A. Example Augmented Algorithms

We detail the augmented examples from the generalization section here. We provide the modified pseudocode based on published papers and publically available code for the evaluated algorithms Co-Teaching+ [34], M-DYR-H [1], and DivideMix [14]. We bold the region where augmentation is inserted. No hyperparameters were changed in any of the experiments. All models used are the same as those used in the originally published papers.

A.1. Augmented CoTeaching+

We provide the full implementation of the Co-Teaching+ [34] algorithm with our augmentations below. Co-Teaching+ is similar to the original Co-Teaching [10], but adds a different disagreement component when the two network predictions are not similar. If the predictions are similar, training is conducted on the lower loss samples. We find that adding strong augmentation to this part of the training improves performance. Co-teaching uses a threshold \( R(\varepsilon) \) instead of a mixture model fitting to take advantage of the memorization effect. The two network setup that it uses is an effective technique in existing algorithms.

A.2. Augmenting M-DYR-H

We provide the full implementation of M-DYR-H algorithm from [1]. M-DYR-H warmup, and uses mixup train-

A.3. Augmenting DivideMix

A full version of the algorithm outlined in DivideMix is provided here (Algorithm 4). The technique is a combination of co-training, MixUp, loss modeling, and is trained in a semi-supervised learning manner. Notation and algorithm are as presented in the original paper [14]. We insert our changes in bold.
\textbf{Algorithm 4:} Augmented DivideMix.

\textbf{Input:} $\theta^{(1)}$ and $\theta^{(2)}$, training dataset $(\mathcal{X}, \mathcal{Y})$, clean probability threshold $\tau$, number of augmentations $M$, augmentation policies Augment$_1$ and Augment$_2$, sharpening temperature $T$, unsupervised loss weight $\lambda_u$, Beta distribution parameter $\alpha$ for MixMatch.

$\theta^{(1)}, \theta^{(2)} = \text{WarmUp}(\mathcal{X}, \mathcal{Y}, \theta^{(1)}, \theta^{(2)})$  
\hspace{0.5cm} // standard training (with confidence penalty)
\vspace{0.2cm}
while $e < \text{MaxEpoch}$ do
\hspace{0.5cm} $W^{(2)} = \text{GMM}(\mathcal{X}, \mathcal{Y}, \theta^{(1)})$  
\hspace{0.8cm} // model per-sample loss with $\theta^{(1)}$ to obtain clean probability
\hspace{0.5cm} for $\theta^{(2)}$ do
\hspace{0.8cm} $W^{(1)} = \text{GMM}(\mathcal{X}, \mathcal{Y}, \theta^{(2)})$  
\hspace{1.1cm} // model per-sample loss with $\theta^{(2)}$ to obtain clean probability
\hspace{0.8cm} end for
\hspace{0.5cm} for $k = 1, 2$ do
\hspace{0.8cm} $\mathcal{X}^{(k)} = \{ (x_i, y_i, w_i) | w_i \geq \tau, \forall (x_i, y_i, w_i) \in (\mathcal{X}, \mathcal{Y}, W^{(k)}) \}$  
\hspace{1.1cm} // labeled training set for $\theta^{(k)}$
\hspace{0.8cm} $\mathcal{U}^{(k)} = \{ x_i | w_i < \tau, \forall (x_i, w_i) \in (\mathcal{X}, W^{(k)}) \}$  
\hspace{1.1cm} // unlabeled training set for $\theta^{(k)}$
\hspace{0.8cm} for iter = 1 to num_iters do
\hspace{1.1cm} From $\mathcal{X}^{(k)}$, draw a mini-batch $\{(x_{b}, y_{b}, w_{b}); b \in \{1, \ldots, B\}\}$
\hspace{1.1cm} From $\mathcal{U}^{(k)}$, draw a mini-batch $\{(u_{b}; b \in \{1, \ldots, B\}\}$
\hspace{1.1cm} for $b = 1 \text{ to } B$ do
\hspace{1.4cm} $x^{d_{\text{rec}}} = \text{Augment}_2 (x_{b})$
\hspace{1.4cm} $u^{d_{\text{rec}}} = \text{Augment}_2 (u_{b})$
\hspace{1.4cm} for $m = 1 \text{ to } M$ do
\hspace{1.7cm} $\hat{x}_{b,m} = \text{Augment}_1 (x_{b})$
\hspace{1.7cm} $\hat{u}_{b,m} = \text{Augment}_1 (u_{b})$
\hspace{1.7cm} $p_{b} = \frac{1}{M} \sum_{m} p_{\text{model}} (\hat{x}_{b,m}; \theta^{(k)})$  
\hspace{1.9cm} // average the predictions across augmentations of $x_{b}$
\hspace{1.7cm} $\hat{y}_{b} = w_{b}y_{b} + (1 - w_{b})p_{b}$  
\hspace{1.9cm} // refine ground-truth label guided by the clean probability produced by the other network
\hspace{1.7cm} $\bar{y}_{b} = \text{Sharpen}(\hat{y}_{b}, T)$  
\hspace{1.9cm} // apply temperature sharpening to the refined label
\hspace{1.7cm} $\bar{q}_{b} = \frac{1}{2T} \sum_{m} \{ p_{\text{model}} (\hat{u}_{b,m}; \theta^{(1)}) + p_{\text{model}} (\hat{u}_{b,m}; \theta^{(2)}) \}$  
\hspace{1.9cm} // co-guessing: average the predictions from both networks across augmentations of $u_{b}$
\hspace{1.7cm} $\hat{q}_{b} = \text{Sharpen}(\bar{q}_{b}, T)$  
\hspace{1.9cm} // apply temperature sharpening to the guessed label
\hspace{1.1cm} end for
\hspace{1.1cm} // train using a different augmentation
\hspace{1.1cm} $\mathcal{X} = \{ (x, y) | x \in x^{d_{\text{rec}}}, y \in \hat{y} \}$  
\hspace{1.1cm} // train with different augmentation
\hspace{1.1cm} $\hat{\mathcal{U}} = \{ (u, q) | u \in u^{d_{\text{rec}}}, q \in \bar{q} \}$
\hspace{1.1cm} // train with different augmentation
\hspace{1.1cm} $L_{\mathcal{X}, \hat{\mathcal{U}}} = \text{MixMatch}(\mathcal{X}, \hat{\mathcal{U}})$
\hspace{1.1cm} // apply MixMatch
\hspace{1.1cm} $L = L_{\mathcal{X}} + \lambda_u L_{\mathcal{U}} + \lambda_v L_{\text{reg}}$  
\hspace{1.1cm} // total loss
\hspace{1.1cm} $\theta^{(k)} = \text{SGD}(L, \theta^{(k)})$  
\hspace{1.1cm} // update model parameters