Learning Deep ResNet Blocks Sequentially using Boosting Theory

Yamen Habib

Mindset

September 2021

ResNet

A residual neural network (ResNet) is composed of stacked entities referred to as residual blocks. Each residual block consists of a neural network module and an identity loop (shortcut). Commonly used modules include MLP and CNN.

A Residual Block of ResNet

ResNet consists of residual blocks. Each residual block contains a module and an identity loop. Let each module map its input x to $f_t(x)$ where t denotes the level of the modules. Each module f_t is a nonlinear unit with n channels, i.e., $f_t(x) \in R^n$.
In constitutional neural network residual network (CNN-ResNet), $f_t(x)$ represents the t-th constitutional module. Then the t-th residual block outputs $g_{t+1}(x)$

$$g_{t+1}(x) = f_t(g_t(x)) + g_t(x) \tag{1}$$

where x is the input fed to the ResNet.
Due to the recursive relation specified in Equation (1), the output of the T-th residual block is equal to the summation over lower module outputs,

$$g_{T+1}(x) = \sum_{t=0}^{T} f_t(g_t(x))$$

Output of ResNet

where $g0(x) = 0$ and $f_0(g_0(x)) = x$. For binary classification tasks, the final output of a ResNet given input x is rendered after a linear classifier $w \in R^n$ on representation $g_{T+1}(x)$ (In the multiclass setting, let C be the number of classes; the linear classifier $W \in R^{nC}$ is a matrix instead of a vector.):

$$\hat{y} = \tilde{\sigma}(F(x)) = \tilde{\sigma}(W^T g_{T+1}(x)) = \tilde{\sigma}(W^T \sum_{t=0}^{T} f_t(g_t(x)) \tag{2}$$

The parameters of a depth-T ResNet are $w, f_t(), \forall t \in T$. A ResNet training involves training the classifier w and the weights of modules $f_t() \forall t \in [T]$ when training examples $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$ are available.

The key difference between boosting and ResNet is that the first one is an ensemble of estimated hypotheses whereas ResNet is an ensemble of estimated feature: $\sum_{t=0}^{T} f_t(g_t(x))$ To solve this we will use an auxiliary linear classifier w_t on top of each residual block to construct a hypothesis module.

$$o_t(x) = w_t^T g_t(x) \in R(Binary) \tag{3}$$

We emphasize that given $g_t(x)$, we only need to train f_t and w_{t+1} to train $o_{t+1}(x)$ as $g_{t+1}(x) = f_t(g_t(x)) + g_t(x)$. As a result we now have: $o_t(x) = \sum_{t'}^{t-1} w_t^T f_{t'}(g_{t'}(x)$, and we can notice that the auxiliary classifier is common for all layers underneath.

Weak Module classifier

A weak module classifier is defined as:

$$h_t(x) = \alpha_{t+1} o_{t+1}(x) - \alpha_t o_t(x) \tag{4}$$

where $o_t(x) = w_t^T g_t(x)$ is a hypothesis module, and α_t is a scalar. We call it a "telescoping sum boosting" framework if the weak learners are restricted to the form of the weak module classifier

Let the input $g_t(x)$ of the t-th module be the output of the previous module, i.e., $g_{t+1}(x) = f_t(g_t(x)) + g_t(x)$. Then the summation of T weak module classifiers divided by α_{t+1} is identical to the output, $F(x)$, of the depth-T ResNet.

$$F(x) = w_t^T g_{T+1}(x) = \sum_{t=0}^{T} h_t(x) \tag{5}$$

Weak Learning Condition

Defining $\tilde{\gamma}_t = \mathbb{E}_{i \sim D_{t-1}}[y_i o_t(x_i)] > 0$ where D_{t-1} is the weight of the examples.

$\tilde{\gamma}_t$ characterizes the performance of the hypothesis module $o_t(x_i)$. A natural requirement would be that $o_{t+1}(x_i)$ improves slightly upon $o_t(x_i)$, so we need: $\tilde{\gamma}_{t+1} - \tilde{\gamma}_t \geq \tilde{\gamma} > 0$

(γ-Weak Learning Condition)

A weak module classifier $h_t(x) = \alpha_{t+1} o_{t+1}(x) - \alpha_t o_t(x)$ satisfies the γ-weak learning condition if $\dfrac{\tilde{\gamma}_{t+1}^2 - \tilde{\gamma}_t^2}{1 - \tilde{\gamma}_t^2} \geq \gamma^2 > 0$.

Interpretation of weak learning condition For each weak module classifier ht (x), $\gamma_t = \sqrt{\dfrac{\tilde{\gamma}_{t+1}^2 - \tilde{\gamma}_t^2}{1 - \tilde{\gamma}_t^2}}$ characterizes the normalized improvement of the correlation between the true labels y and the hypothesis modules $o_{t+1}(x)$ over the correlation between the true labels y and the hypothesis modules $o_t(x)$.

Algorithm 1 BoostResNet: telescoping sum boosting for binary-class classification

Input: m labeled samples $[(x_i, y_i)]_m$ where $y_i \in \{-1, +1\}$ and a threshold γ

Output: $\{f_t(\cdot), \forall t\}$ and $\mathbf{w}_{T+1}$ ▷ Discard $\mathbf{w}_{t+1}, \forall t \neq T$

1: Initialize $t \leftarrow 0$, $\tilde{\gamma}_0 \leftarrow 0$, $\alpha_0 \leftarrow 0$, $o_0(x) \leftarrow 0$

2: Initialize sample weights at round 0: $D_0(i) \leftarrow 1/m, \forall i \in [m]$

3: **while** $\gamma_t > \gamma$ **do**

4: $f_t(\cdot), \alpha_{t+1}, \mathbf{w}_{t+1}, o_{t+1}(x) \leftarrow$ Algorithm 2$(g_t(x), D_t, o_t(x), \alpha_t)$

5: Compute $\gamma_t \leftarrow \sqrt{\frac{\tilde{\gamma}_{t+1}^2 - \tilde{\gamma}_t^2}{1-\tilde{\gamma}_t^2}}$ ▷ where $\tilde{\gamma}_{t+1} \leftarrow \mathbb{E}_{i\sim D_t}[y_i o_{t+1}(x_i)]$

6: Update $D_{t+1}(i) \leftarrow \frac{D_t(i)\exp(-y_i h_t(x_i))}{\sum_{i=1}^{m} D_t(i)\exp[-y_i h_t(x_i)]}$ ▷ where $h_t(x) = \alpha_{t+1} o_{t+1}(x) - \alpha_t o_t(x)$

7: $t \leftarrow t + 1$

8: **end while**

9: $T \leftarrow t - 1$

Algorithm 2 BoostResNet: oracle implementation for training a ResNet block

Input: $g_t(x)$,D_t,$o_t(x)$ and α_t

Output: $f_t(\cdot)$, α_{t+1}, $\mathbf{w}_{t+1}$ and $o_{t+1}(x)$

1: $(f_t, \alpha_{t+1}, \mathbf{w}_{t+1}) \leftarrow \arg\min_{(f,\alpha,\mathbf{v})} \sum_{i=1}^{m} D_t(i)\exp\left(-y_i \alpha \mathbf{v}^\top [f(g_t(x_i)) + g_t(x_i)] + y_i \alpha_t o_t(x_i)\right)$

2: $o_{t+1}(x) \leftarrow \mathbf{w}_{t+1}^\top [f_t(g_t(x)) + g_t(x)]$

Algorithm 3 BoostResNet: telescoping sum boosting for multi-class classification

Input: Given $(x_1, y_1), \ldots (x_m, y_m)$ where $y_i \in \mathcal{Y} = \{1, \ldots, C\}$ and a threshold γ

Output: $\{f_t(\cdot), \forall t\}$ and W_{T+1} $\triangleright$ Discard $\mathbf{w}_{t+1}, \forall t \neq T$

Initialize $t \leftarrow 0$, $\tilde{\gamma}_0 \leftarrow 1$, $\alpha_0 \leftarrow 0$, $o_0 \leftarrow \mathbf{0} \in \mathbb{R}^C$, $s_0(x_i, l) = 0, \forall i \in [m], l \in \mathcal{Y}$

Initialize cost function $\mathbf{C}_0(i, l) \leftarrow \begin{cases} 1 & \text{if } l \neq y_i \\ 1 - C & \text{if } l = y_i \end{cases}$

while $\gamma_t > \gamma$ **do**

  $f_t(\cdot), \alpha_{t+1}, W_{t+1}, o_{t+1}(x) \leftarrow$ Algorithm 4$(g_t(x), \mathbf{C}_t, o_t(x), \alpha_t)$

  Compute $\gamma_t \leftarrow \sqrt{\frac{\tilde{\gamma}_{t+1}^2 - \tilde{\gamma}_t^2}{1 - \tilde{\gamma}_t^2}}$ $\triangleright$ where $\tilde{\gamma}_{t+1} \leftarrow \frac{-\sum_{i=1}^{m} \mathbf{C}_t(i,:) \cdot o_{t+1}(x_i)}{\sum_{i=1}^{m} \sum_{l \neq y_i} \mathbf{C}_t(i,l)}$

  Update $s_{t+1}(x_i, l) \leftarrow s_t(x_i, l) + h_t(x_i, l)$ $\triangleright$ where $h_t(x_i, l) = \alpha_{t+1} o_{t+1}(x_i, l) - \alpha_t o_t(x_i, l)$

  Update cost function $\mathbf{C}_{t+1}(i, l) \leftarrow \begin{cases} e^{s_{t+1}(x_i, l) - s_{t+1}(x_i, y_i)} & \text{if } l \neq y_i \\ -\sum_{l' \neq y_i} e^{s_{t+1}(x_i, l') - s_{t+1}(x_i, y_i)} & \text{if } l = y_i \end{cases}$

  $t \leftarrow t + 1$

end while

$T \leftarrow t - 1$

Algorithm 4 BoostResNet: oracle implementation for training a ResNet module (multi-class)

Input: $g_t(x)$,s_t,$o_t(x)$ and α_t

Output: $f_t(\cdot)$, α_{t+1}, W_{t+1} and $o_{t+1}(x)$

$(f_t, \alpha_{t+1}, W_{t+1}) \leftarrow \arg \min_{(f,\alpha,V)} \sum_{i=1}^{m} \sum_{l \neq y_i} e^{\alpha V^\top [f(g_t(x_i), l) - f(g_t(x_i), y_i) + g_t(x_i, l) - g_t(x_i, y_i)]}$

$o_{t+1}(x) \leftarrow W_{t+1}^\top [f_t(g_t(x)) + g_t(x)]$