ESSENTIALS FOR CLASS INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Contemporary neural networks are limited in their ability to learn from evolving streams of training data. When trained sequentially on new or evolving tasks, their accuracy drops sharply, making them unsuitable for many real-world applications. In this work, we shed light on the causes of this well known yet unsolved phenomenon - often referred to as *catastrophic forgetting* - in a class-incremental setup. We show that a combination of simple components and a loss that balances intra-task and inter-task learning can already resolve forgetting to the same extent as more complex measures proposed in literature. Moreover, we identify poor quality of the learned representation as another reason for catastrophic forgetting in class-IL. We show that performance is correlated with secondary class information (*dark knowledge*) and it can be improved by an appropriate regularizer. With these lessons learned, class-incremental learning results on CIFAR-100 and ImageNet improve over the state-of-the-art by a large margin, while keeping the approach simple.

1 INTRODUCTION

The ability to learn from continuously evolving data is important for many real-world applications. Latest machine learning models, especially artificial neural networks, have shown great ability to learn the task at hand, but when confronted with a new task, they tend to override the previous concepts. Deep networks suffer heavily from this catastrophic forgetting (Mccloskey & Cohen (1989)) when trained with a sequence of tasks, impeding continual or lifelong learning.

In this work, we focus on class-incremental learning (class-IL) (Rebuffi et al., 2017). It is one of the three scenarios of continual learning as described in van de Ven & Tolias (2019), where the objective is to learn a unified classifier over incrementally occurring sets of classes. Since all the incremental data cannot be retained for unified training, the major challenge is to avoid forgetting previous classes while learning new ones.

The three crucial components of a class-IL algorithm include a memory buffer to store few exemplars from old classes, a forgetting constraint to keep previous knowledge while learning new tasks, and a learning system that balances old and new classes. Although several methods have been proposed to address each of these components, there is not yet a common understanding of best practices.

Contributions. In this work, we propose a compositional class-IL (CCIL) model that isolates the underlying reasons for catastrophic forgetting in class-IL and combines the most simple and most effective components to build a robust base model. It employs plain knowledge distillation (Hinton et al., 2015) as a forgetting constraint and selects samples simply randomly. For the loss evaluation, we propose important changes in the output normalization. This simple model already exceeds state-of-the-art results.

In addition, we study the influence of the learned representation's generalization properties on forgetting and find that the degree of feature specialization (overfitting) correlates with the degree of forgetting. We study some common regularization techniques and find that only those that keep, or even improve, the so-called secondary class information – also referred as dark knowledge by Hinton et al. (2015) – have a positive influence on class-incremental learning, whereas others make things much worse.

2 RELATED WORK

iCaRL was the first approach that formally introduced the class-IL problem (Rebuffi et al. (2017)). iCaRL is a decoupled approach for feature representation learning and classifier learning. It alleviates catastrophic forgetting via knowledge distillation and a replay-based approach. Later Castro et al. (2018) extended it to an end-to-end learning model based on a combination of distillation and cross-entropy loss to show improved results over iCaRL. Successive works usually dedicated their contribution to one of the three components in class-IL.

Exemplar selection: A memory buffer is allocated to store exemplar samples for replaying old classes. Many works (Rebuffi et al. (2017); Castro et al. (2018); Hou et al. (2019); Wu et al. (2019)) use herding heuristics (Welling (2009)) for exemplar selection. Herding selects and retains samples closest to the mean sample for each class. Liu et al. (2020) parameterized the exemplars to make optimize them jointly with the model. Iscen et al. (2020) introduced a memory efficient approach to store feature descriptors instead of images. In our work, we simply sample from each class randomly to compile the exemplar set.

Forgetting-constraint: Knowledge distillation (KD) was first introduced by Li & Hoiem (2018) for multi-task incremental learning. Thereafter, various works (Rebuffi et al., 2017; Castro et al., 2018; Wu et al., 2019) have adopted it in class-IL to restore previous knowledge. Lately, several works have proposed new forgetting constraints with an objective to preserve the structure of old-class embeddings. Hou et al. (2019) proposed the usage of feature-level distillation by penalizing change is the feature representation from the old model. Yu et al. (2020) utilized an embedding network to rectify the semantic drift, Tao et al. (2020) proposed a Hebbian graph-based approach to retain the topology of the feature space. We utilize plain knowledge distillation in this work.

Bias removal methods: Various works (Wu et al., 2019; Hou et al., 2019; Zhao et al., 2020) have pointed out that class-imbalance between old and new classes creates a bias in the class weight vectors in the last linear layer, due to which the network predictions are biased towards new classes. To rectify this bias, Wu et al. (2019) trained an extra bias-correction layer using the validation set, Belouadah & Popescu (2019) proposed to rectify the final activations using the statistics of the old task predictions, Zhao et al. (2020) adjusted the norm of new class-weight vectors to those of the old class-weight vectors, and Hou et al. (2019) applied cosine normalization in the last layer. The focus of these works is limited to the bias in the last layer, but ultimately catastrophic forgetting is an issue that affects the entire network: class imbalance causes the model to overfit to the new task, deteriorating the performance on the old ones. Some works (Castro et al., 2018; Lee et al., 2019) also fine-tune the model to avoid overfitting to the current task. We propose a learning system that resolves this bias without the need of any post-processing, by fixing the underlying issues; see Section 4.

3 CLASS-INCREMENTAL LEARNING

3.1 PROBLEM DEFINITION

The objective of class-incremental learning (class-IL) is to learn a unified classifier from a sequence of data from different classes. Data arrives incrementally as a batch of per-class sets $\mathcal{X}$ *i.e.* $(X^1, X^2, ..., X^t)$, where X^y contains all images from class y. Learning from a batch of classes can be considered as a task $\mathcal{T}$. At each incremental step, the data for the new task $\mathcal{T}_i$ arrives, which contains samples of the new set of classes. At each step, complete data is only available for new classes $\mathcal{X}$ *i.e.* $(X^{s+1}, ..., X^t)$. Only a small amount of exemplar data $\mathcal{P}_{old}$ *i.e.* $(P^1, ..., P^s)$ from previous classes *i.e.* $(X^1, ..., X^s)$ is retained in a memory buffer of limited size. The model is expected to classify all the classes seen so far.

The problem definition with strictly separated batches may appear a bit specific. In many practical applications, the data will arrive in a more mixed-up fashion. However, this strict protocol allows the comparison of techniques and it covers the key issues with class-incremental learning. Improvements on this protocol also serve less strict applied settings.

3.2 EVALUATION METRICS FOR CLASS-IL

Class-IL models are evaluated using three metrics: average incremental accuracy and two forgetting rate metrics. After each incremental step, all classes seen so far are evaluated using the latest model. After N incremental tasks, the accuracy $\mathcal{A}_n$ over all $(N+1)$ steps is averaged and reported. It is termed as *average incremental accuracy* (**Avg Acc**), introduced by Rebuffi et al. (2017). We also evaluate the *forgetting rate* $\boldsymbol{\mathcal{F}}$ proposed by Liu et al. (2020). The forgetting rate measures the performance drop on the first task. It is the accuracy difference on the classes of the first task $X_{test}^{0:s}$, using Θ_0 and Θ_N. Therefore, it is independent of the absolute performance on the initial task $\mathcal{T}_0$. We introduce another metric, referred as $\boldsymbol{\mathcal{F}}_\phi$, to measure forgetting in the feature extractor $\phi(\cdot)$. After the final incremental step, parameters of the feature extractor are frozen and the last linear layer is learned using all the data from all the classes. $\boldsymbol{\mathcal{F}}_\phi$ is the accuracy difference between this model and a model where the whole network is trained on all the classes.

3.3 A BASIC CLASS-IL FRAMEWORK

The network model Θ consists of a feature extractor $\phi(\cdot)$ and a fully-connected layer $fc(\cdot)$ for classification. Similar to a standard multi-class classifier, the output logits o are processed through a softmax activation function $\sigma(\cdot)$ before cross-entropy loss $\mathcal{L}^{CE}$ is evaluated corresponding to the correct class. For the initial base task $\mathcal{T}_0$, the model Θ^s learns a standard classifier for the first ($y \in y[1:s]$) classes. In the incremental step, the fc layer is adapted to learn new classes ($y \in y[s+1:t]$) by adding new output nodes, whereas the other part of the network remains unchanged, resulting into a new model Θ^t. The three main elements of class-IL are set up as follows.

Exemplar selection: We compile the exemplar set by randomly selecting an equal number of samples (m) for each class. The samples are sorted in ascending order according to the distance from the mean of the feature vectors μ_i for each class separately. Since the size of the limited memory is fixed (K), some samples of old classes are removed to accommodate exemplars from new classes. Samples with larger distances to the mean vector are removed first.

Algorithm 1: CCIL: INCREMENTAL_STEP

Input: $\mathcal{X} = (X^{s+1}, ..., X^t), \mathcal{P}^s = (P_1, ..., P_s)$ // new classes data, old exemplar sets
Input: $K, \Theta^s, \hat{\Theta}^s$ // memory size, current model, frozen current model
Output: Θ^t // model trained on t classes

$m \leftarrow K/t$ // number of exemplars per class
$\Theta^t \leftarrow \Theta^s$ // add output nodes for new classes
$\mathcal{P}^t \leftarrow$ UPDATEEXEMPLARSETS$(X^{s+1}, ..., X^t; \mathcal{P}^s, m, \Theta^s)$
for $(x, y) \in \mathcal{X}$ **do** // **update for mini-batch data in** $\mathcal{X}$
  $o = \Theta^t(x)$ // output logits $o = \{o_{old}, o_{new}\}$
  $p_{new} = \sigma_{new}(o_{new})$ // softmax over new class logits only
  $\mathcal{L}_{\mathcal{X}}^{CE} = -\sum_{i=s+1}^{t} y[i] \cdot \log(p_{new}[i])$ // cross-entropy loss
  $p_{old} = \sigma_{old}(o_{old})$ // softmax over old class logits only
  $\hat{p} = \sigma_{old}(\hat{\Theta}^s(x))$
  $\mathcal{L}_{\mathcal{X}}^{KD} = D_{KL}(p_{old}||\hat{p})$ // distillation loss (old classes)
  $(x', y') \sim \mathcal{P}^t$ // **load a mini-batch from exemplars set**
  $o = \Theta^t(x')$
  $q = \sigma(o)$ // softmax over all class logits
  $\mathcal{L}_{\mathcal{P}}^{CE} = -\sum_{i=1}^{t} y'[i] \cdot \log(q[i])$
  $\hat{q} = \sigma_{old}(\hat{\Theta}^s(x'))$
  $q_{old} = \sigma_{old}(o_{old})$ // softmax over old class logits only
  $\mathcal{L}_{\mathcal{P}}^{KD} = D_{KL}(q_{old}||\hat{q})$
  $\mathcal{L} = (\mathcal{L}_{\mathcal{X}}^{CE} + \mathcal{L}_{\mathcal{P}}^{CE}) + \lambda * (\mathcal{L}_{\mathcal{X}}^{KD} + \mathcal{L}_{\mathcal{P}}^{KD})$
end

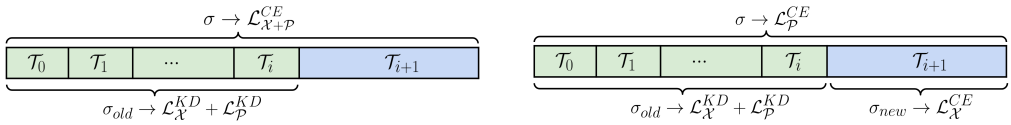


Figure 1: The comparison between a standard loss system (left) and our proposed compositional loss system (right). σ shows the softmax function span over all the network output logits. σ_{old} and σ_{new} shows softmax span over the set of old and new class logits respectively.

Forgetting constraint: Our model uses knowledge distillation as the forgetting constraint. Knowledge distillation penalizes the change with respect to the output of the old model (Θ^s) using KL-divergence, thus preserving the network's knowledge about the old classes. The distillation loss ($\mathcal{L}^{KD}$) is computed for the exemplar sets as well as for samples from the new classes. The final loss for our CCIL model is a combination of cross-entropy loss $\mathcal{L}^{CE}$ for classification and distillation loss $\mathcal{L}^{KD}$ for mitigating catastrophic forgetting as shown in Algorithm 1-Line 18.

Learning system: We propose a new compositional learning system which addresses the bias issue in class-IL. The proposed loss isolates inter-task and intra-task learning for a balanced processing of data by appropriately normalizing the output logits. The task-agnostic parts are shared to yield improved efficiency; see Figure 1. The details are presented in the next section.

4 Compositional Learning System

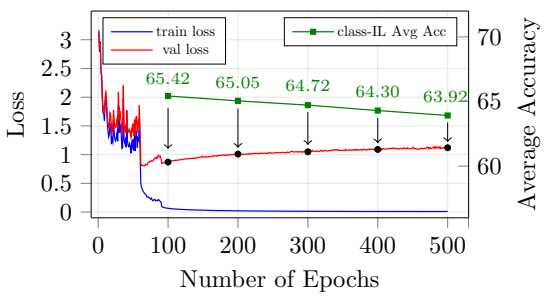
Intra-task Learning: For each gradient update, the CCIL model receives data in separate batches from the set of new classes $\mathcal{X}$ and the set of exemplars $\mathcal{P}$. Instead of merging the batches, we propose to compute two separate losses ($\mathcal{L}^{CE}_{\mathcal{X}}$, $\mathcal{L}^{CE}_{\mathcal{P}}$). The loss for the new classes ($\mathcal{L}^{CE}_{\mathcal{X}}$) is computed using a dedicated softmax function σ_{new} comprising logits of new classes only (Figure 1, Algorithm 1:Line 4-10). This allows the classifier weights for the new classes to be learned independently of the previous classes - while sharing the feature extractor, effectively eliminating the weight bias. In case of a unified softmax, the weights of the old classes are suppressed by the larger amount of new class samples during training. A similar analysis has been shown by Ahn & Moon (2020) for a fine-tuning setup.

Inter-task Learning: The separate softmax helps intra-task learning for the new classes, but this does not yet discriminate the new from the old classes. Therefore, we compile an exemplar set which contains equal numbers of samples from all classes including old and new classes. However small, such exemplar set enables the model to capture the inter-task relationship through the loss $\mathcal{L}^{CE}_{\mathcal{P}}$, which uses a combined softmax function σ evaluated on all classes (shown in Algorithm 1:Line 13). This exemplar set is compiled before learning the incremental task, contrary to previous works, where it is always compiled after the incremental step. Figure 1 shows how the loss terms are calculated using a separate softmax function and also compares it to the unified model used in previous works.

Transfer Learning: We observed that a separate softmax does not remove the bias completely. Another cause for unbalanced class-weight vectors, and catastrophic forgetting in general, is the change in the data distribution between different tasks. We hypothesize that the effect of this distribution shift in the training data is more harmful to the previous knowledge when the transfer learning from old to new classes is poor, resulting in strong alteration of the parameters of the network. We propose to reduce the learning rate for the incremental steps as a simple way to improve transfer learning and mitigate the adverse effect of distribution shift. This further helps reduce the weight bias. Although lowering the learning rate is a standard technique when fine-tuning a network on a new dataset, its importance is underestimated and often missing in incremental learning works.

5 Improving Feature Representations for Incremental Learning

Intuitively, poorly transferable embeddings will force the model to alter its parameters significantly in order to learn new concepts. This destroys the knowledge accumulated for the previous tasks. In this section, we further explore this direction - which, to the best of our knowledge, has not yet been addressed in literature - aiming to further improve the feature representations for class-IL. In



(a)

Epoch	SS-NLL ↓	SS-Acc ↑	$\mathcal{F}\downarrow$	$\mathcal{F}_\phi\downarrow$
100	2.54	38.68	16.03	9.04
200	2.89	32.88	16.04	9.27
300	3.03	30.09	16.94	9.51
400	3.09	29.04	18.38	9.68
500	3.11	27.97	18.57	10.00

(b)

Figure 2: The effect of overfitting on class-IL performance and its correlation with secondary information, on the CIFAR-100 dataset. Figure (a) shows the overfitting behavior as the validation loss (red curve) starts increasing after the 100^{th} epoch. The class-IL performance decreases monotonically (green curve). Table (b) shows the performance of the snapshots taken at every 100^{th} epoch. SS-Acc decreases and SS-NLL increases as more overfitted models are evaluated. Forgetting rates $\mathcal{F}$ and $\mathcal{F}_\phi$also correlate with overfitting.

particular, we study the detrimental effects of overfitting and loss of secondary class information. We find that: 1) both phenomena strongly correlate with catastrophic forgetting; 2) regularization methods can significantly improve robustness against forgetting, but only as long as they enhance the secondary class information of the learned model. All the experiments presented in Sections 5.2 and 5.3 are obtained on the CIFAR-100 dataset using a ResNet-32 architecture.

5.1 Measuring the Quality of Secondary Logits

In literature, the term *secondary information* is interchangeably used to denote the non-target and non-maximum scores of a classifier (Chenglin Yang & Yuille (2019)). Here, for evaluation purposes, the term denotes the non-maximum scores produced by the networks. In both cases the underlying idea is that the secondary output scores reflect the semantic relationships among classes in the data, as learned by the model. No proper annotations exist for secondary information, therefore we define a proxy evaluation objective, exploiting the *coarse*-labeling of the CIFAR-100 dataset, which partitions the 100 *fine*-classes into 20 superclasses. The 5 classes belonging to each superclass are mostly semantically related, and have been previously used for evaluating secondary information (Chenglin Yang & Yuille (2019)). As a proxy evaluation measure for secondary class information we propose to use the classification performance on the superclasses, restricting the network output to the non-maximum logits. We define two new metrics for this purpose: Secondary Superclass NLL and Secondary Superclass Accuracy.

Secondary Superclass-NLL (SS-NLL): Negative Log Likelihood is a commonly used cost function for classification, also known as *Cross-Entropy Loss*. Here we compute the NLL induced by the secondary (non-maximum) logits on the superclass classification problem. Given a set of superclasses $\mathcal{S}$, we can group the fine-grained classes into subsets $\mathcal{C}$ according to their coarse-label, and compute:

$$SS\text{-}NLL(x, y) = -\sum_{j \in \mathcal{S}} \left[\mathbb{1}_{\mathcal{C}_j}(y) \log \sum_{k \in \mathcal{C}_j} \hat{\sigma}\big(f_k(x)\big) \right], \tag{1}$$

where $\mathbb{1}_{\mathcal{C}_j}(y)$ is an indicator function which evaluates to 1 if the true class y belongs to superclass j, $\hat{\sigma}$ is a softmax function over the secondary fine-logits (i.e. it suppresses the maximum logit), and finally $f_k(x)$ is the k-th logit computed by the model f. A lower SS-NLL indicates better superclass classification and thus higher secondary information quality.

Secondary Superclass-Accuracy (SS-Acc): Secondary superclass accuracy computes the percentage of correct superclass predictions. As for SS-NLL, the largest logit score is excluded from the prediction to focus the measure on the quality of secondary information. Higher SS-Acc values indicate higher quality of the secondary information.

5.2 FORGETTING STARTS BEFORE THE INCREMENTAL STEP

In this section, we study how the quality of the representations learned during the initial base task correlates with incremental learning performance. We show how a decline in quality of the learned features - measured as overfitting and loss of secondary information - leads to higher catastrophic forgetting, motivating our following search for a suitable regularizer.

We set up a standard class-IL experiment (with 5 incremental tasks) on CIFAR-100, using a ResNet-32 model. The initial base network is trained for up to 500 epochs. Figure 2a shows that the validation loss (red curve) starts increasing after about 100 epochs, showing an overfitting effect. Thereafter, we perform five different class-IL experiments, each based on a different snapshot of the base network (every 100^{th} epoch). As the validation loss of the snapshot increases, incremental learning performance of the corresponding class-IL model drops (green curve) and both forgetting metrics ($\mathcal{F}$ and $\mathcal{F}_\phi$) worsen (Figure 2b). The worsening $\mathcal{F}_\phi$ metric indicates that the issue is rooted in the feature representations, and cannot be mitigated by acting on the last layer bias. Along with the forgetting metrics, we observe that overfitting causes the quality of secondary information to deteriorate (SS-Acc decreases and the SS-NLL increases, Figure 2b). This loss of secondary information could also be linked to increasing overconfidence of the network, measured as Expected Calibration Error (ECE) (Guo et al., 2017) (details in Appendix A).

These results indicate that: 1) the quality of the features learned during the first base task influences the performance of the class-IL model, and as such it should be expressly addressed. 2) secondary information can be considered as an indicator of the features' quality and their fitness for incremental learning. In the next section we will show experimental evidence in support of these hypotheses.

5.3 ANALYZING CATASTROPHIC FORGETTING WITH REGULARIZATION

Having established a link between early feature quality and catastrophic forgetting, we hypothesize that the application of adequate regularization techniques can improve model performance on the task at hand. We apply four common regularization techniques to our CCIL model: self-distillation (Furlanello et al., 2018), data-augmentation (including cropping, cutout (DeVries & Taylor (2017)) and an extended set of AutoAugment (Cubuk et al., 2019) policies), label smoothing (Szegedy et al., 2016), and mixup (Zhang et al., 2018). All these regularizers have been shown to improve generalization on the held-out validation data. We report details about the application of said regularization methods in Appendix A.

Analysis Table 1 shows the Average Accuracy after finishing the last incremental step, secondary information quality of the first task model, forgetting rates (Section 3.2) and Expected Calibration Error (Guo et al. (2017)). We can divide the regularization methods into two groups: the ones which improve class-IL performance (self-distillation, augmentation) and the ones which harm it (label smoothing, mixup). The first group also shows consistent improvements in secondary information and reduction in forgetting, with augmentation performing the best across all metrics - by a significant margin. In the second group, label smoothing harms secondary information the most. It has been observed that label smoothing encourages representations to be closer to their respective class centroid and equidistant to the other class centroids (Müller et al., 2019), and this comes at the expense of inter-class sample relationships, i.e., secondary information. Mixup also harms the

Model	**Avg. Acc.↑**		**SS Metrics**		**Forgetting**		**ECE↓**
	5 tasks	10 tasks	SS-NLL ↓	SS-Acc. ↑	$\mathcal{F}\downarrow$	$\mathcal{F}_\phi\downarrow$	
CCIL	66.44	64.86	2.784	34.83	17.13	9.7	0.100
CCIL + SD	67.17	65.86	2.675	37.26	16.81	8.88	0.094
CCIL + H-Aug	71.66	69.88	2.051	47.69	13.37	6.73	0.018
CCIL + LS	63.08	61.99	3.103	24.25	18.79	12.83	0.049
CCIL + Mixup	62.31	57.75	2.791	31.57	24.56	16.01	0.024

Table 1: Effect of regularization class-IL average accuracy, secondary information (on the first-task model) and forgetting rates (5 tasks), on CIFAR-100. All the values are averaged over 3 runs. Values that are better than the CCIL baseline are marked in green whereas the worse ones are marked in red. SD:self-distillation, LS:label-smoothing. Standard deviation in Appendix A.

quality of secondary information: we believe this is because it artificially forces arbitrary distances between classes, which modifies the natural output distribution - similarly to label smoothing. Interestingly, all regularizers improve network calibration, but ECE is not a good indicator of class-IL performance, unlike secondary information.

In summary, label smoothing and mixup - despite their proven regularization effects - harm secondary class information and have clear negative consequences for class-incremental learning. On the other hand, regularization methods that enhance secondary class information (self distillation and data augmentation) boost the average incremental accuracy. Analogously to the analysis of Section 5.2 we show that the quality of secondary information negatively correlates to the forgetting rate (Table 1), further indicating the importance of secondary class information.

6 RESULTS

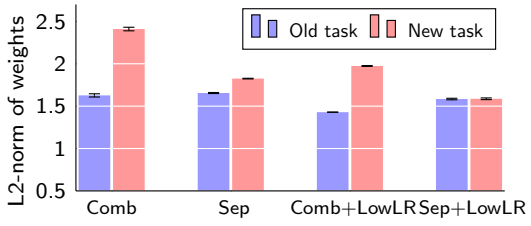
6.1 TRAINING DETAILS

We follow the protocol used in previous works (Hou et al., 2019; Liu et al., 2020). The protocol involves learning of 1 initial base task followed by N incremental tasks. We evaluate with two incremental settings: where the model learns $N = 5$ and $N = 10$ incremental tasks. We conduct experiments on CIFAR-100 (Krizhevsky, 2009), ImageNet-100 and ImageNet (Deng et al., 2009) datasets. For CIFAR-100 and ImageNet-100, 50 classes are selected as the base classes for the initial task and the remaining classes are equally divided over the incremental steps. A similar format is followed for ImageNet with 500 base classes. ResNet-32 is trained for CIFAR, and ResNet-18 is trained for ImageNet. Exemplar memory size is set to $K = 2k$ for 100 class datasets and $K = 20k$ for the full ImageNet dataset. We will publish the code after acceptance.

6.2 ABLATION STUDIES

Elements of the compositional learning system We evaluate the contributions of each element in the proposed learning system by training multiple class-IL models featuring them. The incremental learning in these experiments is conducted in a simple fine-tuning setup (without distillation), in order to single out the effects of the proposed changes. In Figure 3a we compare the average L2 norm of the class weight vectors for old and new classes after 5 incremental training steps, while in Figure 3b we provide the average accuracies and forgetting rates of the respective models. We notice a major difference in the weight norms of old and new classes for the default combined softmax (Comb) setting (Figure 1(left)). Using separate-softmax (Sep) substantially reduces this difference and improves class-IL performance, but does not resolve the problem completely. Lower learning rate (Comb+LowLR) also removes the bias and improves the performance, although to a lesser extent. When both approaches are combined (Sep+Low-LR) most of the bias is resolved and the best class-IL results are produced.

Drawing parallels with iCaRL We compare different components of our CCIL model with the baseline approach proposed by Rebuffi et al. (2017). Table 2 summarizes these changes. We first



(a)

Operations	Avg Acc ↑	$\mathcal{F}$ ↓
Comb	47.97	46.44
Sep	52.86	38.2
Comb+LowLR	52.79	43.46
Sep+LowLR	**58.60**	**32.3**

(b)

Figure 3: (a) compares the average L_2 norm of the classification weight vectors for old and new classes. We evaluate standard combined softmax (Comb) against proposed separate softmax (Sep) and we assess the effect of reduced learning rate (LowLR). (b) contains the corresponding class-IL results without distillation (KD) in terms of average accuracy and forgetting rate. All experiments use the linear classification layer. Results shown on CIFAR-100.

Method	Layer		Softmax		Low LR	AW	Classifier		KD	Avg Acc
	Cos	Dot	Sep	Comb			NME	CNN		
Comb		✓		✓				✓		47.97
iCaRL		✓		✓			✓		✓	56.50
iCaRL++	✓			✓		✓		✓	✓	59.78
CCIL	✓		✓		✓	✓		✓	✓	66.44

Table 2: Drawing parallels between iCaRL and our proposed model. Average accuracy is reported for 5-task class-IL experiments on CIFAR-100 dataset. Last row highlights our proposed changes. All methods use random exemplar selection, Dot: linear layer, KD: knowledge distillation, NME: nearest-mean-of-exemplars.

isolate the contributions of some follow-up methods by creating another baseline as iCaRL++. It consists of a (1) cosine-normalized layer (cos) (Gidaris & Komodakis, 2018; Luo et al., 2018), where the class-weight vectors in the final layer are normalized, and (2) adaptive weighting (AW), where the weight of the distillation loss increases with incremental steps. The last row shows that replacing the combined-softmax (comb) with the proposed separate-softmax (sep) and reducing the learning rate (LowLR) yields a major improvement.

6.3 Comparison to SOTA

Results for CIFAR-100, ImageNet-100 and ImageNet datasets are shown in Table 3. Results are also compared to 'Joint-training', where at every incremental step all the data for the classes seen until then is accessible. Our simple CCIL model compares favorably to previous results on all datasets. Our regularized CCIL-SD closes the gap to joint training further and achieves state-of-the-art performance across all datasets. Since the CCIL model is based only on simple components, we believe that the application of advanced methods for mitigating forgetting (Hou et al., 2019; Tao et al., 2020) and more informative exemplar selection (Liu et al., 2020) can further improve the performance.

Method	CIFAR-100		ImageNet-100		ImageNet	
No. of incremental tasks →	5	10	5	10	5	10
iCaRL* (Rebuffi et al. (2017))	57.17	52.57	65.04	59.53	51.5	46.89
BIC (Wu et al. (2019))	59.36	54.20	70.07	64.96	62.65	58.72
WA (Zhao et al. (2020))	63.25	58.57	—	—	—	—
LUCIR (Hou et al. (2019))	63.12	60.14	70.47	68.09	64.34	61.28
Mnemonics (Liu et al. (2020))	63.34	62.28	72.58	71.37	64.54	63.01
TPCIL (Tao et al. (2020))	65.34	63.58	76.27	74.81	64.89	62.88
CCIL (ours)	66.44	64.86	77.65	74.80	65.11	62.98
CCIL-SD (ours)	**67.17**	**65.86**	**79.39**	**76.68**	**66.73**	**64.88**
Joint-training	74.12	73.80	84.72	84.67	69.72	69.75

Table 3: Comparing average incremental accuracy computed using different methods on CIFAR-100, ImageNet-100 and ImageNet dataset. *as reported in Hou et al. (2019)

7 Conclusions

We presented a straightforward class-incremental learning system that focuses on the essential components and already exceeds the state of the art without integrating sophisticated modules. This makes it a good base model for future research on advancing class-incremental learning.

Moreover, we showed that countering catastrophic forgetting during the incremental step is not enough: the quality of the feature representation prior to the incremental step considerably determines the amount of forgetting. This suggests that representation learning is a promising direction to maximize also incremental performance. In this regard we showed that boosting secondary information is key to improve the transferability of features from old to new tasks without forgetting.

REFERENCES

Hongjoon Ahn and Taesup Moon. A simple class decision balancing for incremental learning. *arXiv preprint arXiv:2003.13947*, 2020.

Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *IEEE International Conference on Computer Vision*, 2019.

Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Siyuan Qiao Chenglin Yang, Lingxi Xie and Alan L. Yuille. Knowledge distillation in generations: More tolerant teachers educate better students. In *Association for the Advancement of Artificial Intelligence*, 2019.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born-again neural networks. In *International Conference on Machine Learning*, 2018.

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL `http://arxiv.org/abs/1503.02531`.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Chunjie Luo, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *International Conference on Artificial Neural Networks*, 2018.

Michael Mccloskey and Neil J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 1989.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, 2019.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *arXiv:1904.07734*, 2019.

S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 2011.

Max Welling. Herding dynamical weights to learn. In *International Conference on Machine Learning*, 2009.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=r1Ddp1-Rb`.

Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

A EXPERIMENT DETAILS

This section includes details concerning experiments included in the paper: hyperparameter details, ablation study for each module in the baseline model and standard deviation for all the results. We report all the evaluation metrics averaged over 3 run trials (unless mentioned otherwise) to capture the variance in the class-IL training process.

A.1 DATASETS

We ran experiments on CIFAR100(Krizhevsky (2009)), ImageNet-100 Subset (Deng et al. (2009)) and full ImageNet datasets. CIFAR-100 contains 60K images from 100 classes of size 32×32, with 50K images for training and 10K for evaluation. CIFAR-100 classes are shuffled using a fixed seed (Numpy van der Walt et al. (2011) seed:1993) across all methods for fair comparison. The ImageNet-100 dataset has 100 randomly sampled classes (using Numpy seed:1993) from ImageNet and further shuffled (using Numpy seed:1993). It contains around 128K images of size 224×224 for training and 5K images for evaluation. The classes of ImageNet dataset are also shuffled using the same seed (Numpy seed:1993) following previous works. We use default data augmentation including random cropping and horizontal flipping for CIFAR-100, and resized-random cropping and horizontal flipping for ImageNet datasets. All the randomization seeds are selected following the experiments in previous works Hou et al. (2019); Liu et al. (2020).

A.2 CCIL CLASS-IL MODEL

Adaptive Weighting (AW) In each incremental step, training a network comprises a classification loss and a distillation loss to preserve knowledge about previous classes. Our baseline contains an adaptive weighting function λ (similar to Hou et al. (2019)) between two losses:

$$\lambda = \lambda_{base}\left(\frac{C_n + C_o}{C_n}\right)^{2/3} \tag{2}$$

,where C_n denotes number of new classes, C_o denotes number of old classes, λ_{base} is fixed constant for each method. It dynamically increases weightage on preserving old knowledge as incremental training continues. It improves the baseline model by 0.45% for 5 task experiment on CIFAR-100. $\lambda_{base} = 5$ is set for CIFAR-100, $\lambda_{base} = 20$ for ImageNet-100 and $\lambda_{base} = 600$ for ImageNet.

Network: We use a 32-layer ResNet He et al. (2016) for CIFAR-100 dataset, and a 18-layer ResNet for ImageNet-100 and ImageNet datasets. The last layer is cosine normalized following the recommendations of Hou et al. (2019).

Optimizer: On CIFAR-100, the base network is trained for 120 epochs using a cosine learning rate schedule, where the base learning rate is 1e-1. Subsequent N tasks are trained for 240 epochs with a base learning rate of 1e-2. The learning rate is decayed until 1e-4. We use a batch size of 100 for CIFAR-100 experiments. Networks for CIFAR-100 dataset is optimized using the SGD optimizer with a momentum of 0.9 and weight decay of 5e-4.

For ImageNet-100, the network is trained for 90 epochs using a step learning rate schedule, where the base learning rate is 1e-1 for the base task and 1e-2 for the subsequent N tasks. The base learning rate is divided by 10 at {30, 60} epochs.

For ImageNet, base task is trained for 70 epochs following a step learning rate, where the base learning is 1e-1. The base learning rate is divided by 10 at {30, 60} epochs. The incremental task is trained for 40 epochs following a step learning rate, where the base learning rate starts from 1e-2. The base learning rate is divided by 10 at {25, 35} epochs. Networks for ImageNet datasets are optimized using the SGD optimizer with a momentum of 0.9 and weight decay of 1e-4. We use a batch size of 128 for both ImageNet datasets.

Epoch	SS-NLL ↓	SS-Acc ↑	Avg Acc ↑	$\mathcal{F}$ ↓	$\mathcal{F}_\phi$↓
100	2.54 ± 0.04	38.68 ± 0.89	65.42 ± 0.06	16.03 ± 0.36	9.04 ± 0.24
200	2.89 ± 0.06	32.88 ± 0.59	65.05 ± 0.08	16.04 ± 0.26	9.27 ± 0.42
300	3.03 ± 0.06	30.09 ± 0.53	64.72 ± 0.07	16.94 ± 0.61	9.51 ± 0.23
400	3.09 ± 0.07	29.04 ± 0.68	64.3 ± 0.12	18.38 ± 0.19	9.68 ± 0.17
500	3.11 ± 0.03	27.97 ± 0.54	62.92 ± 0.11	18.57 ± 0.39	10.00 ± 0.20

Table 4: The effect of overfitting on class-IL performance and its correlation with secondary information. Table shows the performance of the network snapshots taken at every 100^{th} epoch. Accuracy decreases and SS-NLL increases, both monotonically, as more severely overfitted models are evaluated. Forgetting rate $\mathcal{F}$ also correlates with overfitting. Results are computed over 5 runs.

Epoch	ECE
100	0.093±0.003
200	0.118±0.003
300	0.126±0.004
400	0.131±0.005
500	0.137±0.002

Table 5: Expected Calibration Error for different snapshots (every 100^{th} epoch) of the overfitted model.

A.3 Exemplar Selection Algorithm

Algorithm 2: UpdateExemplarSets

Input: $\mathcal{X}, \mathcal{P}_{old}$ // new class data, old exemplar set
Input: Θ^s, m // old model, new exemplar size per class
Output: $\mathcal{P}_{new}$ // new Exemplar sets

for $i = 1, ..., s$ **do**
  $P_i \leftarrow (p_1, ..., p_m)$ // keep first m samples
end
`/* add new class exemplars */`
for $i = s+1, ..., t$ **do**
  $P_i \leftarrow (p_1, ..., p_m) \subset X^i)$ // randomly pick m samples
  $\mu_i \leftarrow \frac{1}{m} \sum_{j=1}^{m} \phi(p_j)$ // mean feature
  `/* sort exemplars based on distance from` μ_i `*/`
  for $k = 1, ..., m$ **do**
    $p_k \leftarrow \arg\min_{p_k} ||\mu_i - \phi(p_k)||$
  end
end

A.4 Overfitting Experiment

Training Details: In the experiment showing overfitting, we train the network longer than usual with a step learning rate schedule. We employ a SGD optimizer with a base learning rate of 1e-1, weight decay of 5e-4 and momentum 0.9. We use a step learning rate schedule, where the learning rate is divided by 10 at 60^{th} and 90^{th} epochs. The training continues at the learning rate of 1e-3 further up to 500 epochs.

Results with standard deviation Table 4 shows class-IL performance using average accuracy and forgetting rate, and quality of secondary information using SS-NLL and SS-Acc for each class-IL runs using increasingly overfitted model snapshots. Average incremental accuracy and forgetting rate is computed for class-IL model trained over different snapshots (every 100^{th}) from the above run. Table 5 shows expected calibration error (ECE) with standard deviation for different snapshots of the overfitted model. It shows that ECE monotonically increases with the number of training epochs. Tables includes values averaged over 5 runs with respective standard deviation.

A.5 REGULARIZATION

All the regularizers are applied at base and all incremental steps, however major improvement is observed due its usage in the initial base task.

Self-distillation In the experiments, self-distillation is conducted over 4 generations (optimized using validation performance) for CIFAR-100 and ImageNet-100 dataset, and over 2 generations for ImageNet dataset. In the beginning of each self-distillation generation, the network snapshot (student) becomes the teacher network and the student continues to train (fine-tuned) with a combination of classification and distillation loss.

For CIFAR-100 experiments, the first base model is trained for 120 epochs following a cosine learning rate schedule, decaying from a learning rate 1e-1 to 1e-4. From first generation onward the model is trained for 70 epochs with a decaying (cosine) learning rate from 1e-1 to 1e-3. All other optimizer settings are the same as the baseline model.

For ImageNet-100 experiments, first base model is trained for 70 epochs following a step learning rate schedule. Starting from a base learning rate of 1e-1, it is divided by 10 at 30^{th} and 60^{th} epoch. From first generation onward the model is trained for 30 epochs each where base learning rate is 1e-2 and it is divided by 10 at 10, 20 epochs.

For ImageNet experiments, the first base model is trained for 40 epochs following a step learning rate schedule. Starting from a base learning rate of 1e-1, it is divided by 10 at 25^{th} and 35^{th} epoch. From first generation onward the model is trained for 15 epochs each where base learning rate is 1e-2 and it is divided by 10 at 8, 12 epochs.

Results with standard deviations Table 6 shows the effect of different regularization on the quality of secondary class information. Table 7 shows the effect of different regularization on class-IL performance in terms of average incremental accuracy and forgetting rate. All experiments are conducted on CIFAR-100 dataset.

Model	**SS Metrics** (5 tasks)	
	SS-NLL ↓	SS-Acc. ↑
CCIL	2.784 ± 0.014	34.827 ± 0.654
CCIL + SD	2.675 ± 0.037	37.26 ± 0.251
CCIL + H-Aug	2.051 ± 0.013	47.69 ± 0.590
CCIL + LS	3.103 ± 0.013	24.25 ± 0.278
CCIL + Mixup	2.791 ± 0.006	31.57 ± 0.256

Table 6: Effect of regularization on secondary information. All the metrics are evaluated on the network trained on the first task. ↓ and ↑ in the column headings indicate that lower and higher values are better respectively. Values that are better than the baseline CCIL method are marked in green whereas the worse ones are marked in red. SD:self-distillation, LS:label-smoothing.

Model	**Avg. Acc.** ↑		**Forgetting** (5 tasks)	
	5 tasks	10 tasks	$\mathcal{F}$ ↓	$\mathcal{F}_\phi$ ↓
CCIL	66.44 ± 0.31	64.86 ± 0.40	17.13 ± 1.12	9.70 ± 0.15
CCIL + SD	67.17 ± 0.14	65.86 ± 0.29	16.81 ± 0.25	8.88 ± 0.35
CCIL + H-Aug	71.66 ± 0.23	69.88 ± 0.36	13.37 ± 0.60	6.73 ± 0.45
CCIL + LS	63.08 ± 0.21	61.99 ± 0.30	18.79 ± 0.29	12.83 ± 0.41
CCIL + Mixup	62.31 ± 0.46	57.75 ± 1.64	24.56 ± 2.52	16.01 ± 0.16

Table 7: Effect of regularization on class-IL performance. All the metrics are evaluated on the network trained on the first task. ↓ and ↑ in the column headings indicate that lower and higher values are better respectively. Values that are better than our baseline method (CCIL) are marked in green whereas the worse ones are marked in red. SD:self-distillation, LS:label-smoothing.

B REPRESENTATIONS: QUALITATIVE ANALYSIS

This section provides a qualitative analysis on the effect of different regularizers on the feature representations (penultimate-layer activations). We analyze the representations of the network trained on 50 classes (first task) of CIFAR-100 dataset using ResNet-32 network.

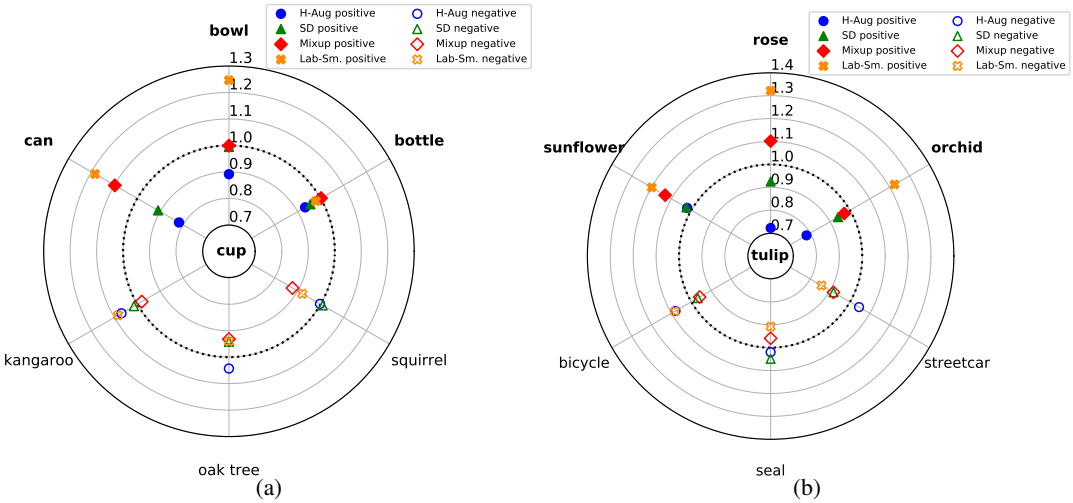


Figure 4: Effect of regularizers on the distance between mean class representations. The numbers shown in the plot are the ratios between the class means distances of each method and of the default CCIL model. Similar classes are marked in **bold**. *Dotted* circle at 1.0 depicts distances between classes in the baseline CCIL model and other distances are depicted relative to the baseline model. *Positive* and *negative* cases indicate similar and dissimilar classes respectively.

B.1 CLASS-MEAN REPRESENTATIONS

We argue that the classes which are semantically similar must be closer in the representation space as compared to the dissimilar classes since they share more features. Based on this argument we analyze the effect of different regularization methods on the relative distances between class-mean representations. We utilize the fine- and coarse-label structure of the CIFAR-100 dataset to compare the effect on the distance between semantically similar and dissimilar classes relative to the default baseline model. Classes associated with the same coarse label or superclass are considered as similar classes, whereas dissimilar classes are picked from different superclasses. L2 distance is used as the distance metric.

Figure 4 show this qualitative analysis for two classes: *cup* and *tulip*. For example cup and can are semantically similar classes. When self-distillation and augmentation are used as regularizers, the relative distance reduces to 0.9 and 0.8 respectively, whereas when label-smoothing and mixup are applied, the relative distance increases to 1.2 and 1.1 respectively. Other similar classes follow a similar trend, whereas dissimilar pairs show an opposite behavior. Overall we find that regularizers: self-distillation and heavy data-augmentation reduce the relative distance between the similar classes (marked in bold) while not affecting or increasing distance between dissimilar classes. Whereas mixup and label smoothing increase the relative distance between similar classes and reduce the relative distance between dissimilar classes. We notice that these observations agree with the findings on secondary class information presented in the main paper.

Earlier in the main paper, we argued that label-smoothing and mixup regularization deteriorate secondary class information since they dismantle the natural output distribution. This qualitative analysis supports our argument showing how they conversely hamper the distances between similar and dissimilar classes.

B.2 REPRESENTATION VISUALIZATION USING *t-SNE*

Figure 5 shows representations learned using mixup and label-smoothing regularization. It can be noticed in the label-smoothing case (on validation set), that the representations of the samples are heavily altered. As discussed in the paper, this happens because label smoothing objective forces samples of the same class to be equidistant from the samples of other classes. Although mixup regularization also alters the representations as argued in the paper and Appendix B.1, such an effect is not clearly visible in the corresponding visualization. Figure 6 shows representations learned using self-distillation and heavy data-augmentation.

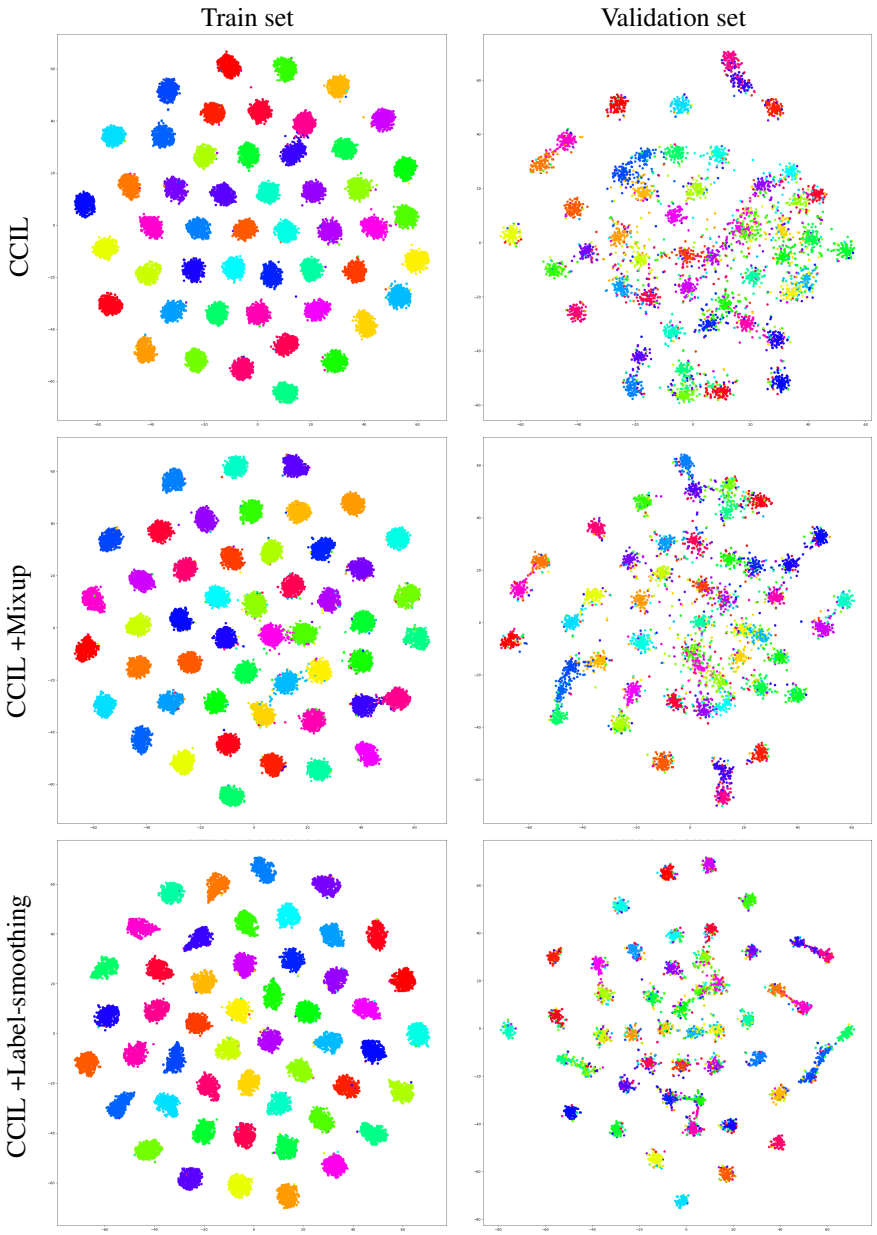


Figure 5: Representations learned using different regularization methods. Top-Bottom: Baseline CCIL , CCIL + mixup, CCIL + label-smoothing. Left and right columns show the representations on the training set and validation set respectively. Visualization is based on 50 classes of CIFAR-100, where each color is a different class.

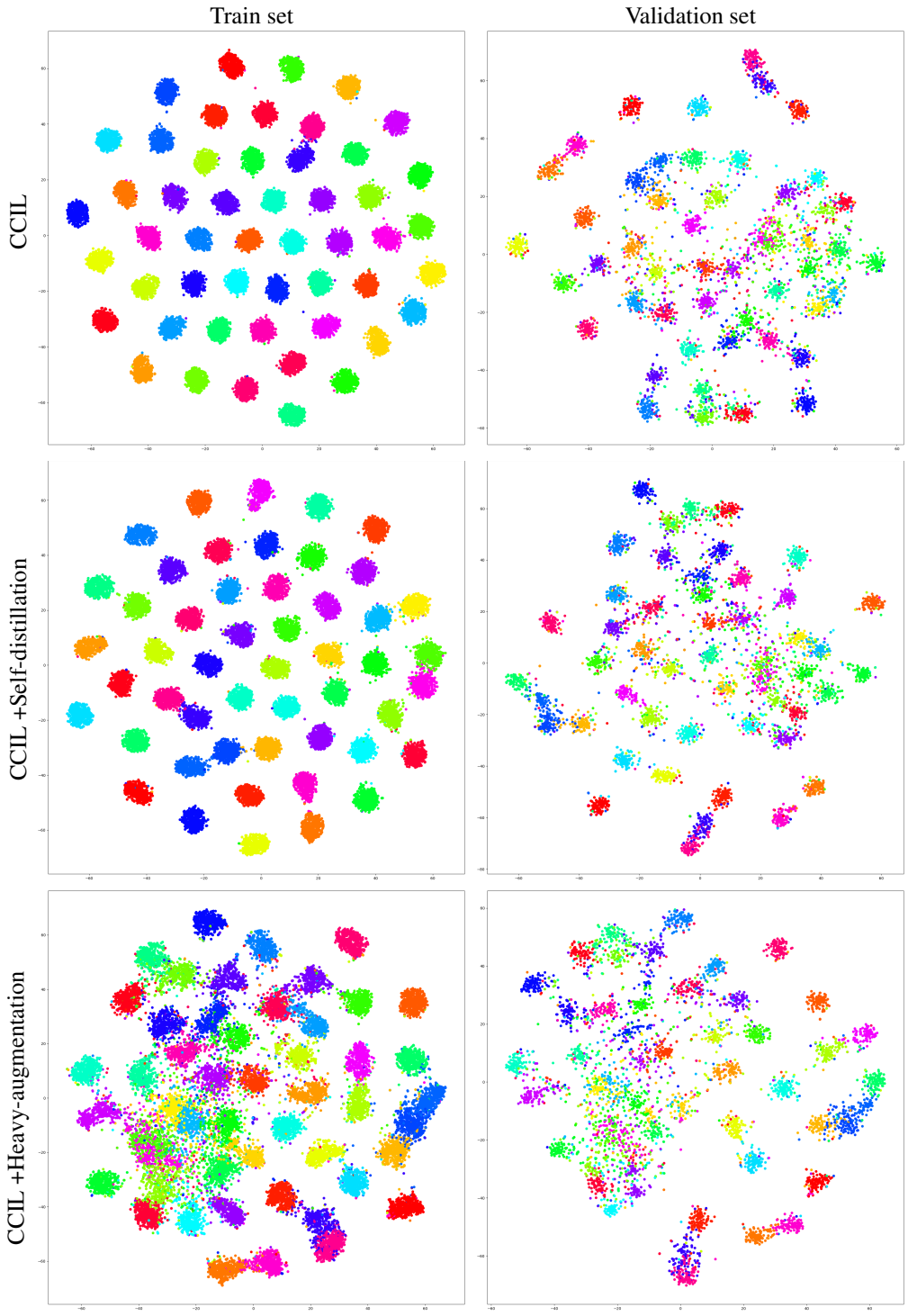


Figure 6: Representations learned using different regularization methods. Top-Left - Bottom-Right: baseline CCIL , CCIL + Self-distillation, CCIL + Heavy-augmentation. Left and right columns show the representations on the training set and validation set respectively.Visualization is based on 50 classes of CIFAR-100, where each color is a different class.