# Rethinking Curriculum Learning with Incremental Labels and Adaptive Compensation

Madan Ravi Ganesh
madantrg@umich.edu
Jason J. Corso
jjcorso@umich.edu

University of Michigan
EECS
Ann Arbor
Michigan, USA

**Abstract**

Like humans, deep networks have been shown to learn better when samples are organized and introduced in a meaningful order or curriculum [37]. Conventional curriculum learning schemes introduce samples in their order of difficulty. This forces models to begin learning from a subset of the available data while adding the external overhead of evaluating the difficulty of samples. In this work, we propose *Learning with Incremental Labels and Adaptive Compensation* (LILAC), a two-phase method that incrementally increases the number of unique output labels rather than the difficulty of samples while consistently using the entire dataset throughout training. In the first phase, *Incremental Label Introduction*, we partition data into mutually exclusive subsets, one that contains a subset of the ground-truth labels and another that contains the remaining data attached to a pseudo-label. Throughout the training process, we recursively reveal unseen ground-truth labels in fixed increments until all the labels are known to the model. In the second phase, *Adaptive Compensation*, we optimize the loss function using altered target vectors for previously misclassified samples. The target vectors of such samples are modified to a smoother distribution to help models learn better. On evaluating across three standard image benchmarks, CIFAR-10, CIFAR-100, and STL-10, we show that LILAC outperforms all comparable baselines. Further, we detail the importance of pacing the introduction of new labels to a model as well as the impact of using a smooth target vector.

## 1 Introduction

Deep networks are a notoriously hard class of models to train effectively [10, 13, 21, 22]. A combination of high-dimensional problems, characterized by a large number of labels and a high volume of samples, a large number of free parameters and extreme sensitivity to experimental setups are some of the main reasons for the difficulty in training deep networks. The go-to solution for deep network optimization is Stochastic Gradient Descent with mini-batches [30] (batch learning) or its derivatives. There are two alternative lines of work which offer strategies to guide deep networks to better solutions than batch learning: Curriculum Learning [3, 12, 15] and Label Smoothing [9, 38].

Curriculum learning helps deep networks learn better by gradually increasing the difficulty of samples used to train networks. This idea is inspired by methods used to teach

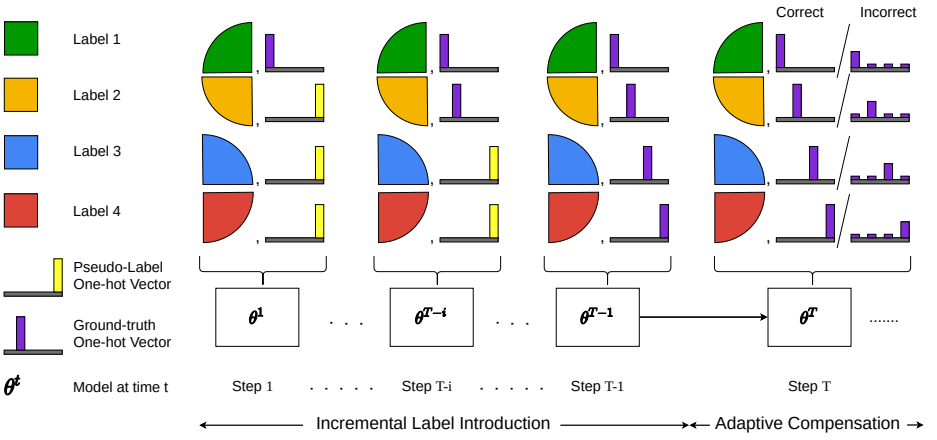



Figure 1: Illustration of the components of LILAC for a four label dataset case. The *Incremental Label introduction* (IL) phase introduces new labels at regular intervals while using the data corresponding to unknown labels (pseudo-label) as negative samples. Once all the labels have been introduced, the *Adaptive Compensation* (AC) phase of training begins. Here, a prior copy of the network is used to classify training data. If a sample is misclassified then a smoother distribution is used as its ground-truth vector in the current epoch.

humans and patterns in human cognition and behaviour [1, 34]. The "difficulty" of samples in the dataset, obtained using either external ranking methods or internal rewards [12, 16], introduces an extra computational overhead while the setup itself restricts the amount of data from which the network begins to learn.

Label smoothing techniques [27, 29, 38] regularize the outcomes of deep networks to prevent over-fitting while improving on existing solutions. They penalize network outputs based on criteria such as noisy labels, overconfident model outcomes, or robustness of a network around a data point in the feature space. Often, such methods penalize the entire dataset throughout the training phase with no regard to the prediction accuracy of each sample.

Inspired by an alternative outlook on Elman's [9] notion of "starting small", we propose LILAC, *Learning with Incremental Labels and Adaptive Compensation*, a novel label-based algorithm that overcomes the issues of the previous methods and effectively combines them. LILAC works in two phases, 1) *Incremental Label Introduction* (IL), which emphasizes gradually learning labels, instead of samples, and 2) *Adaptive Compensation* (AC), which regularizes the outcomes of previously misclassified samples by modifying their target vectors to smoother distributions in the objective function (Fig. 1).

In the first phase, we partition data into two mutually exclusive sets: $\mathbb{S}$, a subset of ground-truth (GT) labels and their corresponding data; and $\mathbb{U}$, remaining data associated with a pseudo-label ($\rho$) and used as negative samples. Once the network is trained using the current state of the data partition for a fixed interval, we reveal more GT labels and their corresponding data and repeat the training process. By contrasting data in $\mathbb{S}$ against the entire remaining dataset in $\mathbb{U}$, we consistently use all the available data throughout training, thereby overcoming one of the key issues of curriculum learning. The setup of the IL phase, inspired by continual learning, allows us to flexibly space out the introduction of new labels and provide the network enough time to develop a strong understanding of each class.

Once all the GT labels are revealed, we initiate the AC phase of training. In this phase,

we replace the target one-hot vector of misclassified samples, obtained from a previous version of the network being trained, with a smoother distribution. The smoother distribution provides an easier value for the network to learn while the use of a prior copy of the network helps avoid external computational overhead and limits the alteration of target vectors to only necessary samples.

To summarize, our main contributions in LILAC are as follows:

- we introduce a novel method for curriculum learning that incrementally learns *labels* as opposed to *samples*,
- we formulate *Adaptive Compensation* as a method to regularize misclassified samples while removing external computational overhead,
- finally, we improve average recognition accuracy across all of the evaluated benchmarks compared to batch learning, a property that is not shared by the other tested curriculum learning and label smoothing methods.

Our code is available at https://github.com/MichiganCOG/LILAC_v2.

## 2 Related Works

**Curriculum Learning** Bengio *et al*. [3], Florensa *et al*. [12], and Graves *et al*. [15] are some important works that have redefined and applied curriculum learning in the context of deep networks. These ideas were expanded upon to show improvements in performance across corrupted [19] and small datasets [11]. More recently, Hacohen and Weinshall [16] explored the impact of varying the pace with which samples were introduced while Weinshall [37] used alternative deep networks to categorize difficult samples. To the best of our knowledge, most previous works have assumed that samples cover a broad spectrum of difficulty and hence need to be categorized and presented in an orderly fashion. The closest relevant work to ours, in terms of learning labels, gradually varies the GT vector from a multimodal distribution to a one-hot vector over the course of the training phase [8].

**Label Smoothing** Label smoothing techniques regularize deep networks by penalizing the objective function based on a pre-defined criterion. Such criteria include using a mixture of true and noisy labels [38], penalizing highly confident outputs [27], and using an alternate deep network's outcomes as GT [29]. Bagherinezhad *et al*. [2] proposed the idea of using logits from trained models instead of just one-hot vectors as GT. Complementary work by Miyato *et al*. [26] used the local distributional smoothness, based on the robustness of a model's distribution around a data point, to smooth labels. The work closest to our method was proposed in Szegedy *et al*. [35], where an alternative target distribution was used across the entire dataset. Instead, we propose to only alter the GT vector for samples that are misclassified. They are identified using a prior copy of the current model, which helps avoid external computational overhead and only uses a small set of operations.

**Incremental Learning and Negative Mining** Incremental and Continual learning are closely related fields that inspired the structure of our algorithm. Their primary concern is learning over evolving data distributions with the addition of constraints on the storage memory [5, 28], distillation of knowledge across different distributions [32, 33], assumption of a single pass over data [6, 25], etc. In our approach, we depart from the assumption of evolving data distributions. Instead, we adopt the experimental pipeline used in incremental learning to introduce new labels at regular intervals. At the same time, inspired by negative

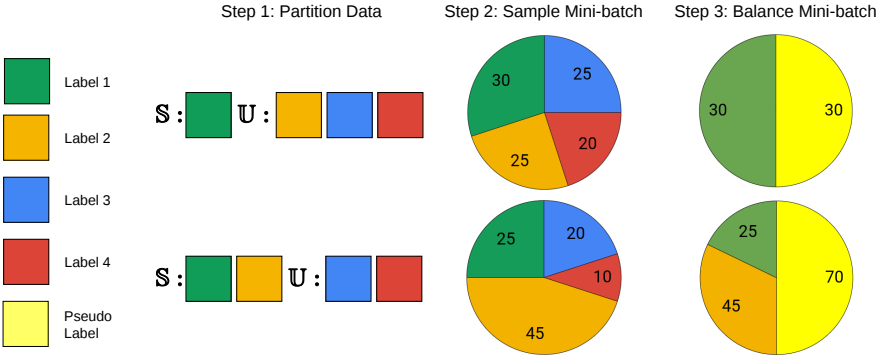



Figure 2: Illustration of the steps in the IL phase when (**Top**) only one GT label is in $\mathbb{S}$ and (**Bottom**) when two GT labels are in $\mathbb{S}$. The steps are 1) partition data, 2) sample a mini-batch of data and 3) balance the number of samples from $\mathbb{U}$ to match those from $\mathbb{S}$ in the mini-batch before training. Samples from $\mathbb{U}$ are assumed to have a uniform prior when being augmented/reduced to match the total number of samples from $\mathbb{S}$. Values inside each pie represent the number of samples. Across both cases, the number of samples from $\mathbb{S}$ determines the final balanced mini-batch size.

mining [4, 23, 36], we use the remaining training data, associated with a pseudo-label, as negative samples. Overall, our setup effectively uses the entire training dataset, thus maintaining the same data distribution.

# 3 LILAC

In LILAC, our main objective is to improve upon batch learning. We do so by first gradually learning labels, in fixed increments, until all GT labels are known to the network (Section 3.1). This behaviour assumes that all samples are of equal difficulty and are available to the network throughout the training phase. Further, we focus on learning strong representations of each class over a dedicated period of time. Once all GT labels are known, we shift to regularizing previously misclassified samples by smoothing the distribution of their target vector while maintaining the peak at the same GT label (Section 3.2). Using a smoother distribution leads to an increase in the entropy of the target vector and helps the network learn better, as we demonstrate in Section 4.2.

## 3.1 Incremental Label Introduction Phase

In the IL phase, we partition data into two sets: $\mathbb{S}$, a subset of GT labels and their corresponding data; and $\mathbb{U}$, the remaining data marked as negative samples using a pseudo-label $\rho$. Over the course of multiple intervals of training, we reveal more GT labels to the network according to a predetermined schedule. Within a given interval of training, the data partition is held fixed and we uniformly sample mini-batches from the entire training set based on their GT label. However, for samples from $\mathbb{U}$, we use $\rho$ as their label. There is no additional change required in the objective function or the outputs of the model when we sample data from $\mathbb{U}$. By the end of this phase, we reveal all GT labels to the network.

For a given dataset, we assume a total of $L$ labels are provided in the ascending order of their value. Based on this ordering, we initialize the first $b$ labels, and their corresponding data, as $\mathbb{S}$, and the data corresponding to the remaining $L-b$ labels as $\mathbb{U}$. Over the course of multiple training intervals, we reveal GT labels in increments of $m$, a hyper-parameter that controls the schedule of new label introduction. Revealing a GT label involves moving the corresponding data from $\mathbb{U}$ to $\mathbb{S}$ and using their GT label instead of $\rho$.

Within a training interval, we train the network for $E$ epochs using the current state of the data partition. First, we sample a mini-batch of data based on a uniform prior over their GT labels. Then, we modify their target vectors based on the partition to which a sample belongs. To ensure the balanced occurrence of samples from GT labels and $\rho$, we augment or reduce the number of samples from $\mathbb{U}$ to match those from $\mathbb{S}$ and use this curated mini-batch to train the network. After $E$ epochs, we move $m$ new GT labels and their corresponding data from $\mathbb{U}$ to $\mathbb{S}$ and repeat the entire process (Fig. 2).

### 3.2 Adaptive Compensation

Once all the GT labels have been revealed and the network has trained sufficiently, we begin the AC phase. In the AC phase, we use a smoother distribution for the target vector of samples which the network is unable to correctly classify. Compared to one-hot vectors, optimizing over this smoother distribution, with an increased entropy, can bridge the gap between the unequal distances in the embedding space and overlaps in the label space [31]. This overlap can occur due to common image content or close proximity in the embedding space relative to other classes. Thus, improving the entropy of such target vectors can help modify the embedding space in the next epoch and compensate for the predictions of misclassified samples.

For a sample $(x_i, y_i)$ in epoch $e \geq T$, we use predictions from the model at $e-1$ to determine the final target vector used in the objective function; specifically, we smoothen the target vector for a sample if and only if it was misclassified by the model at epoch $e-1$. Here, $(x_i, y_i)$ denotes a training sample and its corresponding GT label for sample index $i$, and $T$ represents a threshold epoch value until which the network is trained without adaptive compensation. We compute the final target vector for the $i^{th}$ instance at epoch $e$, $t_i^e$, based on the model $\theta^{e-1}$ using the following equation,

$$t_i^e = \begin{cases} (\frac{\varepsilon L-1}{L-1})\delta_{y_i} + (\frac{1-\varepsilon}{L-1})\mathbb{1}, & \arg\max\left(f_{\theta^{e-1}}(x_i)\right) \neq y_i \\ \delta_{y_i}, & \text{otherwise} \end{cases} . \tag{1}$$

Here, $\delta_{y_i}$ represents the one-hot vector corresponding to GT label $y_i$, $\mathbb{1}$ is a vector of $L$ dimensions with all entries as 1 and $\varepsilon$ is a scaling hyper-parameter.

## 4 Experiments

**Datasets and Metrics** We use three datasets, CIFAR-10, CIFAR-100 ([20]), and STL-10 ([7]), to evaluate our method and validate our claims. CIFAR-10 and CIFAR-100 are 10 and 100 class variants of the popular image benchmark CIFAR while STL-10 is a 10 class subset of ImageNet. *Average Recognition Accuracy (%)* combined with their *Standard Deviation* across 5 trials are used to evaluate the performance of all the algorithms.

**Experimental Setup** For CIFAR-10/100, we use ResNet18 ([17]) as the architectural backbone while for STL-10, we use ResNet34. We set $\rho$ as the last label and $b$ as half the total number of labels of a given dataset. In each interval of LILAC's IL phase, we train the model for 7, 3, and 10 epochs each, at a learning rate of 0.1, 0.01, and 0.1 for CIFAR-10, CIFAR-100, and STL-10, respectively. In the AC phase epochs 150, 220, and 370 are used as thresholds (epoch $T$) for CIFAR-10, CIFAR-100, and STL-10 respectively. Detailed explanations of the experimental setups are provided in the supplementary materials.

**Baselines**

1. Stochastic Gradient Descent with mini-batches (Batch Learning).
2. Standard Baselines
   - Fixed Curriculum: Following the methodology proposed in Bengio *et al*. [3], we create a "Simple" subset of the dataset using data that is within a value of 1.1 as predicted by a linear one-vs-all SVR model. The deep network is trained on the "Simple" dataset for a fixed period of time, which mirrors the total length of the IL phase, after which the entire dataset is used to train the network.
   - Label Smoothing: We follow the method proposed in Szegedy *et al*. [35].
3. Custom Baselines
   - Dynamic Batch Size (DBS): DBS randomly copies data available within a mini-batch to mimic variable batch sizes, similar to the IL phase. However, all GT labels are available to the model throughout the training process.
   - Random Augmentation (RA): This baseline samples from a single randomly chosen class in $\mathbb{U}$, available in the current mini-batch, to balance data between $\mathbb{S}$ and $\mathbb{U}$ in the current mini-batch. This is in contrast to LILAC, which uses samples from all classes in $\mathbb{U}$ that are available in the current mini-batch.
4. Ablative Baselines
   - *Only IL*: This baseline quantifies the contribution of incrementally learning labels when combined with batch learning.
   - *Only AC*: This baseline shows the impact of adaptive compensation, as a label smoothing technique, when combined with batch learning.

## 4.1 Comparison Against Standard Baselines

Table 1 illustrates the improvement offered by LILAC over batch learning, with comparable setups. Further, we break down the contributions of each phase of LILAC. Both *Only IL* and *Only AC* improve over batch learning, albeit to varying degrees, which highlights their individual strengths and importance. However, only when we combine both phases do we observe a consistently high performance across all benchmarks. This indicates that these two phases complement each other.

The Fixed Curriculum approach does not offer consistent improvements over the batch learning baseline across CIFAR-100 and STL-10 while the Label Smoothing approach does not outperform batch learning on the STL-10 dataset. While both of these standard baselines fall short, LILAC consistently outperforms batch learning across all evaluated benchmarks. Interestingly, Label Smoothing provides the highest performance on CIFAR-100. Since the original formulation of LILAC was based on batch learning, we assume all GT vectors to be one-hot. This assumption is violated in Label Smoothing. When we tailor our GT vectors according to the Label Smoothing baseline, we outperform it with minimal hyper-parameter changes, a testament to LILAC's applicability on top of conventional label smoothing.

| Types | Training | Performance (%) CIFAR 10 | CIFAR 100 | STL 10 |
|---|---|---|---|---|
| | Batch Learning | 95.19 ± 0.190 | 78.32 ± 0.175 | 72.88 ± 0.642 |
| Standard | Fixed Curriculum [3] | 95.27 ± 0.112 | 77.89 ± 0.287 | 72.18 ± 0.601 |
| | Label Smoothing [33] | 95.27 ± 0.111 | 79.06 ± 0.179 | 72.55 ± 0.877 |
| Custom | Random Augmentation | 95.27 ± 0.076 | 75.37 ± 0.480 | 73.67 ± 0.708 |
| | Dynamic Batch Size | 95.22 ± 0.131 | 78.73 ± 0.264 | 72.66 ± 1.081 |
| Ablative | *Only IL* (**ours**) | 95.38 ± 0.135 | 78.73 ± 0.139 | 73.43 ± 0.903 |
| | *Only AC* (**ours**) | 95.38 ± 0.170 | 78.94 ± 0.179 | 72.94 ± 0.530 |
| Overall | LILAC (**ours**) | **95.52 ± 0.072** | 78.88 ± 0.201 | **73.77 ± 0.838** |
| | LS + LILAC (**ours**) | 95.34 ± 0.080 | **79.08 ± 0.307** | 73.59 ± 0.623 |

Table 1: Under similar setups, LILAC consistently achieves higher mean accuracy than batch learning across all evaluated benchmarks, a property not shared by other baselines.

| Method | CIFAR-10 |
|---|---|
| Wide Residual Networks [40] | 96.11 |
| Multilevel Residual Networks [41] | 96.23 |
| Fractional Max-pooling [14] | 96.53 |
| Densely Connected Convolutional Networks [18] | 96.54 |
| Drop-Activation [24] | 96.55 |
| Shake-Drop [39] | 96.59 |
| **Shake-Drop + LILAC (ours)** | **96.79** |

Table 2: LILAC easily outperforms the Shake-Drop network ([39]) as well as other top performing algorithms on CIFAR-10 with *standard pre-processing (random crop + flip)*.

The RA baseline highlights the importance of using all of the data in $\mathbb{U}$ as negative samples in the IL phase as opposed to using data from individual classes. This is reflected in the boost in performance offered by LILAC. The DBS baseline is used to highlight the importance of fluctuating mini-batch sizes, which occur due to the balancing of data in the IL phase. Even with the availability of all labels and fluctuating batch sizes, the DBS baseline is easily outperformed by LILAC. This indicates the importance of the recursive structure used to introduce data in the IL phase as well as the use of data from $\mathbb{U}$ as negative samples. Overall, LILAC consistently outperforms batch learning across all benchmarks while existing comparable methods fail to do so. When we extend LILAC to the Shake-Drop [39] network architecture, with only standard pre-processing, we easily outperform other existing approaches with comparable setups, as shown in Table 2.

## 4.2 Key Properties of LILAC

**Smoothness of Target Vector ($\varepsilon$)** Throughout this work, we maintained the importance of using a smoother distribution as the alternate target vector during the AC phase. Table 3 (**Top**) illustrates the change in performance across varying degrees of smoothness in the alternate target vector. There is a clear increase in performance when $\varepsilon$ values are be-

| Property | Performance (%) CIFAR-10 | CIFAR-100 | STL-10 |
|---|---|---|---|
| $\varepsilon = 0.9$ | $95.30 \pm 0.072$ | $78.48 \pm 0.328$ | $73.57 \pm 0.980$ |
| $\varepsilon = 0.8$ | $95.34 \pm 0.141$ | $78.52 \pm 0.118$ | $73.54 \pm 0.984$ |
| $\varepsilon = 0.7$ | $95.42 \pm 0.189$ | $78.72 \pm 0.356$ | $73.59 \pm 0.872$ |
| $\varepsilon = 0.6$ | $95.36 \pm 0.096$ | $78.75 \pm 0.180$ | $\mathbf{73.77 \pm 0.838}$ |
| $\varepsilon = 0.5$ | $95.49 \pm 0.207$ | $78.88 \pm 0.227$ | $73.61 \pm 0.810$ |
| $\varepsilon = 0.4$ | $\mathbf{95.52 \pm 0.072}$ | $\mathbf{78.88 \pm 0.201}$ | $73.54 \pm 0.959$ |
| $\varepsilon = 0.3$ | $95.31 \pm 0.125$ | $78.66 \pm 78.66$ | $73.59 \pm 0.955$ |
| $\varepsilon = 0.2$ | $95.36 \pm 0.095$ | $78.47 \pm 0.093$ | $73.57 \pm 0.963$ |
| $m$: 1 | $95.32 \pm 0.156$ | $\mathbf{78.73 \pm 0.139}$ | $73.27 \pm 0.220$ |
| $m$: 2 (4) | $\mathbf{95.38 \pm 0.135}$ | $78.34 \pm 0.209$ | $\mathbf{73.43 \pm 0.903}$ |
| $m$: 4 (8) | $95.29 \pm 0.069$ | $78.37 \pm 0.114$ | $72.30 \pm 0.543$ |

Table 3: (**Top**) The mid-range of $\varepsilon$ values, 0.7-0.4, show an increase in performance while the edges, due to either too sharp or too flat a distribution, show decreased performance. (**Bottom**) *Only IL* model results illustrate the importance of introducing a small number of new labels in each interval of the IL phase. Values in brackets are for CIFAR-100.

tween 0.7-0.4 (mid-range). On either side of this band of values, the GT vector is either too sharp or too flat, leading to a drop in performance.

**Size of Label Groups ($m$)** LILAC is designed to introduce as many or as few new labels as desired in the IL phase. We hypothesized that developing stronger representations can be facilitated by introducing a small number of new labels while contrasting it against a large variety of negative samples. Table 3 (**Bottom**) supports our hypothesis by illustrating the decrease in performance with an increase in the number of new labels introduced in each interval of the IL phase. Thus, we introduce two labels each for CIFAR-10 and STL-10 and only one new label per interval for CIFAR-100 throughout the experiments in Table 1.

## 5 Discussion: Impact of Each Phase

In this section, we take a closer look at the impact of each phase of LILAC and how they affect the quality of the learned representations. We extract features from the second to last layer of ResNet18/34 from 3 different baselines (Batch Learning, LILAC, and *Only IL*) and use these features to train a linear SVM model.

Fig. 3 highlights the two important phases in our algorithm. First, the plots on the left-hand side show a steady improvement in the performance of LILAC and the *Only IL* baseline once the IL phase is complete and all the labels have been introduced to the network. When we compare the plots of CIFAR-10 and STL-10 against CIFAR-100, we see that all baselines follow the learning trend shown by batch learning, with CIFAR-100 being slightly delayed. Since there are a large number of epochs required to introduce all the labels of CIFAR-100 to the network, the plots are significantly delayed compared to batch learning. Conversely, since there are very few epochs in the IL phase of CIFAR-10 and STL-10, we observe that the performance trend of *Only IL* and LILAC quickly matches that of batch learning. Overall, the

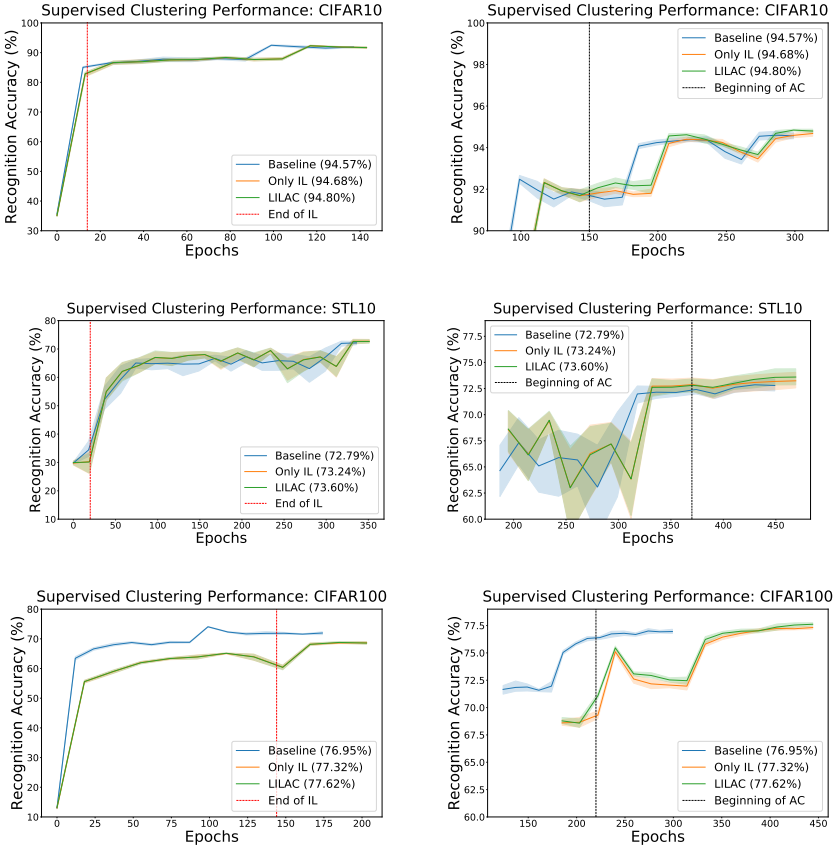
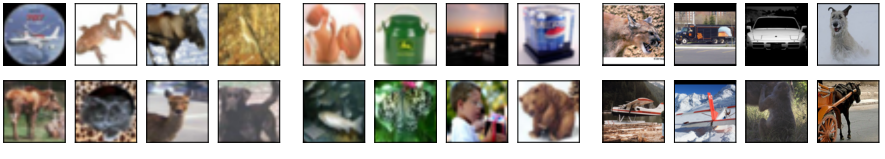



Figure 3: Plots on the (**Left**) show the common learning trend between all baselines, albeit slightly delayed for CIFAR-100, after the IL phase while those on the (**Right**) show steady improvement in performance after applying AC when compared to the *Only IL* baseline. Final supervised classification performances on representations collected from LILAC easily outperform those from batch learning and *Only IL* methods.

final performances of both LILAC and the *Only IL* baseline are higher than batch learning, which supports the importance of the IL phase in learning strong representations.



(a) CIFAR-10 (b) CIFAR-100 (c) STL-10

Figure 4: Illustration of 8 randomly chosen samples that were incorrectly labelled by the *Only IL* baseline and correctly labelled by LILAC. This highlights the importance of AC.

The plots on the right-hand side highlight the similarity in behaviour of *Only IL* and LILAC before AC. However, afterward, we observe that the performance of LILAC overtakes the *Only IL* baseline. This is a clear indicator of the improvement in representation quality when AC is applied. Additionally, from Fig. 3 we observe that inherently the STL-10 dataset results have a high standard deviation, which is reflected in the middle portion of the training phase, between the end of the IL and the beginning of the AC phase and it is not a consequence of our approach. To further support the importance of the AC phase, we provide examples in Fig. 4 of randomly sampled data from the testing set that were incorrectly classified by the *Only IL* baseline and were correctly classified by LILAC.

# 6 Conclusion

In this work, we proposed LILAC, which rethinks curriculum learning based on incrementally learning labels instead of samples. This approach helps kick-start the learning process from a substantially better starting point while making the learned embedding space amenable to adaptive compensation of target vectors. Both these techniques combine well in LILAC to show the highest performance on CIFAR-10 for simple data augmentations while easily outperforming batch and curriculum learning and label smoothing methods on comparable network architectures. The next step in unlocking the full potential of this setup is to include a confidence measure on the predictions of the network so that it can handle the effects of dropout or partial inputs. In further expanding LILAC's ability to handle partial inputs, we aim to explore its effect on standard incremental learning (memory-constrained) while also extending its applicability to more complex neural network architectures.

# 7 Acknowledgements

This work was in part supported by NSF NRI IIS 1522904 and NIST 60NANB17D191. The findings and views represent those of the authors alone and not the funding agencies. The authors would also like to thank members of the COG lab for their invaluable input in putting together and refining this work.

# References

[1] Judith Avrahami, Yaakov Kareev, Yonatan Bogot, Ruth Caspi, Salomka Dunaevsky, and Sharon Lerner. Teaching by examples: Implications for the process of category acquisition. *The Quarterly Journal of Experimental Psychology Section A*, 50(3):586–606, 1997.

[2] Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. *arXiv preprint arXiv:1805.02641*, 2018.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[4] Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. Hard negative mining for metric learning based zero-shot classification. In Gang Hua and Hervé Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 524–531, Cham, 2016. Springer International Publishing. ISBN 978-3-319-49409-8.

[5] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248, 2018.

[6] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019.

[7] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[8] Urun Dogan, Aniket Anand Deshmukh, Marcin Machura, and Christian Igel. Label-similarity curriculum learning. *arXiv preprint arXiv:1911.06902*, 2019.

[9] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

[10] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Artificial Intelligence and Statistics*, pages 153–160, 2009.

[11] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *International Conference on Learning Representations*, 2018.

[12] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 482–495. PMLR, 13–15 Nov 2017.

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[14] Ben Graham. Fractional max-pooling (2014). *arXiv preprint arXiv:1412.6071*, 2014.

[15] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1311–1320. JMLR. org, 2017.

[16] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 2535–2544, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[19] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 2304–2313. PMLR, 10–15 Jul 2018.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[21] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.

[22] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10 (Jan):1–40, 2009.

[23] Xirong Li, CeesG M Snoek, Marcel Worring, Dennis Koelma, and Arnold WM Smeulders. Bootstrapping visual categorization with relevant negatives. *IEEE Transactions on Multimedia*, 15(4):933–945, 2013.

[24] Senwei Liang, Yuehaw Kwoo, and Haizhao Yang. Drop-activation: Implicit parameter reduction and harmonic regularization. *arXiv preprint arXiv:1811.05850*, 2018.

[25] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[26] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing by virtual adversarial examples. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[27] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. *CoRR*, 2017.

[28] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[29] Scott E. Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[30] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[31] Pau Rodríguez, Miguel A Bautista, Jordi Gonzalez, and Sergio Escalera. Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing*, 75: 21–31, 2018.

[32] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pages 348–358, 2019.

[33] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4535–4544, 2018.

[34] Burrhus F Skinner. Reinforcement today. *American Psychologist*, 13(4):94, 1958.

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[36] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

[37] Daphna Weinshall, Gad Cohen, and Dan Amir. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 5238–5246, Stockholmsmässan, Stockholm Sweden, 2018. PMLR.

[38] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4753–4762, 2016.

[39] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedrop regularization for deep residual learning. *arXiv preprint arXiv:1802.02375*, 2018.

[40] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[41] Ke Zhang, Miao Sun, Tony X Han, Xingfang Yuan, Liru Guo, and Tao Liu. Residual networks of residual networks: Multilevel residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(6):1303–1314, 2017.