Contact:
wulijun3@mail2.sysu.edu.cn
fetia@microsoft.com

Learning to Teach with Dynamic Loss Functions

[1]Lijun Wu, [2]Fei Tian, [2]Yingce Xia, [3]Yang Fan, [2]Tao Qin, [1]Jianhuang Lai and [2]Tie-Yan Liu

[1] Sun Yat-sen University [2] Microsoft Research [3] University of Science and Technology of China

Contact:
wulijun3@mail2.sysu.edu.cn
fetia@microsoft.com

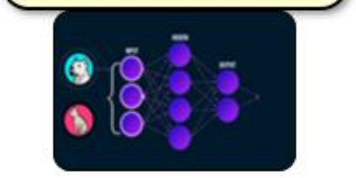
1. Machine Learning

- Typical Machine **Learning** Process

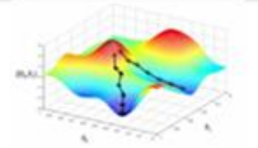
$$\omega^* = \arg\min_{\omega\in\Omega} \sum_{(x,y)\in D} L(f_\omega(x), y)$$

Training data D

Model Space Ω



Loss function L



- Fixed data order
- Fixed model space
- **Fixed loss function**

Why?

2. Loss Function Teaching

- Goal:
  - Automatically discover the **optimal loss functions** for student model training.
- **Student model**:
  - $f_\omega : x \to y$
  - $L(f_\omega, D_{train}) = \sum_{(x,y)\in D_{train}} l(f_\omega(x), y)$
  - $m(f_\omega(x), y)$: measure
- **Teacher model:**
  - u_θ
  - $\max_\theta m(f_\omega, D_{dev})$

Learning of f_ω

Students' Learning Process

Set Loss Functions L

Exams

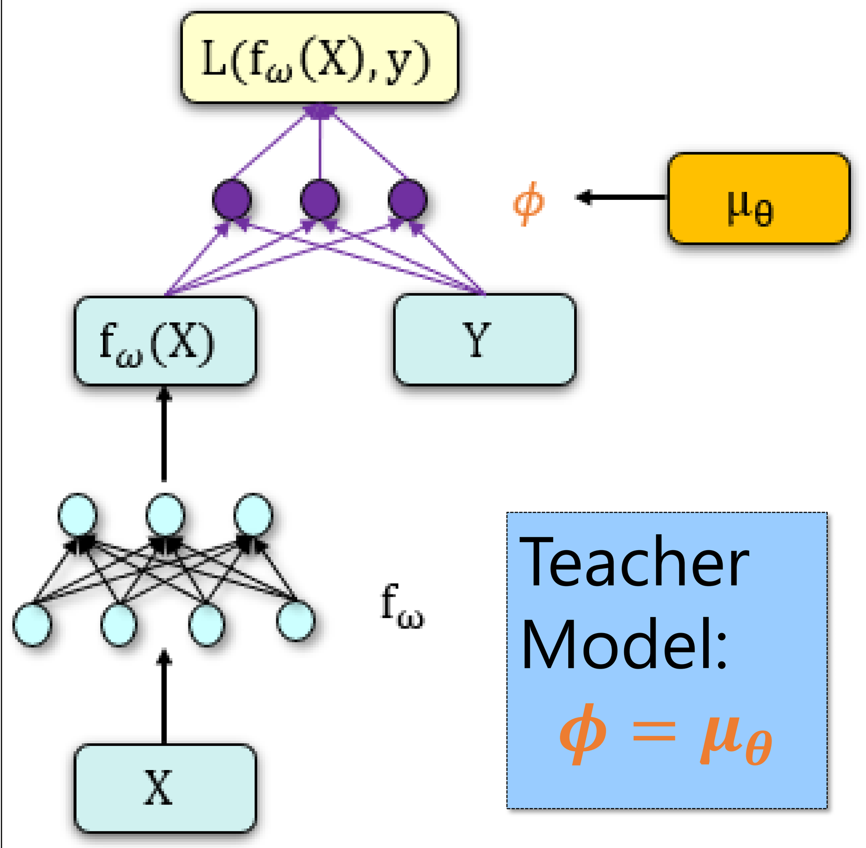
3. Teaching Requirement

- **Requirements** of Loss Function Teaching
  - Adaptive
  - Dynamic

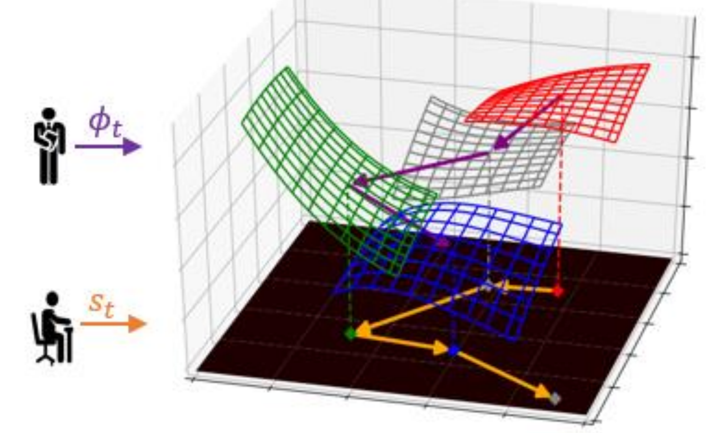
- Qualified human teachers are good at:
- Machine teachers should be :

Qualified human teachers are good at	Machine teachers should be
Adjusting proper exams with the growth of students	**Adaptive**: set different loss functions along different phases of student model training
Self improvement to achieve co-growth with students	**Dynamic**: optimize themselves to constantly enhance the teaching ability

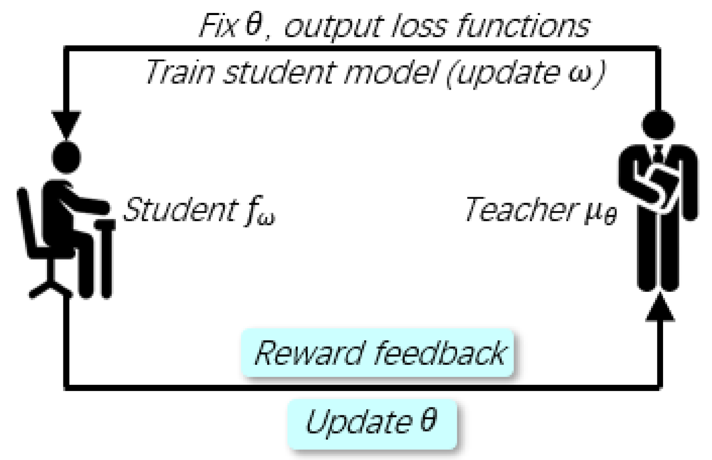
- $L_\phi(f_\omega(x), y)$, with ϕ as its coefficient
- $L_\phi = \sigma(-\log^T p(x)\, W\vec{y} + b)$
- $\phi = \{W, b\}$



- **Adaptive**
- $\phi_t = \mu_\theta(s_t)$



- **Dynamic**
- Reward: dev measure



4. Challenge & Algorithm

- Gradient-based Optimization for Teacher

The measure M is typically *discrete* and *non-smooth*	**Smooth** the measure: $\tilde{M}(f_\omega(x), y) = \sum_{y'} M(y', y)\, p_\omega(y'\vert x)$
The evaluation happens on *develop* dataset while teaching happens on *training* dataset	**Reverse Gradient Flow**: Chaining backwards the gradient from develop dataset to training dataset

- Algorithm/Structure

$$d\omega_T = \frac{\partial \tilde{\mathcal{M}}(f_{\omega_T}, D_{dev})}{\partial \omega_T} = \sum_{(x,y)\in D_{dev}} \frac{\partial \tilde{m}(f_{\omega_T}(x), y)}{\partial \omega_T}. \quad (3)$$

Then looping backwards from T and corresponding to Eqn. (1), at each step $t = \{T-1, \cdots, 1\}$ we have

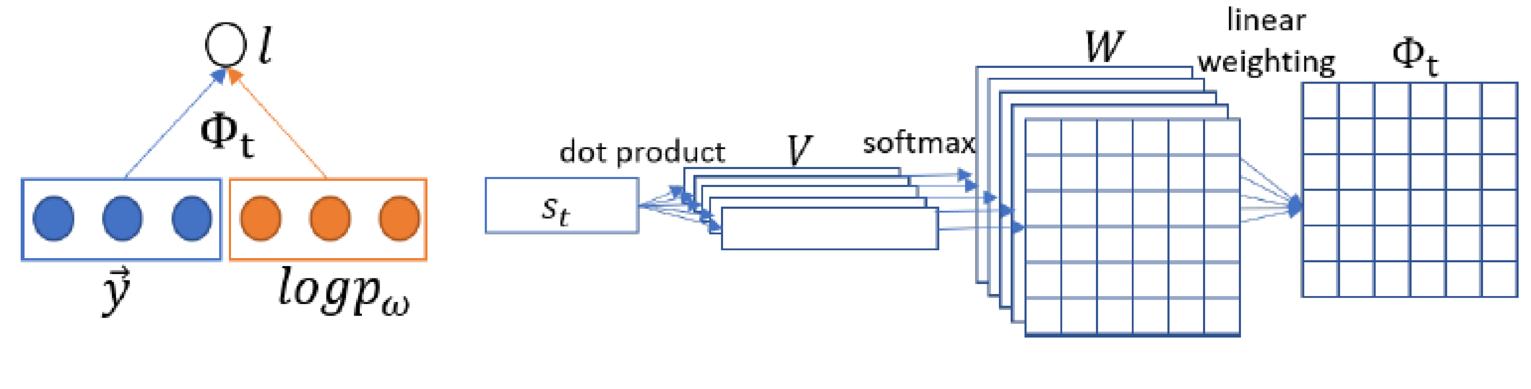
$$d\omega_t = \frac{\partial \tilde{\mathcal{M}}(f_{\omega_t}, D_{dev})}{\partial \omega_t} = d\omega_{t+1} - \eta_t \frac{\partial^2 L_{\mu_\theta(s_t)}(f_{\omega_t}, D^t_{train})}{\partial \omega_t^2} d\omega_{t+1}. \quad (4)$$

At the same time, the gradient of $\tilde{\mathcal{M}}$ w.r.t. θ is accumulated at this time step as:

$$d\theta = d\theta - \eta_t \frac{\partial^2 L_{\mu_\theta(s_t)}(f_{\omega_t}, D^t_{train})}{\partial \theta \partial \omega_t} d\omega_{t+1}. \quad (5)$$

Algorithm 1 Training Teacher Model μ_θ

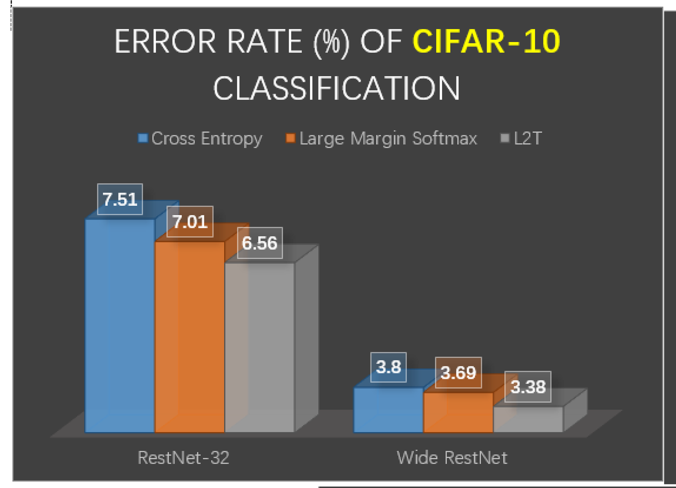
```
Input: Continuous relaxation m̃. Initial value of θ.
while Teacher model parameter θ not converged do          ▷ One teacher optimization step
    Randomly initialize student model parameter ω_0.
    for each time step t = 0, ··· , T − 1 do               ▷ Teach student model
        Conduct student model training step via Eqn. (1).
    end for
    dθ = 0. Compute dω_T via Eqn. (3).
    for each time step t = T − 1, ··· , 0 do               ▷ Reversely calculating the gradient dθ
        Update dθ as Eqn. (5).
        Compute dω_t as Eqn. (4).
    end for
    Update θ using dθ via gradient based optimization algorithm.
end while
Output: the final teacher model μ_θ.
```

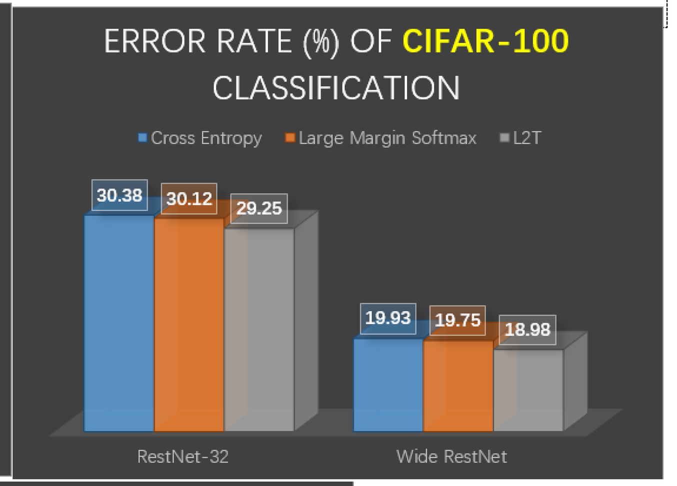


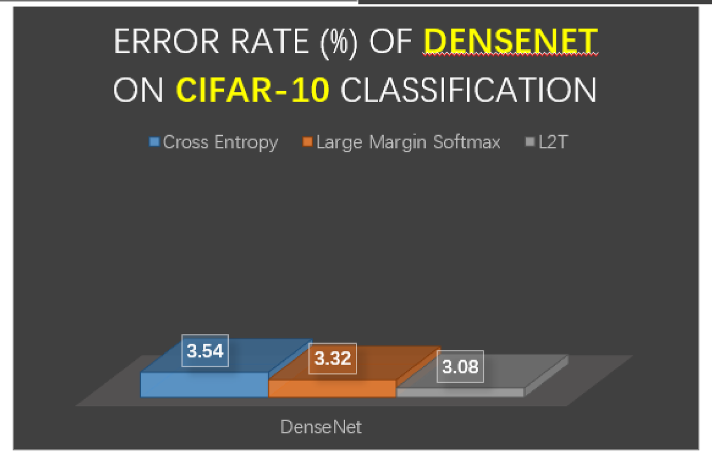
(a) loss function (b) teacher model

5. Experiments

- Image Classification Task







- Neural Machine Translation Task