Label Smoothing++: Enhanced Label Regularization for Training Neural Networks - Supplementary

Sachin Chhabra[1]
sachin.chhabra@asu.edu
Hemanth Venkateswara[2]
hvenkateswara@gsu.edu
Baoxin Li[1]
baoxin.li@asu.edu

[1] Arizona State University
699 S Mill Ave.
Tempe, AZ, USA- 85281

[2] Georgia State University
25 Park Pl NE
Atlanta, GA, USA - 30303

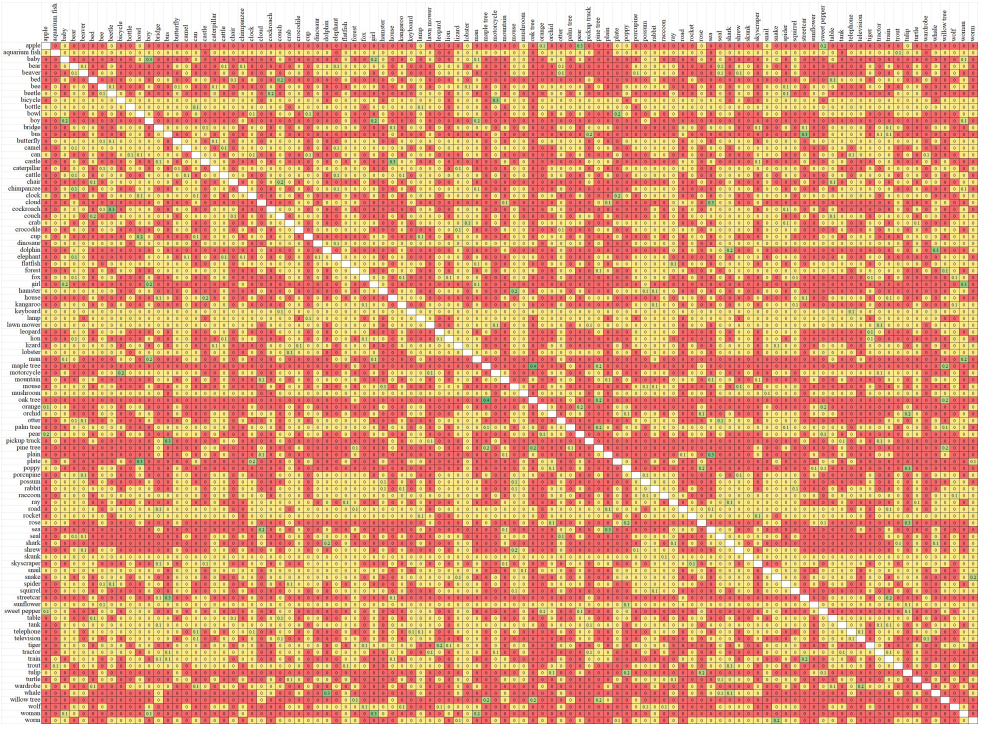

Figure 1: Learned C-Matrix of CIFAR100 dataset.

1 *C*-Matrix

In the main text, we showcased the *C*-Matrix of CIFAR10, FashionMNIST, and Animals10N as they contained only 10 classes and were easy to visualize. Here, we show the *C*-Matrix of CIFAR100 [3] in Figure 1. We can observe that *C*-Matrix captures semantic relationships even for a large number of classifications. For e.g., class *boy* has the highest assignment of 0.21 to class *baby*, 0.23 to class *girl*, 0.17 to class *man*, 0.12 to class *woman*. We can observe that these classes are quite similar to *boy* class, and their assignment is much higher than the other classes. The classes dissimilar to class *boy* are given very low similarity values in the *C*-matrix like class *apple* and class *bee* have 0 value assigned.

2 PyTorch Code

```
# Define LS++ Class
class LSPP(nn.Module):
  def __init__(self, K, alpha=0.1):
    super().__init__()
    self.K = K
    self.alpha = alpha
    self.c_matrix = nn.Parameter(torch.zeros(K, K-1), requires_grad=True)

  def forward(self, logits, y):
    pred = F.softmax(logits, 1)

    y_1hot = F.one_hot(y, num_classes=self.K).float()

    # Convert logits of c_matrix to probs
    c_matrix = F.softmax(self.c_matrix, 1)        # K, K-1

    # Add 0 at y indices to get K X K C-Matrix
    c_matrix = c_matrix.reshape(-1, self.K)       # K, K-1   -> K-1, K
    c_matrix = F.pad(c_matrix, (1, 0, 0, 0))      # K-1, K   -> K-1, K+1
    c_matrix = c_matrix.reshape(-1)               # K-1, K+1 -> K^2 - 1
    c_matrix = F.pad(c_matrix, (0, 1))            # K^2 - 1  ->  K^2
    c_matrix = c_matrix.reshape(self.K, self.K)   # K^2      ->  K, K

    # Compute Targets
    y_tgt = (1 - self.alpha) * y_1hot + self.alpha * c_matrix[y]

    # Symmetric cross-entropy loss with detach
    fwd_ce = cross_entropy_loss(y_tgt, pred.detach())
    bck_ce = cross_entropy_loss(pred, y_tgt.detach())
    loss = (fwd_ce + bck_ce) / 2
    return loss

# Define loss function
loss_fn = LSPP(K, α)

# Add C-Matrix to the training parameters
opt = SGD(list(net.parameters()) + list(loss_fn.parameters()), lr, mom, wd)
```

We showcase the PyTorch code of Label Smoothing++, for which we used the below notations to represent the variables:

- K: Number of classes

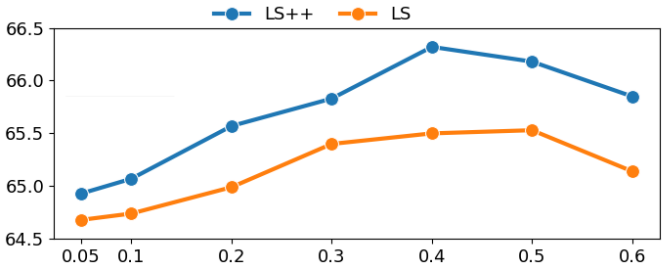

Figure 2: Accuracy vs α comparison of Label Smoothing (LS) and Label Smoothing++ (LS++) on TinyImageNet dataset using ResNet-18.

	ResNet-18		ResNet-101	
	Parameters	Time (mins)	Parameters	Time (mins)
1-hot	11,578,632	142	44,131,080	674
LS [8]	11,578,632	142	44,131,080	674
LS++	11,618,432 (0.3%↑)	144 (1.3%↑)	44,170,880 (0.1%↑)	677 (0.44%↑)

Table 1: Training parameters and training time of different approaches on Tiny-ImageNet on ResNet-18 and ResNet-101.

- net: Neural Network Model
- mom: Momentum value
- wd: Weight decay value
- α: Label Smoothing++ hyperparameter
- CE: Cross-entropy loss (refer eq. 1)

3 Sensitivity on Mixing Coefficient

We conduct a sensitivity analysis on the hyperparameter α using Label Smoothing [8] and Label Smoothing++ on the Tiny-ImageNet dataset with ResNet-18. The results of this experiment are depicted in Figure 2. Label Smoothing++ consistently outperforms label smoothing for all values of α. Tuning α has the potential to enhance performance further, but we adhered to the commonly practiced value of 0.1 for our experiments. Note at α=0, both approaches function as 1-hot vectors.

4 Training Overheads

LS++ introduces additional $K \cdot (K-1)$ training parameters. In this section, we discuss the impact of these additional training parameters on training time and total training parameters. We used the Tiny-ImageNet dataset for this study as it has the highest number of classes (200) in our experiments. The training parameters and time for ResNet18 and ResNet101

are presented in Table 1. We can observe that LS++ adds less than 0.2% parameters and has minimal impact of an extra 1% training time.

5 Limitations of Label Smoothing++

A notable constraint of our approach is its inapplicability to binary classification, necessitating a minimum of three classes. In this scenario, only one non-target class exists, and the absence of inter-class relationships undermines the effectiveness of our method. We also found Label Smoothing++ to be unbeneficial for noisy labels. In such cases, C-Matrix learns a noisy version of the relationship. However, if a clean C-Matrix is available and the network is trained with it, it can alleviate the negative impact of noisy labels.

6 Training Procedure

We stored the C-Matrix as logits (pre-softmax values) and applied softmax when necessary. Additionally, no weight decay was applied to these parameters.

6.1 Training Settings for Image Datasets:

6.1.1 ImageNet-100 [10]:

- **Image size**: 224×224.
- **Augmentations**: Standard augmentation of random resized crops of 224 along with random Horizontal flips.
- **Optimizer**: SGD optimizer with 0.9 momentum and weight decay of 1e-4.
- **Training specifics**: Batch size of 64 for 90 epochs. Learning rate starts at 0.1, undergoes linear warm-up for the first 5 epochs, and decays by a factor of 0.1 at the 30th, 60th, and 80th epochs.

6.1.2 TinyImageNet [1]:

- **Image size**: 64×64.
- **Augmentations**: Padding of size 4 with Random Crops, and random Horizontal flips.
- **Optimizer**: SGD optimizer with 0.9 momentum and weight decay of 1e-4.
- **Training specifics**: Batch size of 64 for 100 epochs. Learning rate starts at 0.1, undergoes linear warm-up for the first 5 epochs, and decays by a factor of 0.1 at the 40th and 60th epochs.

6.1.3 Animals10N [6]:

- **Image size**: 64×64.
- **Augmentations**: Padding of size 4 with Random Crops, and random Horizontal flips.

- **Optimizer**: SGD optimizer with 0.9 momentum and weight decay of 1e-4.
- **Training specifics**: Batch size of 64 for 100 epochs. Learning rate starts at 0.1, undergoes linear warm-up for the first 5 epochs, and decays by a factor of 0.1 at the 40th and 60th epochs.

6.1.4 FER2013 [?]:

- **Image size**: 48×48.
- **Augmentations**: Padding of size 4 with Random Crops, and random Horizontal flips.
- **Optimizer**: SGD optimizer with 0.9 momentum and weight decay of 1e-4.
- **Training specifics**: Batch size of 64 for 100 epochs. Learning rate starts at 0.1, undergoes linear warm-up for the first 5 epochs, and decays by a factor of 0.1 at the 40th and 60th epochs.

6.1.5 CIFAR10 and CIFAR100 [3]:

- **Augmentations**: Padding of size 4 with Random Crops, and random horizontal flips during training.
- **Optimizer**: SGD optimizer with 0.9 momentum and weight decay of 5e-4.
- **Training specifics**: Batch size of 128 for 300 epochs. Learning rate starts at 0.1, warms up linearly for the first 10 epochs, and decays by a factor of 0.1 at the 150th and 225th epochs.

6.1.6 FashionMNIST [12]:

- **Augmentation**: Padding of size 2 with Random crops and Random horizontal flips during training.
- **Network Configuration**: Input channels set to 1 for grayscale images.
- **Optimizer**: SGD optimizer with 0.9 momentum and weight decay of 1e-4.
- **Training specifics**: Batch size of 128 for 200 epochs. Learning rate starts at 0.1, undergoes linear warm-up for the initial 5 epochs, and decays by a factor of 0.1 at the 100th and 150th epochs.

6.1.7 SVHN [5]:

- **Augmentations**: No augmentation.
- **Optimizer**: SGD optimizer with 0.9 momentum and weight decay of 1e-4.
- **Training specifics**: Batch size of 128 for 200 epochs. Learning rate starts at 0.1, has a linear warm-up for the first 5 epochs, and decays by a factor of 0.1 at the 100th and 150th epochs.

6.2 Training Settings for Video Datasets:

6.2.1 Conv*+LSTM for UCF101 [7] and HDMB51 [4]:

- Training and testing on official split 1 of UCF101 and HMDB51.
- Training with a sequence of 40 frames, sampled uniformly between 1 and length of video/40.
- **Input size**: Resized frames to 112 X 112, with random horizontal flips.
- **Network**: Frozen ResNet-50 as backbone, followed by an LSTM layer and 2 fully connected layers for predictions.
- Training for 50 epochs with a batch size of 32. Adam optimizer with a learning rate of 1e-4, decayed per epoch to 0 with a cosine schedule.

6.2.2 C3D network [9] for UCF101 and HDMB51:

- Training and testing on official split 1 of UCF101 and HMDB51.
- Training for 100 epochs with a batch size of 20. SGD optimizer with a learning rate of 1e-3, 0.9 momentum, and weight decay of 5e-4. Learning rate decayed by a factor of 10 every 20 epochs.
- Input formatted as per the original C3D setting. Frames resized to 128×171, with random crops of 112×112 and a 50% probability of horizontal flip. Video sampling frequency set to every 4 frames, sequence length of 16.

6.3 Training Setting for Text Modality:

6.3.1 20-Newsgroups, AGNews, and YahooAnswers Dataset [13]:

- Pretrained BERT [2] network for initialization, finetuned on the target dataset.
- AdamW optimizer with a learning rate of 2e-5 and weight decay of 0.01. Warm-up for the first 20% of training steps, then linear decay.
- Batch size of 64 for 4 epochs (YahooAnswers) and 20 epochs (20-Newsgroups and AGNews).
- Sequence of 128 for training and 512 for testing.

6.4 Training Setting for Audio Modality:

6.4.1 GTZAN [10] and SpeechCommands [11]:

- Training for 70 epochs with a batch size of 32. Adam optimizer with a learning rate of 1e-4 and weight decay of 1e-4. Learning rate decayed by a factor of 10 every 30 epochs.
- ResNet-50 network pretrained on ImageNet or training the network from scratch.
- GTZAN: MelSpectogram and waveform of size 128×1500 for training.

- SpeechCommands: MelSpectogram with augmentations (ChangeAmplitude, ChangeSpeedAndPitchAudio, BackgroundNoise) for training.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[3] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[4] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE, 2011.

[5] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[6] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. SELFIE: Refurbishing unclean samples for robust deep learning. In *ICML*, 2019.

[7] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. URL http://arxiv.org/abs/1212.0402.

[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[9] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[10] George Tzanetakis. Automatic musical genre classification of audio signals. In *ISMIR 2001, 2nd International Symposium on Music Information Retrieval, Indiana University, Bloomington, Indiana, USA, October 15-17, 2001, Proceedings*, 2001. URL http://ismir2001.ismir.net/pdf/tzanetakis.pdf.

[11] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209, 2018. URL http://arxiv.org/abs/1804.03209.

[12] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[13] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.