Notation

Inputs What are we feeding the model with. A single input is denote with $\boldsymbol{x}$, a batch of inputs with $\boldsymbol{X}$. For example in CIFAR-10 $\boldsymbol{x}$ is single image 32×32 with 3 color channel (a tensor $3 \times 32 \times 32$, channel $\times$ height $\times$ width). A batch of 64 images will be denoted $\boldsymbol{X}$ and is a tensor $64 \times 3 \times 32 \times 32$. $|\boldsymbol{X}|$ is the size of batch dimension.

Classes We are concern with classification task. Let K be the set of classes and $|K|$ its cardinality. For example in CIFAR-10 $K = \{\text{airplane}, \dots, \text{truck}\} = \{k_1, \dots, k_{10}\}$ and $|K| = 10$. Let $g_{(\boldsymbol{x})} \in K$ be the class associated to the input $\boldsymbol{x}$ (ground–truth class). When the $\boldsymbol{x}$ is clear from the context the index for g will be dropped.

Labels Labels are the numerical encoding of classes. Each $\boldsymbol{x}$ belong to only one class g. In this setting one popular encoding scheme is *one–hot encoding.*

$$\text{one–hot} : K \longrightarrow \{0,1\}^{|K|} \,:\, g_{(\boldsymbol{x})} \longmapsto \boldsymbol{y}_{(\boldsymbol{x})} \quad \text{with components} \quad \boldsymbol{y}_j := \delta_{g,j}$$

Applying an encoding scheme to the batch of labels produce $\boldsymbol{Y}_{(\boldsymbol{X})}$.

Outputs What model return. Let $\boldsymbol{z}_{(\boldsymbol{x})} \in \mathbb{R}^{|K|}$ be the outputs of the model given the input $\boldsymbol{x}$. To output prediction as probability vector, usually denoted by $\hat{\boldsymbol{y}}$, $\boldsymbol{z}$ must be normalized.

$$P_K : \mathbb{R}^{|K|} \to [0,1]^{|K|} \,:\, \boldsymbol{z}_{(\boldsymbol{x})} \mapsto \hat{\boldsymbol{y}}_{(\boldsymbol{x})} = P_K(\boldsymbol{z}_{(\boldsymbol{x})}) \quad \text{with} \quad P_K(\boldsymbol{z})_j := \frac{e^{\boldsymbol{z}_j}}{\sum_{k \in K} e^{\boldsymbol{z}_k}}$$

$P_K(.)$ is an alternative notation for the usual *softmax* function. Be more explicit about the underlying classes K give us more flexibility for constructing custom loss function.

Losses In the following losses are define as a pointwise operation that take $\boldsymbol{x}$ and its corresponding label $\boldsymbol{y}$ as input and return a scalar. Pointwise function apply to a batch can be combined with a reduction operator $\bigoplus$ (e.g. sum or mean).

$$\ell : (\boldsymbol{x}, \boldsymbol{y}) \longmapsto \ell(\boldsymbol{x}, \boldsymbol{y}) \qquad \mathcal{L} : (\boldsymbol{X}, \boldsymbol{Y}) \longmapsto \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}) := \bigoplus_{(\boldsymbol{x},\boldsymbol{y}) \in (\boldsymbol{X},\boldsymbol{Y})} \ell(\boldsymbol{x}, \boldsymbol{y})$$

Cross Entropy Loss (XE)

Standard Cross–Entropy only uses the output vector component corresponding to the ground–truth label (i.e. the g component of the output vector). Training process force $\hat{\boldsymbol{y}}_g$ to increase and, due to normalization, other components shrinks.

$$\ell_{\text{XE}} := -\boldsymbol{y}^\intercal \log \hat{\boldsymbol{y}} = -\log \hat{\boldsymbol{y}}_g \quad \longrightarrow \quad \mathcal{L}_{\text{XE}} := \frac{1}{|\boldsymbol{X}|} \sum \ell_{\text{XE}} \tag{1}$$

Complement Entropy Loss (CE)

[Che+19a] introduce *Complement Entropy Loss*. CE take as input the output vector $\boldsymbol{z}$, remove its g component and renormalise it obtaining $P_{K\setminus\{g\}}(\boldsymbol{z})$. CE is minus the Shannon Entropy $H(.)$ of this new vector.

$$\ell_{\mathrm{CE}} := -H(P_{K\setminus\{g\}}(\boldsymbol{z})) \qquad \longrightarrow \qquad \mathcal{L}_{\mathrm{CE}} := \frac{1}{|\boldsymbol{X}|}\sum \ell_{\mathrm{CE}} \tag{2}$$

Maximise the entropy of a distribution force it towards a flatter one (in these setting the distribution with maximum entropy is the uniform distribution). CE is paired with XE using a custom training loop:

Algorithm 1 Custom Training Loop for CE

1: **for** *step* in *steps* **do**
2: Update parameters of the model using $\mathcal{L}_{\mathrm{XE}}$
3: Update parameters of the model using $\mathcal{L}_{\mathrm{CE}}$
4: **end for**

They also proposed an empirical modification to balance the contribution that came from CE [1]

$$\ell'_{\mathrm{CE}} := \frac{\ell_{\mathrm{CE}}}{|K|-1} \qquad \longrightarrow \qquad \mathcal{L}'_{\mathrm{CE}} := \frac{1}{|\boldsymbol{X}|}\sum \ell'_{\mathrm{CE}}$$

Guided Complement Entropy Loss (GCE)

[Che+19b] improved upon CE by adding an additional term in the ℓ_{CE} that leverage the variation in model confidence during the training phase, i.e.

$$\ell_{\mathrm{GCE}} := [\hat{\boldsymbol{y}}_g]^{\alpha}\,\ell_{\mathrm{CE}} \qquad \longrightarrow \qquad \mathcal{L}_{\mathrm{GCE}} := \frac{1}{|\boldsymbol{X}|}\sum \ell_{\mathrm{GCE}} \tag{3}$$

The new factor $[\hat{\boldsymbol{y}}_g]^{\alpha}$ is called the *guiding factor*. At the beginning of training it is small (the model outputs low probability for the g-component of $\hat{\boldsymbol{y}}$) and then it increase with training (the model get better and the g-component of $\hat{\boldsymbol{y}}$ will be higher). α is a fixed hyperparameter ($\alpha = 0.2$ works reasonably well).

GCE is the only loss used in a standard training loop, i.e.

Algorithm 2 Standard Training Loop for GCE

1: **for** *step* in *steps* **do**
2: Update parameters of the model using $\mathcal{L}_{\mathrm{GCE}}$
3: **end for**

Similar to [Che+19a], they modify $\mathcal{L}_{\mathrm{GCE}}$ to account for the number of classes

$$\ell'_{\mathrm{GCE}} := \frac{\ell_{\mathrm{GCE}}}{\log(|K|-1)} \qquad \longrightarrow \qquad \mathcal{L}'_{\mathrm{GCE}} := \frac{1}{|\boldsymbol{X}|}\sum \ell'_{\mathrm{GCE}}$$

[1] In the paper they propose a rescaling factor of $|K|-1$ but in the source code they use $|K|$. The similar inconsistencies appear in others normalized loss functions.

Hierarchical Complement Entropy (HCE)

[Che+19c] try to exploit hierarchical labels in CIFAR-100. Let G be a set that contains the siblings classes that belong to the same parental class of the ground–truth class, that is $g \in G$ and $G \subseteq K$. The *Hierarchical Complement Entropy* is

$$\ell_{\text{HCE}} := -H(P_{G\setminus\{g\}}(\boldsymbol{z})) - H(P_{K\setminus\{G\}}(\boldsymbol{z})) \quad \longrightarrow \quad \mathcal{L}_{\text{HCE}} := \frac{1}{|\boldsymbol{X}|}\sum \ell_{\text{HCE}} \tag{4}$$

[Che+19c] empolyed HCE in two training loops: the classic single step training loop and in the double steps training loop for direct comparison with [Che+19a]

Algorithm 3 Standard Training Loop for XE + HCE

1: **for** *step* in *steps* **do**
2: Update parameters of the model using $\mathcal{L}_{\text{XE}} + \mathcal{L}_{\text{HCE}}$
3: **end for**

Algorithm 4 Custom Training Loop for HCE

1: **for** *step* in *steps* **do**
2: Update parameters of the model using $\mathcal{L}_{\text{XE}}$
3: Update parameters of the model using $\mathcal{L}_{\text{HCE}}$
4: **end for**

Normalized version of ℓ_{HCE} can be obtained dividing each $H(.)$ in (4) by the number of classes involved

$$\ell'_{\text{HCE}} := -\frac{H(P_{G\setminus\{g\}}(\boldsymbol{z}))}{|G|-1} - \frac{H(P_{K\setminus\{G\}}(\boldsymbol{z}))}{|K|-|G|} \qquad \longrightarrow \qquad \mathcal{L}'_{\text{HCE}} := \frac{1}{|\boldsymbol{X}|}\sum \ell'_{\text{HCE}}$$

Hierarchical Guided Complement Entropy Loss (HGCE)

Following the same reasoning in [Che+19b], we take HCE and add the guiding factor $[\hat{\boldsymbol{y}}_g]^\alpha$.

$$\ell_{\text{HGCE}} := [\hat{\boldsymbol{y}}_g]^\alpha \, \ell_{\text{HCE}} \quad \longrightarrow \quad \mathcal{L}_{\text{HGCE}} := \frac{1}{|\boldsymbol{X}|}\sum \ell_{\text{HGCE}} \tag{5}$$

Standard (single step) training loop is employed using HGCE as the only criterion

Algorithm 5 Standard Training Loop for HGCE

1: **for** *step* in *steps* **do**
2: Update parameters of the model using $\mathcal{L}_{\text{HGCE}}$
3: **end for**

Combining ideas in GCE and HCE the normalized version will be

$$\mathcal{L}'_{\text{HGCE}} := \frac{1}{|\boldsymbol{X}|}\sum [\hat{\boldsymbol{y}}_g]^\alpha \left(-\frac{H(P_{G\setminus\{g\}}(\boldsymbol{z}))}{\log(|G|-1)} - \frac{H(P_{K\setminus\{G\}}(\boldsymbol{z}))}{\log(|K|-|G|)}\right)$$

Results

Missing results are not available at the moment due to a bug in the training script.

Ref	Loss	Baseline	FGSM	I-FGSM	PGD
Che+19a	XE	0.9176/1.0000	0.1044/1.0000	0.0795/1.0000	0.0173/1.0000
Che+19a	XE,CE	-	-	-	-

Table 1: ResNet110 on CIFAR-10, Accuracy/LCA.

Ref	Loss	Baseline	FGSM	I-FGSM	PGD
Che+19a	XE	0.7011/1.6577	0.0112/1.9649	0.0260/1.9175	0.0077/1.9438
Che+19a	XE,CE	-	-	-	-

Table 2: ResNet110 on CIFAR-100, Accuracy/LCA.

Ref	Loss	Baseline	FGSM	I-FGSM	PGD
Che+19b	XE	0.9269/1.0000	0.0991/1.0000	0.0426/1.0000	0.0038/1.0000
Che+19b	GCE	0.9297/1.0000	0.1195/1.0000	0.1306/1.0000	0.0959/1.0000

Table 3: ResNet56 on CIFAR-10, Accuracy/LCA.

Ref	Loss	Baseline	FGSM	I-FGSM	PGD
Che+19b	XE	0.6677/1.6759	0.0128/1.9599	0.0259/1.9255	0.0097/1.9555
Che+19c	XE,CE	0.6866/1.6643	0.0102/1.9580	0.0258/1.9431	0.0102/1.9592
Che+19b	GCE	0.6749/1.6626	0.0153/1.9503	0.0669/1.9130	0.0315/1.9394
Che+19c	XE,HCE	0.6971/1.6418	0.0098/1.9655	0.0278/1.9355	0.0126/1.9600
Che+19c	HGCE	0.6777/1.6131	0.0165/1.9514	0.0741/1.9155	0.0313/1.9270

Table 4: ResNet56 on CIFAR-100, Accuracy/LCA.

References

[Che+19a] Hao-Yun Chen et al. “Complement Objective Training”. In: (Mar. 2019). eprint: 1903.01182v2. URL: http://arxiv.org/abs/1903.01182v2.

[Che+19b] Hao-Yun Chen et al. “Improving Adversarial Robustness via Guided Complement Entropy”. In: (Mar. 2019). eprint: 1903.09799v3. URL: http://arxiv.org/abs/1903.09799v3.

[Che+19c] Hao-Yun Chen et al. “Learning with Hierarchical Complement Objective”. In: (Nov. 2019). eprint: 1911.07257v1. URL: http://arxiv.org/abs/1911.07257v1.