META-AGGREGATING NETWORKS FOR CLASS-INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Class-Incremental Learning (CIL) aims to learn a classification model with the number of classes increasing phase-by-phase. The inherent problem in CIL is the stability-plasticity dilemma between the learning of old and new classes, i.e., high-plasticity models easily forget old classes but high-stability models are weak to learn new classes. We alleviate this issue by proposing a novel network architecture called Meta-Aggregating Networks (MANets) in which we explicitly build two residual blocks at each residual level (taking ResNet as the baseline architecture): a stable block and a plastic block. We aggregate the output feature maps from these two blocks and then feed the results to the next-level blocks. We meta-learn the aggregating weights in order to dynamically optimize and balance between two types of blocks, i.e., between stability and plasticity. We conduct extensive experiments on three CIL benchmarks: CIFAR-100, ImageNet-Subset, and ImageNet, and show that many existing CIL methods can be straightforwardly incorporated on the architecture of MANets to boost their performance.

1 INTRODUCTION

AI systems are expected to work in an incremental manner when the amount of knowledge increases over time. They should be capable to learn new concepts while maintaining the ability to recognize previous ones. However, deep-neural-network-based systems often suffer from serious forgetting problems (usually called “catastrophic forgetting”) when continuously updated using new coming data. This is due to two facts: (i) the updates can override the knowledge acquired from the previous data (McCloskey & Cohen, 1989; McRae & Hetherington, 1993; Ratcliff, 1990; Shin et al., 2017; Kemker et al., 2018); and (ii) the model can not replay the entire previous data to regain the old knowledge. To encourage the study to addressing the forgetting problems, Rebuffi et al. (2017) defined a class-incremental learning (CIL) protocol that requires the model to do image classification for which the training data of different classes come in a sequence of phases. In each phase, the classifier is re-trained on new class data, and then evaluated on the test data of both old and new classes. To prevent trivial algorithms such as storing all old data for replaying, there is a strict memory budget — only a tiny set of exemplars of old classes can be saved in the memory.

This memory constraint causes the serious data amount imbalance between old and new classes, and indirectly causes the main problem of CIL – stability-plasticity dilemma (Mermillod et al., 2013). Higher plasticity results in the forgetting of old classes (McCloskey & Cohen, 1989), while higher stability weakens the model from learning the data of new classes (containing a larger number of samples). Existing methods try to balance stability and plasticity using simple data strategies. As illustrated in Figure 1, they directly train the model on the imbalanced dataset (Rebuffi et al., 2017; Li & Hoiem, 2018), and some other works include a fine-tuning step using a balanced subset of exemplars (Castro et al., 2018; Hou et al., 2019; Douillard et al., 2020). However, these methods turn out to be not particularly effective. For example, LUCIR (Hou et al., 2019) sees an accuracy drop of around 16% in predicting 50 previous classes in the last phase (compared to the upper-bound accuracy when all old samples are available) on the CIFAR-100 dataset (Krizhevsky et al., 2009).

In this paper, we address the stability-plasticity dilemma by introducing a novel network architecture called Meta-Aggregating Networks (MANets) for CIL. Taking the ResNet (He et al., 2016b) as an example of baseline architectures, in MANets, we explicitly build two residual blocks (at each residual level): one for maintaining the knowledge of old classes (i.e., the stability) and the other...
Figure 1: Conceptual illustrations of different CIL methods. (a) Conventional methods use all available data (imbalanced classes) to train the model (Rebuffi et al., 2017; Hou et al., 2019) (b) Castro et al. (2018), Hou et al. (2019) and Douillard et al. (2020) follow the convention but add a fine-tuning step using the balanced set of exemplars. (c) Our MANets approach uses all available data to update the plastic and stable blocks, and use the balanced set of exemplars to meta-learn the aggregating weights. We continuously update these weights such as to dynamically balance between plastic and stable blocks, i.e., between plasticity and stability. *: herding is the method to choose exemplars (Welling, 2009), and can be replaced by other methods, e.g., Mnemonics Training (Liu et al., 2020).

for learning new classes (i.e., the plasticity), as shown in Figure 1(c). We achieve these by allowing different numbers of learnable parameters in these two blocks, i.e., less learnable parameters in the stable but more in the plastic. We apply aggregating weights to the output feature maps from these two blocks, sum them up, and pass the results to the next residual level. In this way, we are able to dynamically balance between the stable and plastic features, i.e., stability and plasticity, by updating the aggregating weights. To achieve auto updating, we take these weights as hyperparameters and use meta-learning (Finn et al., 2017; Wu et al., 2019; Liu et al., 2020) to optimize them.

Technically, the optimization of MANets includes two steps at each CIL phase: (1) learn the network parameters for two types of residual blocks, and (2) meta-learn their aggregating weights. Step 1 is the standard training for which we use all the data available at the phase. Step 2 aims to balance between two types of blocks for which we downsample the new class data to build a balanced subset as the meta-training data, as illustrated in Figure 1(c). We formulate these two steps in a bilevel optimization program (BOP) and conduct the optimizations alternatively, i.e., update network parameters with aggregating weights fixed, and then switch (Sinha et al., 2018; MacKay et al., 2019; Liu et al., 2020). For evaluation, we conduct extensive CIL experiments on three benchmarks, CIFAR100, ImageNet-Subset, and ImageNet. We find that many existing CIL methods, e.g., iCaRL (Rebuffi et al., 2017), LUCIR (Hou et al., 2019), Mnemonics Training (Liu et al., 2020), and PODNet (Douillard et al., 2020), can be straightforwardly incorporated on the architecture of MANets, yielding consistent performance improvements.

Our contributions are thus three-fold: (1) a novel and generic network architecture consisting of stable and plastic blocks, specially designed for tackling the problems of CIL; (2) a BOP-based formulation and the corresponding end-to-end optimization solution that enables dynamic and auto balancing between stable and plastic blocks; and (3) extensive experiments by incorporating the proposed architecture into different baseline methods of CIL.

2 RELATED WORK

Incremental learning studies the problem of learning a model from the data that come gradually in sequential training phases. It is also referred to as continual learning (De Lange et al., 2019a; Lopez-Paz & Ranzato, 2017) or lifelong learning (Chen & Liu, 2018; Aljundi et al., 2017). Recent approaches are either in task-incremental setting (classes from different datasets) (Li & Hoiem, 2018; Shin et al., 2017; Hu et al., 2019; Chaudhry et al., 2019; Riemer et al., 2019), or in class-
In Figure 2(a), we provide an illustrative example of our MANets with three residual levels. The maps respectively from two blocks, and aggregate them by weight parameters $\alpha$ to learn the knowledge from old classes. After feeding images to Level 1, we obtain two sets of features. One (orange) will be actively adapted to tackle new tasks while meta-learning a generic model to retain the knowledge of all training tasks. The details of this architecture of its residual levels is composed of two different blocks: a plastic one to adapt to the new class data and a stable one to maintain the knowledge learned from old classes. The optimization process can be formulated as a bilevel optimization program where model parameters are updated at the base level and hyperparameters at the meta level (Von Stackelberg & Von, 1952; Wang et al., 2018; Goodfellow et al., 2014). Recently, there emerge a few of meta-learning based incremental learning methods. Wu et al. (2019) meta-learned a bias correction layer for incremental learning models. Liu et al. (2020) parameterized data exemplars and optimized them by meta gradient descent. Rajasegaran et al. (2020) incrementally learned new tasks while meta-learning a generic model to retain the knowledge of all training tasks.

### 3 Meta-Aggregating Networks (MANets)

Class incremental learning (CIL) usually assumes there are $(N + 1)$ learning phases in total, i.e., one initial phase and $N$ incremental phases during which the number of classes gradually increases (Hou et al., 2019; Liu et al., 2020; Douillard et al., 2020). In the initial phase, only data $D_0$ is available to train the first model $\Theta_0$. There is a strict memory budget in CIL systems, so after the phase, only a small subset of $D_0$ (exemplars denoted as $E_0$) can be stored in the memory to be used as replay samples in later phases. In the $i$-th ($i \geq 1$) phase, we load the exemplars of old classes $E_{c_i-1} = \{E_{c_i}, \ldots, E_{c_{i-1}}\}$ to train model $\Theta_i$ together with new class data $D_i$. Then, we evaluate the trained model on the test data containing both old and new classes. We repeat such training and evaluation through all phases.

The major challenge of CIL is that the model trained at new phases easily “forgets” old classes. To tackle this, we introduce a novel architecture called MANets. MANets is based on ResNet and each of its residual levels is composed of two different blocks: a plastic one to adapt to the new class data and a stable one to maintain the knowledge learned from old classes. The details of this architecture are provided in Section 3.1. The optimization steps of MANets are elaborated in Section 3.2.

#### 3.1 The Architecture of MANets

In Figure 3(a), we provide an illustrative example of our MANets with three residual levels. The inputs $x^{[0]}$ are the images and the outputs $x^{[3]}$ are the features for training classifiers. Every residual level in-between consists of two parallel residual blocks: one (orange) will be actively adapted to new coming classes while the other one (blue) has its parameters partially fixed to maintain the knowledge learned from old classes. After feeding images to Level 1, we obtain two sets of feature maps respectively from two blocks, and aggregate them by weight parameters $\alpha^{[1]}$. Then, we feed
Figure 2: (a) The architecture of MANets. For each residual level, we derive the feature maps from stable blocks ($\phi \odot \theta_{\text{base}}$, blue) and plastic blocks ($\eta$, orange), respectively, aggregate them with meta-learned weights, and feed the result in the next level. (b) An improved version of MANets by including a highway connection block ($h$, green) at each level.

Stable and plastic blocks at each residual level. We aim to balance between the plasticity (for learning new classes) and stability (for maintaining the knowledge of old classes) using a pair of stable and plastic blocks at each residual level. We achieve this by allowing different numbers of learnable parameters in two blocks, i.e., less learnable parameters in the stable but more in the plastic. We detail the operations in the following. We denote the learnable parameters as $\eta$ and $\phi$ for the plastic and stable blocks respectively (at any CIL phase). $\eta$ contains all the convolutional weights, while $\phi$ contains only the neuron-level scaling weights (Sun et al., 2019) which are applied on the frozen convolutional neural network $\theta_{\text{base}}$ pre-learned at the $0$-th phase. As a result, the number of learnable parameters $\phi$ is much less than that of $\eta$. For example, when using the neurons of size $3 \times 3$ in $\theta_{\text{base}}$, the number of learnable parameters $\phi$ is reduced to only $\frac{1}{9}$ of the original number (i.e. the number of learnable parameters in $\eta$). More details are provided in the appendices.

Feature extraction and aggregation. Let $F^k_{\mu}(\cdot)$ denote the transformation function corresponding to the residual block with parameters $\mu$ at Level $k$. Given a batch of training images $x^{[0]}$, we feed them to MANets and compute the feature maps at the $k$-th level (through the stable and plastic blocks respectively) as follows,

$$x^{[k]}_{\phi} = x^{[k]}_{\phi \odot \theta_{\text{base}}} \cdot x^{[k-1]}; \quad x^{[k]}_{\eta} = F^k_{\eta}(x^{[k-1]}).$$

The transferabilities (of the knowledge learned from old classes) are different at different levels of neural networks (Yosinski et al., 2014). Therefore, it is important to apply different aggregating weights for different levels. Let $\alpha^{[k]}_{\phi}$ and $\alpha^{[k]}_{\eta}$ denote the aggregating weights of the stable and plastic blocks respectively at the $k$-th level, based on which we compute the weighted sum of $x^{[k]}_{\phi}$ and $x^{[k]}_{\eta}$ as follows,

$$x^{[k]} = \alpha^{[k]}_{\phi} \cdot x^{[k]}_{\phi} + \alpha^{[k]}_{\eta} \cdot x^{[k]}_{\eta}.$$  

In our illustrative example in Figure 2(a), there are three pairs of weights at each phase. Hence, it becomes increasingly challenging to determine all the weights/hyperparameters if multiple phases are
involved. In this paper, we propose a meta-learning strategy to automatically adapt these weights, i.e., meta-optimizing the weights for different blocks at each phase, see details in Section 3.2.

**Highway connection blocks.** Highway network aims to address the vanishing gradients problem in deep neural networks (Srivastava et al., 2015). From the view of the network architecture, adding highway connection modifies our dual-block architecture to be a residual one where the highway plays the role of an identity branch (except that it has a gating mechanism) (He et al., 2016a). In specific, at each residual level, we add a block of $1 \times 1$ convolution layers (stride=2) and denote it as $h$. We thus can rewrite Eq. 2 as follows,

$$x^{[k]} = x^{[k]} \cdot \alpha^{[k]} + x^{[k]} + x^{[k]}, \quad \text{where} \quad x^{[k]} = x^{[k]}(x^{[k-1]}).$$

When there are $K$ levels in MANets (with or without highway at each level), we use the feature maps after the highest level $x^{[K]}$ to train classifiers.

### 3.2 Bilevel optimization program for MANets

In each incremental phase, we optimize two groups of learnable parameters in MANets: (a) the scaling weights $\phi$ on stable blocks, the convolutional weights $\eta$ on plastic blocks, and the convolutional weights $h$ on highway blocks; (b) the aggregating weights $\alpha$. The former is for network parameters and the latter is for hyperparameters. Therefore, we formulate the overall optimization process as a bilevel optimization program (BOP) (Goodfellow et al., 2014; Liu et al., 2020).

**BOP formulation.** In our MANets, the network parameters $[\phi, \eta, h]$ are trained using the aggregating weights $\alpha$ as hyperparameters. In turn, $\alpha$ can be updated based on the learned network parameters $[\phi, \eta, h]$. In this way, the optimality of $[\phi, \eta, h]$ imposes a constraint on $\alpha$ and vise versa. Ideally, in the $i$-th phase, the CIL system aims to learn the optimal $\alpha_i$ and $[\phi_i, \eta_i, h_i]$ that minimize the classification loss on all training samples seen so far, i.e., $D_i \cup D_{0:i-1}$, so the (ideal) BOP can be formulated as,

$$\min_{\alpha_i} L(\alpha_i, \phi_i^*, \eta_i^*, h_i^*; D_{0:i-1} \cup D_i)$$

s.t. $[\phi_i^*, \eta_i^*, h_i^*] = \arg \min_{[\phi, \eta, h]} L(\alpha_i, \phi, \eta, h; D_{0:i-1} \cup D_i), \quad (4a)$

where $L(\cdot)$ denotes the loss function, e.g., cross-entropy loss. Please note that for the conciseness of the formulation, we use $\phi_i$ to represent $\phi_i \circ \theta_{base}$ (same in the follows). Following Liu et al. (2020), we call Problem 4a and Problem 4b as *meta-level* and *base-level* problems, respectively.

**Data strategy.** To solve Problem 4, we need to use $D_{0:i-1}$. However, in the setting of CIL (Rebuffi et al., 2017; Hou et al., 2019; Douillard et al., 2020), we cannot access $D_{0:i-1}$ but only a small set of exemplars $E_{0:i-1}$, e.g., 20 samples of each old class. Directly replacing $D_{0:i-1} \cup D_i$ with $E_{0:i-1} \cup D_i$ in Problem 4 will lead to the forgetting problem for the old classes. To alleviate this, we propose a new data strategy in which we use different training data splits to learn different parameters: (i) in the *meta-level* problem, $\alpha_i$ is used to balance the stable and the plastic blocks, so we use the balanced subset to update it, i.e., meta-training $\alpha_i$ on $E_{0:i-1} \cup E_i$; (ii) in the *base-level* problem, $[\phi_i, \eta_i, h_i]$ are the network parameters used for feature extraction, so we leverage all the available data to learn them, i.e., base-training $[\phi_i, \eta_i, h_i]$ on $E_{0:i-1} \cup D_i$. In this way, we reformulate the ideal BOP in Problem 4 as a solvable BOP provided below,

$$\min_{\alpha_i} L(\alpha_i, \phi_i^*, \eta_i^*, h_i^*; E_{0:i-1} \cup E_i) \quad (5a)$$

s.t. $[\phi_i^*, \eta_i^*, h_i^*] = \arg \min_{[\phi, \eta, h]} L(\alpha_i, \phi, \eta, h; E_{0:i-1} \cup D_i). \quad (5b)$

**Updating parameters.** We solve the BOP by updating the two groups of parameters $(\alpha_i$ and $[\phi_i, \eta_i, h_i]$ respectively) alternately across epochs, i.e., the $j$-th epoch for learning one group and the $(j+1)$-th epoch for the other group until both groups converge. First, we initialize $\alpha_0$, $\phi_0$, $\eta_0$, and $h_0$ with $\alpha_{i-1}$, $\phi_{i-1}$, $\eta_{i-1}$, and $h_{i-1}$, respectively. Please note that $\theta_0$ is initialized with ones, following Sun et al. (2019). $\eta_0$ is initialized with $\theta_{base}$, and $\alpha_0$ is initialized with $0.5$. Based on our Data strategy, we use all available data in the $i$-th phase to solve the *base-level* problem, i.e., base-learning $[\phi_i, \eta_i, h_i]$ as follows,

$$[\phi_i, \eta_i, h_i] \leftarrow [\phi_i, \eta_i, h_i] - \gamma_1 \nabla_{[\phi_i, \eta_i, h_i]} L(\alpha_i, \phi_i, \eta_i, h_i; E_{0:i-1} \cup D_i). \quad (6)$$
Then, we use a balanced exemplar set to solve meta-level problem, i.e., meta-learning $\alpha_i$ as follows,

$$\alpha_i \leftarrow \alpha_i - \gamma_2 \nabla_{\alpha_i} \mathcal{L}(\alpha_i, \phi_i, \eta_i, h_i; \mathcal{E}_{0:i-1} \cup \mathcal{E}_i),$$

where $\gamma_1$ and $\gamma_2$ are the base-level and meta-level learning rates, respectively. Algorithm 1 summarizes the training algorithm of the proposed MANets, taking the $i$-th CIL phase as an example.

4 EXPERIMENTS

We evaluate the proposed MANets on three CIL benchmarks, i.e., CIFAR-100 (Krizhevsky et al., 2009), ImageNet-Subset (Rebuffi et al., 2017) and ImageNet (Russakovsky et al., 2015), and achieve the state-of-the-art performance. Below we describe the datasets and implementation details (Section 4.1), followed by the results and analyses (Section 4.2) including comparisons to related methods, ablation studies and visualization results.

4.1 DATASETS AND IMPLEMENTATION DETAILS

Datasets. We conduct CIL experiments on two datasets, CIFAR-100 (Krizhevsky et al., 2009) and ImageNet (Russakovsky et al., 2015), following related works (Hou et al., 2019; Liu et al., 2020; Douillard et al., 2020). ImageNet is used in two CIL settings: one based on a subset of 100 classes (ImageNet-Subset) and the other based on the entire 1,000 classes. The 100-class data for ImageNet-Subset are randomly sampled from ImageNet with an identical random seed (1993) by NumPy, following Hou et al. (2019); Liu et al. (2020).

Implementation details. Following the uniform setting (Douillard et al., 2020; Liu et al., 2020), we use a 32-layer ResNet for CIFAR-100 and an 18-layer ResNet for ImageNet. The learning rates $\gamma_1$ and $\gamma_2$ are initialized as 0.1 and $1 \times 10^{-5}$, respectively. We impose a constraint on $\alpha$, i.e., $\alpha_{\eta} + \alpha_{\phi} = 1$ for each block. For CIFAR-100 (ImageNet), we train the model for 160 (90) epochs in each phase, and the learning rates are divided by 10 after 80 (30) and 120 (60) epochs. We use an SGD optimizer with momentum to train the model.

Benchmark protocol. This work follows the protocol in Hou et al. (2019), Liu et al. (2020), and Douillard et al. (2020). Given a dataset, the model is firstly trained on half of the classes. Then, it learns the remaining classes evenly in the subsequent phases. Assume there is an initial phase and $N$ incremental phases for the CIL system. $N$ is set to be 5, 10 or 25. At each phase, the model is evaluated on the test data for all seen classes. The average accuracy (over all phases) is reported.

4.2 RESULTS AND ANALYSES

Table 1 shows the results of 4 baselines with and without our MANets as a plug-in architecture, and some other related works. Table 2 demonstrates the results in 8 ablative settings. Figure 3 visualizes the Grad-CAM (Selvaraju et al., 2017) activation maps obtained from different residual blocks. Figure 4 shows our phase-wise results compared to those of baselines. Figure 5 shows the changes of values for $\alpha_{\eta}$ and $\alpha_{\phi}$ across 10 incremental phases.

Comparing with the state-of-the-arts. Table 1 shows that taking our MANets as a plug-in architecture on 4 baseline methods (Rebuffi et al., 2017; Hou et al., 2019; Liu et al., 2020; Douillard et al., 2020) consistently improves their performance. E.g., on CIFAR-100, “LUCIR + MANets” achieves 3% of improvement on average. In Figure 4, we can observe that our method achieves the highest accuracy in all settings, compared to the state-of-the-arts. Interestingly, we find that our MANets can boost the more performance for the simpler baseline methods, e.g., iCaRL. “iCaRL + MANets” achieves better results than those of LUCIR on ImageNet-Subset, even though the latter method uses a series of regularization techniques.
Method | CIFAR-100 | ImageNet-Subset | ImageNet |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LwF [Li &amp; Hoiem, 2018]</td>
<td>49.59</td>
<td>46.98</td>
<td>45.51</td>
</tr>
<tr>
<td>BiC [Wu et al., 2019]</td>
<td>59.36</td>
<td>54.20</td>
<td>50.00</td>
</tr>
<tr>
<td>TPCLR [Tao et al., 2020]</td>
<td>65.34</td>
<td>63.58</td>
<td>–</td>
</tr>
<tr>
<td>LUCIR [Hou et al., 2019]</td>
<td>57.12</td>
<td>52.66</td>
<td>48.22</td>
</tr>
<tr>
<td>+ MANets (ours)</td>
<td><strong>64.11</strong></td>
<td><strong>60.22</strong></td>
<td><strong>56.40</strong></td>
</tr>
<tr>
<td>iCaRL [Rebuffi et al., 2017]</td>
<td>65.34</td>
<td>63.58</td>
<td>–</td>
</tr>
<tr>
<td>+ MANets (ours)</td>
<td><strong>67.12</strong></td>
<td><strong>65.21</strong></td>
<td><strong>64.29</strong></td>
</tr>
<tr>
<td>Mnemonics [Liu et al., 2020]</td>
<td>63.34</td>
<td>62.28</td>
<td>60.96</td>
</tr>
<tr>
<td>+ MANets (ours)</td>
<td><strong>67.37</strong></td>
<td><strong>65.64</strong></td>
<td><strong>63.29</strong></td>
</tr>
<tr>
<td>PODNet-CNN [Douillard et al., 2020]</td>
<td>64.83</td>
<td>63.19</td>
<td>60.72</td>
</tr>
<tr>
<td>+ MANets (ours)</td>
<td><strong>66.12</strong></td>
<td><strong>64.11</strong></td>
<td><strong>62.12</strong></td>
</tr>
</tbody>
</table>

Table 1: Average incremental accuracy (%) of four CIL methods with and without our MANets as a plug-in architecture, and the related methods. Please note (1) Douillard et al. (2020) didn’t report the results for N=25 on the ImageNet, so we produce the results using their public code; (2) Liu et al. (2020) updated the results on arXiv version (after fixing a bug in their code), different from its conference version; (3) Highway connection blocks are applied in our MANets; and (4) For CIFAR-100, we use “all”+“scaling” blocks. For ImageNet-Subset and ImageNet, we use “scaling”+“frozen” blocks. Please refer to Section 4.2 Ablation settings for details.

Ablation settings. Table 2 demonstrates the ablation study. **Block types:** by differentiating the numbers of learnable parameters, we have 3 block types: (1) “all” means learning all the convolutional weights and biases; (2) “scaling” means learning neuron-level scaling weights (Sun et al., 2019) on the top of a frozen base model $\theta_{base}$; and (3) “frozen” means using $\theta_{base}$ (frozen) as the feature extractor of the stable block. Rows 1 is the baseline model of LUCIR (Hou et al., 2019). Row 2 is a double-block version of LUCIR. They are without meta-learning. Rows 3-5 are our MANets using different pairs of blocks. Row 6-8 use “all”+“scaling”, and under the setting of: (1) Row 6 includes highway connection blocks; (2) Row 7 uses imbalanced data $D_{i-1} \cup D_i$ to meta-train $\alpha$; and (3) Row 8 simply uses fixed weights $\alpha = 0.5$ and $\alpha = 0.5$ at each residual level.

Ablation results. Comparing the second block of results (Rows 3-5) to the first block (baseline), it is obvious that using the proposed MANets can significantly improve the performance of incremental learning, e.g. an average of over 6% gain on ImageNet-Subset (N = 25). From Rows 3-5, we can observe that on ImageNet-Subset, the model with fewer learnable parameters (“scaling”+“frozen”) works the best. This is because we use a shallower network for the larger dataset following the benchmark protocol (ResNet-32 for CIFAR-100; ResNet-18 for ImageNet-Subset), so $\theta_{base}$ for ImageNet-Subset is well-learned in the initial phase and can offer high-quality features for later phases. Comparing Row 6 to Row 3, it is clear that highway connection is helpful for all settings. Comparing Row 7 to Row 3, it shows the importance of using balanced subset to meta-optimize $\alpha$. Comparing Row 8 to Row 3, it shows the superiority of meta-learned $\alpha$ (that is dynamic and optimal) over manually-chosen $\alpha$. 

<table>
<thead>
<tr>
<th>Row</th>
<th>Ablation Setting</th>
<th>CIFAR-100</th>
<th>ImageNet-Subset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>only single “all”</td>
<td>63.17</td>
<td>60.14</td>
</tr>
<tr>
<td>2</td>
<td>“all”+“all”</td>
<td>64.99</td>
<td>61.89</td>
</tr>
<tr>
<td>3</td>
<td>“all”+“scaling”</td>
<td>66.21</td>
<td>65.17</td>
</tr>
<tr>
<td>4</td>
<td>“all”+“frozen”</td>
<td>65.62</td>
<td>64.05</td>
</tr>
<tr>
<td>5</td>
<td>“scaling”+“frozen”</td>
<td>64.71</td>
<td>63.65</td>
</tr>
<tr>
<td>6</td>
<td>w/ highway</td>
<td><strong>67.12</strong></td>
<td><strong>65.21</strong></td>
</tr>
<tr>
<td>7</td>
<td>w/o balanced $E$</td>
<td>65.91</td>
<td>64.70</td>
</tr>
<tr>
<td>8</td>
<td>w/o meta-learned $\alpha$</td>
<td>65.89</td>
<td>64.49</td>
</tr>
</tbody>
</table>
The visualization of activation maps.
Figure 3 shows the activation maps visualized by Grad-CAM for Phase 5 (last phase) model on ImageNet-Subset ($N=5$). The visualized samples (on the left and right) are from the classes coming in Phase 0 and Phase 5, respectively. When input Phase 0 samples to the Phase 5 model, it activates the object regions on the stable block but fails on the plastic block. It is easy to explain as the plastic block already forgets the knowledge learned in Phase 0 while the stable block successfully retains it. This situation is reversed when input Phase 5 samples to that model. It is because the stable block is far less learnable and fails to adapt to the new coming data. While for all samples, our MANets can capture the right object features, as it aggregates the feature maps from two types of blocks and its meta-learned aggregating weights ensure the effective adaptation (balancing between two types of blocks) in both early and late phases.

The values of $\alpha_\eta$ and $\alpha_\phi$. Figure 5 shows the changes of values for $\alpha_\eta$ and $\alpha_\phi$ on CIFAR-100 ($N=10$). We can see that Level 1 tends to get larger values of $\alpha_\phi$, while Level 3 tends to get larger values of $\alpha_\eta$, i.e., lower-level residual block learns to be stabler which is intuitively correct in deep models. Actually, the CIL system is continuously transferring its learned knowledge to subsequent phases. Different layers (or levels) of the model have different transferabilities (Yosinski et al., 2014). Level 1 encodes low-level features that are more stable and shareable among classes. Level 3 nears the classifiers, and it tends to be more plastic such as to fast to adapt to new coming data.

5 Conclusions
In this paper, we introduce a novel network architecture MANets for class-incremental learning (CIL). Our main contribution lies in addressing the issue of stability-plasticity dilemma by aggregating the feature maps from two types of residual blocks (i.e., stable and plastic blocks) for which the aggregating weights are optimized by meta-learning in an end-to-end manner. Our approach is generic and can be easily incorporated into existing CIL methods to boost the performance.
REFERENCES


