Why Weight Averaging ?

CS Chen

考試如何考高分

- 考卷有 3 題, $\{x_1, x_2, x_3\}$
- 班上同學有 5 (N 表示) 人, $\{g_1(x), g_2(x), \ldots, g_5(x)\}$
  - 例如 $g_3(x_1)$ 表示第 3 位同學的第 1 題答案
- 正確答案: $f(x)$
- 假使你: $G(x)$ 可以看到 5 位同學的答案
- 請問你採取什麼策略可以高分?

投票、平均


Why?

用數學分析看看

- 首先定義某位同學 $g(x)$ 的錯誤率:

$$Err(g) = \mathrm{E}_x\left[\left(g(x) - f(x)\right)^2\right]$$

- 平均同學符號: $avg_n := \frac{1}{N}\sum_{n=1}^{N}$
- 直接講推導結果:

$$avg_n\left(Err(g_n)\right) = avg_n\left(\mathrm{E}_x\left[\left(g_n(x) - G(x)\right)^2\right]\right) + Err(G)$$
$$\geq Err(G)$$

- L.H.S.:「每位同學錯誤率」的平均
- R.H.S.: 你(採取平均策略)的錯誤率
- 平均策略好棒棒!

- L.H.S. 表示 Left Hand Side
- R.H.S. 表示 Right Hand Side

- 推導參考: https://bobondemon.github.io/2017/03/13/Why-Aggregation-Work/

獨裁 V.S. 民主？

$$avg_n\left(Err(g_n)\right) \geq Err(G)$$

- 獨裁可以想像成選擇哪位同學當領袖
- 運氣好選到高分同學
- 運氣差就 ...
- 最糟糕的是還沒有糾錯機會!

- 民主 G 至少不會太差

用數學分析看看 (Conti.)

$$avg_n\left(Err(g_n)\right) = avg_n\left(\mathrm{E}_x\left[\left(g_n(x) - G(x)\right)^2\right]\right) + Err(G)$$
$$\geq Err(G)$$

- 「每位同學錯誤率」的平均 ≥ 採取平均策略的你的錯誤率
  - 平均策略好棒棒!
- 假設有兩個班級 $\{s_1(x), s_2(x), \ldots\}$ and $\{h_1(x), h_2(x), \ldots\}$, 他們的 L.H.S. 一樣

$$avg_n\left(Err(s_n)\right) = avg_n\left(Err(h_n)\right)$$

- 哪個班級的平均策略方法較好？$S(x)$ or $H(x)$？
  - R.H.S. 愈小的那個班級愈好 → 意見遇不同的那個班級愈好 (式子的藍色部分)
  - 意見不同好棒棒!

民主好亂喔, 我討厭吵吵鬧鬧

- 假設有兩個班級 $\{s_1(x), s_2(x), \dots\}$ and $\{h_1(x), h_2(x), \dots\}$, 他們的 L.H.S. 一樣

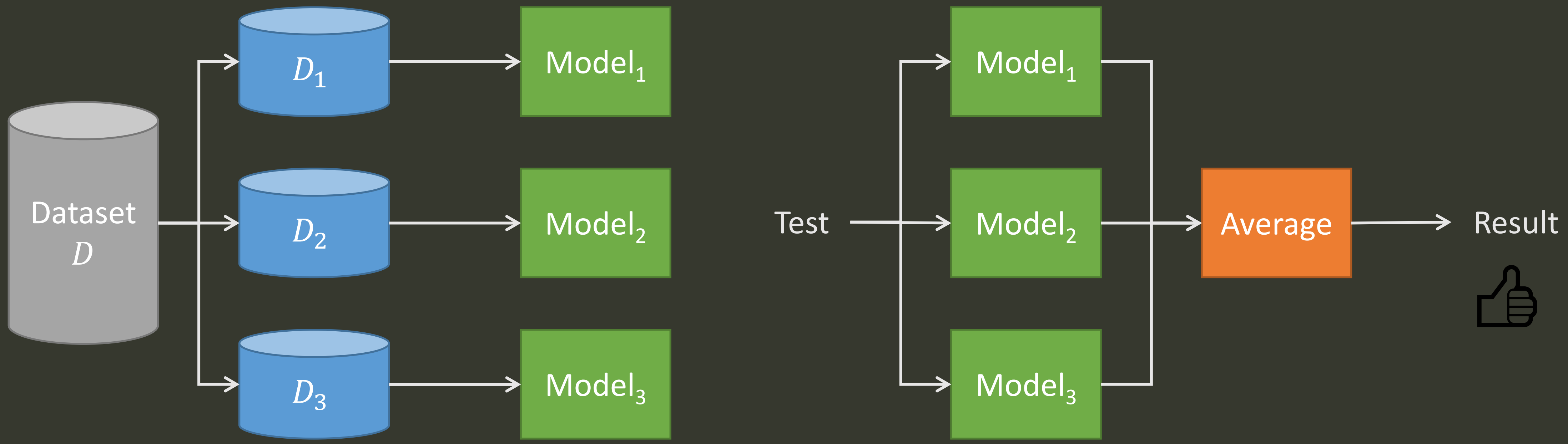
$$avg_n(Err(s_n)) = avg_n(Err(h_n))$$

- 哪個班級的平均策略方法較好 ? $S(x)$ or $H(x)$?
  - 意見不同好棒棒!

- 數學告訴我們, 意見不同才好!

回到 Machine Learning

- 如何利用上面平均方法取得更好的 prediction 結果?
- Bootstrap: 對 dataset 採樣出 datasets
  - E.g: Adaboost (DNN 之前人臉辨識最佳方法)



Bootstrap 缺點

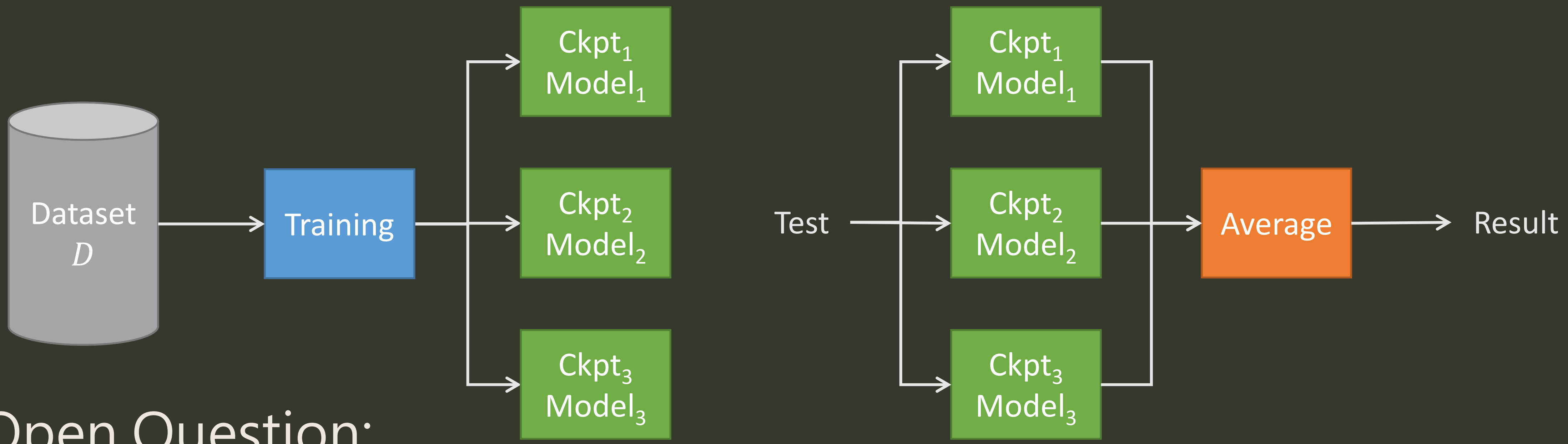
- N 表示訓練出來的 models
- Training 時間 * N
- Inference 時間 * N

- 打比賽可以 (ensemble), 實用上不行

- 那怎麼改進? 有辦法先降低 Training 的開銷嗎?

DNN Training 時的 checkpoints?

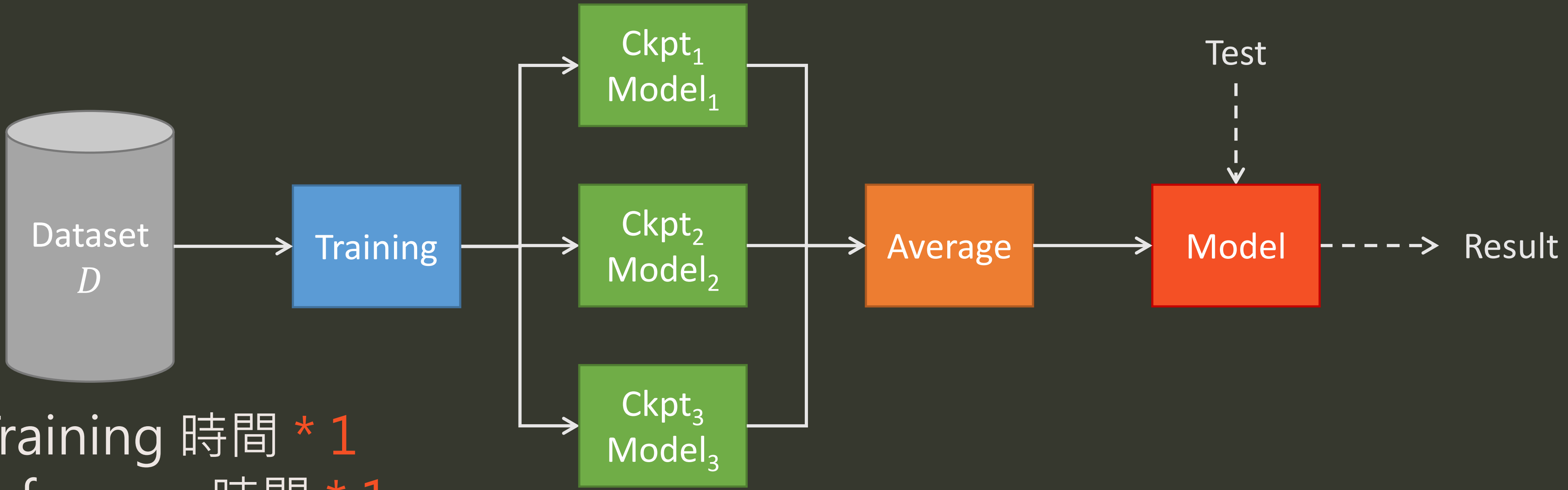
- 改成使用不同 checkpoints
- Training 時間 * 1
- Inference 時間 * N



- Open Question:
  - 如何挑選 checkpoints? 想想我們希望意見不同 ...

能不能 Inference 也降低開銷?

- 有! 使用 weight averaging



- Training 時間 * 1
- Inference 時間 * 1
- Question:
  - 我們之前只保證對結果 averaging, 現在對 weights averaging 也行嗎?
  - 兩者的關聯是?

Stochastic Weight Averaging

- n 個 NN weights $\{w_1, w_2, \dots, w_n\}$
- Weight averaging: $\bar{w} = \frac{1}{n}\sum_{i=1}^{n} w_i$
- 定義 weight 差為: $\Delta_j = w_j - \bar{w}$
- $f(w, x)$ 表示 NN 使用參數 w, 對 input x 的 prediction 結果
- 先當 x 是固定的來分析, 因此簡寫 $f(w)$ 即可
- Output 平均: $\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f(w_i)$

- 目的是觀察: $\bar{f} - f(\bar{w})$

Paper: [Averaging Weights Leads to Wider Optima and Better Generalization](), 2018

- 目的是觀察: $\bar{f} - f(\overline{w})$
- Taylor expansion 在 $\overline{w}$ 處展開:

$$f(w) = f(\overline{w}) + \langle \nabla f(\overline{w}), w - \overline{w} \rangle + \mathcal{O}(\|w - \overline{w}\|^2)$$

- 對 $f(w_j)$ 逼近

$$f(w_j) = f(\overline{w}) + \langle \nabla f(\overline{w}), \Delta_j \rangle + \mathcal{O}\left(\|\Delta_j\|^2\right)$$

- 所以

$$\bar{f} - f(\overline{w}) = \frac{1}{n}\sum_{j=1}^{n}\left(f(w_j) - f(\overline{w})\right)$$

$$= \frac{1}{n}\sum_{j=1}^{n}\left(\langle \nabla f(\overline{w}), \Delta_j \rangle + \mathcal{O}\left(\|\Delta_j\|^2\right)\right)$$

$$= \left\langle \nabla f(\overline{w}), \frac{1}{n}\sum_{j=1}^{n}\Delta_j \right\rangle + \mathcal{O}(\Delta^2), \text{ where } \Delta = \max_j \|\Delta_j\| \text{ (離平均最大的距離)}$$

$$= \langle \nabla f(\overline{w}), 0 \rangle + \mathcal{O}(\Delta^2)$$

$$= \mathcal{O}(\Delta^2)$$

- 為了說明 $\mathcal{O}(\Delta^2)$ 很小, 對比 $f(w_i) - f(w_j) = \langle \nabla f(\overline{w}), \Delta_i - \Delta_j \rangle + \mathcal{O}(\Delta^2) = \mathcal{O}(\Delta)$
  - 不同 models w_j 之間的 prediction 差異 $\mathcal{O}(\Delta)$

- Q: 我們之前只保證對結果 averaging, 現在對 weights averaging 也行嗎?
- A: 可以, 他們 outputs 之間差異不大: $\mathcal{O}(\Delta^2)$

Paper: Averaging Weights Leads to Wider Optima and Better Generalization, 2018

論文的一個實驗

- 不同 models w_j 之間的 prediction 差異 $\mathcal{O}(\Delta)$
  - 20 epochs of proposal models $\{w_j\}_{j=1}^{20}$
  - $\{w_j\}_{j=1}^{20}$ 的 class probabilities 平均 norm 差異為 0.126

- 結果 average 與 weights average 他們的 outputs 差異 $\mathcal{O}(\Delta^2)$
  - 0.079, 確實小了一個 order

Paper: [Averaging Weights Leads to Wider Optima and Better Generalization](), 2018

SWA 算法

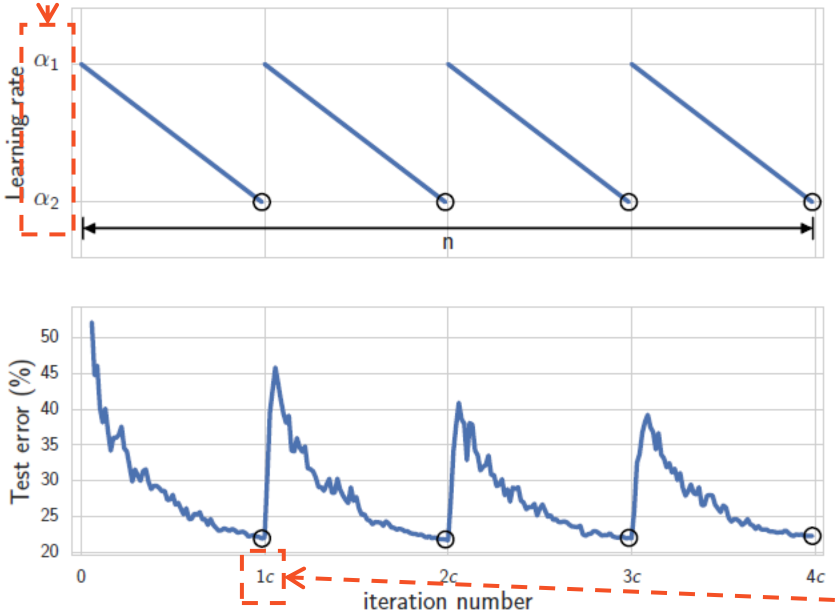


Figure 2: **Top**: cyclical learning rate as a function of iteration. **Bottom**: test error as a function of iteration for cyclical learning rate schedule with Preactivation-ResNet-164 on CIFAR-100. Circles indicate iterations corresponding to the minimum learning rates.

Algorithm 1 Stochastic Weight Averaging

Require:
weights $\hat{w}$, LR bounds α_1, α_2, cycle length c (for constant learning rate $c = 1$), number of iterations n

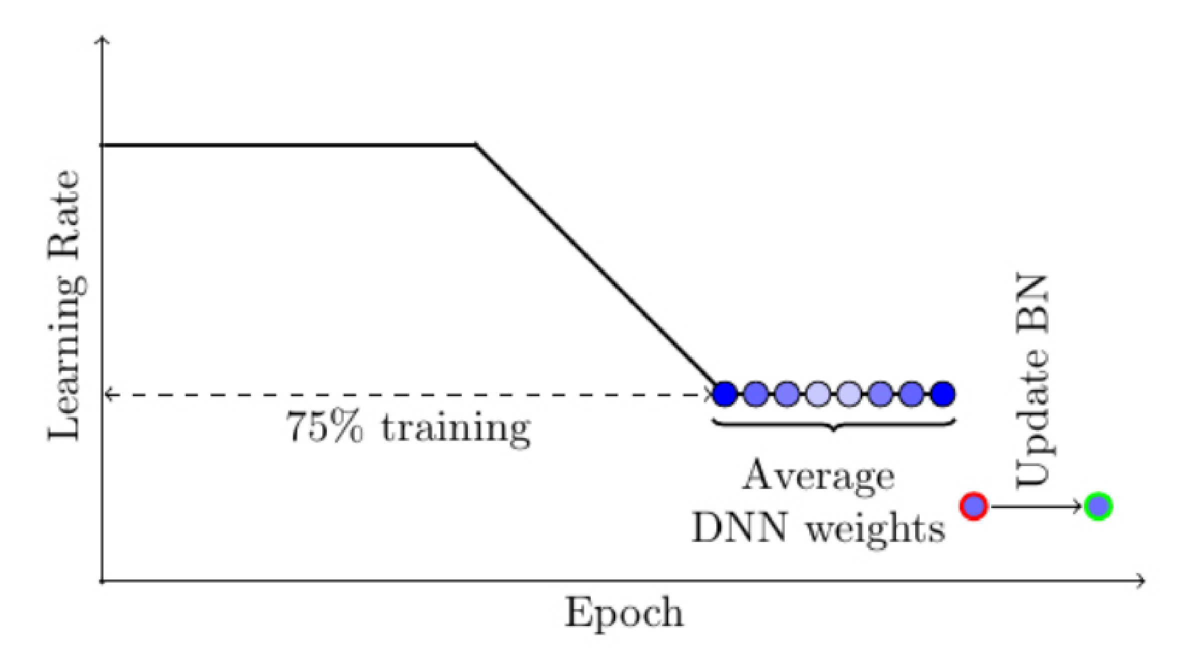
Ensure: w_{SWA}

$w \leftarrow \hat{w}$ {Initialize weights with $\hat{w}$}
$w_{\text{SWA}} \leftarrow w$
for $i \leftarrow 1, 2, \dots, n$ **do**
  $\alpha \leftarrow \alpha(i)$ {Calculate LR for the iteration}
  $w \leftarrow w - \alpha \nabla \mathcal{L}_i(w)$ {Stochastic gradient update}
  if $\mathrm{mod}(i, c) = 0$ **then**
    $n_{\text{models}} \leftarrow i/c$ {Number of models}
    $w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1}$ {Update average}
  end if
end for
{Compute BatchNorm statistics for w_{SWA} weights}

Paper: Averaging Weights Leads to Wider Optima and Better Generalization, 2018

August 18, 2020

PyTorch 1.6 now includes Stochastic Weight Averaging




```
from torch.optim.swa_utils import AveragedModel, SWALR
from torch.optim.lr_scheduler import CosineAnnealingLR

loader, optimizer, model, loss_fn = ...
swa_model = AveragedModel(model)
scheduler = CosineAnnealingLR(optimizer, T_max=100)
swa_start = 5
swa_scheduler = SWALR(optimizer, swa_lr=0.05)

for epoch in range(100):
      for input, target in loader:
          optimizer.zero_grad()
          loss_fn(model(input), target).backward()
          optimizer.step()
      if epoch > swa_start:
          swa_model.update_parameters(model)
          swa_scheduler.step()
      else:
          scheduler.step()

# Update bn statistics for the swa_model at the end
torch.optim.swa_utils.update_bn(loader, swa_model)
# Use swa_model to make predictions on test data
preds = swa_model(test_input)
```

論文其他觀點和實驗

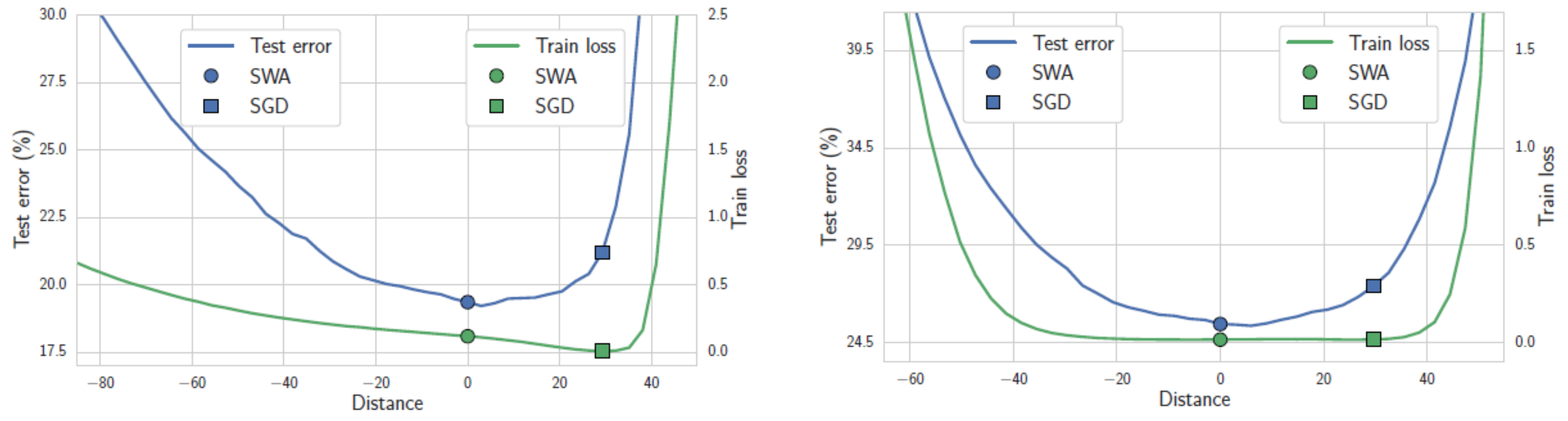


Figure 5: L_2-regularized cross-entropy train loss and test error as a function of a point on the line connecting SWA and SGD solutions on CIFAR-100. **Left**: Preactivation ResNet-164. **Right**: VGG-16.

- SWA 傾向找到“最廣”的低點
- SGD 容易找到低點區域的邊界
- 細節請參考論文

Paper: [Averaging Weights Leads to Wider Optima and Better Generalization](), 2018

Support in PyTorch and Lightning

- PyTorch 1.6 now includes Stochastic Weight Averaging
- Stochastic Weight Averaging in PyTorch Lightning
- 在 WeNet 上, 也有類似 SWA 機制
- 直接對 pre-trained model 用 SWA 跑上幾個 epochs 就可以!