Appendix A. Noise Generation

Uniform noise was generated by replacing a true label with a random label through uniform sampling for all datasets.

Class-conditional noise was generated by mapping TRUCK → AUTOMOBILE, BIRD → AIRPLANE, DEER → HORSE, and CAT ↔ DOG with probability $\eta$ for CIFAR-10. For CIFAR-100 and TinyImageNet, we simulated class-conditional noise by flipping each class into the next circularly with probability $\eta$.

Appendix B. Mixup Augmentation

We use Mixup technique to train DNNs on convex combinations of sample pairs ($x_1$ and $x_2$) and their corresponding labels ($y_1$ and $y_2$). Specifically, a mixed sample pair can be computed by:

$$
\begin{align*}
x' &= \lambda' x_1 + (1 - \lambda') x_2 \\
y' &= \lambda' y_1 + (1 - \lambda') y_2
\end{align*}
$$

(1)

where $\lambda'$ is randomly sampled from a beta distribution: $\lambda' = \text{Beta} (\sigma, \sigma)$. In this paper, we set $\sigma = 4$ for all data sets and noise levels.

Appendix C. Noise Reduction

In Methodology Section, we have present entropy distributions of correctly and incorrectly labeled samples on CIFAR-100. Here we also present entropy distributions on CIFAR-10 and TinyImageNet datasets in Figure 1. Furthermore, we report the mislabeling rates of the generated pseudo supervision over different phases in Table 1. As we can see, after filtering out a small set of hard samples with large entropy, the mislabeling rates drop to a low to relatively low level. Moreover, as shown in Table 1, the mislabeling rates at the end of training phase are relatively lower than that at the end of warm-up phase, this indicates that our method can iteratively reduce the mislabeling rate in learning phase.

Appendix D. Additional Comparison with DivideMix

DivideMix need to tune the hyperparameters $\lambda_u$ for different noise types and rates. In our experiments, we use the suggested values of the original paper. The results of DivideMix can be significantly influence by the changing of $\lambda_u$. For example, in the case of 0.2 uniform noise on CIFAR-10, if we set $\lambda_u = 25$ then the obtained accuracy is 91.50, which drops about 4%. DivideMix trains two networks simultaneously, thus we can choose to average the predictions from both networks in test phase. For fair comparison, we use the prediction from a single network, and here we show that we can improve our method by using same averaging strategy. We train two networks...
independently using complementary-loss on same synthetic noisy dataset and test the performance of both the first network and the ensemble of these two networks. As is shown in Table 2, the results of our method can be also improved by using this averaging strategy. In fact, for the 4 cases presented in Table 2, the single network learned using our method outperforms the ensemble strategy of DivideMix.

**Appendix E. Performance Curves**

We record the test accuracy at each training epoch to provide more detailed visualized results in Figure 5 (we randomly choose 1 trial from 3 runs). Note that for our method, we reduce the learning rate by a factor of 10 after 200 epochs, and we maintain the learning policy of DivideMix reported in their papers (in which the learning rate is reduced after 150 epochs). As shown in Figure 2, our proposed method consistently outperform other compared methods. After reducing the learning rate at epoch 200, the test accuracies of our method over different epochs are stable.