LERAC: LEARNING RATE CURRICULUM – SUPPLEMENTARY

Anonymous authors
Paper under double-blind review

ABSTRACT

In the supplementary, we give a theoretical proof of the underlying assumption of LeRaC. Furthermore, we present a series of additional experiments to validate several choices and statements in our paper. We also clarify several important aspects and discuss the limitations of our work.

1 THEORETICAL PROOF

The motivation behind using LeRaC stems from our conjecture stating that the level of noise inside features gradually increases with every layer of a neural network. Regardless of the type of layer (convolutional, transformer or fully connected), the operation performed inside a neural layer boils down to matrix or vector multiplications. To this end, we set out to demonstrate that the signal resulting from the multiplication of two signals has a lower signal-to-noise ratio (SNR) than the multiplied factors. We start with the definition of the variance of a signal, which is given below:

Definition 1. *The variance of a signal s is given by:*

$$\mathrm{Var}(s) = E[s^2] - E[s]^2. \tag{1}$$

From Definition 1, it results that the expected value of s^2, which represents the power of signal s, is equal to:

$$E[s^2] = E[s]^2 + \mathrm{Var}(s) = \mu_s^2 + \sigma_s^2, \tag{2}$$

where μ_s is the mean of s, and σ_s^2 is the variance of s. We use Eq. (2) to define the SNR of a signal as follows:

Definition 2. *The signal-to-noise ratio (SNR) of a signal $s = u + z$, where u is the clean signal and z is the noise component, is the ratio between the power of u and the power of z, which is given by:*

$$\mathrm{SNR}(s) = \frac{E[u^2]}{E[z^2]} = \frac{\mu_u^2 + \sigma_u^2}{\mu_z^2 + \sigma_z^2}, \tag{3}$$

where μ_u and μ_z are the means of u and z, and σ_u^2 and σ_z^2 are the variances of u and z, respectively.

The noise contained by data samples given as input to neural networks is usually uncorrelated, *e.g.* the noise in images is assumed to come from a random normal distribution of zero mean. Moreover, the weights of a neural network are usually initialized by sampling them from a random normal distribution of zero mean (Glorot & Bengio, 2010). Hence, without loss of generality, we can naturally assume that the noise component has zero mean. This means that we can simplify Eq. (3) to:

$$\mathrm{SNR}(s) = \frac{\mu_u^2 + \sigma_u^2}{\sigma_z^2}. \tag{4}$$

If the power of the signal u is higher than the power of the noise z, then $\mathrm{SNR}(s)$ is higher than 1. If the signal is dominated by noise, then $\mathrm{SNR}(s)$ is between 0 and 1. Note that the SNR does not take negative values. To avoid discussing edge cases, we assume that the SNR of any signal is always defined, *i.e.* the power of the noise is never 0.

Theorem 1. *Let $s_1 = u_1 + z_1$ and $s_2 = u_2 + z_2$ be two signals, where u_1 and u_2 are the clean components, and z_1 and z_2 are the noise components. The signal-to-noise ratio of the product between the two signals is lower than the signal-to-noise ratios of the two signals,* i.e.*:*

$$\mathrm{SNR}(s_1 \cdot s_2) \leq \mathrm{SNR}(s_i), \forall i \in \{1, 2\}. \tag{5}$$

Proof. To demonstrate our proposition, we rely on the formula of variance for the sum of two signals with zero mean:

$$\mathrm{Var}(s_1 + s_2) = \mathrm{Var}(s_1) + \mathrm{Var}(s_2). \tag{6}$$

We also rely on the formula of variance for the product of two signals:

$$\mathrm{Var}(s_1 \cdot s_2) = \mathrm{Var}(s_1) \cdot \mathrm{Var}(s_2) + \mathrm{Var}(s_1) \cdot E[s_2]^2 + \mathrm{Var}(s_2) \cdot E[s_1]^2. \tag{7}$$

Let s denote the product of the two signals, *i.e.* $s = s_1 \cdot s_2$. Expanding the signals s_1 and s_2 leads to the following formulation of s:

$$s = s_1 \cdot s_2 = (u_1 + z_1) \cdot (u_2 + z_2) = u_1 \cdot u_2 + u_1 \cdot z_2 + u_2 \cdot z_1 + z_1 \cdot z_2, \tag{8}$$

where the clean component is $u = u_1 \cdot u_2$, and the noise component is $z = u_1 \cdot z_2 + u_2 \cdot z_1 + z_1 \cdot z_2$. Hence, $s = u + z$.

An example given as input to a neural network and the initial weights of the respective neural network are not correlated under any practical circumstances. Hence, without loss of generality, we can assume that the signals s_1 and s_2 are independent of each other, *i.e.* their covariance is equal to 0. This assumption allows us to simplify the signal power of u to:

$$E[u^2] = E[u_1^2 \cdot u_2^2] = E[u_1^2] \cdot E[u_2^2] = \left(\mu_{u_1}^2 + \sigma_{u_1}^2\right) \cdot \left(\mu_{u_2}^2 + \sigma_{u_2}^2\right). \tag{9}$$

The signal power of z is given by:

$$E[z^2] = E[z]^2 + \mathrm{Var}(z) = \mathrm{Var}(z), \tag{10}$$

since the noise is of zero mean, *i.e.* $E[z] = 0$. By employing Eq. (6), we can compute the power of z as follows:

$$\begin{aligned} E[z^2] = \mathrm{Var}(z) &= \mathrm{Var}(u_1 \cdot z_2 + u_2 \cdot z_1 + z_1 \cdot z_2) \\ &= \mathrm{Var}(u_1 \cdot z_2) + \mathrm{Var}(u_2 \cdot z_1) + \mathrm{Var}(z_1 \cdot z_2). \end{aligned} \tag{11}$$

By applying Eq. (7) in Eq. (11), and considering that z_1 and z_2 have zero mean, we obtain:

$$\begin{aligned} \mathrm{Var}(u_1 \cdot z_2) &= \left(\mu_{u_1}^2 + \sigma_{u_1}^2\right) \cdot \sigma_{z_2}^2, \\ \mathrm{Var}(u_2 \cdot z_1) &= \left(\mu_{u_2}^2 + \sigma_{u_2}^2\right) \cdot \sigma_{z_1}^2, \\ \mathrm{Var}(z_1 \cdot z_2) &= \sigma_{z_1}^2 \cdot \sigma_{z_2}^2. \end{aligned} \tag{12}$$

Replacing Eq. (9) and Eq. (12) inside Definition 2 leads to the following expression of the signal-to-noise ratio of signal s:

$$\begin{aligned} \mathrm{SNR}(s) = \frac{E[u^2]}{E[z^2]} &= \frac{\left(\mu_{u_1}^2 + \sigma_{u_1}^2\right) \cdot \left(\mu_{u_2}^2 + \sigma_{u_2}^2\right)}{\left(\mu_{u_1}^2 + \sigma_{u_1}^2\right) \cdot \sigma_{z_2}^2 + \left(\mu_{u_2}^2 + \sigma_{u_2}^2\right) \cdot \sigma_{z_1}^2 + \sigma_{z_1}^2 \cdot \sigma_{z_2}^2} \\ &= \frac{\left(\mu_{u_1}^2 + \sigma_{u_1}^2\right) \cdot \left(\mu_{u_2}^2 + \sigma_{u_2}^2\right)}{\sigma_{z_1}^2 \cdot \sigma_{z_2}^2 \cdot \left(\frac{\mu_{u_1}^2 + \sigma_{u_1}^2}{\sigma_{z_1}^2} + \frac{\mu_{u_2}^2 + \sigma_{u_2}^2}{\sigma_{z_2}^2} + 1\right)} = \frac{\frac{\mu_{u_1}^2 + \sigma_{u_1}^2}{\sigma_{z_1}^2} \cdot \frac{\mu_{u_2}^2 + \sigma_{u_2}^2}{\sigma_{z_2}^2}}{\frac{\mu_{u_1}^2 + \sigma_{u_1}^2}{\sigma_{z_1}^2} + \frac{\mu_{u_2}^2 + \sigma_{u_2}^2}{\sigma_{z_2}^2} + 1} \\ &= \frac{\mathrm{SNR}(s_1) \cdot \mathrm{SNR}(s_2)}{\mathrm{SNR}(s_1) + \mathrm{SNR}(s_2) + 1}. \end{aligned} \tag{13}$$

To simplify our notations in the remainder of this proof, we define $a = \mathrm{SNR}(s_1)$ and $b = \mathrm{SNR}(s_2)$. By introducing these notations in Eq. (13), we obtain the following:

$$\mathrm{SNR}(s) = \frac{a \cdot b}{a + b + 1}. \tag{14}$$

Now, it remains to prove that:

$$\frac{a \cdot b}{a + b + 1} \leq a, \quad \frac{a \cdot b}{a + b + 1} \leq b. \tag{15}$$

However, since a and b are commutable in Eq. (14), it is sufficient to prove only one of the inequalities. We choose to provide the complete proof for the first inequality in Eq. (15) (as the proof for the other is analogous). We consider two separate cases, $a = 0$ and $a > 0$.

• Case (i): When $a = 0$, we obtain the following inequality:

$$\frac{0}{b + 1} \leq 0, \tag{16}$$

Training Regime	Distance	
	First Conv Layer	Last Conv Layer
conventional	38.36	709.93

Table 1: Distances between feature maps at epoch $k = 0$ and feature maps after the final epoch for ResNet-18 on CIFAR-10, while using the conventional training regime. Distances are independently computed for the first and last convolutional layers.

which clearly holds for any $b \geq 0$.

• Case (ii): When $a > 0$, we can divide both terms of the inequality by a and arrive to:

$$\frac{b}{a+b+1} \leq 1. \tag{17}$$

Next, we multiply both terms by $a + b + 1$, obtaining that:

$$b \leq a + b + 1. \tag{18}$$

We can subtract b from both terms and obtain the following:

$$0 \leq a + 1. \tag{19}$$

Since $a > 0$, it results that Eq. (19) is true. Moreover, the inequality is strict when $a > 0$. This concludes our proof. □

Corollary 1. *Let $\{s_1, s_2, ...s_n\}$ be a set of n signals, where each signal $s_i = u_i + z_i$ is formed of a clean component u_i and a noise component z_i. The following equation is true:*

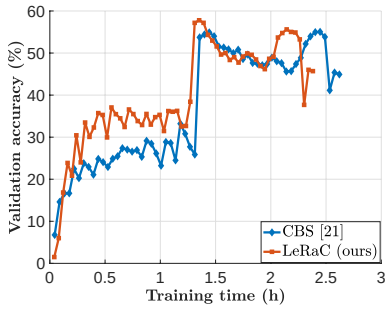
$$\text{SNR}\left(\prod_{i=1}^{p} s_i\right) \leq \text{SNR}\left(\prod_{j=1}^{p-1} s_j\right), \forall p \in \{2, ..., n\}. \tag{20}$$

Proof. The proof results immediately by induction from Theorem 1. Note that the inequality is strict when $\text{SNR}(s_i) > 0, \forall i \in \{1, 2, ..., p\}$. □

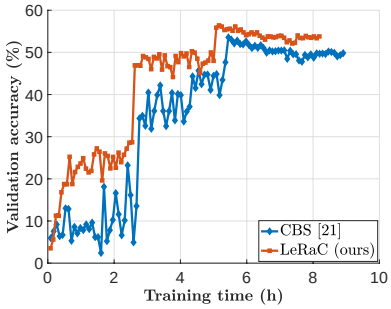
We employ Corollary 1 in the context of neural networks, where the input signal, which is expected to bear meaningful information and thus have a high SNR, is initially multiplied with random weights, which are expected to have low SNR values just after initialization. According to Corollary 1, the SNR of the resulting signal (features) is gradually decreasing, layer by layer. In this context, we conjecture that optimizing the weights θ_i of layer i to learn patterns from the signal (features) given as input to layer i is suboptimal for layers that are sufficiently far away from the input. This happens because the respective features (passed to layer i) can contain a large amount of noise, which can derail the network towards adapting the weights θ_i to the noise instead of the clean signal. This phenomenon becomes more and more prevalent as the layer i is placed farther away from the input. To regulate this phenomenon during the initial stages of the learning process, we propose to employ LeRaC and gradually decrease the learning rate as layers get deeper, allowing the network to optimize the earlier weights sooner. We underline that training the earlier layers also reduces the amount of noise in later layers, since the amount of noise in later layers is bounded by the amount of noise in earlier layers (according to Corollary 1). As the amount of noise in later layers is progressively diminished, we can gradually increase the learning rates of later layers, allowing them to optimize their weights to cleaner signals (meaningful patterns).

2 Empirical Proof

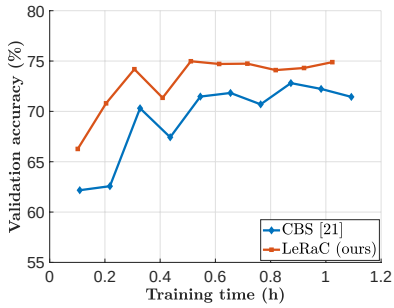
Noise quantification of early and later layers. The application of LeRaC is justified by the fact that the level of noise gradually grows with each layer during a forward pass through a neural network with randomly initialized weights. To empirically confirm this statement, we have computed the distances for the low-level (first conv) and high-level (last conv) layers between the activation maps at iteration 0 (based on random weights) and the last iteration (based on weights optimized until convergence) for ResNet-18 on CIFAR-10, while using the conventional training regime. The computed distances shown in Table 1 confirm our conjecture, namely that shallow layers contain less noise than deep layers when applying the conventional training regime.



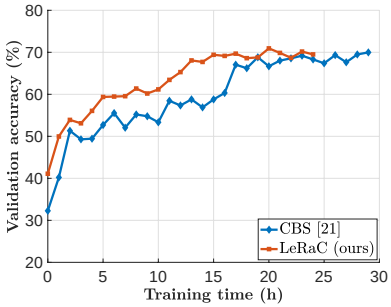
(a) ResNet-18 on Tiny ImageNet.



(b) Wide-ResNet-50 on Tiny ImageNet.



(c) BERT on BoolQ.



(d) SepTr on CREMA-D.

Figure 1: Validation accuracy (on the y-axis) versus training time (on the x-axis) for four distinct architectures. The number of training epochs is the same for both LeRaC and CBS, the observable time difference being caused by the overhead of CBS due to the Gaussian kernel layers.

Training Regime	Entropy	
	First Conv Layer	Last Conv Layer
conventional	0.9965	0.9905
LeRaC (ours)	0.9970	0.9968

Table 2: Entropy after $k = 6$ epochs for ResNet-18 on CIFAR-10, while alternating between the conventional and LeRaC training regimes.

Entropy of low-level versus high-level features. We show a few examples of training dynamics in Figure 1. All four graphs exhibit a higher gap between standard training and LeRaC in the first half of the training process, suggesting that LeRaC has an important role towards faster convergence. To assess the comparative quality of low-level versus high-level feature maps obtained either with conventional or LeRaC training, we compute the entropy of the first and last conv layers of ResNet-18 on CIFAR-10, after $k = 6$ iterations. We report the computed entropy levels in Table 2. Conventional training seems to update deeper layers faster, observing a higher difference between the entropy levels of low-level and high-level features obtained with conventional training than with LeRac. This shows that LeRaC balances the training pace of low-level and high-level features. We conjecture that updating the deeper layers too soon could lead to overfitting to the noise still present in the early layers. This statement is supported by our empirical results on 10 data sets, showing that giving a chance to the early layers to converge before introducing large updates to the later layers leads to superior performance.

Aside from computing the global entropy over all training samples, in Figure 2, we illustrate some activation maps with the highest and lowest entropy from the first and last conv layers for three randomly chosen examples from ImageNet. The activation maps are extracted at epoch $k = 6$ from the ResNet-18 model trained on CIFAR-10 either with the conventional regime or the LeRaC regime. In general, we observe that the low-level activation maps corresponding to LeRaC exhibit a higher degree of variability (being more distinct from each other), regardless of the entropy level (low or high). We believe the higher degree of variability comes from the fact that, having lower learning rates for the deeper layers, the model based on LeRaC is likely focused on finding a higher

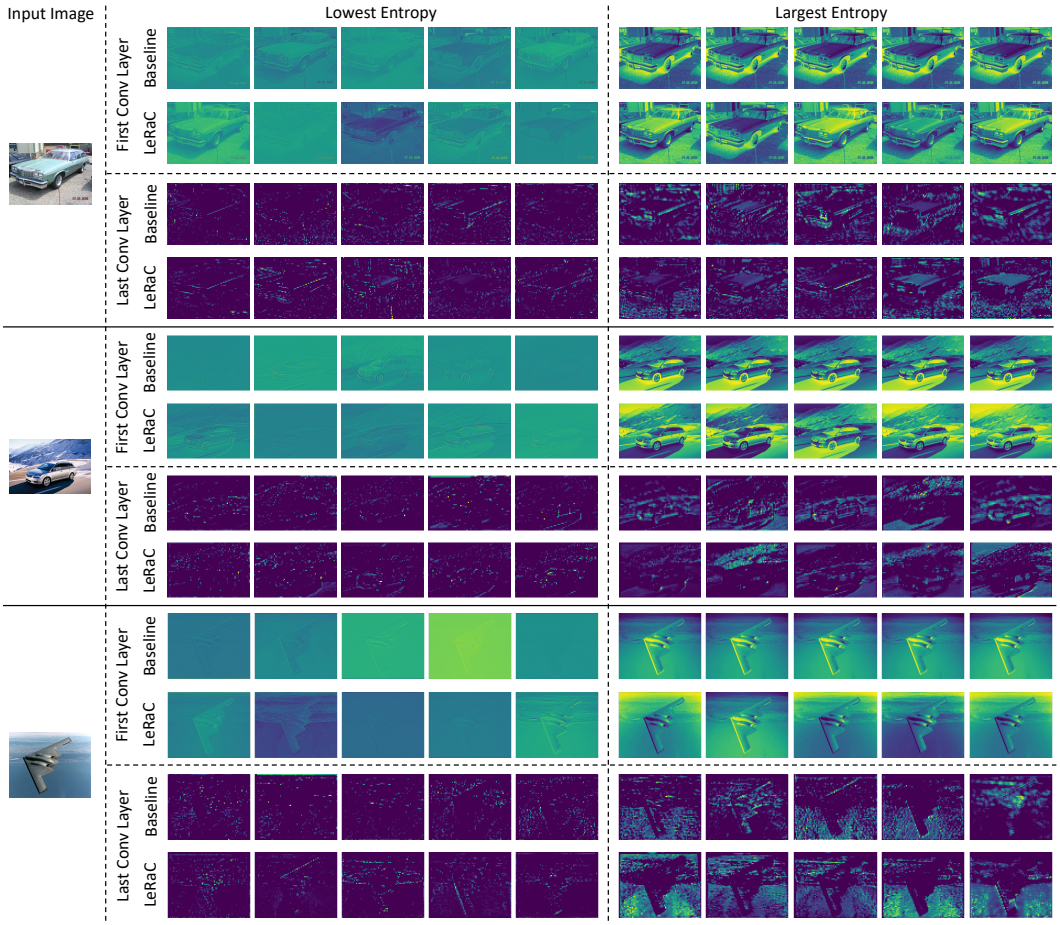


Figure 2: Activation maps with low and high entropy from the first and last conv layers of ResNet-18 trained on CIFAR-10 for $k = 6$ epochs with the conventional (baseline) and LeRaC (ours) regimes. The input images are taken from ImageNet. Best viewed in color.

Training Regime	Distance	
	First Conv Layer	Last Conv Layer
conventional	0.60	0.37
LeRaC (ours)	0.61	0.66

Table 3: Distances between feature maps at epoch $k = 6$ and feature maps after the final epoch for ResNet-18 on CIFAR-10, while alternating between the conventional and LeRaC training regimes. Distances are independently computed for the first and last convolutional layers.

variety of patterns within the first layers to minimize the loss. For the third example (the image of an airplane), we observe that the activation maps with the highest entropy from the last conv layer produced by LeRaC have a higher entropy than the activation maps with the highest entropy produced by the conventional regime. This observation is in line with the results reported in Table 2, confirming that LeRaC is able to better balance the entropy of low-level and high-level features by preventing the faster convergence of the deeper layers.

Distances at epoch k versus final epoch. As discussed above, in Table 2, we report the entropy of the low-level and high-level layers after $k = 6$ epochs, before and after using LeRaC to train ResNet-18 on CIFAR-10. However, we consider that using the distance to the final feature maps provides additional useful insights about how LeRaC works. To this end, we compute the Euclidean distances of both low-level and high-level features between epoch k and the final epoch, before and after using LeRaC. We report the distances in Table 3. The computed distances confirm our

Data Set	Model	Training Regime	Accuracy
CIFAR-100	ResNet-18	conventional	71.70±0.06
		anti-LeRaC	71.24±0.11
		LeRaC (ours)	**72.72**±0.12
	Wide-ResNet-50	conventional	68.14±0.16
		anti-LeRaC	67.47±0.15
		LeRaC (ours)	**69.38**±0.26
CREMA-D	SepTr	conventional	70.47±0.67
		anti-LeRaC	68.33±0.61
		LeRaC (ours)	**70.95**±0.56

Table 4: Average accuracy rates (in %) over 5 runs for ResNet-18 and Wide-ResNet-50 on CIFAR-100, as well as SepTr on CREMA-D, based on different training regimes: conventional, anti-LeRaC and LeRaC. The accuracy of the best training regime in each experiment is highlighted in bold.

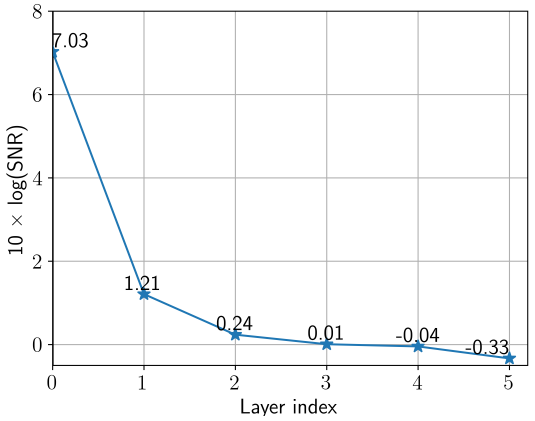


Figure 3: Average SNR of the feature maps at each layer of the randomly initialized LeNet architecture. The SNR at each layer is averaged for 100 randomly picked images from the CIFAR-100 data set. For the later layers, the SNR is negative because the signal is dominated by noise.

previous observations, namely that LeRaC is capable of balancing the training pace of low-level and high-level layers.

Motivation for exponential scheduler. Our ablation study from the main paper shows that the exponential scheduler leads to higher gains than the linear or the logarithmic schedulers. In general, a suitable scheduler is one that adjusts the learning rate at each layer proportionally to the estimated signal-to-noise drop from one layer to the next. To understand how the average SNR drops from one neural layer to the next, we plot the average SNR of the features maps at each layer of the randomly initialized LeNet architecture, computed over 100 images from CIFAR-100 in Figure 3. As anticipated, the average SNR decreases along with the layer index. Notably, we observe that the drop in SNR follows an exponential trend. This can explain why the exponential scheduler is a more suitable choice.

To further justify our preference towards the exponential scheduler, we analyze the training progress of the ResNet-18 and the pre-trained CvT-13 models using various schedulers (logarithmic, linear, exponential) for LeRaC. Figure 4 shows the results for ResNet-18, while Figure 5 shows the results for CvT-13. In both cases, the exponential scheduler leads to a better training progress than the conventional regime, but the linear and logarithmic schedulers are worse. These results confirm that the exponential scheduler is a better choice.

3 Additional Experiments

Training time comparison. For a particular model and data set, all training regimes are executed for the same number of epochs, for a fair comparison. However, the CBS strategy adds the smoothing operation at multiple levels inside the architecture, which increases the training time. To this end,

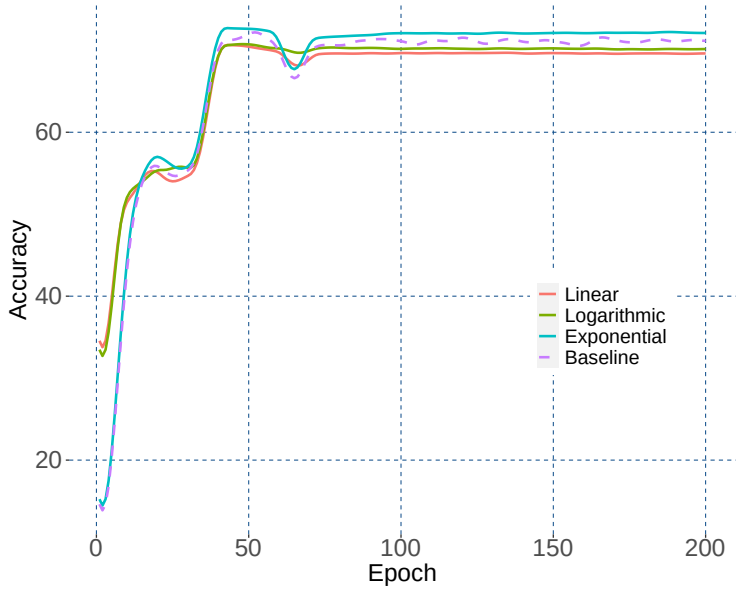


Figure 4: Test accuracy (on the y-axis) versus training time (on the x-axis) for ResNet-18 on CIFAR-100 with various curriculum schedulers. The dashed line corresponds to the conventional regime, while the continuous lines correspond to LeRaC with various schedulers. Best viewed in color.

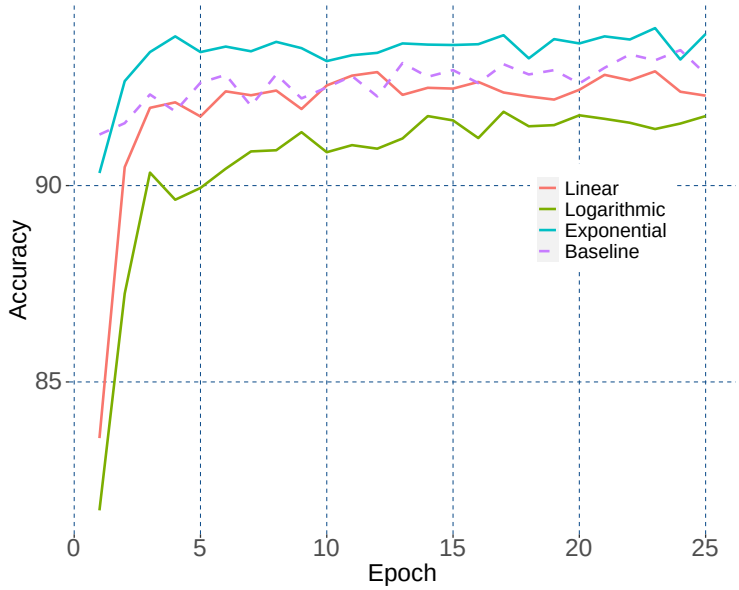


Figure 5: Test accuracy (on the y-axis) versus training time (on the x-axis) for the pre-trained CvT-13 on CIFAR-10 with various curriculum schedulers. The dashed line corresponds to the conventional regime, while the continuous lines correspond to LeRaC with various schedulers. Best viewed in color.

we compare the training time (in hours) versus the validation error of CBS and LeRaC. For this experiment, we selected four neural models and illustrate the evolution of the validation accuracy over time in Figure 1. We underline that LeRaC introduces faster convergence times, being around 7-12% faster than CBS. It is trivial to note that LeRaC requires the same time as the conventional regime.

Model	Optimizer	Training Regime	Accuracy
Wide-ResNet-50	Adam	conventional	66.48 ± 0.50
	SGD	conventional	68.14 ± 0.16
	SGD	LeRaC (ours)	$\mathbf{69.38} \pm 0.26$

Table 5: Average accuracy rates (in %) over 5 runs for Wide-ResNet-50 on CIFAR-100 using different optimizers and training regimes (conventional versus LeRaC). The accuracy of the best training regime is highlighted in bold.

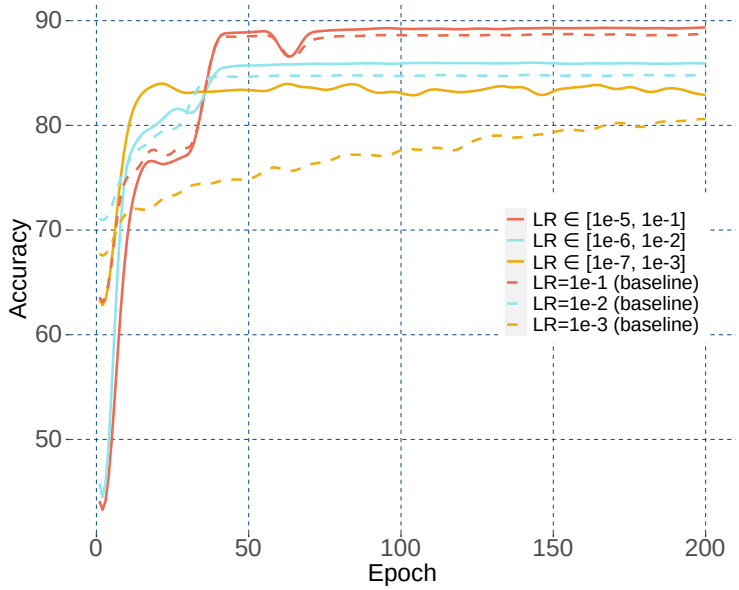


Figure 6: Test accuracy (on the y-axis) versus training time (on the x-axis) for ResNet-18 on CIFAR-10 with various configurations for the initial learning rates. Dashed lines correspond to the conventional regime, while continuous lines correspond to LeRaC. The different colors correspond to different initial learning rates. Best viewed in color.

Significance testing. To determine if the reported accuracy gains observed for LeRaC with respect to the baseline are significant, we apply McNemar significance testing (Dietterich, 1998) to the results reported in the main article on all 10 data sets. In 20 of 26 cases, we found that our results are significantly better than the corresponding baseline, at a p-value of 0.001. This confirms that our gains are statistically significant in the majority of cases.

Anti-curriculum. Since our goal is to perform curriculum learning (from easy to hard), we restrict the settings for $\eta_j, \forall j \in \{1, 2, ..., n\}$, such that deeper layers start with lower learning rates. However, another strategy is to consider the opposite setting, where we use higher learning rates for deeper layers. If we train later layers at a faster pace (anti-curriculum), we conjecture that the later layers get adapted to the noise from the early layers, which could likely lead to local optima or difficult training (due to the need of readapting to the earlier layers, once these layers start learning useful features). We tested this approach (anti-LeRaC), which belongs to the category of anti-curriculum learning strategies (Soviany et al., 2022), in a set of new experiments with ResNet-18 and Wide-ResNet-50 on CIFAR-100, as well as SepTr on CREMA-D. We report the corresponding results with LeRaC and anti-LeRaC in Table 4. Although anti-curriculum, *e.g.* hard negative sample mining, was shown to be useful in other tasks (Soviany et al., 2022), our results indicate that learning rate anti-curriculum attains inferior performance compared with our approach. Furthermore, anti-LeRaC is also below the conventional regime, confirming our conjecture regarding this strategy.

Training progress for various initial learning rates. We compare the training progress of the conventional and LeRaC training regimes. We first comparatively consider the progress of ResNet-18 on CIFAR-10, shown in Figure 6, and CIFAR-100, shown in Figure 7, respectively. LeRaC is consistently better than the conventional regime for all initial learning rate configurations, on both

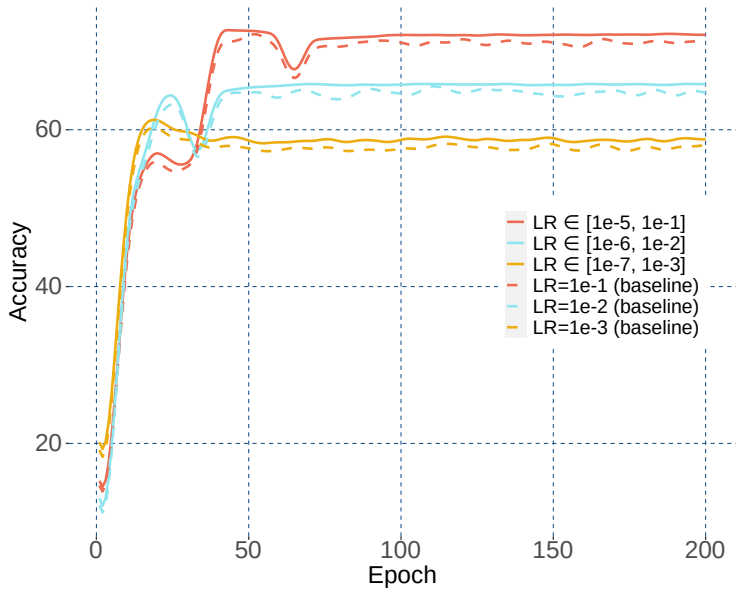


Figure 7: Test accuracy (on the y-axis) versus training time (on the x-axis) for ResNet-18 on CIFAR-100 with various configurations for the initial learning rates. Dashed lines correspond to the conventional regime, while continuous lines correspond to LeRaC. The different colors correspond to different initial learning rates. Best viewed in color.

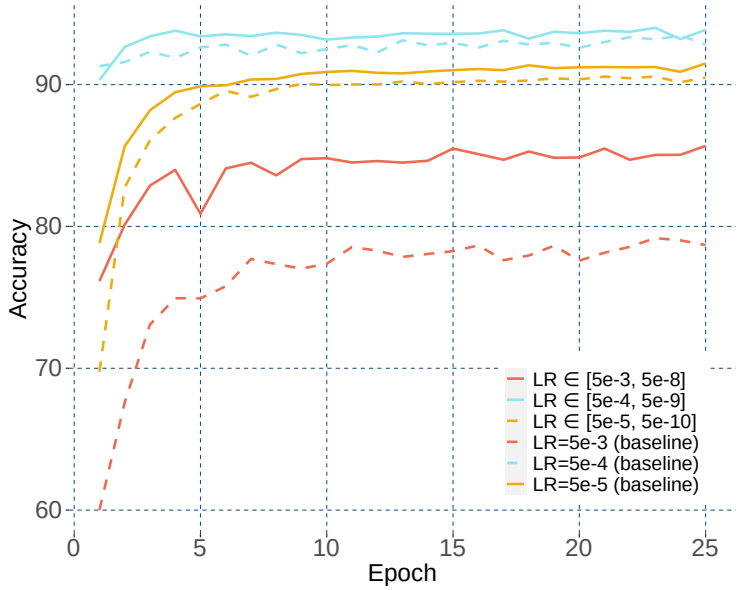


Figure 8: Test accuracy (on the y-axis) versus training time (on the x-axis) for the pre-trained CvT-13 on CIFAR-10 with various configurations for the initial learning rates. Dashed lines correspond to the conventional regime, while continuous lines correspond to LeRaC. The different colors correspond to different initial learning rates. Best viewed in color.

data sets. We next compare the progress on CIFAR-10 for ResNet-18, illustrated in Figure 6, and CvT-13 (pre-trained), illustrated in Figure 8. The training progress of LeRaC is consistently above the training progress of the conventional regime, for both ResNet-18 and CvT-13. In summary, the results showcase the benefits on the training progress of using LeRaC across distinct models and data sets.

Model	Training Regime	Accuracy
ResNet-18	conventional	72.25 ± 0.04
	LeRaC (ours)	$\mathbf{73.51} \pm 0.22$
Wide-ResNet-50	conventional	65.42 ± 0.66
	LeRaC (ours)	$\mathbf{67.00} \pm 0.55$

Table 6: Average accuracy rates (in %) over 5 runs for ResNet-18 and Wide-ResNet-50 on CIFAR-100 using data augmentation and different training regimes (conventional versus LeRaC). The accuracy of the best training regime in each experiment is highlighted in bold.

Data Set	Training Regime	Accuracy
CIFAR-10	conventional	89.20 ± 0.43
	LSCL (Dogan et al., 2020)	88.28 ± 0.14
	LeRaC (ours)	$\mathbf{89.56} \pm 0.16$
CIFAR-100	conventional	71.70 ± 0.06
	LSCL (Dogan et al., 2020)	72.59 ± 0.25
	LeRaC (ours)	$\mathbf{72.72} \pm 0.12$

Table 7: Average accuracy rates (in %) over 5 runs for ResNet-18 on CIFAR-10 and CIFAR-100 using different training regimes: conventional, LSCL (Dogan et al., 2020) and LeRaC. The accuracy of the best training regime on each data set is highlighted in bold.

SGD+LeRaC versus Adam. In Table 5, we present results showing that SGD and SGD+LeRaC obtain better accuracy rates than Adam (Kingma & Ba, 2015) for the Wide-ResNet-50 model on CIFAR-100. This indicates that a simple optimizer combined with LeRaC can obtain better results than a state-of-the-art optimizer such as Adam. This justifies our decision to use a different optimizer for each neural model (see Table 1 in the main article).

Data augmentation on vision data sets. Following Sinha et al. (2020), we did not use data augmentation for the vision data sets. We consider training data augmentation as an orthogonal method for improving results, expecting improvements for both baseline and LeRaC models. Nevertheless, since we extended the experimental settings considered in Sinha et al. (2020) to other domains, we took the liberty to use data augmentation in the audio domain (see the results in Table 4 from the main paper). The same augmentations (noise perturbation, time shifting, speed perturbation, mix-up and SpecAugment) are used for all audio models, ensuring a fair comparison. Moreover, we next present additional results with ResNet-18 and Wide-ResNet-50 on CIFAR-100 using the following augmentations: horizontal flip, rotation, solarization, blur, sharpening and auto-contrast. The results reported in Table 6 confirm that the performance gaps in the vision domain are in the same range after introducing data augmentation. In addition, we note that data augmentation seems to be rather harmful for the Wide-ResNet-50 model, which attains better results without data augmentation.

Comparing with domain-specific curriculum learning strategies. Although we consider CBS (Sinha et al., 2020) as our closest competitor in terms of applicability across architectures and domains, there are domain-specific curriculum learning methods reporting promising results. To this end, we perform additional experiments with ResNet-18 on CIFAR-10 and CIFAR-100 with another recent curriculum learning strategy (Dogan et al., 2020) applied in the image domain. Dogan et al. (2020) proposed Label-Similarity Curriculum Learning (LSCL), a strategy that relies on hierarchically clustering the classes (labels) based on inter-label similarities determined with the help of document embeddings representing the Wikipedia pages of the respective classes. The corresponding results shown in Table 7 indicate that label-similarity curriculum is useful for CIFAR-100, but not for CIFAR-10. This suggests that the method needs a sufficiently large number of classes to benefit from the constructed hierarchy of classes. In contrast, LeRaC does not rely on external components, such as the similarity measure used by Dogan et al. (2020) in their strategy. Another important limitation of LSCL (Dogan et al., 2020) is its restricted use, *e.g.* LSCL is not applicable to regression tasks, where there are no classes. Therefore, we consider LeRaC as a more versatile alternative.

Limited data regime. In all our experiments carried out so far, the evaluated models were trained on the complete training sets. However, it is interesting to find out how our strategy behaves in a limited data regime. To this end, we conduct another experiment to compare LeRaC with the

Training Set Size	Training Regime	Accuracy
5%	conventional	23.86 ± 0.32
	CBS	24.79 ± 0.17
	LeRaC (ours)	$\mathbf{25.04} \pm 0.22$

Table 8: Average accuracy rates (in %) over 5 runs for ResNet-18 on CIFAR-100 using limited training data (only 5% of the full training set) and different training regimes: conventional, CBS (Sinha et al., 2020) and LeRaC. The accuracy of the best training regime is highlighted in bold.

Model	Training Regime	CIFAR-10	CIFAR-100	Tiny ImageNet
CvT-13	conventional	71.84 ± 0.37	41.87 ± 0.16	33.38 ± 0.27
	CBS	72.64 ± 0.29	44.48 ± 0.40	33.56 ± 0.36
	LeRaC	72.90 ± 0.28	43.46 ± 0.18	33.95 ± 0.28
	CBS + LeRaC	73.25 ± 0.19	44.90 ± 0.41	34.20 ± 0.61

Table 9: Average accuracy rates (in %) over 5 runs on CIFAR-10, CIFAR-100 and Tiny ImageNet for CvT-13 based on different training regimes: conventional, CBS (Sinha et al., 2020), LeRaC with linear update, LeRaC with exponential update (proposed), and a combination of CBS and LeRaC.

conventional and CBS regimes in a limited data scenario, considering only 5% of the training data. We present the results for ResNet-18 on CIFAR-100 in Table 8. The results indicate that LeRaC keeps its performance edge in the limited data regime. We therefore conclude that LeRaC can also be useful when limited training data is available.

Combining CBS and LeRaC. Another interesting aspect worth studying is to determine if putting the CBS and LeRaC regimes together could bring further performance gains. We study the effect of combining CBS and LeRaC for CvT-13, since both CBS and LeRaC improve this model. In Table 9, we present the results with CvT-13 on CIFAR-10, CIFAR-100 and Tiny ImageNet. The reported results show that the combination brings accuracy gains across all three data sets. We thus conclude that the combination of curriculum learning regimes is worth a try, whenever the two independent regimes boost performance.

Results on ImageNet-1K. One question that naturally arises is if the results reported on ImageNet-200 in the main paper are transferable to the full ImageNet-1K. To this end, we conducted an experiment with ResNet-18 on ImageNet-1K to compare LeRaC with CBS (Sinha et al., 2020) and EfficientTrain (Wang et al., 2023). Consistent with the results on ImageNet-200, the ImageNet-1K results reported in Table 10 show that LeRaC outperforms CBS (Sinha et al., 2020). Furthermore, LeRaC also surpasses a more recent approach reporting results in the same setting, namely EfficientTrain (Wang et al., 2023).

Comparing with other efficient training strategies. We further compare with EfficientTrain (Wang et al., 2023) on CIFAR-10 and CIFAR-100 in Table 11, considering two architectures: ResNet-18 and Wide-ResNet-50. Our method surpasses EfficientTrain (Wang et al., 2023) in 3 out of 4 evaluation scenarios. These results confirm the competitiveness of LeRaC in comparison to very recent methods, such as EfficientTrain (Wang et al., 2023).

4 Discussion

Relation to curriculum learning. Consider the extreme case when the learning rate is set to zero for later layers, which is equivalent to freezing those layers. This reduces the learning capacity of the model. If layers are unfrozen one by one, the capacity of the model grows. LeRaC can be seen as a soft version of the model-level curriculum method described above. We thus consider that LeRaC is a model-level curriculum method. However, our method can also be seen as a curriculum learning strategy that simplifies the optimization in the early training stages by restricting the model updates (in a soft manner) to certain directions (corresponding to the weights of the earlier layers). Due to the imposed soft restrictions (lower learning rates for deeper layers), the optimization is easier at the beginning. As the training progresses, all directions become equally important, and the network is permitted to optimize the loss function in any direction. As the number of directions grows, the optimization task becomes more complex (it is harder to find the optimum). Another relationship to curriculum learning can be discovered by noting that the complexity of the optimization increases over time, just as in curriculum learning.

Data Set	Model	Training Regime	Accuracy
ImageNet-1K	ResNet-18	CBS (Sinha et al., 2020)	71.02
		EfficientTrain (Wang et al., 2023)	71.00
		LeRaC (ours)	**71.96**

Table 10: Accuracy rates (in %) for ResNet-18 on ImageNet-1K using different training regimes: CBS (Sinha et al., 2020), EfficientTrain (Wang et al., 2023) and LeRaC (ours). The accuracy of the best training regime is highlighted in bold.

Model	Training Regime	CIFAR-10	CIFAR-100
ResNet-18	conventional	89.20 ± 0.43	71.70 ± 0.06
	EfficientTrain (Wang et al., 2023)	89.51 ± 0.13	$\mathbf{72.83} \pm 0.12$
	LeRaC (ours)	$\mathbf{89.56} \pm 0.16$	72.72 ± 0.12
Wide-ResNet-50	conventional	91.22 ± 0.24	68.14 ± 0.16
	EfficientTrain (Wang et al., 2023)	91.03 ± 0.28	69.14 ± 0.20
	LeRaC (ours)	$\mathbf{91.58} \pm 0.16$	$\mathbf{69.38} \pm 0.26$

Table 11: Average accuracy rates (in %) over 5 runs for ResNet-18 and Wide-ResNet-50 on CIFAR-10 and CIFAR-100 using different training regimes: conventional, EfficientTrain (Wang et al., 2023) and LeRaC. The accuracy of the best training regime on each data set is highlighted in bold.

Relation to learning rate schedulers. There are some contributions (Singh et al., 2015; You et al., 2017) showing that using adaptive learning rates can lead to improved results. We explain how our method is different below. In (Singh et al., 2015), the main goal is increasing the learning rate of certain layers as necessary, to escape saddle points. Different from Singh et al. (2015), our strategy reduces the learning rates of deeper layers, introducing soft optimization restrictions in the initial training epochs. You et al. (2017) proposed to train models with very large batches using a learning rate for each layer, by scaling the learning rate with respect to the norms of the gradients. The goal of You et al. (2017) is to specifically learn models with large batch sizes, *e.g.* formed of 8K samples. Unlike You et al. (2017), we propose a more generic approach that can be applied to multiple architectures (convolutional, recurrent, transformer) under unrestricted training settings.

Gotmare et al. (2019) point out that *learning rate with warmup and restarts* is an effective strategy to improve stability of training neural models using large batches. Different from LeRaC, this approach does not employ a different learning rate for each layer. Moreover, the strategy restarts the learning rate at different moments during the entire training process, while LeRaC is applied only during the first few training epochs. Aside from these technical differences, our experiments already include a direct comparison of the two strategies for the CvT architecture, *i.e.* the baseline CvT uses warmup and restarts. The results show that introducing LeRaC brings consistent improvements. We thus conclude that our strategy is a viable and distinct alternative to learning rate with warmup and restarts.

Relation to optimizers. We consider Adam (Kingma & Ba, 2015) and related optimizers as orthogonal approaches that perform the optimization rather than setting the learning rate. Our approach, LeRaC, only aims to guide the optimization during the initial training iterations by reducing the relevance of optimizing deeper network layers. Most of the baseline architectures used in our experiments are already based on Adam or some of its variations, *e.g.* AdaMax, AdamW (Loshchilov & Hutter, 2019). LeRaC is applied in conjunction with these optimizers, showing improved performance over various architectures and application domains. This supports our claim that LeRaC is an orthogonal contribution to the family of Adam optimizers.

Interaction with other curriculum learning strategies. Our simple and generic curriculum learning scheme can be integrated into any model for any task, not relying on domain or task dependent information, *e.g.* the data samples. We already showed that combining LeRaC and CBS can boost performance (see the results presented in Table 9). In a similar fashion, LeRaC can be combined with data-level curriculum strategies for improved performance. We leave this exploration for future work.

Interaction with optimization algorithms. Throughout our experiments, we always keep using the same optimizer for a certain neural model, for all training regimes (conventional, CBS, LeRaC). The best optimizer for each neural model is established for the conventional training regime. We

underline that our initial learning rates and scheduler are used independently of the optimizers. Although our learning rate scheduler updates the learning rates at the beginning of every iteration, we did not observe any stability or interaction issues with any of the optimizers (SGD, Adam, AdaMax, AdamW).

Interaction with other learning rate schedulers. Whenever a learning rate scheduler is used for training a model in our experiments, we simply replace the scheduler with LeRaC until epoch k. For example, all the baseline CvT results are based on Linear Warmup with Cosine Annealing, this being the recommended scheduler for CvT (Wu et al., 2021). When we introduce LeRaC, we simply deactivate Linear Warmup with Cosine Annealing between epochs 0 and k. In general, we recommend deactivating other schedulers while using LeRaC for simplicity in avoiding stability issues.

Setting the initial learning rates. We should emphasize that the different learning rates $\eta_j^{(0)}$, $\forall j \in \{1, 2, ..., n\}$, are not optimized nor tuned during training. Instead, we set the initial learning rates $\eta_j^{(0)}$ through validation, such that $\eta_n^{(0)}$ is around five or six orders of magnitude lower than $\eta^{(0)}$, and $\eta_1^{(0)} = \eta^{(0)}$. After initialization, we apply our exponential scheduler, until all learning rates become equal at iteration k. In addition, we would like to underline that the difference δ between the initial learning rates of consecutive layers is automatically set based on the range given by $\eta_1^{(0)}$ and $\eta_n^{(0)}$. For example, let us consider a network with 5 layers. If we choose $\eta_1^{(0)} = 10^{-1}$ and $\eta_5^{(0)} = 10^{-2}$, then the intermediate initial learning rates are automatically set to $\eta_2^{(0)} = 10^{-1.25}$, $\eta_3^{(0)} = 10^{-1.5}$, $\eta_4^{(0)} = 10^{-1.75}$, *i.e.* δ is used in the exponent and is equal to -0.25 in this case. To obtain the intermediate learning rates according to this example, we actually apply an exponential scheduler (similar to the one defined in Eq. (9) from the main paper). This reduces the number of tunable hyperparameters from n (the number layers) to two, namely $\eta_1^{(0)}$ and $\eta_n^{(0)}$. We go even further, setting $\eta_1^{(0)} = \eta^{(0)}$ without tuning, in all our experiments. Hence, tuning is only performed for the initial learning rate of the last layer, namely $\eta_n^{(0)}$. However, tuning all $\eta_j^{(0)}$, $\forall j \in \{1, 2, ..., n\}$, might lead to better results. We leave this exploration for future work.

Setting c without tuning. Learning rates are usually expressed as a power of $c = 10$, *e.g.* 10^{-4}. If we start with a learning rate of 10^{-8} for some layer j and we want to increase it to 10^{-4} during the first $k = 5$ epochs, the intermediate learning rates are 10^{-7}, 10^{-6} and 10^{-5}. We thus believe it is more intuitive to understand what happens when setting $c = 10$ in Eq. (9), as opposed to using some tuned value for c. To this end, we refrain from tuning c and fix it to $c = 10$.

Avoiding too large learning rates. In principle, a larger learning rate implies a larger update. However, if the learning rate is too high, the model can actually diverge. This is because the gradient describes the loss function in the vicinity of the current location, providing no guarantee for the value of the loss outside this vicinity. Our implementation takes this aspect into account. Instead of increasing the learning rate for earlier layers, we reduce the learning rate for the deeper layer to avoid divergence. More precisely, we set the learning rate for the first layer $\eta_1^{(0)}$ to the original learning rate $\eta^{(0)}$ and the other initial learning rates are gradually reduced with each layer. During training, the lower learning rates are gradually increased, until epoch k. Hence, LeRaC actually slows down the learning for deeper layers, until the earlier layers have learned "good enough" features.

Number of hyperparameters. LeRaC adds only two additional tunable hyperparameters with respect to the conventional training regime. These are the lowest learning rate $\eta_n^{(0)}$ and the number of iterations k to employ LeRaC. We reduce the number of hyperparameters that require tuning by using a fixed rule to adjust the intermediate learning rates, *e.g.* by employing an exponential scheduler, or by fixing some hyperparameters, *e.g.* $c = 10$. We emphasize that CBS (Sinha et al., 2020) has three additional hyperparameters, thus having one extra hyperparameter compared with LeRaC. Furthermore, we note that data-level curriculum methods also introduce additional hyperparameters. Even a simple method that splits the examples into easy-to-hard batches that are gradually added to the training set requires at least two hyperparameters: the number of batches, and the number of iterations before introducing a new training batch. We thus believe that, in terms of the number of additional hyperparameters, LeRaC is comparable to CBS and other curriculum learning strategies. We emphasize that the same happens if we look at new optimizers, *e.g.* Adam (Kingma & Ba, 2015) adds three additional hyperparameters compared with SGD.

Limitations of our work. One limitation is the need to disable other learning rate schedulers while using LeRaC. We already tested this scenario with CvT (the baseline CvT uses Linear Warmup with Cosine Annealing, which is removed when using LeRaC), observing consistent performance gains (see Table 2 from the main paper). However, disabling alternative learning rate schedulers might bring performance drops in other cases. Hence, this has to be decided on a case by case basis. Another limitation is the possibility of encountering longer training times or poor convergence when the hyperparameters are not properly configured. We recommend hyperparameter tuning on the validation set to avoid this outcome.

5 More Details About Experiments

5.1 Data Set Descriptions

CIFAR-10. CIFAR-10 (Krizhevsky, 2009) is a popular data set for object recognition in images. It consists of 60,000 color images with a resolution of 32×32 pixels. An image depicts one of 10 object classes, each class having 6,000 examples. We use the official data split with a training set of 50,000 images and a test set of 10,000 images.

CIFAR-100. The CIFAR-100 (Krizhevsky, 2009) data set is similar to CIFAR-10, except that it has 100 classes with 600 images per class. There are 50,000 training images and 10,000 test images.

Tiny ImageNet. Tiny ImageNet is a subset of ImageNet-1K (Russakovsky et al., 2015) which provides 100,000 training images, 25,000 validation images and 25,000 test images representing objects from 200 different classes. The size of each image is 64×64 pixels.

ImageNet-200. ImageNet-200 is a part of ImageNet-1K (Russakovsky et al., 2015) with images from a subset of 200 classes, where the original resolution of the images is preserved.

PASCAL VOC 2007+2012. One of the most popular benchmarks for object detection is PASCAL VOC (Everingham et al., 2010). The datasets consists of 21,503 images which are annotated with bounding boxes for 20 object categories. The official split has 16,551 training images and 4,952 test images.

BoolQ. BoolQ (Clark et al., 2019) is a question answering data set for yes/no questions containing 15,942 examples. The questions are naturally occurring, being generated in unprompted and unconstrained settings. Each example is a triplet of the form: $\{$question, passage, answer$\}$. We use the data split provided in the SuperGLUE benchmark (Wang et al., 2019a), containing 9,427 examples for training, 3,270 for validation and 3,245 for testing.

QNLI. The QNLI (Question-answering NLI) data set (Wang et al., 2019b) is a natural language inference benchmark automatically derived from SQuAD (Rajpurkar et al., 2016). The data set contains $\{$question, sentence$\}$ pairs and the task is to determine whether the context sentence contains the answer to the question. The data set is constructed on top of Wikipedia documents, each document being accompanied, on average, by 4 questions. We consider the data split provided in the GLUE benchmark (Wang et al., 2019b), which comprises 104,743 examples for training, 5,463 for validation and 5,463 for testing.

RTE. Recognizing Textual Entailment (RTE) (Wang et al., 2019b) is a natural language inference data set containing pairs of sentences with the target label indicating if the meaning of one sentence can be inferred from the other. The training subset includes 2,490 samples, the validation set 277, and the test set 3,000 examples.

CREMA-D. The CREMA-D multi-modal database (Cao et al., 2014) is formed of 7,442 videos of 91 actors (48 male and 43 female) of different ethnic groups. The actors perform various emotions while uttering 12 particular sentences that evoke one of the 6 emotion categories: anger, disgust, fear, happy, neutral, and sad. Following Ristea & Ionescu (2021), we conduct experiments only on the audio modality, dividing the set of audio samples into 70% for training, 15% for validation and 15% for testing.

ESC-50. The ESC-50 (Piczak, 2015) data set is a collection of 2,000 samples of 5 seconds each, comprising 50 classes of various common sound events. Samples are recorded at a 44.1 kHz sam-

pling frequency, with a single channel. In our evaluation, we employ the 5-fold cross-validation procedure, as described in related works (Piczak, 2015; Ristea et al., 2022).

5.2 Domain-Specific Preprocessing

Image preprocessing. For the image classification experiments, we apply the same data preprocessing approach as Sinha et al. (2020). Hence, we normalize the images and maintain their original resolution, 32×32 pixels for CIFAR-10 and CIFAR-100, and 64×64 pixels for Tiny ImageNet. Similar to Sinha et al. (2020), we do not employ data augmentation.

Text preprocessing. For the text classification experiments with BERT, we lowercase all words and add the classification token ([CLS]) at the start of the input sequence. We add the separator token ([SEP]) to delimit sentences. For the LSTM network, we lowercase all words and replace them with indexes from vocabularies constructed from the training set. The input sequence length is limited to 512 tokens for BERT and 200 tokens for LSTM.

Speech preprocessing. The speech preprocessing steps are carried out following Ristea et al. (2022). We thus transform each audio sample into a time-frequency matrix by computing the discrete Short Time Fourier Transform (STFT) with N_x FFT points, using a Hamming window of length L and a hop size R. For CREMA-D, we first standardize all audio clips to a fixed dimension of 4 seconds by padding or clipping the samples. Then, we apply the STFT with $N_x = 1024$, $R = 64$ and a window size of $L = 512$. For ESC-50, we keep the same values for N_x and L, but we increase the hop size to $R = 128$. Next, for each STFT, we compute the square root of the magnitude and map the values to 128 Mel bins. The result is converted to a logarithmic scale and normalized to the interval $[0, 1]$. Furthermore, in all our speech classification experiments, we use the following data augmentation methods: noise perturbation, time shifting, speed perturbation, mix-up and SpecAugment (Park et al., 2019).

References

Houwei Cao, David G. Cooper, Michael K. Keutmann, Ruben C. Gur, Ani Nenkova, and Ragini Verma. CREMA-D: Crowd-sourced emotional multimodal actors dataset. *IEEE Transactions on Affective Computing*, 5(4):377–390, 2014.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *Proceedings of NAACL*, pp. 2924–2936, 2019.

Thomas G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

Ürün Dogan, Aniket Anand Deshmukh, Marcin Bronislaw Machura, and Christian Igel. Label-similarity curriculum learning. In *ECCV*, pp. 174–190, 2020.

Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Intenational Journal of Computer Vision*, 88(2):303–338, 2010.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pp. 249–256, 2010.

Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. In *Proceedings of ICLR*, 2019.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *Proceedings of ICLR*, 2015.

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

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *Proceedings of ICLR*, 2019.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *Proceedings of INTERSPEECH*, pp. 2613–2617, 2019.

Karol J. Piczak. ESC: Dataset for Environmental Sound Classification. In *Proceedings of ACMMM*, pp. 1015–1018, 2015.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of EMNLP*, pp. 2383–2392, 2016.

Nicolae-Catalin Ristea and Radu Tudor Ionescu. Self-paced ensemble learning for speech and audio classification. In *Proceedings of INTERSPEECH*, pp. 2836–2840, 2021.

Nicolae-Catalin Ristea, Radu Tudor Ionescu, and Fahad Shahbaz Khan. SepTr: Separable Transformer for Audio Spectrogram Processing. In *Proceedings of INTERSPEECH*, pp. 4103–4107, 2022.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Bharat Singh, Soham De, Yangmuzi Zhang, Thomas Goldstein, and Gavin Taylor. Layer-specific adaptive learning rates for deep networks. In *Proceedings of ICMLA*, pp. 364–368, 2015.

Samarth Sinha, Animesh Garg, and Hugo Larochelle. Curriculum by smoothing. In *Proceedings of NeurIPS*, pp. 21653–21664, 2020.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *Proceedings of NeurIPS*, volume 32, pp. 3266–3280, 2019a.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of ICLR*, 2019b.

Yulin Wang, Yang Yue, Rui Lu, Tianjiao Liu, Zhao Zhong, Shiji Song, and Gao Huang. EfficientTrain: Exploring Generalized Curriculum Learning for Training Visual Backbones. In *Proceedings of ICCV*, pp. 5852–5864, 2023.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing Convolutions to Vision Transformers. In *Proceedings of ICCV*, pp. 22–31, 2021.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.