Incremental Object Detection via Meta-Learning

K J Joseph, J Rajasegaran, S H Khan, F S Khan and V N Balasubramanian

Abstract—In a real-world setting, object instances from new classes can be continuously encountered by object detectors. When existing object detectors are applied to such scenarios, their performance on old classes deteriorates significantly. A few efforts have been reported to address this limitation, all of which apply variants of knowledge distillation to avoid catastrophic forgetting. We note that although distillation helps to retain previous learning, it obstructs fast adaptability to new tasks, which is a critical requirement for incremental learning. In this pursuit, we propose a meta-learning approach that learns to reshape model gradients, such that information across incremental tasks is optimally shared. This ensures a seamless information transfer via a meta-learned gradient preconditioning that minimizes forgetting and maximizes knowledge transfer. In comparison to existing meta-learning methods, our approach is task-agnostic, allows incremental addition of new-classes and scales to high-capacity models for object detection. We evaluate our approach on a variety of incremental learning settings defined on PASCAL-VOC and MS COCO datasets, where our approach performs favourably well against state-of-the-art methods.

Index Terms—Object Detection, Incremental Learning, Deep Neural Networks, Meta-learning, Gradient preconditioning.

1 Introduction

Deep learning has brought about remarkable improvements on numerous vision tasks, including object detection [1], [2], [3]. Most existing detection models make an inherent assumption that examples of all the object classes are available during the training phase. In reality, new classes of interest can be encountered on the go, due to the dynamic nature of the real-world. This makes the existing methods brittle in an incremental learning setting, wherein they tend to forget old task information when trained on a new task [4].

In this work, we study the *class-incremental* object detection problem, where new classes are sequentially introduced to the detector. An intelligent learner must not forget previously learned classes, while learning to detect new object categories. To this end, knowledge distillation [5] has been utilized as a de facto solution [6], [7], [8], [9]. While learning a new set of classes, distillation based methods ensure that the classification logits and the regression targets of the previous classes, are not altered significantly from the earlier state of the model. As a side effect, distillation enforces intransigence in the training procedure, making it hard to learn novel classes. An essential characteristic for incremental object detectors is to have optimal plasticity, which aids in quick adaptability to new classes without losing grasp of previously acquired knowledge.

Learning to learn for quick adaptability forms the basis of current meta-learning methods [10], [11], [12]. These methods have generally been successful in few-shot learning settings. Directly adopting such methodologies to incremental object detection is challenging due to the following reasons: (a) A meta-learner is explicitly trained with a *fixed* number of classes (N-way classification) and does not generalize to an incremental setting. (b) Each task in the meta-train and meta-test stages is carefully designed to avoid task-overfitting [13]. This is prohibitive in an object detection setting as each image can possibly have multi-class instances. (c) The meta-learners require knowledge about the end-task for fine-tuning, while in the incremental setting a test sample can belong to any of the classes observed so-far, implicitly demanding task-agnostic inference. (d) Network architectures meta-learned in a traditional setting are in the order of a few convolutional layers. This comes in stark contrast to an object detector which involves multiple sub-networks for generating backbone features, object proposals and final classification and localization outputs.

We propose a methodology that views incremental object detection through the lens of meta-learning that can effectively deal with the above challenges. Our meta-learning procedure learns to modify the gradients such that quick adaptation across multiple incremental learning tasks is possible. This is efficiently realised by meta-learning a set of gradient preconditioning matrices, interleaved between layers of a standard object detector (Sec. 3.2). Further, we formulate a meta-training objective to learn these gradient preconditioning matrices (Sec. 3.3.2). In this way, our meta-training procedure captures properties of all the incrementally presented task distributions, effectively alleviating forgetting and preparing the networks for quick adaptation.

The key contributions of our work are:

- We propose a gradient-based meta-learning approach which learns to reshape gradients such that optimal updates are achieved for both the old and new tasks, for class-incremental object detection problem.
- We propose a novel loss formulation that counters the intransigence enforced due to knowledge distillation, by learning a generalizable set of gradient directions that alleviates forgetting and improves adaptability.
- Our extensive evaluations on two benchmark datasets against three competitive baseline methods shows the utility of our meta-learning based methodology.

- *K. J. Joseph and V. N. Balasubramanian are with the Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad, India. E-mail: cs17m18p100001@iith.ac.in*
- *K. J. Joseph, J. Rajasegaran, S. H. Khan and F. S. Khan are with MBZ University of AI, Abu Dhabi, UAE.*

(a) Learning $\mathcal{T}_1$ (b) Learning $\mathcal{T}_2$ (c) Incrementally learning $\mathcal{T}_2$

Fig. 1. (a) In Task 1 ($\mathcal{T}_1$), a standard object detector (Faster R-CNN [2]) is trained to detect the `'bird'` class and it can accurately detect `'bird'` instances on a test image. (b) In Task 2 ($\mathcal{T}_2$), the same model is trained to detect `'person'` class and it accurately detects a `'person'` instance. However, the detector forgets the `'bird'` class ($\mathcal{T}_1$) which was not present during $\mathcal{T}_2$ training. (c) Our meta-learning based incremental Faster R-CNN detector accurately detects most instances of *both* the classes.

2 Related Work

Our proposed methodology lies at the intersection of incremental learning and meta-learning. Hence, we review the literature from both these paradigms here.

Incremental Learning: In comparison to the incremental setting for image-classification [14], [15], [16], [17], class-incremental object detection and class-incremental semantic segmentation [18], [19] has been relatively less explored. Most of the methods proposed so far [6], [7], [8], [9] use knowledge distillation [5] to address catastrophic forgetting. These methods mainly vary in the base object detector or parts of the network that are distilled. Shmelkov *et al*. [6] proposed an incremental version of Fast R-CNN [20], which uses pre-computed Edge Box object proposal algorithm [21], making the setting simpler than what we consider, where the proposal network is also learned. While training for a new task, the classification and regression outputs are distilled from a copy of the model trained on the previous task. Li *et al*. [7] and Chen *et al*. [9] proposed to distill the intermediate features, along with the network outputs. Hao *et al*. [8] expands the capacity of the proposal network along with distillation. Despite all these efforts, the methods fail to improve the benchmark [6] on the standard evaluation criteria, which calls for thoughts on the effectiveness of the distillation methods and the complexity of the task at hand. More recently, Acharya *et al*. [22] proposed to used memory replay to solve a newly formulated online object detection setting, while Joseph *et al*. [23] explicitly characterised the unknown objects, which was found to be effective in alleviating forgetting. Peng *et al*. [24] improved the state-of-the-art by using Faster R-CNN with the distillation methodology from [6]. Zhang *et al*. [25] proposed to distill a consolidated model from a base and incremental detector, using unlabelled auxiliary data. In this work, we hypothesise that the restraining effect of distillation may lead to hindrance towards learning new tasks. Therefore to learn a better incremental object detector, the distillation must be carefully modulated with learning for generalizability to new-tasks. To this end, meta-learning offers an attractive solution.

Meta Learning: Meta-learning algorithms can be broadly classified into optimization based methods [10], [11], [26], black-box adaptation methods [27], [28] and non-parametric methods [12], [29]. Adapting these meta-learning methods, which are successful in few-shot image classification setting, to object detection is not straightforward for the reasons enumerated in Sec. 1. Recently, meta-learning has been applied to address k-shot object detection setting. Wang *et al*. [30] learn a meta-model that predicts the weights of the RoI Head, that is finally fine-tuned. Yan *et al*. [31] and Kang *et al*. [32] learn to re-weight the RoI features and backbone features of Faster R-CNN [2] and YOLO [3], respectively. Unlike these methodologies, we adopt to use a gradient based meta-learning technique to tackle incremental object detection. Inspired by Flennerhag *et al*. [33], we propose to meta-learn a gradient preconditioning matrix that encapsulates information across multiple learning tasks. Through extensive experiments, we show that the proposed approach is effective in learning a detector that can be continually adapted to handle new classes.

3 Methodology

The standard object detection frameworks [1], [3], [20], [34] can be characterised as a function ($\mathcal{F}_{OD}$), that takes an input image and transforms it into a set of bounding boxes enclosing objects, each of which is classified into one of the classes, known a priori. $\mathcal{F}_{OD}$ is trained on large amounts of annotated data corresponding to each class, using variants of stochastic gradient descent. A *class-incremental object detector* relaxes the constraint that all the class data is available beforehand. As and when new class information is available, the detector should modify itself to be competent on detecting the new classes along with the old classes by combating itself against catastrophic forgetting [4], [35].

We formally define the problem in Sec. 3.1, introduce how we meta-learn the gradient preconditioning matrix in Sec. 3.2 and finally explain the specifics of our proposed incremental object detector in Sec. 3.3.

3.1 Problem Formulation

Let $\mathcal{C}$ denotes the set of classes that are incrementally introduced to the object detector. A task $\mathcal{T}_t$, is defined as a grouping of these classes, which are exposed to the detector at time t: $\mathcal{T}_t \subset \mathcal{C}$, such that $\mathcal{T}_i \cap \mathcal{T}_j = \emptyset$, for any $i, j \leq t$. Let $\mathcal{D}_t$ denote the images containing annotated objects of classes in $\mathcal{T}_t$. Each image can contain multiple objects of different classes, however annotations are available only for those object instances that belong to classes in $\mathcal{T}_t$.

Let $\boldsymbol{I} \in \mathcal{D}_t$ denotes an input image. An object detector $\mathcal{F}_{OD}(\boldsymbol{I})$, is a composition of functions: $(\mathcal{F}_{RoI\_Head} \circ \mathcal{F}_{RPN} \circ$

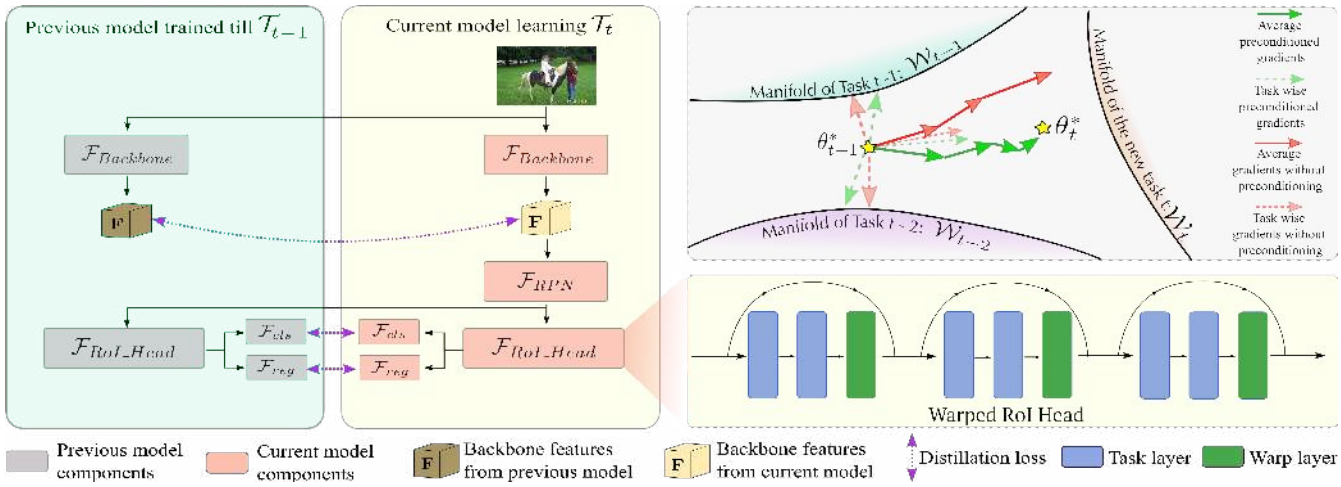

Fig. 2. The figure outlines the overall architectural components and an illustration of how gradient preconditioning controls learning (*top right*). While learning the current task $\mathcal{T}_t$, the gradient preconditioning induced by the warp layers (green rectangles in the RoI Head of the detector) effectively modulates the dotted-red task-wise gradients to the dotted-green gradients, which inherently guides the average gradients (solid green arrows) to a better optima that respects all three task manifolds $\mathcal{W}_t, \mathcal{W}_{t-1}$ and $\mathcal{W}_{t-2}$. Additionally, backbone features and RoI Head outputs are distilled (purple arrows) from the previous model. *(best viewed in color)*

$\mathcal{F}_{Backbone})(\boldsymbol{I})$. Here, without loss of generality, we focus on R-CNN family [2], [20], [34] of two-stage detectors. $\mathcal{F}_{Backbone}$ takes $\boldsymbol{I}$ as input and generates a feature map $\boldsymbol{F} \in \mathbb{R}^{C\times H\times W}$ where C, H and W refer to the number of channels, height and width of the feature map respectively. $\mathcal{F}_{RPN}$ takes these features and proposes areas which can possibly contain an object. Concretely, it outputs a class-agnostic objectness score and the bounding box location for N proposals. Each of these proposals (which is RoI pooled feature from $\boldsymbol{F}$) is classified into one of $\mathcal{T}_{i\leq t}$ classes and its bounding box locations are regressed by F_{RoI_Head}. Let $\mathcal{F}_{OD}$ be parameterised by $\boldsymbol{\theta}$. The challenge in class incremental object detection, is to continually adapt $\boldsymbol{\theta}$ to learn new tasks $\mathcal{T}_{t+1}$, without access to all of $\{\mathcal{D}_0 \cdots \mathcal{D}_t\}$, while maintaining original performance on $\{\mathcal{T}_0 \cdots \mathcal{T}_t\}$.

3.2 Meta-Learning the Gradient Preconditioning

The standard update rule while training an object detector $\mathcal{F}_{OD}$ parameterised by $\boldsymbol{\theta}$ is: $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} - \mu\nabla\mathcal{L}(\boldsymbol{\theta})$, where $\mathcal{L}$ is the loss function and μ is the learning rate. We intend to meta-learn a parameterised preconditioning matrix $P(\boldsymbol{\theta};\boldsymbol{\phi})$, which warps the gradient to the steepest direction accounting for the different tasks introduced to the object detector till then. The parameter update is thus given by: $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} - \mu P(\boldsymbol{\theta};\boldsymbol{\phi})\nabla\mathcal{L}(\boldsymbol{\theta})$, where $\boldsymbol{\phi}$ are parameters of P.

A scalable way to achieve such a gradient preconditioning is to embed it directly into the task learner [33], [36], [37]. Following Flennerhag *et al.* [33], we dedicate some of the network layer for preconditioning (called warp layers), which are meta-learned across incremental learning tasks. Hence, the parameter set $\boldsymbol{\theta}$ of $\mathcal{F}_{OD}$ is split into task parameters $\boldsymbol{\psi}$ and warp parameters $\boldsymbol{\phi}$, such that $\boldsymbol{\theta} = \boldsymbol{\psi} \cup \boldsymbol{\phi}$ and $\boldsymbol{\psi} \cap \boldsymbol{\phi} = \emptyset$. During back-propagation, gradient preconditioning is inherently induced on the task layers by the Jacobians of the warp-layers. As the warp layers are non-linear transformations, it allows modelling rich relationships between gradients. This allows for stronger representational capability than previous works that consider preconditioning via block diagonal matrices [38], [39].

The warp parameters which precondition the gradients are learned using information from all the tasks seen till then. The carefully defined loss function $\mathcal{L}_{warp}$ (Eqn. 5) helps achieve this objective. These layers implicitly help to model the joint task distribution of all the tasks introduced to the detector. This results in better generalization to new tasks, faster convergence and alleviates catastrophic forgetting. The illustration in Fig. 2 (top right) explains how gradient preconditioning via warp layers controls learning. Before learning $\mathcal{T}_t$, let θ^*_{t-1} be the optimal parameters that lie close to the previously learned task manifolds: $\mathcal{W}_{t-1}$ and $\mathcal{W}_{t-2}$. While learning the new task $\mathcal{T}_t$, the preconditioning layers effectively warp the task-wise gradients. The transformation of the red dotted arrows to green dotted arrows pictorially shows the change induced by the preconditioning matrix to the task-wise gradients in the first step (the task-wise gradients for the subsequent descent steps are not shown to avoid clutter). Thus, gradient warping helps the average task gradients, shown in green, to converge to a better optima θ^*_t, which respects all the task manifolds, $\mathcal{W}_t, \mathcal{W}_{t-1}$ and $\mathcal{W}_{t-2}$.

3.3 Incremental Object Detector

Fig. 2 illustrates the end-to-end architecture of the meta-learned incremental object detector. Faster R-CNN [2] is adapted to incorporate additional warp layers which preconditions the gradient flow. The input image is passed through the backbone network ($\mathcal{F}_{Backbone}$) to generate a set of features $\boldsymbol{F}$, which are in turn passed on to the RPN and the RoI pooling layers ($\mathcal{F}_{RPN}$) to generate object features. These are passed to the RoI Head ($\mathcal{F}_{RoI\_Head}$) which contains a set of three residual blocks, each with three convolutional layers. Amongst these convolutional layers, we designate one of them as warp layer (colored green), which is meta-learned using warp loss, $\mathcal{L}_{warp}$. All the other layers of the network constitute the task layers, which are learned using task loss, $\mathcal{L}_{task}$. $\mathcal{F}_{RoI\_Head}$ terminates in a multi-class classification head ($\mathcal{F}_{cls}$) and a regression head ($\mathcal{F}_{reg}$). While learning a new task, distilling backbone

features and final heads from the previous task helps to add additional constraint which guides current learning. Additional implementation details are discussed in Sec. 4.3.

Below, we explain how $\mathcal{L}_{task}$ and $\mathcal{L}_{warp}$ are formulated, followed by the learning and inference strategies.

3.3.1 Task Loss ($\mathcal{L}_{task}$):

A standard object detector is learned by minimising the classification error and the bounding box localization error based on the predictions from the classification and regression heads of $\mathcal{F}_{RoI\_Head}$. Simultaneously, it reduces the discrepancy in objectness score predicted by $\mathcal{F}_{RPN}$ and the corresponding bounding box offsets from the ground-truth. Let $\boldsymbol{p} = (p_0, \cdots, p_K)$ denote the class probabilities for $K+1$ classes (K object classes + background class) and let $\boldsymbol{l} = (l_x, l_y, l_w, l_h)$ denote the bounding box locations predicted by $\mathcal{F}_{RoI\_Head}$ for each of RoI pooled feature. Let the ground-truth class and bounding box regression targets be p^* and $\boldsymbol{l}^*$. Then, $\mathcal{L}_{RoI\_Head}$ is defined as follows:

$$\mathcal{L}_{RoI\_Head} = \mathcal{L}_{cls}(\boldsymbol{p}, p^*) + \lambda[p^* \geq 1]\mathcal{L}_{loc}(\boldsymbol{l}, \boldsymbol{l}^*), \quad (1)$$

where $\mathcal{L}_{cls}(\boldsymbol{p}, p^*) = -\log p_{p^*}$ is the log loss for the true class p^* and $\mathcal{L}_{loc}$ is the robust smooth L_1 loss function defined in [20]. $p^* = 0$ denotes the background class, and the localisation loss is not calculated for them. In a similar manner, the loss for training $\mathcal{F}_{RPN}$ which generates an objectness score $o \in [0, 1]$ and the corresponding bounding box predictions $\boldsymbol{l}$ is defined as follows:

$$\mathcal{L}_{RPN} = \mathcal{L}_{cls}(o, o^*) + \lambda o^* \mathcal{L}_{loc}(\boldsymbol{l}, \boldsymbol{l}^*). \quad (2)$$

where o^* is the ground truth which denotes whether the region features contains an object ($= 1$) or not ($= 0$) and $\boldsymbol{l}^*$ is the bounding box regression target. The weighting parameter λ is set to 1 for all experiments following [2].

While adapting the current detector $\mathcal{F}_{OD}^{\boldsymbol{\theta}_t}$, with parameters $\boldsymbol{\theta}_t$ for a new task $\mathcal{T}_t$, we use a frozen copy of the previous model $\mathcal{F}_{OD}^{\boldsymbol{\theta}_{t-1}}$, to distill the backbone features and the RoI_head targets. This will ensure that $\mathcal{F}_{OD}^{\boldsymbol{\theta}_t}$ does not deviate too much from $\mathcal{F}_{OD}^{\boldsymbol{\theta}_{t-1}}$; which indeed acts as a knowledgeable teacher who has expertise in detecting the previously known classes. Each training image is passed through $\mathcal{F}_{Backbone}^{\boldsymbol{\theta}_t}$ and $\mathcal{F}_{Backbone}^{\boldsymbol{\theta}_{t-1}}$ to obtain $\boldsymbol{F}_t$ and $\boldsymbol{F}_{t-1}$. Each RoI pooled feature from $\mathcal{F}_{RPN}^{\boldsymbol{\theta}_t}$ is passed to $\mathcal{F}_{RoI\_Head}^{\boldsymbol{\theta}_t}$ and $\mathcal{F}_{RoI\_Head}^{\boldsymbol{\theta}_{t-1}}$ to obtain $\{\boldsymbol{p}_t, \boldsymbol{l}_t\}$ and $\{\boldsymbol{p}_{t-1}, \boldsymbol{l}_{t-1}\}$ respectively. See purple arrows in Fig. 2 for clarity. Distillation loss is defined as:

$$\mathcal{L}_{Distill} = \mathcal{L}_{Reg}(\boldsymbol{F}_t, \boldsymbol{F}_{t-1}) + \mathcal{L}_{KL}(\boldsymbol{p}_t, \boldsymbol{p}_{t-1}) + \mathcal{L}_{Reg}(\boldsymbol{l}_t, \boldsymbol{l}_{t-1}), \quad (3)$$

where $\mathcal{L}_{Reg}$ is the L_2 regression loss and $\mathcal{L}_{KL}$ is the KL divergence between the probability distribution generated by current and previous classification heads, computed only for the previously seen classes. The final task loss is a convex combination of the detection and the distillation losses,

$$\mathcal{L}_{task} = \alpha\mathcal{L}_{Distill} + (1-\alpha)(\mathcal{L}_{RPN} + \mathcal{L}_{RoI\_Head}) \quad (4)$$

where α is the weighting factor which controls the importance of each of the term in the loss function. We run a sensitivity analysis on the value of α in Sec. 5.

3.3.2 Warp Loss ($\mathcal{L}_{warp}$):

Motivated by the incremental learning methods for image classification [15], [40], we maintain an Image Store (I_{Store}) with a small number of exemplar images per class. We make use of the images from I_{Store} and the current model parameters $\boldsymbol{\theta}_t$, to define the warp loss $\mathcal{L}_{warp}$. We meta-learn the warp layers in $\mathcal{F}_{RoI\_Head}$ (refer Fig. 2), which classifies each of the input RoI pooled features into one of the classes and predicts its bounding box locations, using $\mathcal{L}_{warp}$. Though I_{Store} guarantees a lower bound on the number of images per class, the actual per class instance statistics would be much uneven. This is because each image can contain multiple instances of different classes. This will heavily bias the warp layer training towards those classes which has more instances. To combat this, we propose to use a feature store F_{Store}, which stores N_{feat} features per class. This is realised by maintaining a fixed size queue per class and queuing the RoI pooled features into the corresponding class specific queue. This ensures that even if there are multiple instances of many classes in the training data of the detector, the warp layers are updated with equal priority to all the classes. This is a key component that ensures that the gradient preconditioning that is meta-learned would retain equal importance to all the classes. Further, as I_{Store} contains images and annotations from all the classes seen till then, this implicitly embeds information from not only the current class but also the previous classes into the preconditioning layers and therein the whole network, via the warped gradients, effectively reducing forgetting.

Let $\boldsymbol{f}$ denote a single RoI Pooled feature and p^* and $\boldsymbol{l}^*$ denote the corresponding true class label and the bounding box annotation. $\boldsymbol{f}$ is passed through the RoI head $\mathcal{F}_{RoI\_Head}$, to generate class predictions $\boldsymbol{p}$, and box predictions $\boldsymbol{l}$. Then, the warp loss $\mathcal{L}_{warp}$ from the features and labels stored in F_{Store} is computed as,

$$\mathcal{L}_{warp} = \sum_{(\boldsymbol{f}, p^*, \boldsymbol{l}^*) \sim F_{Store}} \mathcal{L}_{cls}(\boldsymbol{p}, p^*) + [p^* \geq 1]\mathcal{L}_{loc}(\boldsymbol{l}, \boldsymbol{l}^*), \quad s.t., (\boldsymbol{p}, \boldsymbol{l}) = F_{RoI\_Head}(\boldsymbol{f}) \quad (5)$$

Here, $\mathcal{L}_{cls}$ is the log loss and $\mathcal{L}_{loc}$ is a smooth L_1 regression loss. Algorithm 1 illustrates the warp loss computation. Each image in I_{Store} is passed through $\mathcal{F}_{Backbone}$ and $\mathcal{F}_{RPN}$ to generate the RoI pooled features and the associated labels, which is then queued into F_{Store} (Lines 3 - 4). Once all the RoI features are extracted, we accumulate per RoI loss to compute the warp loss $\mathcal{L}_{warp}$ (Lines 6 - 10).

3.3.3 Learning and Inference:

Algorithm 2 summarises the end-to-end learning strategy. A mini-batch of datapoints $\mathcal{D}_{tr}$ from the current task is sampled from $\mathcal{D}_t$ in Line 3. As introduced in Sec. 3.3.2, we maintain an Image Store (I_{Store}) with a small number of exemplar images per class. Specifically, I_{Store} contains one fixed size (N_{img}) queue per class, which is used to meta-learn the wrap layers. The fixed size queue ensures that only the recently seen N_{img} images per class would be maintained in the store. Images from $\mathcal{D}_{tr}$ are added to I_{Store} in Line 5. For images with multiple class objects in it, we associate it with one of its randomly chosen constituent class. The task loss $\mathcal{L}_{task}$ is computed using Eq. 4 and the task parameters $\boldsymbol{\psi}_t$ are updated in Lines 6 and 7. The warp

Algorithm 1 Algorithm GETWARPLOSS

Input: Image store: I_{Store}; Current model params: $\boldsymbol{\theta}_t = \boldsymbol{\psi}_t \cup \boldsymbol{\phi}_t$.

1: Initialise F_{Store} ▷ *A queue of length N_{feat} per class*
2: **for** $\mathcal{I} \in I_{Store}$ **do**
3: $\{(f_{ROI\_pooled}, \text{labels})\} \leftarrow \mathcal{F}_{RPN}(\mathcal{F}_{Backbone}(\mathcal{I}))$
4: Enqueue F_{store} with $\{(f_{ROI\_pooled}, \text{labels})\}$
5: $\mathcal{L}_{warp} \leftarrow 0$
6: **for** $f, labels \in F_{Store}$ **do**
7: $p^*, \boldsymbol{l}^* \leftarrow labels$
8: $\boldsymbol{p}, \boldsymbol{l} \leftarrow \mathcal{F}_{RoI\_Head}(f)$
9: $\mathcal{L}_{per\_RoI} = \mathcal{L}_{cls}(\boldsymbol{p}, p^*) + [p^* \geq 1]\mathcal{L}_{loc}(\boldsymbol{l}, \boldsymbol{l}^*)$ ▷ *Eq. 5*
10: $\mathcal{L}_{warp} \leftarrow \mathcal{L}_{warp} + \mathcal{L}_{per\_RoI}$
11: **return** $\mathcal{L}_{warp}$

parameters are updated with warp loss $\mathcal{L}_{warp}$ after every γ iterations using I_{store} in Line 11. We re-emphasise that while the task layers are updated, the warp layers a kept fixed, which effectively preconditions the task gradients. These preconditioning matrices (layers) are updated using the warp loss by using all the images in I_{Store}, which contains images of all the classes introduced to the detector till then, infusing the global information across classes into the task layers. A key aspect of detection setting is its innate incremental nature. Even two images from the same task $\mathcal{T}_t$ may not share all same classes. We exploit this behaviour to learn our warp layers even while training on the first task.

Similar to other meta-learning approaches [10], [11], [26], at inference time, the images in I_{Store} are used to fine-tune the task parameters. In this process, meta-learned preconditioning matrix effectively guides the gradients in the steepest descent direction resulting in quick adaptation. In our experiments, we find that with just 10 examples per task, the model exhibits superior performance. The rest of the inference pipeline follows standard Faster R-CNN [2].

Algorithm 2 Learning a current task $\mathcal{T}_t$

Input: Current model params: $\boldsymbol{\theta}_t = \boldsymbol{\psi}_t \cup \boldsymbol{\phi}_t$; Previous model params: $\boldsymbol{\theta}_{t-1} = \boldsymbol{\psi}_{t-1} \cup \boldsymbol{\phi}_{t-1}$; Data for $\mathcal{T}_t$: $\mathcal{D}_t$; Image store: I_{Store}; Warp update interval: γ; Step len: μ, υ.

while until required iterations **do**
$i \leftarrow 0$
$\mathcal{D}_{tr} \leftarrow$ Sample a mini-batch from $\mathcal{D}_t$
for $\mathcal{I} \in \mathcal{D}_{tr}$ **do**
Add $\mathcal{I}$ to I_{Store}
$\mathcal{L}_{task} \leftarrow$ Compute using Eq. 4
$\boldsymbol{\psi}_t \leftarrow \boldsymbol{\psi}_t - \mu \nabla \mathcal{L}_{task}$ ▷ *Task-parameters update*
$i \leftarrow i + 1$
if $i\%\gamma == 0$ **then**
$\mathcal{L}_{warp} \leftarrow$ GetWarpLoss($\boldsymbol{\theta}_t, I_{store}$) ▷ *Alg. 1*
$\boldsymbol{\phi}_t \leftarrow \boldsymbol{\phi}_t - \upsilon \nabla \mathcal{L}_{warp}$ ▷ *Meta-parameters update*
return $\boldsymbol{\theta}_t, I_{Store}$

4 EXPERIMENTS AND RESULTS

We evaluate the proposed approach on a variety of class incremental settings across two prominent detection datasets. We compare against the state-of-the-art methods, consistently outperforming them in all the settings. Below, we introduce the datasets, explain the experimental settings, provide implementation details and report our results.

4.1 Datasets and Evaluation Metrics

To benchmark our method, we evaluate on PASCAL VOC 2007 [41] and MS COCO 2014 [42] datasets following Shmelkov *et al*. [6]. PASCAL VOC 2007 contains 9,963 images containing 24,640 annotated instances of 20 categories. Following the standard setting [41], 50% of data is split into train/val splits and the rest for testing on PASCAL VOC. MS COCO 2014 contains objects from 80 different categories with 83,000 images in its training set and 41,000 images in the validation set. Since the MS COCO test set is not available, we use the validation set for evaluation.

The mean average precision at 0.5 IoU threshold (mAP@50) is used as the primary evaluation metric for both datasets. For MS COCO, we additionally report average precision and recall across scales, the number of detections and IoU thresholds, in line with its standard protocol.

4.2 Experimental Settings

Following [6], we simulate incremental versions of both PASCAL VOC and MS COCO datasets. As introduced in Sec. 3.1, a group of classes constitute a task $\mathcal{T}_t$, which is presented to the learner at time t. Let C denotes the set of classes that are part of $\mathcal{T}_t$. The data $\mathcal{D}_t$ for task $\mathcal{T}_t$ is created by selecting those images which have any of the classes in C. Instances of those classes that are not part of C, but still co-occur in a selected image would be left unlabelled.

For PASCAL VOC, we order the classes alphabetically and create multiple tasks by grouping them. We consider four different settings, in the decreasing order of difficulty: (a) the first task $\mathcal{T}_1$ containing initial 15 classes and the next five successive tasks ($\mathcal{T}_2 \cdots \mathcal{T}_6$) containing a new class each. (b) $\mathcal{T}_1$ containing first 15 classes and $\mathcal{T}_2$ containing the rest of 5 classes. (c) $\mathcal{T}_1$ containing first 10 classes and $\mathcal{T}_2$ containing the other 10 classes. (d) Grouping all the initial 19 classes in $\mathcal{T}_1$ and the last class into $\mathcal{T}_2$. For MS COCO, we use the first 40 classes as task $\mathcal{T}_1$ and the rest as $\mathcal{T}_2$.

4.3 Implementation Details

We build our incremental object detector based on Faster R-CNN [2]. Continually learning a Faster R-CNN is more challenging than the case of Fast R-CNN (as deployed in Shmelkov *et al*. [6]), since it makes use of pre-computed Edge Box proposals [21], while we also learn the RPN. To maintain fairness in comparison with [6], we use ResNet-50 [43] with frozen batch normalization layers as the backbone.

The classification head of Faster R-CNN handles only the classes seen so far. Following other class incremental works [16], [44], [45], this is done by setting logits of the unseen class to a very high negative value (-10^{10}). This makes the contribution of the unseen classes in the softmax function negligible ($e^{-10^{10}} \rightarrow 0$), while computing the class probabilities (referred to as p in Eq. 1). While updating the task parameters ψ, the warp parameters ϕ are kept fixed and vice-versa. This is achieved by selectively zeroing out the gradients during the backward pass of the corresponding loss functions; $\mathcal{L}_{task}$ and $\mathcal{L}_{warp}$. Other than these two modifications, the architectural components and the training methodology is the same as standard Faster R-CNN.

We use stochastic gradient descent (SGD) with momentum 0.9. The initial learning rate is set to 0.02 and subsequently reduced to 0.0002, with a warm-up period of 100 iterations. Each task is trained for 18,000 and 90,000 iterations

Class Split		aero	cycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	bike	person	mAP-old	plant	sheep	sofa	train	tv	mAP
1-20		79.4	83.3	73.2	59.4	62.6	81.7	86.6	83.0	56.4	81.6	71.9	83.0	85.4	81.5	82.7	76.8	49.4	74.4	75.1	79.6	73.6	75.2
1-15		78.1	82.6	74.2	61.8	63.9	80.4	87.0	81.5	57.7	80.4	73.1	80.8	85.8	81.6	83.9	76.9	-	-	-	-	-	76.9
(1-15)+16	[6]	70.5	78.3	69.6	60.4	52.4	76.8	79.4	79.2	47.1	70.2	56.7	77.0	80.3	78.1	70.0	69.7	26.3	-	-	-	-	67.0
	Ours	78.8	79.0	65.9	51.8	57.3	76.1	84.2	80.3	47.3	77.0	60.4	76.0	81.8	76.9	80.6	**71.6**	33.3	-	-	-	-	**69.2**
(1-15)+	[6]	70.3	78.9	67.7	59.2	47.0	76.3	79.3	77.7	48.0	58.8	60.2	67.4	71.6	78.6	70.2	67.4	27.9	46.8	-	-	-	63.9
16+17	Ours	80.1	79.9	66.2	53.4	58.4	79.6	84.9	79.0	47.3	72.9	60.7	73.7	81.2	78.3	80.4	**71.7**	35.5	56.3	-	-	-	**68.7**
(1-15)+	[6]	69.8	78.2	67.0	50.4	46.9	76.5	78.6	78.0	46.4	58.6	58.6	67.5	71.8	78.5	69.9	66.4	26.1	56.2	45.3	-	-	62.5
16+..+18	Ours	80.1	79.4	65.0	53.2	58.0	78.4	85.0	77.5	46.3	72.9	59.2	74.1	81.4	75.7	79.9	**71.1**	35.4	55.8	50.0	-	-	**67.1**
(1-15)+	[6]	70.4	78.8	67.3	49.8	46.4	75.6	78.4	78.0	46.0	59.5	59.2	67.2	71.8	71.3	69.8	66	25.9	56.1	48.2	65.0	-	62.4
16+..+19	Ours	77.7	78.5	67.1	51.9	56.9	74.9	84.1	79.1	46.4	71.8	59.4	72.5	81.6	75.6	79.5	**70.5**	33.5	58.2	49.4	65.5	-	**66.5**
(1-15)+	[6]	70.0	78.1	61.0	50.9	46.3	76.0	78.8	77.2	46.1	66.6	58.9	67.7	71.6	71.4	69.6	66	25.6	57.1	46.5	70.7	58.2	62.4
16+..+20	Ours	77.5	77.1	66.8	53.6	55.0	73.7	83.6	76.7	45.2	74.0	57.7	72.4	81.3	77.0	79.1	**70.0**	34.9	58.1	49.6	67.5	53.7	**65.7**

TABLE 1
Per-class AP and overall mAP values when five classes from PASCAL VOC dataset are added one-by-one to a model initially trained on 15 categories on the left. 'mAP-old' column refers to the mAP of all the base classes, while 'mAP' column refers to the mAP of all the classes seen till then. Our approach achieves consistent improvement in detection performance, compared to [6].

for PASCAL VOC and MS COCO respectively. The training is carried out on a single machine with 8 GPUs, each of them processing two images at a time. Hence, the effective batch size is 16. During evaluation, 100 detections per image are considered with an NMS threshold of 0.4. N_{feat} and N_{img}, which control the queue size of the feature store (F_{Store}) and image store (I_{Store}) respectively, are set to 10. Note that both F_{Store} and I_{Store} are class specific queues of fixed length. Hence they do not grow in size as the oldest item will be dequeued as a new item is enqueued to an already full queue. The warp update interval γ and α in Eq. 4, is set to 20 and 0.2 respectively. Our implementation is based on Detectron2 [46] library. We would be releasing the code and trained models for further clarity and reproducibility.

4.4 Results

In the following, we organise the results for each of the experimental settings outlined in Sec. 4.2. The classes introduced in each task are color coded for clarity.

Adding Classes Sequentially: In the first experiment, we consider incrementally adding one new class at a time to the object detector that is trained to detect all the previously seen classes. We simulate this scenario by training the detector on images from the first 15 classes of PASCAL VOC and then adding $16^{th} - 20^{th}$ classes one by one.

Table 1 shows the class-wise average precision (AP) at IoU threshold 0.5 and the corresponding mean average precision (mAP). The first row is the upper-bound where the detector is trained on data from all 20 classes. The AP values when $\mathcal{F}_{OD}$ is trained on first 15 class examples and evaluated on test data from the same 15 classes is reported in the second row. The following five pair of rows showcase the result when a new class is added. The notation $(1-15)+16+..+20$ is a shorthand for this setting. Our meta-learned model performs favourably well against the previous best method [6] on all the sequential tasks.

Adding Groups of Classes Together: Next, we test our method in a dual task scenario, where $\mathcal{T}_1$ contains one set of classes and $\mathcal{T}_2$ contains the remaining classes. We consider $10+10$, $15+5$ and $19+1$ settings for PASCAL VOC. Tables 2, 3 and 4 show the corresponding results. The second row in each of the tables shows the upper-bound when all class data is available for training. The third row reports the AP values when we train and evaluate $\mathcal{F}_{OD}$ on $\mathcal{T}_1$. When the second task $\mathcal{T}_2$ is added with standard training, we see that performance on classes of the first task drops significantly (fourth row). This evaluation is carried out on test examples from all 20 classes. In the subsequent rows, we report the accuracies of our proposed methodology when compared against Shmelkov *et al*. [6], Faster ILOD [24] and ORE [23] . We see that our approach comfortably outperforms [6] and Faster ILOD [24] in terms of mAP in all the settings. ORE [23] has a slightly better performance in the 15+5 setting. ORE's capability to model unknown objects explicitly is orthogonal to our work and can be incorporated into ours. We will explore this direction in a future work. Class-wise AP values are also reported, showing our improvements on majority of the classes. We compare with DMC [25] in the Supplementary. Qualitative results are showcased in Fig. 3.

Table 5 reports the results on MS COCO in a $40+40$ setting (results are reported on entire val-set). For the sake of comparison with [6], [24], we also report results on mini-val, which contains the first 5000 images from the validation split. Following the standard COCO evaluation, we report average precision across multiple IoUs (AP-(.50:.05:.95), AP^{50}-.50, AP^{75}-.75) and scales (AP^S-small, AP^M-medium and AP^L-large) and average recall while using 1, 10 and 100 detections per image (AR^1, AR^{10}, AR^{100}) and different scales (AR^S-small, AR^M-medium and AR^L-large).

5 Discussions and Analysis

Here, we report ablation results and the justification for our design choices. All the experiments are conducted in the $15+5$ setting, where 5 new classes are added to the detector trained on the first 15 classes of PASCAL VOC [41].

Ablation Experiment: We design a set of experiments to clearly understand the contribution of each of the constituent components of the proposed methodology: distillation, gradient preconditioning and fine-tuning. Table 6 shows the results of this ablation study where each of these components are selectively switched on (✓) and off (×) in a $15+5$ setting. We note that using only the distillation loss to avoid forgetting during the training for a new task helps to achieve 58.5% mAP. Just using gradient preconditioning during incremental learning results in a lower mAP of 47.6%, but fine-tuning it with the same amount of data results in 13.5% mAP improvement when compared to fine-tuning a distilled model, which results in only 5.1% mAP improvement. This clearly brings out the effectiveness of meta-learning the gradient preconditioning layers, for quick adaptability. Finally, although using both distillation and gradient preconditioning helps to achieve an mAP of 54.3%, the best performance is achieved when fine-tuning is also applied resulting in an mAP of 67.8%.

Train with	aero	cycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	bike	person	plant	sheep	sofa	train	tv	mAP
All 20	79.4	83.3	73.2	59.4	62.6	81.7	86.6	83.0	56.4	81.6	71.9	83.0	85.4	81.5	82.7	49.4	74.4	75.1	79.6	73.6	75.2
First 10	78.6	78.6	72.0	54.5	63.9	81.5	87.0	78.2	55.3	84.4	-	-	-	-	-	-	-	-	-	-	73.4
Std Training	35.7	9.1	16.6	7.3	9.1	18.2	9.1	26.4	9.1	6.1	57.6	57.1	72.6	67.5	73.9	33.5	53.4	61.1	66.5	57.0	37.3
Shmelkov *et al.* [6]	69.9	70.4	69.4	54.3	48.0	68.7	78.9	68.4	45.5	58.1	59.7	72.7	73.5	73.2	66.3	29.5	63.4	61.6	69.3	62.2	63.1
Faster ILOD [24]	72.8	75.7	71.2	60.5	61.7	70.4	83.3	76.6	53.1	72.3	36.7	70.9	66.8	67.6	66.1	24.7	63.1	48.1	57.1	43.6	62.2
ORE [23]	63.5	70.9	58.9	42.9	34.1	76.2	80.7	76.3	34.1	66.1	56.1	70.4	80.2	72.3	81.8	42.7	71.6	68.1	77.0	67.7	64.6
Ours	76.0	74.6	67.5	55.9	57.6	75.1	85.4	77.0	43.7	70.8	60.1	66.4	76.0	72.6	74.6	39.7	64.0	60.2	68.5	60.5	**66.3**

TABLE 2
Per-class AP and overall mAP on PASCAL VOC when 10 new classes are added to a detector trained on the first 10 classes.

Train with	aero	cycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	bike	person	plant	sheep	sofa	train	tv	mAP
All 20	79.4	83.3	73.2	59.4	62.6	81.7	86.6	83.0	56.4	81.6	71.9	83.0	85.4	81.5	82.7	49.4	74.4	75.1	79.6	73.6	75.2
First 15	78.1	82.6	74.2	61.8	63.9	80.4	87.0	81.5	57.7	80.4	73.1	80.8	85.8	81.6	83.9	-	-	-	-	-	53.2
Std Training	12.7	0.6	9.1	9.1	3.0	0.0	8.5	9.1	0.0	3.0	9.1	0.0	3.3	2.3	9.1	37.6	51.2	57.8	51.5	59.8	16.8
Shmelkov *et al.* [6]	70.5	79.2	68.8	59.1	53.2	75.4	79.4	78.8	46.6	59.4	59.0	75.8	71.8	78.6	69.6	33.7	61.5	63.1	71.7	62.2	65.9
Faster ILOD [24]	66.5	78.1	71.8	54.6	61.4	68.4	82.6	82.7	52.1	74.3	63.1	78.6	80.5	78.4	80.4	36.7	61.7	59.3	67.9	59.1	67.9
ORE [23]	75.4	81.0	67.1	51.9	55.7	77.2	85.6	81.7	46.1	76.2	55.4	76.7	86.2	78.5	82.1	32.8	63.6	54.7	77.7	64.6	**68.5**
Ours	78.4	79.7	66.9	54.8	56.2	77.7	84.6	79.1	47.7	75.0	61.8	74.7	81.6	77.5	80.2	37.8	58.0	54.6	73.0	56.1	67.8

TABLE 3
Per-class AP and overall mAP when last 5 classes from PASCAL VOC are added to a detector trained on the initial 15 classes.

Train with	aero	cycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	bike	person	plant	sheep	sofa	train	tv	mAP
All 20	79.4	83.3	73.2	59.4	62.6	81.7	86.6	83.0	56.4	81.6	71.9	83.0	85.4	81.5	82.7	49.4	74.4	75.1	79.6	73.6	75.2
First 19	76.3	77.3	68.4	55.4	59.7	81.4	85.3	80.3	47.8	78.1	65.7	77.5	83.5	76.2	77.2	46.6	71.4	65.8	76.5	-	67.5
Std Training	16.6	9.1	9.1	9.1	9.1	8.3	35.3	9.1	0.0	22.3	9.1	9.1	9.1	13.7	9.1	9.1	23.1	9.1	15.4	50.7	14.3
Shmelkov *et al.* [6]	69.4	79.3	69.5	57.4	45.4	78.4	79.1	80.5	45.7	76.3	64.8	77.2	80.8	77.5	70.1	42.3	67.5	64.4	76.7	62.7	68.3
Faster ILOD [24]	64.2	74.7	73.2	55.5	53.7	70.8	82.9	82.6	51.6	79.7	58.7	78.8	81.8	75.3	77.4	43.1	73.8	61.7	69.8	61.1	68.6
ORE [23]	67.3	76.8	60.0	48.4	58.8	81.1	86.5	75.8	41.5	79.6	54.6	72.8	85.9	81.7	82.4	44.8	75.8	68.2	75.7	60.1	68.9
Ours	78.2	77.5	69.4	55.0	56.0	78.4	84.2	79.2	46.6	79.0	63.2	78.5	82.7	79.1	79.9	44.1	73.2	66.3	76.4	57.6	**70.2**

TABLE 4
Per-class AP and overall mAP when tvmonitor class from PASCAL VOC is added to the detector, trained on all other classes.

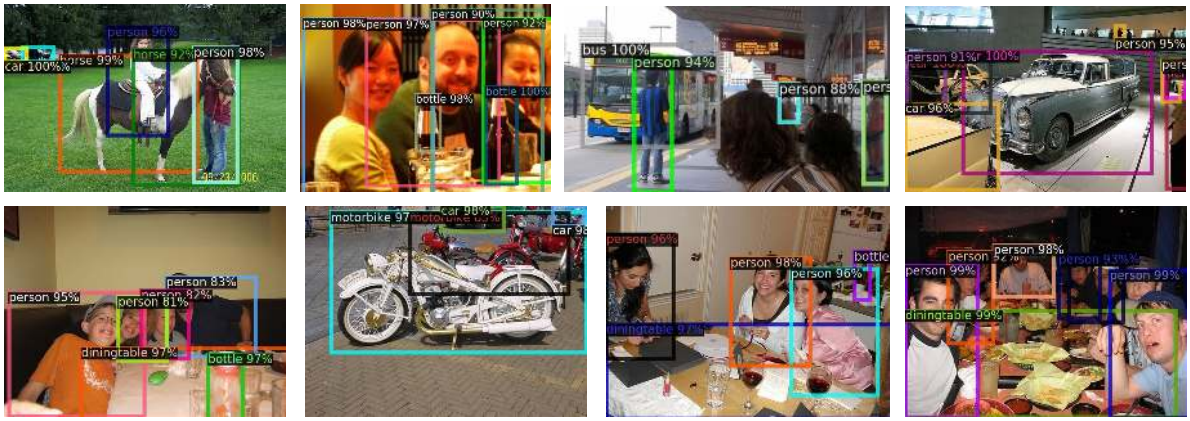

Fig. 3. Qualitative results of our incremental object detector trained in a 10+10 setting where $\mathcal{T}_1 = \{$aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow$\}$ and $\mathcal{T}_2 = \{$diningtable, dog, horse, motorbike, person, pottedplant, sheep, sofa, train, tvmonitor$\}$.

		AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L	AR^1	AR^{10}	AR^{100}	AR^S	AR^M	AR^L
mini-val	All 80	31.5	50.9	33.5	15.4	35.6	43.5	28.7	44.3	45.9	26.1	51.2	61.8
	[6]	21.3	37.4	-	-	-	-	-	-	-	-	-	-
	[24]	20.6	40.1	-	-	-	-	-	-	-	-	-	-
	Ours	**23.8**	**40.5**	24.4	12	26.6	32	24.2	38.4	39.7	20.7	44.6	52.4
full-val	All 80	31.2	51	33.1	14.8	34.6	41.5	28.9	44.8	46.4	25	51.5	61.7
	Ours	23.7	40.4	24.5	11.8	26.2	30	24.3	38.6	40	20.5	44.3	52.7

TABLE 5
AP and AR values when 40 new classes are added to a model trained with the first 40 classes on MS COCO dataset.

Choice of Preconditioning Layers: We characterise the forgetting in a two stage object detector and identify that the backbone, the RPN ($\mathcal{F}_{RPN}$) and the RoI Head ($\mathcal{F}_{RoI\_Head}$) have different forgetting rates. This can be attributed to the category-agnostic nature of $\mathcal{F}_{RPN}$. It learns to predict regions that can possibly contain an object (expressed by the objectness score), making it independent of the object class. We hypothesise that keeping the RPN fixed or training it along for the new classes would not lead to significant performance change. Our experimental results corroborate with this, where we see a similar overall mAP of 58.5% vs. 58.1% for a distilled detector with and without trained RPN.

We choose to add the preconditioning layers in $\mathcal{F}_{RoI\_Head}$ because: (a) Its category-specific nature as opposed to category-agnostic nature of $\mathcal{F}_{RPN}$ makes it the most forgetful component in the object detection pipeline. (b) Since each RoI-pooled feature (which is an input to $\mathcal{F}_{RoI\_Head}$) corresponds to only a single object, this helps to meta-learn the preconditioning layers in a class balanced manner using the F_{Store}. (c) The earlier the preconditioning is applied to the gradients in the back-propagation path, the larger portion of the network is adapted effectively. We choose one layer from each of the residual blocks in $\mathcal{F}_{RoI\_Head}$ as preconditioning layers (refer Fig. 2). Empirically, we find that adding the preconditioning layers to the

D	G	F	$\mathcal{T}_1$	$\mathcal{T}_2$	All
✓	×	×	64.5	40.8	58.5
✓	×	✓	71.0	41.6	63.6
×	✓	×	45.6	53.6	47.6
×	✓	✓	69.7	55.2	66.1
✓	✓	×	58.2	42.7	54.3
✓	✓	✓	71.7	55.9	67.8

TABLE 6
An ablation study to understand the contribution of Distillation (D), Gradient preconditioning (G) and Fine-tuning (F).

α	$\mathcal{T}_1$	$\mathcal{T}_2$	All
0.1	64.5	40.8	58.54
0.2	66.5	35.2	58.67
0.4	68.5	27.0	58.08
0.6	69.1	16.8	56.03
0.8	70.0	3.0	53.23

TABLE 7
Sensitivity analysis on the hyper-parameter α, in Eq. 4. α controls the importance of distillation and detection losses.

last residual block yields the best result (58.5% mAP), when compared to adding it to the other two layers (55.6% and 54.1% mAP respectively).

Sensitivity Analysis: We run a sensitivity analysis on the weighting factor α which weighs the importance of the distillation loss and the object detection loss in Eq. 4. The results are reported in Table 7, where we clearly see that increasing the importance for distillation, reduces the ability to learn new tasks. Only distillation technique is used to reduce forgetting in these results.

6 Conclusion

The existing incremental object detection approaches are based on knowledge distillation, which helps in retaining old learning at the cost of a reduced adaptability to new tasks. In this work, we propose a meta-learning approach to object detection, that learns to precondition the gradient updates such that information across incremental tasks is automatically shared. This helps the model not only to retain old knowledge but also to adapt flexibly to new tasks. The meta-learned incremental object detector outperforms the current best methods on two benchmark datasets. Further, our extensive ablation experiments brings out the contributions of each constituent components of the methodology.

References

[1] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017, pp. 2980–2988.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NeurIPS*, 2015, pp. 91–99.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[4] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, 1989, vol. 24, pp. 109–165.

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

[6] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *ICCV*, 2017.

[7] D. Li, S. Tasci, S. Ghosh, J. Zhu, J. Zhang, and L. Heck, "Rilod: near real-time incremental learning for object detection at the edge," in *4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 113–126.

[8] Y. Hao, Y. Fu, Y.-G. Jiang, and Q. Tian, "An end-to-end architecture for class-incremental object detection with knowledge distillation," in *ICME*. IEEE, 2019, pp. 1–6.

[9] L. Chen, C. Yu, and L. Chen, "A new knowledge distillation for incremental object detection," in *IJCNN*. IEEE, 2019, pp. 1–7.

[10] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[11] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[12] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *NeurIPS*, 2017, pp. 4077–4087.

[13] M. Yin, G. Tucker, M. Zhou, S. Levine, and C. Finn, "Meta-learning without memorization," *arXiv preprint arXiv:1912.03820*, 2019.

[14] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *PNAS*, vol. 114, no. 13, pp. 3521–3526, 2017.

[15] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *CVPR*, 2017.

[16] J. Rajasegaran, M. Hayat, S. H. Khan, F. S. Khan, and L. Shao, "Random path selection for continual learning," in *NeurIPS*, 2019.

[17] J. K J and V. Nallure Balasubramanian, "Meta-consolidation for continual learning," *NeurIPS*, 2020.

[18] F. Cermelli, M. Mancini, S. R. Bulo, E. Ricci, and B. Caputo, "Modeling the background for incremental learning in semantic segmentation," in *CVPR*, 2020.

[19] U. Michieli and P. Zanuttigh, "Incremental learning techniques for semantic segmentation," in *CVPR Workshops*, 2019.

[20] R. Girshick, "Fast r-cnn," in *ICCV*, 2015, pp. 1440–1448.

[21] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *ECCV*. Springer, 2014, pp. 391–405.

[22] M. Acharya, T. L. Hayes, and C. Kanan, "Rodeo: Replay for online object detection," in *BMVC*, 2020.

[23] K. J. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian, "Towards open world object detection," in *CVPR*, 2021.

[24] C. Peng, K. Zhao, and B. C. Lovell, "Faster ilod: Incremental learning for object detectors based on faster rcnn," *Pattern Recognition Letters*, Dec 2020.

[25] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo, "Class-incremental learning via deep model consolidation," in *WACV*, 2020.

[26] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," *arXiv preprint arXiv:1807.05960*, 2018.

[27] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *ICML*, 2016, pp. 1842–1850.

[28] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv:1707.03141*, 2017.

[29] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *CVPR*, 2018, pp. 1199–1208.

[30] Y.-X. Wang, D. Ramanan, and M. Hebert, "Meta-learning to detect rare objects," in *ICCV*, 2019, pp. 9925–9934.

[31] X. Yan, Z. Chen, A. Xu, X. Wang, X. Liang, and L. Lin, "Meta r-cnn: Towards general solver for instance-level low-shot learning," in *ICCV*, 2019, pp. 9577–9586.

[32] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *ICCV*, 2019.

[33] S. Flennerhag, A. A. Rusu, R. Pascanu, F. Visin, H. Yin, and R. Hadsell, "Meta-learning with warped gradient descent," in *ICLR*, 2020.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014, pp. 580–587.

[35] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[36] Y. Lee and S. Choi, "Meta-learning with adaptive layerwise metric and subspace," in *ICML*, 2017.

[37] G. Desjardins, K. Simonyan, R. Pascanu *et al.*, "Natural neural networks," in *NeurIPS*, 2015, pp. 2071–2079.

[38] E. Park and J. B. Oliva, "Meta-curvature," in *NeurIPS*, 2019.

[39] Y. Lee and S. Choi, "Gradient-based meta-learning with learned layerwise metric and subspace," *arXiv:1801.05558*, 2018.

[40] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *ECCV*, 2018.

[41] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *IJCV*, vol. 88, no. 2, pp. 303–338, 2010.

[42] 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 *ECCV*. Springer, 2014, pp. 740–755.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[44] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *NeurIPS*, 2017, pp. 6467–6476.

[45] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *ICLR*, 2019.

[46] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.