On label confidence estimation

11.11

陈明猜

PRELIMINARIES

we consider a c-class classification problem using a DNN with a softmax output layer. Let $\mathcal{X} \subset \mathbb{R}^d$ be the feature space and $\mathcal{Y} = \{0,1\}^c$ be the ground-truth label space in a *one-hot* manner. In a typical classification problem, we are provided with a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ obtained from an unknown joint distribution $P_{\mathcal{D}}$ over $\mathcal{X} \times \mathcal{Y}$, where each (x_i, y_i) is *independent and identically distributed*. The goal of the task is to learn the mapping function $f(\,\cdot\,;\Theta) : \mathcal{X} \to [0,1]^c$ of the DNN parameterized by Θ such that the parameter Θ minimizes the empirical risk $\mathcal{R}_{\mathcal{D}}(f)$,

$$\mathcal{R}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}[\ell\big(f(x;\Theta), y\big)] = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell\big(f(x;\Theta), y\big)$$

where ℓ is a certain loss function.

As data labels are corrupted in various real-world scenarios, we aim to train the DNN from noisy labels. Specifically, we are provided with a noisy training dataset $\tilde{\mathcal{D}} = \{(x_i, \tilde{y}_i)\}_{i=1}^N$ obtained from a noisy joint distribution $P_{\tilde{\mathcal{D}}}$ over $\mathcal{X} \times \tilde{\mathcal{Y}}$, where $\tilde{y}$ is a *noisy* label which may not be true. Hence, following the standard training procedure, a mini-batch $\mathcal{B}_t = \{(x_i, \tilde{y}_i)\}_{i=1}^b$ comprising b examples is obtained randomly from the noisy training dataset $\tilde{\mathcal{D}}$ at time t. Subsequently, the DNN parameter Θ_t at time t is updated along the descent direction of the empirical risk on mini-batch $\mathcal{B}_t$,

$$\Theta_{t+1} = \Theta_t - \eta\nabla\Big(\frac{1}{|\mathcal{B}_t|} \sum_{(x,\tilde{y})\in\mathcal{B}_t} \ell\big(f(x;\Theta_t), \tilde{y}\big)\Big)$$

Methods

- ***Loss Reweighting***: Loss reweighting aims to assign smaller weights to the examples with false labels and greater weights to those with true labels.

$$\Theta_{t+1} = \Theta_t - \eta \nabla \Big(\frac{1}{|\mathcal{B}_t|} \sum_{(x,\tilde{y}) \in \mathcal{B}_t} \overbrace{w(x,\tilde{y}) \ell\big(f(x;\Theta_t), \tilde{y}\big)}^{\text{Reweighted Loss}} \Big)$$

- ***Label Refurbishment***: Refurbishing a noisy label $\tilde{y}$ effectively prevents overfitting to false labels. Let $\hat{y}$ be the current prediction.

$$y^{refur} = \alpha\hat{y} + (1-\alpha)\tilde{y}$$

Where is the weight/label confidence?

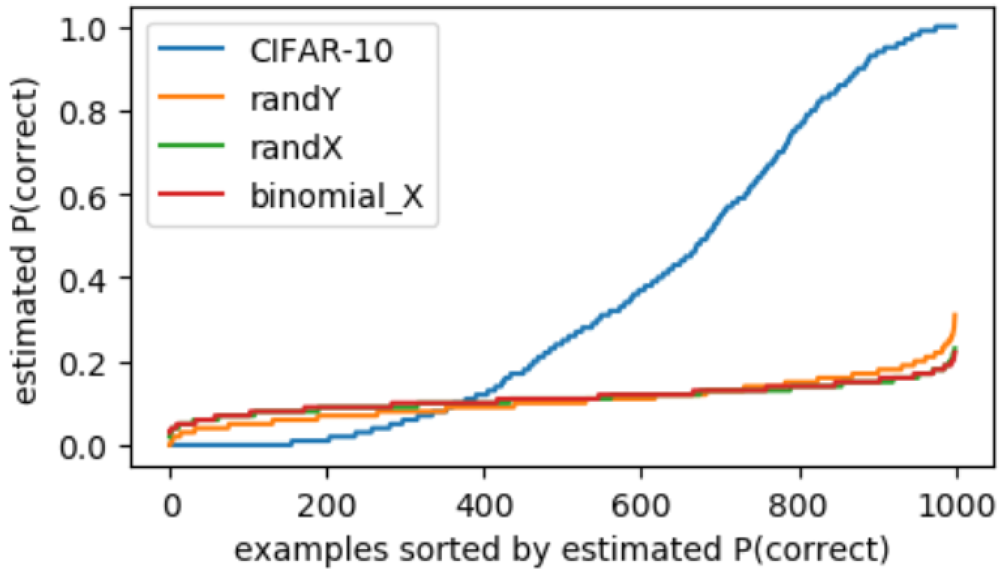
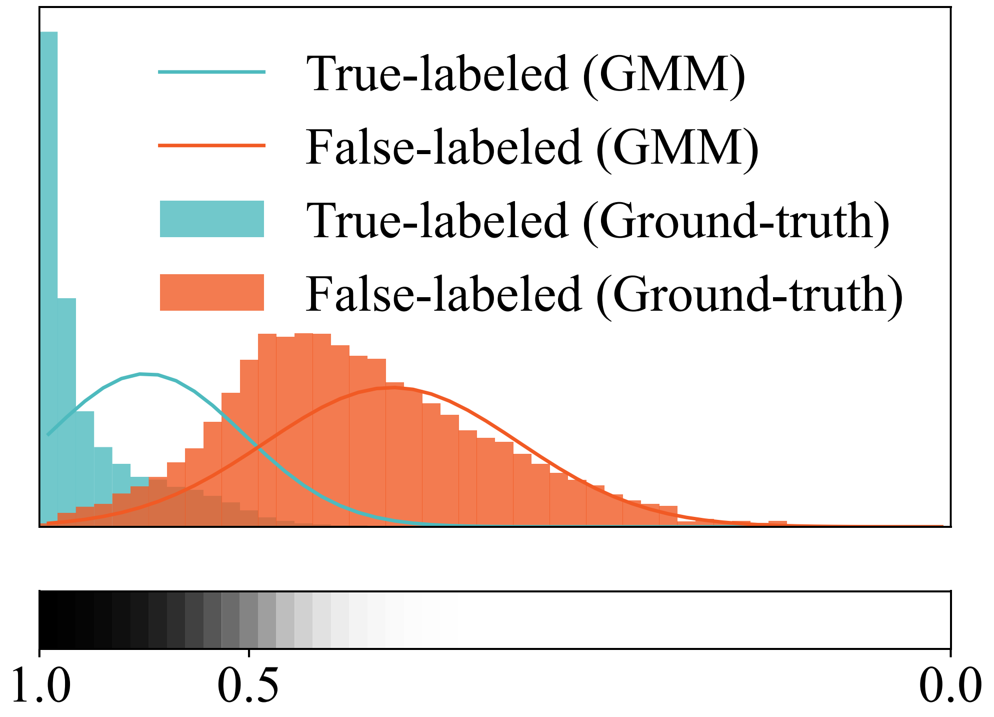


Figure 1. Average (over 100 experiments) misclassification rate for each of 1000 examples after one epoch of training. This measure of an example's difficulty is much more variable in real data. We conjecture this is because the easier examples are explained by some simple patterns, which are reliably learned within the first epoch of training.

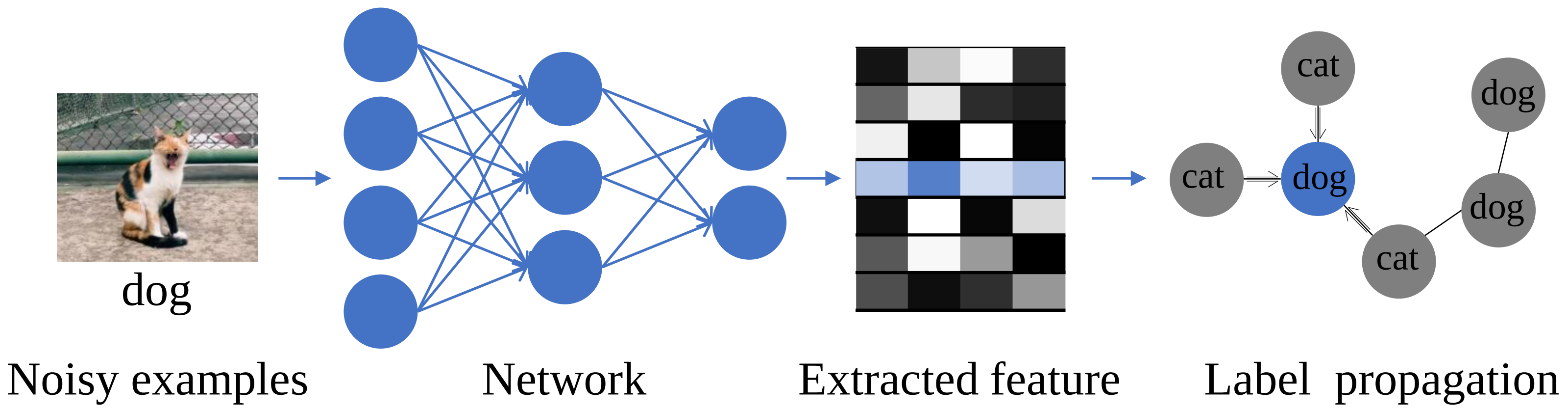
One reason for this behavior could be that, within a batch, gradient updates from randomly sampled noisy labels cancel out, while gradients from correct examples that are marginally more frequent sum together and contribute to learning.

A Closer Look at Memorization in Deep Networks

Memorization effect/ small-loss trick



Loss distribution of training examples on noisy CIFAR-10.



In the next, an undirected k-NN graph is constructed using penultimate layer features $V = \{g^*(x_1), \ldots, g^*(x_n)\}$, where the weighted adjacency matrix W:

$$W_{ij} = \begin{cases} \langle v_i, v_j \rangle, & \text{if } z_i \in \mathcal{N}_k(z_j) \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

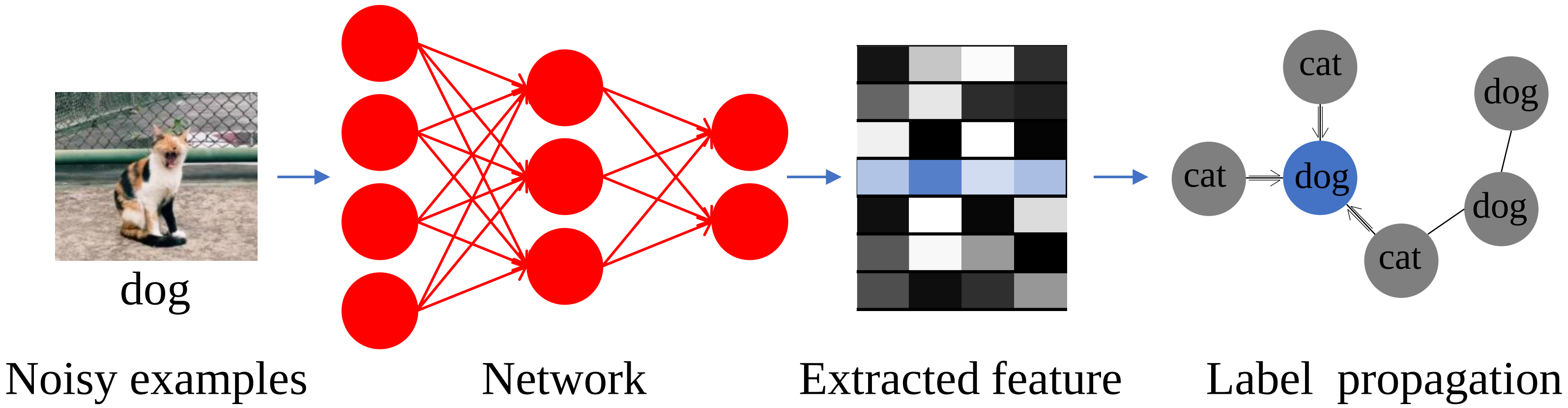
where $\langle \cdot, \cdot \rangle$ is the inner product between the representations of examples. $\mathcal{N}_k$ denotes the k nearest neighbors. The diagonal degree matrix D is used to normalize W: $d_{ii} = \sum_{j=1}^{N} W_{ij}$.

$$\begin{aligned} \mathcal{W} =& D^{-1/2} W D^{-1/2} \\ D =& \mathrm{diag}(W \mathbb{1}_N) \end{aligned} \tag{3}$$

Then LC propagates the information in the observed labels across the graph by minimizing the graphical Laplacian of the prediction matrix F:

$$\mathcal{Q}(F) = \frac{1}{2} \sum_{i,j=1}^{N} \mathcal{W}_{ij} \left\| \frac{F_i}{\sqrt{d_{ii}}} - \frac{F_j}{\sqrt{d_{jj}}} \right\|^2 + \frac{\mu}{2} \sum_{i=1}^{N} \| F_i - \tilde{Y}_i \|^2 \tag{4}$$

where $F = [F_1, \cdots, F_N] \in \mathbb{R}^{N \times C}$ is the refined soft labels and $\tilde{Y} = [\tilde{Y}_1, \cdots, \tilde{Y}_N] \in \mathbb{R}^{N \times C}$ is the one-hot observed labels. The second term is a fidelity term which avoids the refined labels changes too much from the observed labels. μ is a coefficient. This is a common minimization problem. To side-step the calculation of matrix inverse, we instead use conjugate gradient approach to solve the linear system $(I - (1+\mu)^{-1} \mathcal{W}) F = \tilde{Y}$. The algorithm follows the common practice, so we don't elaborate the details here. Having the refined soft labels, LC directly uses $F_i[\tilde{y}_i]$ as label confidence for i-th example.


dog
Noisy examples
Network
Extracted feature
cat
cat
dog
dog
dog
cat
Label propagation

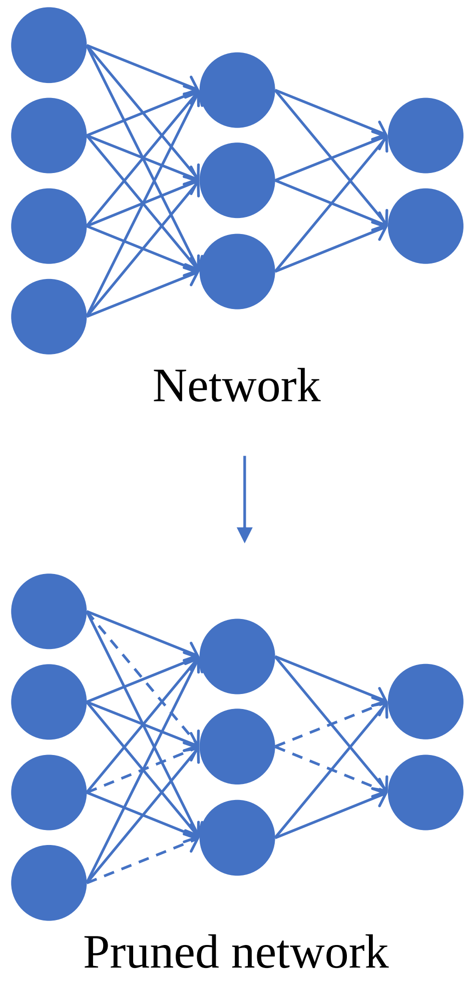
Extract robust features?

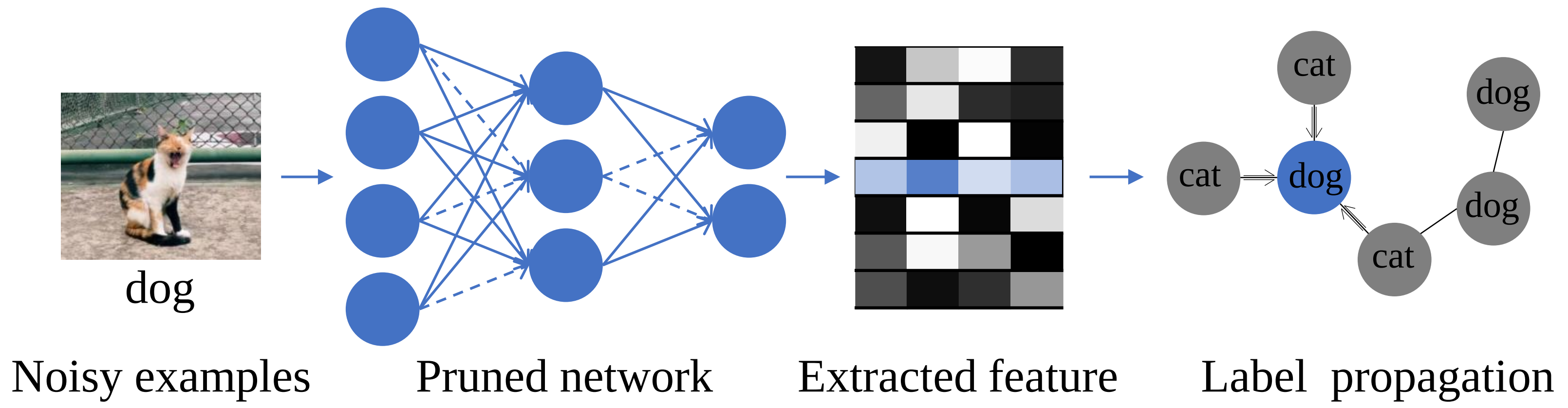
（把你idea给我交了）

Is pruning an answer?

SOME PRELIMINARIES OF NETWORK PRUNING

Following Occam's razor, network pruning that aims to reduce parameter counts could also be regarded as some kind of regularization on model capacity. By restricting a subset of model parameters to a value of zero, pruning imposes sparsity constraints on neural networks and penalizes its redundant expressive power. Furthermore, there are other conjectures on how pruning can benefit generalization, e.g., pruning creates sparsified versions of data representation, which introduce noise and encourage flatness into neural networks, as flatness of minima is usually correlated with good generalization.




dog
Noisy examples
Pruned network
Extracted feature
cat
cat
dog
dog
dog
cat
Label propagation

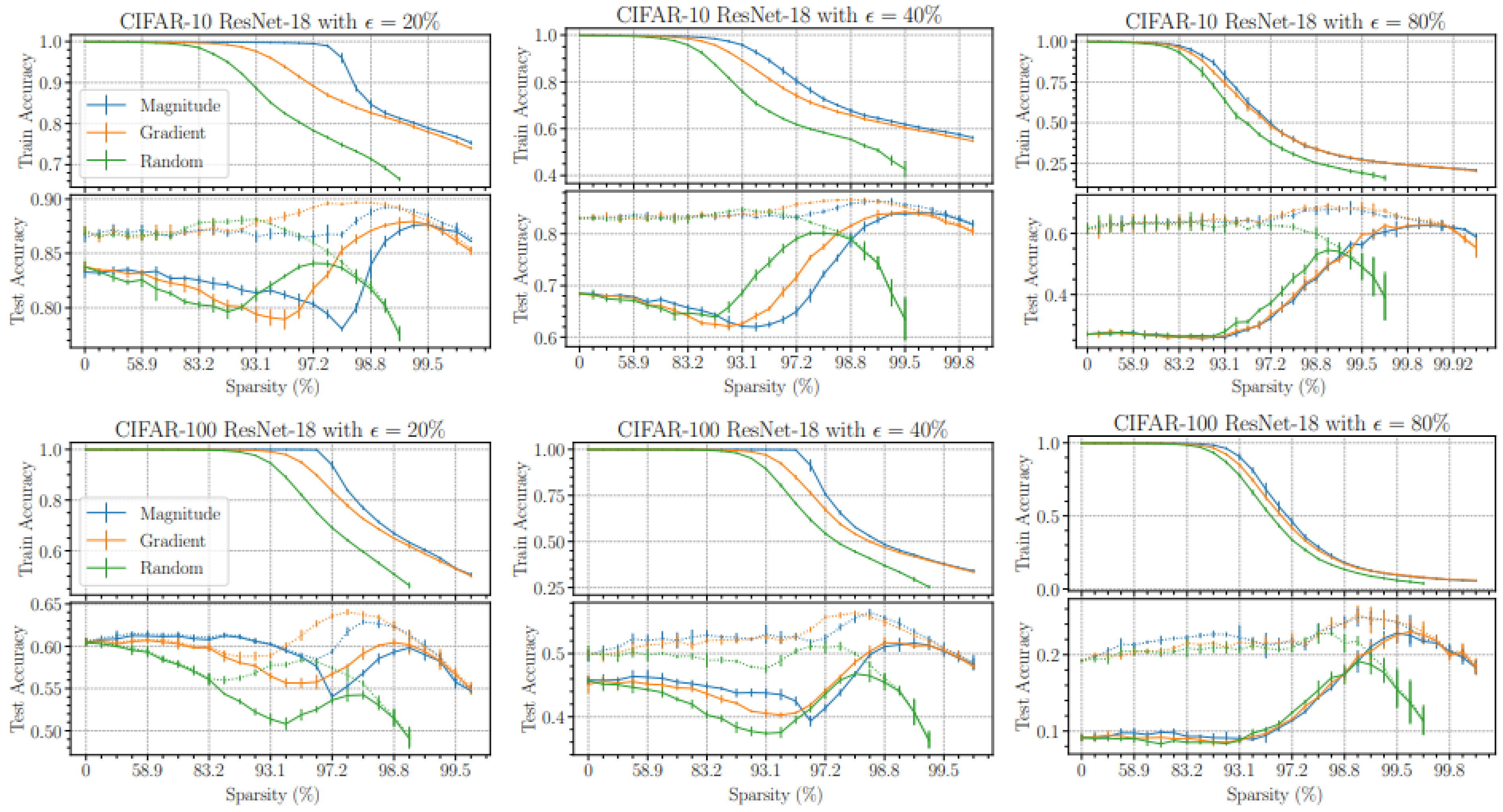


Figure 1: Double descent phenomenon in sparse regimes for ResNet-18 with three pruning strategies and varying permuted fraction ϵ. **Top**: CIFAR-10. **Bottom**: CIFAR-100. We plot **train accuracy** (solid lines in the upper sub-figures) and the **last test accuracy** of the final epoch (solid lines in the lower sub-figures), as well as the **best test accuracy** across all epochs (dotted lines).