Supplementary Material: Bootstrapping the Relationship Between Images and Their Clean and Noisy Labels

Brandon Smart
Australian Institute for Machine Learning
University of Adelaide, Australia
a1743623@adelaide.edu.au

Gustavo Carneiro
Centre for Vision, Speech and Signal Processing
University of Surrey, United Kingdom
g.carneiro@surrey.ac.uk

1. Additional Training Information

1.1. Self-Supervised Pretraining

For CIFAR-10, CIFAR100 and Animal10N we use SimCLR [1] for self-supervised pre-training. Pretraining is done for 1000 epochs, with the learning rate starting at 0.5, and decaying by a factor of 0.1 after 700, 800 and 900 epochs. We use stochastic gradient descent as our optimizer, with Nesterov momentum of 0.9 and weight decay of 1×10^{-4}. We use a temperature of 0.5 for SimCLR and a batch size of 512.

For Webvision, we adopt MoCo-v2 [2], trained for 100 epochs (with 1 epoch of warmup) and with a batch size of 128. We use stochastic gradient descent as our optimizer, with momentum of 0.9 and weight decay of 1×10^{-4}. The learning rate starts at 0.015, decaying by a factor of 0.1 at epoch 50.

The feature embeddings generated by our models have 512 dimensions.

1.2. Bootstrapping Training

For CIFAR10, CIFAR100 and Animal10N, we do 60 epochs of bootstrapping training with MixUp. We use a learning rate of 0.02, which decays to 0.002 after 5 epochs and to 0.0002 after 50 epochs. Stochastic gradient descent is used as the optimizer, with Nesterov momentum of 0.9 and weight decay of 5×10^{-4}.

For Webvision, we do 300 epochs of bootstrapping training with MixUp. We use a starting learning rate of 0.005, which increases linearly for the first 30 epochs until it reaches 0.1, and then follows a cosine learning rate decay (capped at a minimum of 1×10^{-5}). Stochastic gradient descent is used as the optimizer, with Nesterov momentum of 0.9 and weight decay of 1×10^{-5}.

In all cases, we use a batch size of 64 and a MixUp alpha of 0.2.

1.3. Semi-Supervised Learning

For semi-supervised learning, we use FixMatch for all our experiments, with the temperature set at 0.5, the confidence threshold for pseudo-label generation set at 0.95, and the unlabelled loss ratio set at 1.0. We train for 100,000 iterations in all cases, and use an exponential moving average momentum of 0.999.

We use a cosine learning rate, starting at 0.02 (capped at a minimum of 1×10^{-5} for Webvision, and at 1×10^{-4} for all other experiments). We use stochastic gradient descent with nesterov momentum of 0.9, and weight decay of 1×10^{-5} for Webvision and 5×10^{-4} for all other experiments.

For CIFAR10, CIFAR100 and Animal10N, we use a batch size of 64 clean samples, and 3×64 noisy samples per batch. For Webvision, we use a batch size of 32 clean samples, and 3×32 noisy samples per batch.

1.4. Final Model Training

For final model training, we do 300 epochs of training for all experiments.

We use a cosine learning rate, starting at 0.02 (capped at a minimum of 1×10^{-5} for Webvision, and at 1×10^{-4} for all other experiments). We use stochastic gradient descent with nesterov momentum of 0.9, and weight decay of 1×10^{-5} for Webvision and 5×10^{-4} for all other experiments.

We use a batch size of 64 for Webvision and 128 for all other experiments,

1.5. Creating the Clean, Noisy and Final Datasets

For all of our experiments, we generate predictions for samples by averaging over 25 weak augmentations of each sample, and use the 90% most confident predictions to estimate the noise transition matrix for the dataset.

For CIFAR10 and Animal10N, we set $K = 0.1$ and $\tau = 0.99$. For CIFAR100 and Webvision, we set $K = 0.25$ and $\tau = 0.99$.

1.6. Model Architecture

In the modified networks that we used to learn the relationship between images, noisy labels and true labels, we project both the images and noisy labels to have an encoding size of 128 before concatenating them together, with our hidden layer also having a size of 128. We use a dropout layer with $p = 0.2$ after each of these linear projections (except the final classifier head), and we use batch normalization before the final classifier head.

2. Effect of Null Label Type

In our method, we describe the use of a 'null' label to represent the case where no noisy label is present. For all of our experiments, if there are k classes in the training set, we use a k-wide zero vector as our 'null' label. Here, we experiment with two alternative choices:

- One Vectors: Using a k-wide vector filled with ones;
- $\frac{1}{k}$ Vectors: Using a k-wide vector where every value equals $\frac{1}{k}$ (so that the sum of all values is 1).

We show the results of using these alternative 'null' label representations in Table 1.

Null Label Type	Accuracy
Zero Vectors	95.70
One Vectors	95.82
$\frac{1}{k}$ Vectors	95.50

Table 1. Accuracy using different null label methods for Asym. 40% noise on CIFAR10

In our experiments, we find that the choice of null label representation has little impact on the final performance of the model. In all cases, the model is able to learn to make predictions when a noisy label is and is not present.

3. Ablation of Model Construction

For all of our results, we use a 'concatenation' based model architecture, where image features and noisy labels are combined by projecting them to the same dimensionality, and then concatenating them together before passing them through the remaining linear, ReLU and batch normalisation layers of the network. This form of combining noisy labels and image features together with concatenation is the standard method used by contemporary works [3, 4, 6].

Here, we briefly explore two other potential architectures for combining image features and noisy label information together.

- Mixture of Experts: A separate classification head is created for every noisy label class, with the noisy label controlling which noisy label head is used for the prediction. In the case of mixed noisy labels (such as when performing MixUp between two noisy labels of different classes), the model output is the linear combination of each of the classification heads, weighted by their corresponding value in the noisy label;
- Attention: Scaled dot-product attention, as described by Vaswani et al. [5], is used to allow different noisy labels to attend to different image features. For our experiments on CIFAR10, we generate a query by projecting the noisy label to a 1×16 tensor, generate keys by projecting image features to a 128×16 tensor (representing a set of 128 keys), and generate values by projecting image features to a 128×16 tensor (representing a set of 128 values). Scaled dot-product attention is then used to compute a 1×16 feature tensor, with a final linear layer acting as a classification head.

In Table 2, we show the results obtained by our training method using all three of these model architectures on 40% Asymmetric noise on CIFAR10. We see that the concatenation and mixture of experts models perform similarly well, with the attention based model performing $\sim 1.5\%$ worse.

Model Type	Accuracy
Concatenation	95.65
Mixture of Experts	95.53
Attention	94.02

Table 2. Accuracy using different model constructions for Asym. 40% noise on CIFAR10

We note however that our exploration into using these model types is limited, and there may be opportunities to further optimise for these architectures.

4. Effect of Dropping Labels for Pseudo-Label Generation

In Section 3.2, we discuss how label-dropping is used during semi-supervised learning to allow our model to make predictions with and without noisy labels present. However, we do not use label dropping for the weakly augmented samples used for pseudo-label generation, with the justification that the model loss is not backpropagated through the weakly augmented samples in the FixMatch algorithm, and that always using noisy labels improves pseudo-label accuracy. Here, we experimentally justify this decision by comparing the final model accuracy when label dropping is and is not used for weakly augmented samples during semi-supervised learning.

-	Accuracy
With Label Dropping	95.33
Without Label Dropping	95.74

Table 3. Effect of Label Dropping on Final Accuracy for Asym. 40% noise on CIFAR10

Here, we see that using label dropping for weakly augmented samples decreases the accuracy of the model. Thus, we always use noisy labels for pseudo-label generation.

5. Generating Plausible Noisy Labels for Testing Samples

In our experiments, we explore using 'null' labels in place of a noisy label for samples at testing time. However, rather than passing a null label into the model alongside the testing sample, we could attempt to generate plausible 'noisy' labels from testing samples.

In this experiment, we attempt to generate plausible noisy labels from samples using the model as it was at the end of the bootstrapping phase. For a given testing sample, we use the bootstrapping model to generate the 'noisy' label, then we pass the testing sample and the 'noisy' label into the final trained model to generate the final prediction.

-	Accuracy
With Null Labels	95.74
With Label Generation	95.09

Table 4. Comparison of Null Labels and Label Generation on Final Accuracy for Asym. 40% noise on CIFAR10

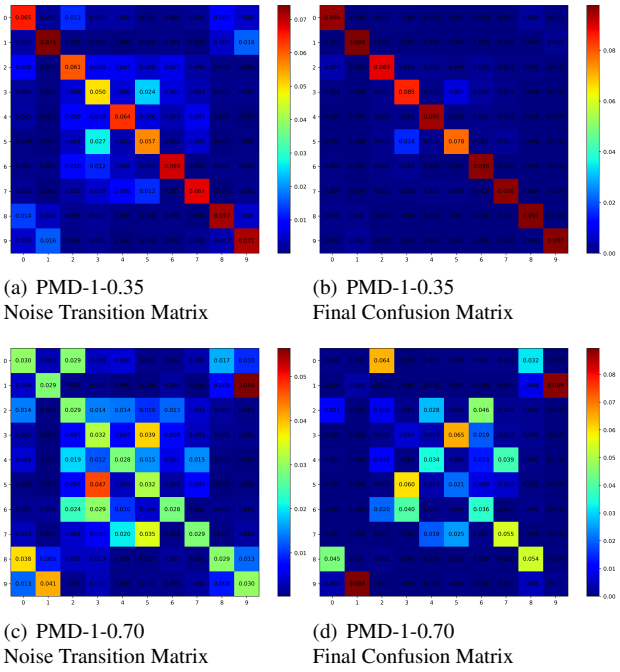
Here, we see that attempting to generate plausible noisy labels is a less effective strategy that using null labels to represent samples without associated noisy labels.

6. 70% PMD Noise on CIFAR10 and CIFAR100

In this section, we investigate the results of our method on 70% PMD noise for CIFAR10 and CIFAR100. In Table 5, we show the accuracy of our method on these datasets. We see that on CIFAR100 we get SOTA results, greatly surpassing the existing PLC method. However, on CIFAR10, we perform poorly.

To understand this, in Figures 1 and 2 we show the noise transition matrix and the final confusion matrix of our model for 35% and 70% PMD-1 noise for CIFAR10 and CIFAR100.

In Figure 1(c), we see the noise transition matrix for PMD-1-0.70 Noise on CIFAR10, and we note that by introducing 70% noise in a non-symmetric way, several of the classes have 'flipped'. For example, more cats have become labelled as dogs than are labelled as cats, and vice



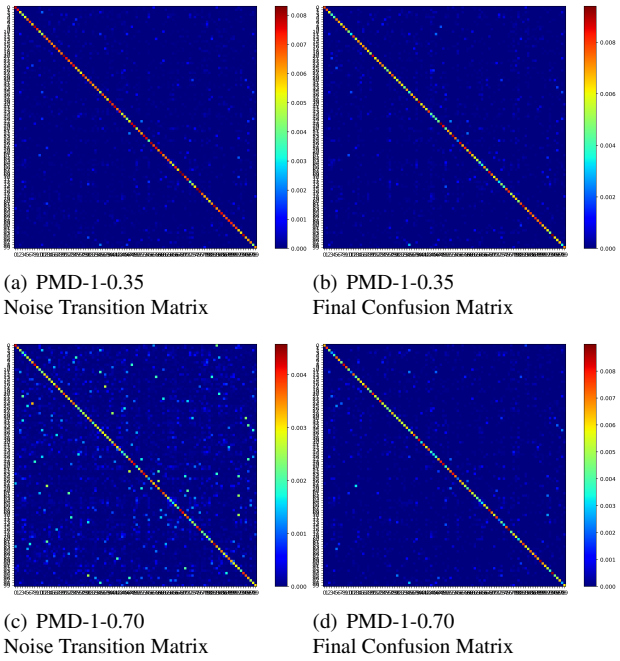
(a) PMD-1-0.35 Noise Transition Matrix

(b) PMD-1-0.35 Final Confusion Matrix

(c) PMD-1-0.70 Noise Transition Matrix

(d) PMD-1-0.70 Final Confusion Matrix

Figure 1. Noise Transition and Confusion Matrices for PMD-1 Noise on CIFAR10



(a) PMD-1-0.35 Noise Transition Matrix

(b) PMD-1-0.35 Final Confusion Matrix

(c) PMD-1-0.70 Noise Transition Matrix

(d) PMD-1-0.70 Final Confusion Matrix

Figure 2. Noise Transition and Confusion Matrices for PMD-1 Noise on CIFAR100

Dataset	CIFAR-10			CIFAR-100		
Noise Type	Type-I 70%	Type-II 70%	Type-III 70%	Type-I 70%	Type-II - 70%	Type-III - 70%
Cross-Entropy	41.98	45.57	43.42	39.32	39.30	40.01
PLC [8]	42.74	46.04	45.05	45.92	45.03	44.52
Ours (Regular Model)	21.02	27.55	21.27	58.15	53.77	57.81
+ Test-Time Aug.	*21.17*	*27.38*	*20.99*	*59.27*	*54.44*	*58.77*
Ours (Modified Model)	19.18	27.28	19.94	**58.69**	**58.03**	**57.90**
+ Test-Time Aug.	*19.71*	*27.17*	*20.21*	*59.39*	*58.95*	*58.95*

Table 5. Test accuracy (%) for Polynomial Margin Diminishing Noise [7]. Top methods are in **bold**.

versa (classes 3 and 5). Trucks and airplanes have similarly become flipped. Because of this, our method attempts to 'correct' samples to the wrong class, which we show in Figure 1(d).

If we measure the performance of our model with respect to the flipped classes (by associating each label with the modal class it represents in the noise transition matrix), we find that our model has an accuracy of 45.7%, slightly surpassing the accuracy of PLC.

On CIFAR100, this form of class flipping happens much more rarely due to the 70% of mislabelled samples being 'spread out' among over more classes. Because of this, our method is able to achieve much higher accuracy on CIFAR100 than it does in CIFAR10. In Figure 2(d), we show that the final confusion matrix for our trained model on CIFAR100 PMD-1-0.70 noise is much cleaner than it is for CIFAR10 PMD-1-0.70 noise (shown in Figure 1(d)).

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[2] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[3] Keren Gu, Xander Masotto, Vandana Bachani, Balaji Lakshminarayanan, Jack Nikodem, and Dong Yin. An instance-dependent simulation framework for learning with label noise, 2021.

[4] Naoto Inoue, Edgar Simo-Serra, Toshihiko Yamasaki, and Hiroshi Ishikawa. Multi-label fashion image classification with minimal human supervision. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2261–2267, 2017.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[6] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 839–847, 2017.

[7] Le Zhang, Ryutaro Tanno, Kevin Bronik, Chen Jin, Parashkev Nachev, Frederik Barkhof, Olga Ciccarelli, and Daniel C Alexander. Learning to segment when experts disagree. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 179–190. Springer, 2020.

[8] Yikai Zhang, Songzhu Zheng, Pengxiang Wu, Mayank Goswami, and Chao Chen. Learning with feature-dependent label noise: A progressive approach. In *International Conference on Learning Representations*, 2021.