialized
        parameters $\boldsymbol{\theta}_0^{\mathrm{C}}$ and $\boldsymbol{\theta}_0^{\mathrm{F}}$, step size $t_0$;
1  $t \leftarrow 0$;
2  while $t < 1$ do
3      Sample: mini-batch $\mathcal{B}_t \sim \mathcal{D}$;
4      for $(\mathbf{x}_n,\mathbf{y}_n^{\star}) \in \mathcal{B}_t$ do
5          Compute $\mathbf{y}_n^{\mathrm{C}} \leftarrow \mathbf{f}^{\mathrm{C}}\left(\mathbf{x}_n;\boldsymbol{\theta}^{\mathrm{C}}\right)$;
6          Compute $\tilde{\mathbf{y}}_n$ using Eqn (4);
7          Compute $\mathbf{z}_n = \mathbf{g}(\tilde{\mathbf{y}}_n)$;
8      end
9      Update $\boldsymbol{\theta}^{\mathrm{C}}$ and $\boldsymbol{\theta}^{\mathrm{F}}$ with Eqn (5);
10     $t \leftarrow t + t_0$;
11 end
Output: Trained parameters $\boldsymbol{\theta}_1^{\mathrm{C}}$ and $\boldsymbol{\theta}_1^{\mathrm{F}}$.
```

**Algorithm 2:** Progressive Recurrent Learning: Testing

```
Input : Input $\mathbf{x}$, trained parameters $\boldsymbol{\theta}_1^{\mathrm{C}}$ and $\boldsymbol{\theta}_1^{\mathrm{F}}$,
        maximal number of iterations $L$;
1  Coarse testing: $\mathbf{y}^{(0)} \leftarrow \mathbf{f}^{\mathrm{C}}\left(\mathbf{x};\boldsymbol{\theta}_1^{\mathrm{C}}\right)$, $l \leftarrow 0$;
2  while $l < L$ do
3      $l \leftarrow l + 1$;
4      $\mathbf{y}^{(l)} \leftarrow \mathbf{f}^{\mathrm{F}}\left(\mathbf{x},\mathbf{g}\left(\mathbf{y}^{(l-1)}\right);\boldsymbol{\theta}_1^{\mathrm{F}}\right)$;
5  end
Output: Output $\mathbf{y}^{(L)}$.
```

until convergence or a pre-defined maximum number of iterations is reached. Except for the *first* time, $\mathbb{M}^{\mathrm{F}}$ takes the output from $\mathbb{M}^{\mathrm{F}}$ itself rather than $\mathbb{M}^{\mathrm{C}}$, which is not consistent with the training process. However, since both $\mathbb{M}^{\mathrm{C}}$ and $\mathbb{M}^{\mathrm{F}}$ are supervised by $\mathbf{y}^{\star}$, the difference between their outputs is relatively small, so this pipeline works well in practice. The testing process is illustrated in Algorithm 2.

## 3.3. Implementation Details

In all our experiments, both $\mathbb{M}^{\mathrm{C}}$ and $\mathbb{M}^{\mathrm{F}}$ are deep neural networks, use the same architecture for simplicity, but are allowed to have different parameters. The only difference lies in the input layer, where $\mathbf{f}^{\mathrm{C}}(\cdot)$ takes the original input $\mathbf{x}$ and $\mathbf{f}^{\mathrm{F}}(\cdot)$ takes both $\mathbf{x}$ and $\mathbf{z}=\mathbf{g}(\tilde{\mathbf{y}})$.

In practice, $\mathbf{g}(\cdot)$ has two convolutional layers, each with 3 output channels (same as the input image), and a ReLU activation layer to provide non-linearity [23]. This transformation module, though only containing less than 100 parameters, plays an important role of weakening the information provided by $\tilde{\mathbf{y}}$, otherwise $\mathbb{M}^{\mathrm{F}}$ can easily learns an identity input-output mapping especially in the early training stage ($\tilde{\mathbf{y}}$ is very close to $\mathbf{y}^{\star}$). The parameters in the first convolutional layers of $\mathbb{M}^{\mathrm{C}}$ and $\mathbb{M}^{\mathrm{F}}$ are adjusted according to the dimensionality of input data.

## 3.4. Theoretical Analysis

We analyze the property of PRL before applying it to different vision problems. First, we note that PRL is not a curriculum learning process, as PRL gradually reduces the probability of sampling easy cases during the training process, and eventually discards them – this does not align with curriculum learning. However, PRL shares a similar property with curriculum learning, that the entropy of data distribution keeps increasing during training.

Let $\mathbf{u} = (\mathbf{x}, \tilde{\mathbf{y}})$ be a training sample of $\mathbb{M}^{\mathrm{F}}$, which is sampled from the *coarse-prediction distribution* $\mathcal{P}_{\mathbb{M}_t^{\mathrm{C}}}$:

$$P_{\mathbb{M}_t^{\mathrm{C}}}\left(\mathbf{u}; \boldsymbol{\theta}_t^{\mathrm{C}}\right) = P(\mathbf{x}) \cdot \mathcal{N}\left(\tilde{\mathbf{y}} \mid \mathbf{f}^{\mathrm{C}}\left(\mathbf{x}; \boldsymbol{\theta}_t^{\mathrm{C}}\right), \sigma_t \mathbf{I}\right), \quad (6)$$

where $P(\mathbf{x})$ is determined by the training set and $\mathcal{N}(\cdot)$ is an isotropic Gaussian distribution, which degenerates to the Kronecker $\delta$-function when its variance $\sigma \to 0$. Similarly, we define the *ground-truth distribution* $\mathcal{P}_{\mathrm{GT}}$:

$$P_{\mathrm{GT}}(\mathbf{u}) = P(\mathbf{x}) \cdot \mathcal{N}(\tilde{\mathbf{y}} \mid \mathbf{y}^{\star}, \sigma_t \mathbf{I}). \quad (7)$$

In the training process, $\mathcal{P}_{\mathbb{M}_t^{\mathrm{C}}}$ changes with $\boldsymbol{\theta}_t^{\mathrm{C}}$. Thus, the distribution of $\mathbf{u}$ at time $t$, denoted by $\mathcal{P}_t$, has the following formulation:

$$P_t(\mathbf{u}) = (1-t) \cdot P_{\mathrm{GT}}(\mathbf{u}) + t \cdot P_{\mathbb{M}_t^{\mathrm{C}}}\left(\mathbf{u}; \boldsymbol{\theta}_t^{\mathrm{C}}\right). \quad (8)$$

Here we make a simple assumption, that the difference between $\mathcal{P}_{\mathrm{GT}}$ and $\mathcal{P}_{\mathbb{M}_t^{\mathrm{C}}}$ is relatively large. This can be explained as (i) in the early training stage, coarse prediction is often less accurate, *i.e.*, $\mathbf{y}^{\star}$ is often far away from $\mathbf{f}^{\mathrm{C}}\left(\mathbf{x}; \boldsymbol{\theta}_t^{\mathrm{C}}\right)$, while (ii) in the late training stage, coarse prediction becomes more accurate but also more deterministic, *i.e.*, $\sigma_t$ becomes very small. Thus, we can approximate the Shannon entropy of $\mathcal{P}_t$ as:

$$\mathbb{H}[\mathcal{P}_t] \approx (1-t) \cdot \mathbb{H}[\mathcal{P}_{\mathrm{GT}}] + t \cdot \mathbb{H}\left[\mathcal{P}_{\mathbb{M}_t^{\mathrm{C}}}\right] + \mathbb{H}(t), \quad (9)$$

where $\mathbb{H}(t) = -t \ln t - (1-t) \ln(1-t)$, which has an upper bound of $\ln 2$. On the other hand, $\mathbb{H}[\mathcal{P}_{\mathrm{GT}}]$ is smaller than $\mathbb{H}\left[\mathcal{P}_{\mathbb{M}_t^{\mathrm{C}}}\right]$ by a margin. So during the training process, $\mathbb{H}[\mathcal{P}_t]$ is mostly increasing, which implies that training difficulty becomes larger.

## 3.5. Discussions and Relationship to Prior Work

PRL provides a tradeoff between training stability and generalization ability. On the one hand, PRL allows a warm start in training the fine model. In the training process especially the early epochs, $\mathbb{M}^{\mathrm{C}}$ is often less optimized, and thus the coarse prediction $\mathbf{y}^{\mathrm{C}}$ may be less stable and introduce noise to the fine model[3]. On the other hand, although learning from $\mathbf{y}^{\star}$ is easier as it provides more accurate cues, it can easily lead to over-fitting as in the testing process, we are actually obtaining $\mathbf{y}^{\mathrm{C}}$ which is much more noisy. PRL alleviates this issue via a gradual transition from $\mathbf{y}^{\star}$ to $\mathbf{y}^{\mathrm{C}}$.

In the previous literature, the most related work is [27] and [43]. [27] considered a sequence learning task in which each cell takes the output of the previous cell as input. In each training epoch, the first part of training data are provided by ground-truth while the second provided by prediction, and the fraction was controlled by the elapsed training time $t$. Differently, PRL allows the data distribution to be changed more smoothly and thus improves training stability. [43] proposed a coarse-to-fine framework for semantic segmentation, and used a weaker version of curriculum learning in which the distribution was changed from $\mathcal{P}_{\mathrm{GT}}$ to $\mathcal{P}_{\mathbb{M}_t^{\mathrm{C}}}$ all at once. This sudden change may cause the model fail to convergence. PRL instead gradually changes the distribution, leading to consistent convergence and accuracy gain in experiments (see Section 4.2).

Last but not least, we find that curriculum learning and coarse-to-fine learning have their own benefits, and PRL combines both of them. This also makes PRL applicable to a wide range of visual recognition tasks, and we study two of them in the next section.

## 4. Applications

In this section, we apply the theory of PRL to two popular vision problems, *i.e.*, object localization (Section 4.1) and semantic segmentation (Section 4.2). We also show that the improvement brought by PRL can be directly transferred from object localization to object parsing.

## 4.1. Object Localization

### 4.1.1 Settings

In the first part, we apply PRL to object localization, which differs from object detection in that we do not need to predict the object class in both training and testing – for each input image, the desired output is a bounding box that indicates the object. While being less specific, this system can assist a wide range of vision tasks including object detection [26] and object parsing, *i.e.*, detecting the semantic parts of an object [45]. Here, we assume that only one object exists in each image, but as shown in [45], this assumption can be easily taken out by applying simple techniques in the testing process.

We collect data from the ILSVRC2012 dataset [28], in which 21 categories with the superclass of *vehicle* are

[3]According to the assumption of a coarse-to-fine approach [8][43], the fine model expects the coarse prediction $\mathbf{y}^{\mathrm{C}}$ to be "good enough", otherwise the iteration process cannot guarantee a stable convergence.

|  | $t_0$ | $E$ | STG | Scl. ↓ | Loc. ↓ | IOU ↑ | Acc. ↑ |
|---|---|---|---|---|---|---|---|
| **BL** | – | – | N/A | 29.67 | 27.50 | 74.85 | 85.74 |
| **IND** | – | – | F | 29.88 | 27.84 | 75.18 | 86.43 |
| **JNT** | – | – | C | 31.75 | 30.23 | 73.97 | 84.38 |
|  |  |  | F | 31.74 | 29.84 | 75.60 | 86.43 |
| **PRL** | 0.0 | 30 | C | 29.22 | 28.70 | 74.77 | 84.96 |
|  |  |  | F | **29.00** | 28.45 | 75.35 | 85.94 |
| **PRL** | 0.5 | 30 | C | 30.75 | 28.59 | 75.07 | 85.64 |
|  |  |  | F | 30.53 | 27.41 | **77.38** | 87.60 |
| **PRL** | 0.5 | 40 | C | 30.39 | 27.75 | 74.63 | 85.94 |
|  |  |  | F | 29.48 | 26.61 | 75.93 | **88.48** |
| **PRL** | 0.5 | 50 | C | 29.16 | 27.20 | 74.31 | 85.94 |
|  |  |  | F | 29.04 | **26.47** | 75.04 | 86.13 |

Table 1. Object localization accuracy (%) on the ILSVRC2012 *vehicle* superclass using different learning approaches and different options of PRL ($t_0$ and $E$ indicate the value of $t$ at the start of training and the number of epochs when $t$ reaches 1.0). STG indicates the stage (coarse or fine). The coarse stage of **IND** is **BL**. There are 50 training epochs in total. For detailed descriptions of these evaluation metrics, see Section 4.1.1. ↑ indicate the higher number is better, and ↓ indicates the opposite.

chosen, because the original method provided reasonable prediction on rigid objects [26]. We only choose those training and testing images with exactly one bounding-box annotated[4]. We ignore those images with more than one objects annotated to avoid confusion. In total, there are around 10,000 training and 1,000 testing images.

We take ScaleNet [26] as our baseline. The entire image is rescaled into $192 \times 192$ with its aspect ratio preserved (empty stripes are added if necessary) and fed into a 50-layer deep residual network [11] (only the middle part with 39 layers is actually used). The output consists of four floating point numbers, indicating the coordinate $(x, y)$ of the central pixel, the width $W$ and the height $H$ of the bounding box, respectively. These numbers are individually compared with the ground-truth using the log-scale $\ell_1$-norm and summed up to the final loss. There are in total 50 training epochs.

In the testing stage, we compute several evaluation metrics. Suppose the ground-truth numbers are $x^\star$, $y^\star$, $W^\star$ and $H^\star$, a prediction with $x$, $y$, $W$ and $H$ has a scale distance of $|H^\star - H| / H^\star + |W^\star - W| / W^\star$, a location distance of $|x^\star - x| / W^\star + |y^\star - y| / H^\star$, an IOU between the ground-truth and predicted bounding boxes, and an accuracy indicating whether IOU is at least 0.5[5].

[4]In the ILSVRC2012 dataset, about half of training image were not annotated with bounding-boxes, but all testing images were annotated with bounding-boxes.

[5]In both training and testing, we perform coarse and fine models exactly once (no iteration) because iteration helps little in this problem (this is partly due to the limited amount of information provided by the bounding box). For an opposite example, see the next subsection for PRL on image segmentation, in which rich information is delivered and so iteration largely improves recognition accuracy.

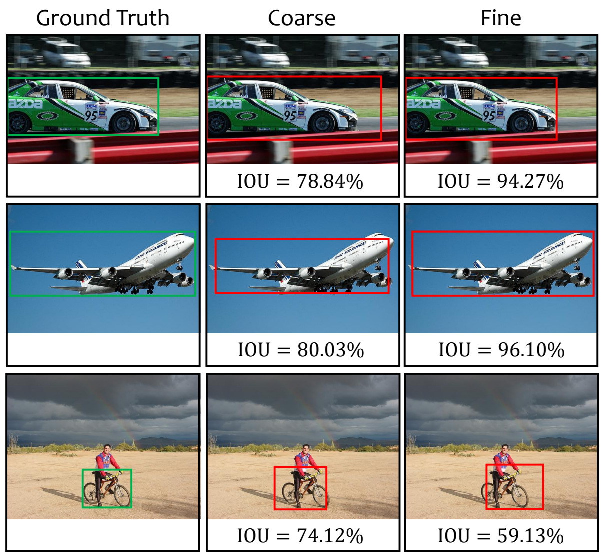


Figure 2. Two successful and one failure cases of PRL in object localization (best viewed in color). The first and second rows show the successful cases. The bottom row shows a failure case. In each row, the green frame indicates the ground-truth, and the red frame indicates the prediction.

To study this task in a coarse-to-fine manner, we first construct a weighted map using the predicted $x$, $y$, $W$ and $H$ from the coarse stage. The values within the bounding box is set to be 1 and those outside set to be 0. This map is then passed through $\mathbf{g}(\cdot)$ which is composed by two $3 \times 3$ convolutional layers and appended to $\mathbf{x}$. Although this box only provides a limited amount of information, we shall see the improvement it brings to object localization.

### 4.1.2 Different Learning Options

We compare progressive recurrent learning (**PRL**) with four other training strategies. The baseline (**BL**) simply trains one single network (*a.k.a.*, the coarse model). The individual learning (**IND**) and joint learning (**JNT**) methods trains the coarse and fine model simultaneously, but in individual and joint manner, respectively. Here, by joint optimization we mean to provide $\mathbf{y}^{\mathrm{C}}$ to $\mathbb{M}^{\mathrm{F}}$ from the beginning of training *i.e.*, $t \equiv 1$. We study different options of **PRL** defined by $t_0$ (the $t$ value at the start of training) and $E$ (the number of epochs when $t$ reaches 1), and we assume that $t$ always grows linearly with training time.

Results are summarized in Table 1. Two interesting phenomena are observed. First, starting training with a non-zero $t$ often improves performance, since when $t = 0$, the extra information is too strong so that the fine model can be severely biased towards such "cheating" information and

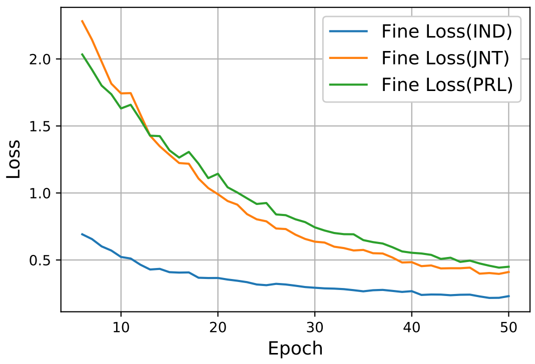


Figure 3. Learning curves of **IND**, **JNT** and **PRL** with $t_0 = 0.5$ and $E = 30$ (best viewed in color).

thus learns a weaker connection between image data and output label. In addition, it is always better for the model to be trained on $t \equiv 1$ (the same setting as in testing) for several epochs, so that the model can adjust to this scenario. Several examples showing how PRL works in localization, including a failure case, are shown in Figure 2.

In Figure 3, we compare the learning curves of **IND**, **JNT** and the best **PRL**, in terms of mean IOU, with $t_0 = 0.5$ and $E = 30$. We can see that the fine phase of **IND** achieves a very low training error by heavily over-fitting training data, in particular, with the "cheating" information from $\mathbf{y}^\star$. The loss of **JNT** is much higher at the beginning, because the fine stage is confused by the coarse stage. As training continues, the loss term becomes smaller because it starts fitting the coarse prediction. This does not bring benefit, because the potential errors in coarse prediction are not fixed. **PRL** alleviates this issue by starting with a relatively easy task in which part of data are assisted by ground-truth, and gradually moving onto the real distribution, during which ground-truth data are still provided to prevent the model from being impacted by inaccurate coarse predictions.

### 4.1.3 Application to Object Parsing

Finally, we apply the result of object localization to DeepVoting [45] for object parsing, *i.e.*, detecting the so-called semantic parts in objects. Here, each semantic part refers to a verbally describable pattern in the object, *e.g.*, the *wheels* of a *car* or the *pedal* of a *bike*. As DeepVoting required all training objects to have a fixed scale, accurate object localization (either scale and location) can help a lot in the testing process.

We use the VehicleSemanticParts (VSP) dataset introduced in [45], which was created from the *vehicle* images in Pascal3D+ [41]. There are six types of vehicles, namely, *airplane*, *bike*, *bus*, *car*, *motorbike* and *train*. There are different numbers of semantic parts annotated for each class, and we directly use the trained model for these six classes individually, *i.e.*, DeepVoting itself is not modified, and we only change the object localization module which aims at providing a better input for DeepVoting. In the testing stage, we also add random occlusion by extracting pixel-wise masks from irrelevant objects (*e.g.*, *cat* or *dog*) in the PascalVOC 2007 dataset [7] and pasting them to the input images. By controlling the number of occluders and the fraction of occlusion, we construct four levels of difficulties denoted by **L0**, **L1**, **L2** and **L3**, with **L0** indicating no occlusion, and **L3** the heaviest occlusion.

| SC | **L0** | | **L1** | | **L2** | | **L3** | |
|---|---|---|---|---|---|---|---|---|
| | **BL** | **PRL** | **BL** | **PRL** | **BL** | **PRL** | **BL** | **PRL** |
| *ai.* | 62.4 | **74.6** | 46.1 | **60.9** | 46.6 | **59.4** | 44.6 | **56.4** |
| *bi.* | 66.8 | **74.5** | 64.0 | **72.4** | 52.3 | **68.0** | 46.7 | **59.1** |
| *bu.* | 81.5 | **87.7** | 57.2 | **76.6** | 54.0 | **66.6** | 47.8 | **60.9** |
| *ca.* | 89.7 | **95.4** | **67.2** | 65.9 | **58.2** | 53.3 | **49.6** | 40.1 |
| *mo.* | 59.9 | **74.1** | 52.8 | **63.1** | 42.8 | **55.9** | 38.5 | **55.3** |
| *tr.* | 55.1 | **73.5** | 47.4 | **60.4** | 42.0 | **57.1** | 37.5 | **47.3** |
| **avg** | 69.2 | **80.0** | 55.8 | **66.5** | 49.3 | **60.1** | 44.1 | **53.2** |

| SP | **L0** | | **L1** | | **L2** | | **L3** | |
|---|---|---|---|---|---|---|---|---|
| | **BL** | **PRL** | **BL** | **PRL** | **BL** | **PRL** | **BL** | **PRL** |
| *ai.* | 60.3 | **61.3** | 40.6 | **42.6** | 32.3 | **34.8** | 25.4 | **27.4** |
| *bi.* | 90.8 | **92.3** | 85.2 | **88.4** | 79.6 | **81.6** | 62.5 | **67.8** |
| *bu.* | 81.3 | **82.1** | 65.8 | **65.9** | **54.6** | 54.3 | **40.5** | 39.9 |
| *ca.* | 80.6 | **81.1** | 57.3 | **57.6** | **41.7** | 41.3 | **29.4** | 28.0 |
| *mo.* | **69.7** | 69.3 | 55.5 | **56.5** | 43.4 | **46.5** | 31.2 | **34.3** |
| *tr.* | 61.2 | **65.3** | 43.7 | **45.8** | 29.8 | **31.6** | **22.2** | 22.1 |
| **avg** | 74.0 | **75.2** | 58.0 | **59.4** | 46.9 | **48.4** | 35.2 | **36.6** |

Table 2. Scale (SC) prediction and semantic part (SP) detection accuracies (%) on the VSP dataset, measured by a threshold of 10% relative difference and mean average precision (mAP), respectively. **L0**, **L1**, **L2** and **L3** indicate different occlusion levels.

Result are summarized in Table 2. We train two scale prediction models **BL** and **PRL** ($t_0 = 0.5$, $E = 30$) on the training set of VSP (each image provides a bounding box for the only *vehicle* in it). On the testing set, we compute both scale prediction accuracy (measured by whether it differs from the ground-truth by more than 10%, which follows the original work [45]) and semantic part detection accuracy (measured by mAP). Results are summarized in Table 2. We can see that, PRL generalizes from ILSVRC2012 to Pascal3D+ well for scale prediction, and the more accurate scale prediction indeed helps object parsing, *i.e.*, the improvement of mAP, averaged over six classes, exceeds 1% at all occlusion levels. This demonstrates the wide application of PRL.

## 4.2. Medical Imaging Segmentation

The second task is medical imaging segmentation, which serves as an important prerequisite for computer-assisted diagnosis (CAD). We investigate the scenario of CT scans which are easy to acquire yet raise the problem of organ segmentation. We follow [43] to evaluate the dataset containing 16 organs and blood vessels in 200 abdominal CT scans. These scans have different numbers of slices along the long axis of the body (the distance between neighboring slices is $0.5\mathrm{mm}$), but the spatial resolution of each slice is the same ($512 \times 512$). We evaluate each organ individually, where $150$ cases are used for training and the remaining $50$ for testing. In both datasets, we measure the segmentation accuracy by computing the Dice-Sørensen coefficient (DSC) for each sample, and report the average and standard deviation over all tested cases.

The baseline model is RSTN [43], a coarse-to-fine approach which deals with each target individually. It is a 2D-based approach, which cuts each 3D volume into slices and processes each slice separately. Three viewpoints, *i.e.*, the *coronal*, *sagittal* and *axial* views are individually trained and tested, and finally combined into prediction. In RSTN, Both $\mathbb{M}^{\mathrm{C}}$ and $\mathbb{M}^{\mathrm{F}}$ are fully-convolutional networks (FCN) [22], and $\mathbf{g}(\cdot)$ contains two convolutional layers on the segmentation mask $\tilde{\mathbf{y}}$, blurring it into a saliency map and adds it to the original image. To filter out less useful input contents, a minimal bounding box is built to cover all pixels with a probability of at least $0.5$, and the input image is cropped accordingly before fed into the fine stage. In the original paper [43], to improve the stability of RSTN, the authors designed a stepwise training strategy which first feeds the ground-truth mask $\mathbf{y}^{\star}$ into $\mathbb{M}^{\mathrm{F}}$, and changes it to $\mathbf{y}^{\mathrm{C}}$ at a fixed point of the training process. However, it still consistently failed to converge on three out of 16 targets (see Table 3), and had a probability to fail on other five.

By applying PRL, we allow the supervision signal to change gradually from $\mathbf{y}^{\star}$ to $\mathbf{y}^{\mathrm{C}}$, not suddenly. There are in total $120\mathrm{K}$ iterations with a mini-batch size of $1$. Because semantic segmentation is much more difficult than object localization, we use $t \equiv 0$ in the first $40\mathrm{K}$ iterations otherwise coarse prediction may provide a meaningless mask and thus totally confuse the fine stage. We change $t$ gradually from $0$ to $1$ in the next $40\mathrm{K}$ iterations, and set $t \equiv 1$ in the last $40\mathrm{K}$ iterations, which is learned from previous experiments. The learning rate starts with $10^{-5}$ and is divided by 2 after 90K, 100K and 110K iterations. We also tried to change $t$ linearly throughout all 120K iterations, but this achieved a worse success rate in convergence.

Results are summarized in Table 3. We can see that, after PRL is applied, RSTN achieves convergence on all 16 targets, including the 3 blood vessels which failed to converge. Among all 16 targets, PRL saves 3 of them from non-convergence, improves the segmentation accuracy of other 10, with 5 of them over $+1\%$, 3 of them over $+3\%$ and 1 of them over $+10\%$ (*small bowel*). Slight accuracy drop ($-0.01\%$) is reported on 2 out of 16 targets, and the maximal drop is $-0.80\%$ (*adrenal gland*). In overall, the average accuracy over 13 converged targets is boosted by over $2\%$, which is significant given such a high baseline and that PRL merely changes the training strategy of RSTN.

| Organ | [43]-**C** | [43]-**F** | PRL-**C** | PRL-**F** |
|---|---|---|---|---|
| *aorta*$^*$ | 90.78 | $^\sharp$90.76 | 92.09 | **93.67** |
| *adrenal gland* | 60.70 | **63.76** | 59.13 | 62.96 |
| *celiac a.a.*$^*$ | – | – | 58.29 | **61.97** |
| *colon* | 74.69 | $^\sharp$74.14 | 77.60 | **80.03** |
| *duodenum* | 71.40 | 73.42 | 67.85 | **73.46** |
| *gallbladder* | 87.08 | 87.10 | 88.21 | **89.38** |
| *inferior v.c.*$^*$ | 79.12 | 79.69 | 81.42 | **83.66** |
| *kidney left* | 96.08 | **96.21** | 95.88 | **96.23** |
| *kidney right* | 95.80 | 95.97 | 95.54 | 95.96 |
| *liver* | 96.70 | 96.75 | 96.28 | **96.93** |
| *pancreas* | 86.09 | 87.60 | 83.89 | **87.74** |
| *superior m.a.*$^*$ | – | – | 66.29 | **74.19** |
| *small bowel* | 63.86 | $^\sharp$63.52 | 71.60 | **75.05** |
| *spleen* | 96.61 | **96.78** | 96.22 | 96.77 |
| *stomach* | 94.82 | 94.98 | 94.09 | **95.12** |
| *veins*$^*$ | – | – | 72.94 | **74.77** |
| **average**: *organs* | 83.98 | 84.57 | 84.21 | **86.33** |
| **average**: *vessels* | – | – | 74.21 | **77.65** |
| **average**: *all* | – | – | 81.08 | **83.62** |

Table 3. Comparison of coarse (**C**) and fine (**F**) segmentation by [43] and the improved version based on PRL. A target is marked by an asterisk if it is a blood vessel. The original version of RSTN [43] cannot achieve convergence on three blood vessels (marked by $-$). A fine-scaled accuracy is indicated by $\sharp$ if it is lower than the coarse-scaled one.

## 5. Conclusions

In this paper, we generalize curriculum learning to a wide range of vision problems. Our approach, named progressive recurrent learning (PRL), is motivated by that curriculum learning can be integrated with coarse-to-fine learning, so that the former provides a more stable training scheme, and the latter provides a natural way of constructing training data with varying difficulties. PRL is evaluated in two vision problems, namely, object localization and semantic segmentation. In both scenarios, PRL consistently improves the accuracy of visual recognition.

This paper leaves many topics for future research. For example, it remains unclear if there exists some specific strategies for each particular task which take full advantage of its properties. Also, the strategy of monotonically increasing the difficulty of training data may not be perfect, as some prior work [12] verified that disturbing the training process can lead to better model ensemble. Studying these topics may provide new perspectives to understand machine learning, in particular deep learning methods.

## References

[1] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015. 2

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009. 1, 2, 3

[3] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *International Conference on Computer Vision*, 2015. 2

[4] H. Chen, Q. Dou, X. Wang, J. Qin, and P. A. Heng. Mitosis detection in breast cancer histology images via deep cascaded networks. In *AAAI Conference on Artificial Intelligence*, 2016. 2

[5] Q. Chen, W. Qiu, Y. Zhang, L. Xie, and A. Yuille. Sampleahead: Online classifier-sampler communication for learning from synthesized data. *arXiv preprint arXiv:1804.00248*, 2018. 2

[6] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, 2009. 2

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 7

[8] S. Gangaputra and D. Geman. A design principle for coarse-to-fine classification. In *Computer Vision and Pattern Recognition*, 2006. 2, 5

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010. 1

[10] J. Gu, J. Cai, G. Wang, and T. Chen. Stack-captioning: Coarse-to-fine learning for image captioning. 2018. 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, 2016. 1, 2, 6

[12] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*, 2018. 9

[13] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition*, 2017. 2

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015. 1, 2

[15] F. Khan, B. Mutlu, and X. Zhu. How do humans teach: On curriculum learning and teaching dimension. In *Advances in Neural Information Processing Systems*, 2011. 2

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 1, 2

[17] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015. 2

[18] C. Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, 2015. 4

[19] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016. 2

[20] Q. Li, J. Wang, D. Wipf, and Z. Tu. Fixed-point model for structured labeling. In *International Conference on Machine Learning*, 2013. 2

[21] C. Liu, B. Zoph, J. Shlens, W. Hua, L. J. Li, L. Fei-Fei, A. L. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *European Conference on Computer Vision*, 2018. 4

[22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*, 2015. 8

[23] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, 2010. 1, 2, 4

[24] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition*, 2015. 1, 2

[25] A. Pentina, V. Sharmanska, and C. H. Lampert. Curriculum learning of multiple tasks. In *Computer Vision and Pattern Recognition*, 2015. 2

[26] S. Qiao, W. Shen, W. Qiu, C. Liu, and A. L. Yuille. Scalenet: Guiding object proposal generation in supermarkets and beyond. In *International Conference on Computer Vision*, 2017. 5, 6

[27] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. 2016. 2, 5

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[29] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 4

[30] N. Sarafianos, T. Giannakopoulos, C. Nikou, and I. A. Kakadiaris. Curriculum learning for multi-task classification of visual attributes. *arXiv preprint arXiv:1708.08728*, 2017. 2

[31] R. Sharma, S. Barratt, S. Ermon, and V. Pande. Improved training with curriculum gans. *arXiv preprint arXiv:1807.09295*, 2018. 2

[32] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Computer Vision and Pattern Recognition*, 2016. 2

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 2

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 1, 2

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, 2015. 2, 4

[36] Y. Tang, X. Wang, A. P. Harrison, L. Lu, J. Xiao, and R. M. Summers. Attention-guided curriculum learning for weakly supervised classification and localization of thoracic diseases on chest radiographs. In *International Workshop on Machine Learning in Medical Imaging*, 2018. 2

[37] Z. Tu. Auto-context and its application to high-level vision tasks. In *Computer Vision and Pattern Recognition*, 2008. 2

[38] S. E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *Computer Vision and Pattern Recognition*, 2016. 2

[39] D. Weinshall and G. Cohen. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International Conference on Machine Learning*, 2018. 2

[40] Y. Wu and Y. Tian. Training agent for first-person shooter game with actor-critic curriculum learning. 2017. 2

[41] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, 2014. 7

[42] Y. Yang, Y. Wang, Q. M. J. Wu, X. Lin, and M. Liu. Progressive learning machine: A new approach for general hybrid system approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 26(9):1855–1874, 2015. 4

[43] Q. Yu, L. Xie, Y. Wang, Y. Zhou, E. K. Fishman, and A. L. Yuille. Recurrent saliency transformation network: Incorporating multi-stage visual cues for small organ segmentation. In *Computer Vision and Pattern Recognition*, 2018. 2, 4, 5, 8

[44] W. Zaremba and I. Sutskever. Reinforcement learning neural turing machines-revised. *arXiv preprint arXiv:1505.00521*, 2015. 2

[45] Z. Zhang, C. Xie, J. Wang, L. Xie, and A. L. Yuille. Deepvoting: An explainable framework for semantic part detection under partial occlusion. In *Computer Vision and Pattern Recognition*, 2018. 5, 7

[46] T. Zhou and J. Bilmes. Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *International Conference on Learning Representations*, 2018. 2

[47] Y. Zhou, L. Xie, W. Shen, Y. Wang, E. K. Fishman, and A. L. Yuille. A fixed-point model for pancreas segmentation in abdominal ct scans. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2017. 2