## 1 Extended Implementation Details

In our proposed graphical model, as illustrated in Fig. 1b, the term $p(y|x;\theta_y)$ utilises the baseline classifier. Therefore, the architecture of this classifier remains in alignment with the integrated baseline corresponding to this term. For the terms $q(y|x,\hat{y})$ and $p(\hat{y}|x,y;\theta_{\hat{y}},\epsilon)$, we utilise a network architecture analogous to the baseline classifier present in the state-of-the-art integrated methodologies. In our methodology, we utilised PyTorch's auto-cast feature to optimise computational efficiency (Tab. 10). The training uses stochastic gradient descent (SGD) with a momentum of 0.9. The classifiers' learning rate is kept the same as their baseline model. The noise rate $\epsilon$ is learned using the learnable parameter of the sigmoid activation function, where training uses SGD with a learning rate of 0.001 and momentum of 0.9. The WarmUp stage also follows the baselines with the following number of epochs: 30 for CIFAR100 [22] and red mini-ImageNet [17], 1 for Clothing1M [45], and 5 for mini-WebVision [26]. The batch sizes used are 64 for CIFAR100 [22] and red mini-ImageNet [17], 32 for Clothing1M [45] and mini-WebVision [26]. Additionally, for the self-supervision variant of red mini-ImageNet [17], we use DINO [4], where all the settings of DINO [4] are unchanged from its original work. DINO is trained on red mini-ImageNet [17] and the WarmUp stage is reduced to 10 epochs. For Clothing1M [45], pre-trained ResNet-50 is used for DivideMix [26] and CC [58], and "clean data is not used while training". Similarly, the variant without class balance is used for SSR [11] on Clothing1M [45], and no pre-trained network is used. Moreover, while training C2D [59] on mini-WebVision [26], we use the provided pre-trained classifier ResNet-50 with SimCLR [20] for self-supervision.

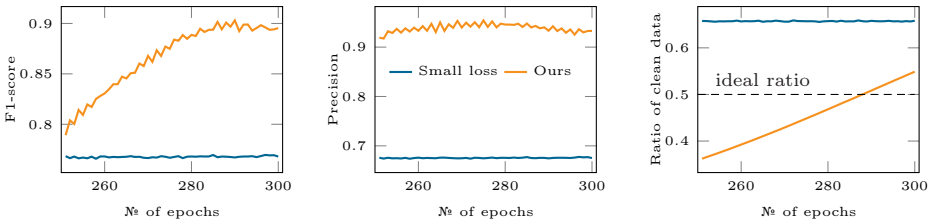
## 2 Empirical Analysis of our Approach

In this section, we compare our sample selection approach with the sample selection methods in DivideMix [26] (Fig. 4) and SSR [11] (Fig. 5) on CIFAR100 [22] at 0.5 IDN [44]. We show the F1 score, precision, and ratio of clean samples classified on the last 50 training epochs.

*F1-Score:* Fig. 4a shows DivideMix's baseline small loss approach [26] resulting in $\approx 0.70$, whilst our sample selection approach integrated with DivideMix [26] reaches around 0.92. Our approach integrated with SSR [11] is shown to achieve around 0.93 whereas the baseline SSR shows 0.94 in Fig. 5a.

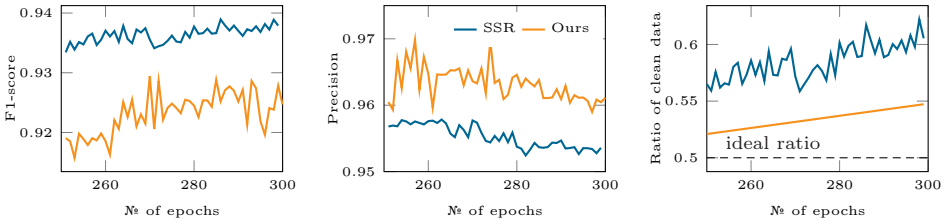
*Precision:* Fig. 4b shows the precision comparison, where DivideMix's small loss [26] reaches around 0.65, whereas our approach with DivideMix produces a result around 0.95. Additionally, Fig. 5b shows that our approach with SSR has a precision of 0.97 that is slightly larger than the SSR's precision of around 0.95.

*Ratio of samples classified as clean:* Fig. 4c exhibits the proportion of instances identified as clean. This setting employs a noise rate of 0.5. Consequently, in




**(a)** F1 Score **(b)** Precision **(c)** Clean Classified Ratio

**Fig. 4:** The graphs above show (a) F1-score, (b) precision, and (c) ratio of data classified as clean by the sample selection strategy as a function of the last 50 epochs for our approach with DivideMix [26] (orange) and DivideMix's approach [26] based on small loss (blue) on CIFAR-100 [22] with 0.5 IDN [44] noise rate.




**(a)** F1 Score **(b)** Precision **(c)** Clean classified ratio

**Fig. 5:** The graphs above show (a) F1-score, (b) precision, and (c) ratio of data classified as clean by the sample selection strategy as a function of the last 50 epochs for our approach with SSR [11] (orange) and SSR [11] alone (blue) on CIFAR-100 [22] with 0.5 IDN [44] noise rate.

an optimal scenario, the sample selection should classify half of the training samples as clean (ideal case is 0.5). Our approach with DivideMix [26] classifies 0.53 of the training data as clean, in comparison to DivideMix's [26] results of around $0.65 - 0.70$. Also, our approach with SSR [11] estimates 0.53 (Fig. 5c), which is closer to the ideal rate of 0.5 than the baseline SSR [11] which estimates $0.65 - 0.70$.

## 3 Additional Experiments and Discussion

Tab. 6 shows the outcomes of integrating our model with DivideMix [26] on IIN [16] benchmarks under symmetric noise settings for CIFAR10 [22] and CIFAR100 [22] datasets at noise rates of $0.3, 0.5$ and $0.7$. Tab. 7 *(left)* shows the results of SSR [11] and PartT [44] on CIFAR100 [22] at 0.5 IDN [44], and Tab. 7 *(right)* shows the results of DivideMix [26] and InstanceGM [15] on CIFAR100 [22] at 0.6 IDN [44]. The results are locally reproduced. Estimating noise rate and integrating it with baseline models boost their performances. Tab. 8 shows the model performance *(left)* and estimated noise rate *(right)* of our approach with

**Table 6:** *(top)* Test accuracy % and *(bottom)* final estimated noise rate $\epsilon$ on CIFAR10 [22] and CIFAR100 [22] under different symmetric IIN [16]. Here, we integrate DivideMix [26] into our proposed model.

| **Method** | **CIFAR10-IIN** | | | **CIFAR100-IIN** | | |
|---|---|---|---|---|---|---|
| | **0.3** | **0.5** | **0.7** | **0.3** | **0.5** | **0.7** |
| DivideMix [26] | 95.1 | 94.6 | 93.7 | 75.2 | 74.6 | 61.3 |
| **DivideMix-Ours** | 95.9 | 95.2 | 94.1 | 78.9 | 77.0 | 63.5 |

| **Estimated Noise Rates** | | | | | | |
|---|---|---|---|---|---|---|
| DivideMix-Ours | 0.31 | 0.48 | 0.68 | 0.29 | 0.49 | 0.71 |

**Table 7:** *(left)* Test accuracy (%) of baseline SSR [11], PartT [44], and our approach with SSR [11] and PartT [44] on CIFAR100 [22] at 0.5 IDN [44]. *(right)* Test accuracy (%) of baseline DivideMix [26], InstanceGM [15] and our approach with DivideMix [26] and InstanceGM [15] on CIFAR100 [22] at high 0.6 IDN [44]. All the results are locally reproduced by us and we also show our approach's final estimated noise rates in all cases

| **Method** | **Test Accuracy** | **Noise Estimation** |
|---|---|---|
| SSR [11] | 75.8 | |
| **SSR-Ours** | 76.9 | 0.47 |
| PartT [44] | 56.8 | |
| **PartT-Ours** | 58.2 | 0.52 |

| **Method** | **Test Accuracy** | **Noise Estimation** |
|---|---|---|
| DivideMix [26] | 50.12 | |
| **DivideMix-Ours** | 57.20 | 0.57 |
| InstanceGM [15] | 72.01 | |
| **InstanceGM-Ours** | 74.62 | 0.58 |

baseline DivideMix [26] on red mini-ImageNet [17] at low noise rates of $0.1, 0.2$ and 0.3. Tab. 9 presents the noise rate estimation with standard deviation of our model integrated with DivideMix [26] and InstanceGM [15] on CIFAR100 [22] under IDN [44] settings at $0.2, 0.3, 0.4$ and 0.5.

*Discussion (DivideMix vs other models):* In Tab. 1 *(left)*, both DivideMix-Ours and InstanceGM-Ours present improvements of 1.5% to 9% for noise rates$\geq$ 0.4, but larger improvements for DivideMix-Ours suggest that DivideMix is farther from upper-bound result in that benchmark than InstanceGM. In Tab. 1 *(right)*, both DivideMix-Ours and InstanceGM-Ours show comparably significant improvements (from 3% to 9%), particularly at noise rate 0.4, but InstanceGM-SS-Ours shows large improvement only for low noise rate of 0.2. Fig. 2b shows significant improvements in DivideMix [26] compared to F-DivideMix [21]. This is because we focus on the small-loss hypothesis [16] instead of the small distance to class-specific eigenvector [21] for sample selection. Given that DivideMix [26] relies on the small-loss hypothesis instead of F-DivideMix's [21] eigenvalue-based sample selection, it is natural that our approach works better for DivideMix [26]. Similar to IDN, we posit that enhancements offered by our method are correlated with the extent of potential improvement available to the model within the benchmark. Furthermore, we aim to explore its dynamics, revealing significant improvements for some models, while noting more modest enhancements for others.

**Table 8:** *(left)* Test accuracy % and *(right)* final estimated noise rate $\epsilon$ for red mini-ImageNet [17] for low noise rates $0.1, 0.2$ and $0.3$. We integrate DivideMix [26] with our method.

| Method | Noise rate | | |
|---|---|---|---|
| | **0.1** | **0.2** | **0.3** |
| DivideMix [26] | 52.75 | 50.96 | 48.91 |
| **DivideMix-Ours** | **54.38** | **52.89** | **51.67** |

| **Estimated noise rates** | | | |
|---|---|---|---|
| **Method** | **Actual noise rate** | | |
| | **0.1** | **0.2** | **0.3** |
| DivideMix-Ours | 0.11 | 0.19 | 0.32 |

**Table 9:** Noise rate estimation with standard deviation of our model integrated with DivideMix [26] and InstanceGM [15] on CIFAR100 [22] under IDN [44] settings at $0.2, 0.3, 0.4$ and $0.5$.

| **Estimated noise rates ± std dev.** | | | | |
|---|---|---|---|---|
| **Method** | **Actual noise rate** | | | |
| | **0.2** | **0.3** | **0.4** | **0.5** |
| DivideMix-Ours | 0.18 ± 0.002 | 0.34 ± 0.003 | 0.43 ± 0.006 | 0.53 ± 0.004 |
| InstanceGM-Ours | 0.23 ± 0.002 | 0.37 ± 0.001 | 0.42 ± 0.003 | 0.47 ± 0.003 |

## 4 Complexity Analysis

**Table 10:** Computational analysis of the vanilla DivideMix [26] and the DivideMix with our noise rate integration.

| **Model** | **GFLOPs ↓** | **Throughput (img/sec) ↑** |
|---|---|---|
| DivideMix [26] | 1.115 | 465.25 |
| **DivideMix-ours** | 1.120 | 897.62 |

In this section, we present the complexity of our method shown in Algorithm 1 and compare it to some common SOTA methods. To simplify the analysis, we omit the warm-up state and analyse the complexity per an epoch. We also define all the notations used and show in Tab. 11 to ease the analysis. Furthermore, our interest is the complexity induced by the learning algorithm, not the dimension of input data. Thus, we omit the number of data dimension to simplify our analysis.

Our method proposed in Algorithm 1 consists of three main steps. The complexity of each step is analysed as follows:

*Sample selection* This step consists of three sub-steps:

- Forward pass to calculate the loss on $N$ samples: $\mathcal{O}(N \times |\theta_y|)$ (this sub-step can be fast-tracked by parallelism, but we use this to simplify the analysis).,
- Sort those $N$ loss values: $\mathcal{O}(N \ln N)$,
- Threshold the clean and noisy-label samples using the obtained noise rate: $\mathcal{O}(N)$.

**Table 11:** The notations used in the complexity analysis

| Notation | Description |
|---|---|
| $N$ | number of training samples |
| $C$ | the number of classes |
| $\|\theta\|$ | number of parameters $\theta$ |
| $d$ | the number of dimensions of extracted features |

Overall, the complexity of this step is: $\mathcal{O}(N(|\theta_y| + \ln N + 1))$.

*Expectation step* This step is equivalent to training a deep neural network with the loss function defined in Eq. (2). In short, it consists of two sub-steps: forward and backward. Note that for simplicity, we perform a Monte Carlo approximation by sampling a single sample from the variational distribution $q(y|x, \hat{y}; \rho)$.

The complexity of the forward pass can be decomposed into:

- Forward pass for $\ln p(\hat{y}|x, y; \theta_{\hat{y}}, \epsilon)$: $\mathcal{O}(N|\theta_{\hat{y}}|)$,
- Forward pass for $\ln p(y|x; \theta_y)$: $\mathcal{O}(N|\theta_y|)$
- Forward pass for the entropy $\mathbb{H}(q)$ (one forward pass to calculate $y$, then entropy): $\mathcal{O}(N(|\rho| + C))$

The complexity of the backward pass with autodiff is the same as the forward pass. Thus, the overall complexity in this case is: $\mathcal{O}(2N(|\theta_{\hat{y}}| + |\theta_y| + |\rho| + C))$.

*Maximisation step* Similar to the E step, the M step also calculate the lower-bound $Q$ defined in Eq. (2). Thus, it shares a similar complexity with the E step. When adding constraints into the objective function of the M step as mentioned in Sec. 3.3, the complexity in this M step is added another term, denoted as $\mathcal{O}(T_{\text{constraint}})$.

The overall complexity in this case is: $\mathcal{O}(2N(|\theta_{\hat{y}}| + |\theta_y| + |\rho| + C) + T_{\text{constraint}})$.

*Overall complexity per epoch*

$$\mathcal{O}(N(5|\theta_y| + \ln N + 1 + 4|\theta_{\hat{y}}| + 4|\rho| + 4C) + T_{\text{constraint}}).$$

In general, $N \ll \min(|\theta_y|, |\theta_{\hat{y}}|, |\rho|)$ and so is $C$. Thus, we can simplify the overall complexity as follows:

$$\boxed{\mathcal{O}(N(5|\theta_y| + 4|\theta_{\hat{y}}| + 4|\rho|) + T_{\text{constraint}}).} \tag{9}$$

Hence, if a base label noise algorithm is used (e.g., DivideMix [26] or FINE [21]), our method proposed in Algorithm 1 will add up a complexity of $N(5|\theta_y| + 4|\theta_{\hat{y}}| + 4|\rho|)$.

The complexity term $T_{\text{constraint}}$ depends on the base label noise method used and are presented as follows:

### 4.1 DivideMix as Base Method

Note that the DivideMix [26] used in our method does not require to apply Gaussian mixture modelling to cluster loss values. In our case, we rely on $\epsilon$ – the label noise rate – to threshold the loss values as shown in Eq. (7). Thus, the additional complexity included in the M step is mainly due to MixMatch [3].

- Data augmentation: $\mathcal{O}(N)$
- Label augmentation (including predicted label or forward pass): $\mathcal{O}(N(|\theta_y|+1))$
- MixMatch:
  - mixup: $\mathcal{O}(2N)$ (assume that the input $|x|$ is reasonably small compared to the number of model's parameters)
  - loss on mixup data (forward pass): $\mathcal{O}(N|\theta_y|)$
- Back propagation to train model with autodiff: $\mathcal{O}(N|\theta_y|)$

The additional complexity with DivideMix-based approach can then be presented as:

$$T_{\text{DivideMix}} = \mathcal{O}(3N|\theta_y|) \tag{10}$$

If we assume that the models used are similar: $|\theta_{\hat{y}}| \approx |\theta_y| \approx |\rho|$, then substituting into Eq. (9) results in approximately 5 times higher than DivideMix [26]. Note that this calculation ignores the parallelism and mixed-precision. In practice, this number would be much smaller. We also adopt an efficient empirical approach where only one gradient update is executed, effectively reducing computational costs. Although the complexity is higher, it is essential to note that our approach incorporates several techniques to expedite the training process. Such an increase in complexity is justifiable, considering the enhanced performance our model offers.

### 4.2 FINE as Base Method

The FINE-based approach [21] is similar to the DivideMix-based approach, except the step of sample selection where the distance to the eigenvector of the largest eigenvalue is used as a replacement for loss value. In this case, its complexity includes an extra overhead due to eigen decomposition of $C$ Gram matrices: $\mathcal{O}(Cd^3)$.

In general, the FINE-based approach [21] has an additional overhead of $\mathcal{O}(Cd^3)$ compared to the DivideMix-based approach [26].

## 5 Statistical Analysis: Mean Accuracy and Confidence Intervals

Tab. 12 reports the mean classification accuracy and corresponding confidence intervals on the CIFAR100 [22], red mini-ImageNet [18], and Clothing1M [45] datasets, obtained by running the training procedure multiple times with different initialisations. For the CIFAR100 [22] dataset, our proposed method consistently

**Table 12:** Mean test accuracy (%) with 95% confidence interval (in red) over three separate trainings on CIFAR100 [22], red mini-ImageNet [18], and Clothing1M [45].

| **Method** | **CIFAR100- IDN** | | |
|---|---|---|---|
| | **0.2** | **0.4** | **0.5** |
| DivideMix [26] | 76.98 ± 0.12 | 71.01 ± 0.07 | 57.42 ± 0.24 |
| DivideMix-Ours | 77.81 ± 0.04 | 72.60 ± 0.27 | 64.51 ± 0.17 |
| InstanceGM [15] | 79.32 ± 0.14 | 78.26 ± 0.04 | 76.91 ± 0.34 |
| InstanceGM-Ours | 79.60 ± 0.02 | 79.57 ± 0.04 | 78.41 ± 0.22 |
| **Method** | **Red mini-Imagenet [17]** | | |
| | **0.4** | **0.6** | **0.8** |
| DivideMix [26] | 46.02 ± 0.29 | 43.27 ± 0.34 | 33.02 ± 0.54 |
| DivideMix-Ours | 51.21 ± 0.37 | 45.02 ± 0.29 | 38.12 ± 0.46 |
| InstanceGM [15] | 53.01 ± 0.24 | 47.11 ± 0.37 | 38.66 ± 0.47 |
| InstanceGM-Ours | 55.92 ± 0.12 | 51.20 ± 0.29 | 44.02 ± 0.41 |

| | DivideMix [26] | **DivideMix-Ours** | |
|---|---|---|---|
| | % | % | **Noise Rate** |
| Clothing1M [45] | 73.92 ± 0.22 | 74.47 ± 0.24 | 0.41 ± 0.005 |

achieves higher mean classification accuracies compared to existing state-of-the-art methods, DivideMix [26] and InstanceGM [15]. Specifically, at a noise level of 0.5, our method shows an improvement of approximately 7% over DivideMix [26] and around 2% over InstanceGM [15]. Moreover, on the red mini-ImageNet [18] dataset, our approach demonstrates significant improvements over the existing methods. At a noise level of 0.8, our method achieves approximately a 5% improvement over DivideMix [26] and around 6% improvement over InstanceGM [15]. Similarly, for the Clothing1M [45] dataset, our method also surpasses DivideMix [26], with our approach achieving around a 1% performance improvement. These results indicate that our approach is statistically more accurate than the existing state-of-the-art methods, highlighting robustness of estimating noise rate, and effectiveness across multiple datasets and noise conditions.

## 6 Performance Analysis with varying architecture and $\lambda$

The results presented in Tab. 13 demonstrate that our proposed method outperforms the baseline DivideMix [26] approach across a variety of network architectures on the CIFAR100 [22] dataset with 0.5 IDN [44] . In particular, our method achieves the highest test accuracy of 64.02% using the PRN18 architecture [26], accompanied by an estimated noise rate of 0.53. These findings underscore the robustness and effectiveness of our approach in enhancing classification performance and noise estimation capabilities across different model configurations.

**Table 13:** Test accuracy (%) of DivideMix [26], and DivideMix-ours with estimated noise, on CIFAR100 [22] at 0.5 IDN [44] with various network architectures.

| **Arch** | DivideMix [26] (%) | **Dividemix-Ours (%)** | **Noise Estimation** |
|---|---|---|---|
| MLP (4-layers) | 21.78 | 38.90 | 0.68 |
| CNN (9-layers) | 55.11 | 60.02 | 0.55 |
| ResNet18 | 56.20 | 62.56 | 0.52 |
| PRN18 | 58.61 | 64.02 | 0.53 |

**Table 14:** Test accuracy (%) and estimated noise rate of our approach using baseline DivideMix [26] with various $\lambda$ values, at CIFAR100 [22] at 0.5 IDN [44].

| $\lambda$ | **0.01** | **0.1** | **1** | **2** |
|---|---|---|---|---|
| DivideMix-Ours (%) | 49.10 | 56.41 | 64.02 | 57.27 |
| Noise Estimation | 0.71 | 0.67 | 0.53 | 0.59 |

Tab. 14 presents the test accuracy and estimated noise rate of our approach, which employs the baseline DivideMix [26] method with various hyperparameter values on the CIFAR100 [22] dataset under an IDN [44] setting. The results demonstrate that the choice of hyperparameter significantly impacts both the performance and the estimated noise rate. Specifically, a $\lambda$ value of 1 achieves the highest test accuracy of 64.02% and the lowest estimated noise rate of 0.53, demonstrating the importance of tuning $\lambda$ to balance the trade-off between accuracy and noise estimation.

## 7 Performance Comparison based on threshold based approach (S2E)

Tab. 15 presents the test accuracy and estimated noise rate for DivideMix-S2E [49] and our proposed approach on the CIFAR100 [22]- IDN [44] dataset. Our proposed method significantly outperforms DivideMix-S2E [49], achieving a substantially higher test accuracy compared to the baseline. Furthermore, our approach provides a more accurate estimate of the noise rate.

## 8 Further limitations and future work

While our approach demonstrates promising results, it still faces several limitations. Firstly, scaling the method to large-scale datasets with thousands of classes remains a significant challenge that we plan to address in future research, when these large scale datasets would be available. Additionally, exploring different network architectures for processing clean and noisy labels could reveal significant performance variations. Finally, the identifiability issues associated with noisy labels continue to be a limitation. Future investigations will focus on enhancing

**Table 15:** CIFAR100 [22]-0.5 IDN [44] results on threshold based approach called DivideMix-S2E [49] and our approach. We compare the test accuracy (%) and the estimated noise rate.

| | DivideMix-S2E [49] (%) | **DivideMix-Ours (%)** | **Noise Rate** |
|---|---|---|---|
| CIFAR100 [22] | 31.09 | 64.02 | 0.53 |

identifiability techniques to improve the accuracy in distinguishing between clean and noisy data. Despite these challenges, we are optimistic that addressing these areas will lead to the development of even more robust solutions.