Distilling Knowledge (知识蒸馏)

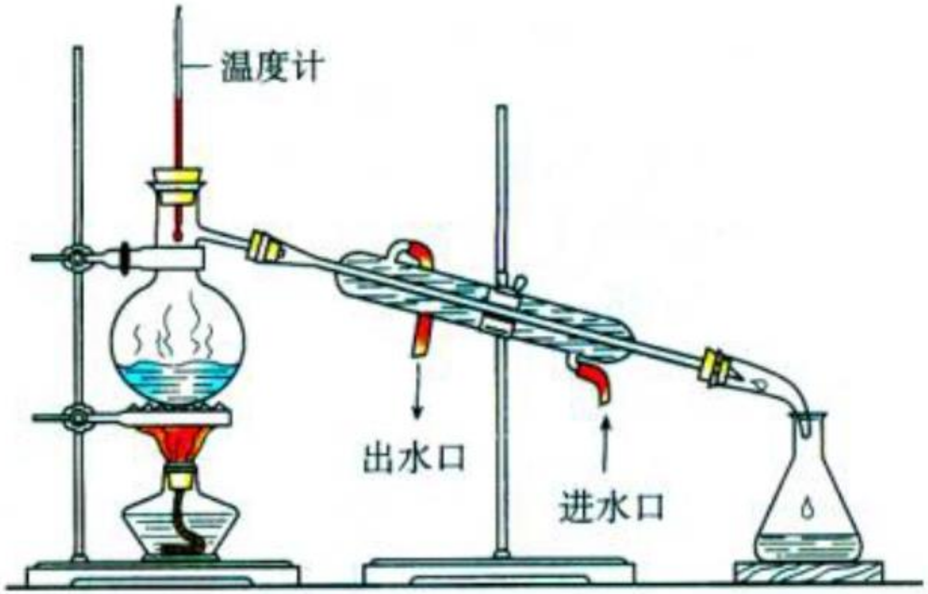
赵洲

浙江大学计算机学院 副教授

什么是蒸馏？

- 在化学中，蒸馏是一种有效的分离不同沸点组成部分的方法。

- 蒸馏的液体是混合物，由于组成部分的沸点不同，蒸馏时要根据目标物质的沸点设置蒸馏温度。



水蒸气蒸馏装置

模型背景

- 在模型训练过程中，需要复杂模型和计算资源，从大量与冗余数据中提取信息。因此训练好的模型存在**推理速度慢**和**推理所需资源高**的问题。
- 在模型部署过程中，对模型**推理延时**和**计算资源**有严格限制。

模型压缩（在保证性能的前提下减少模型的参数量）成为机器学习领域的一个重要问题。

“模型蒸馏”是模型压缩的一种重要方法。

知识蒸馏的提出

Distilling the Knowledge in a Neural Network

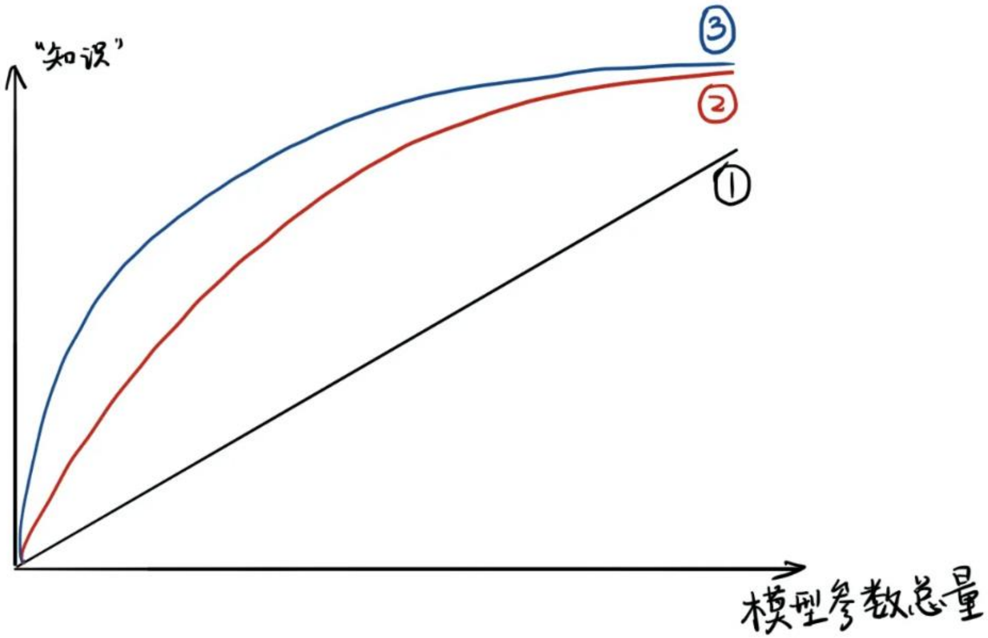
Geoffrey Hinton[*] [†]
Google Inc.
Mountain View
geoffhinton@google.com

Oriol Vinyals[†]
Google Inc.
Mountain View
vinyals@google.com

Jeff Dean
Google Inc.
Mountain View
jeff@google.com

常识与事实

- 常识：一个模型的参数量基本决定了其所能捕获到的数据内蕴含的“知识”量。
- 事实：
  - 模型参数量与捕获“知识”量之间为边际收益减少的增长（非线性）。
  - 相同模型架构和参数量，训练方法不同，所能捕获“知识”量也不同。

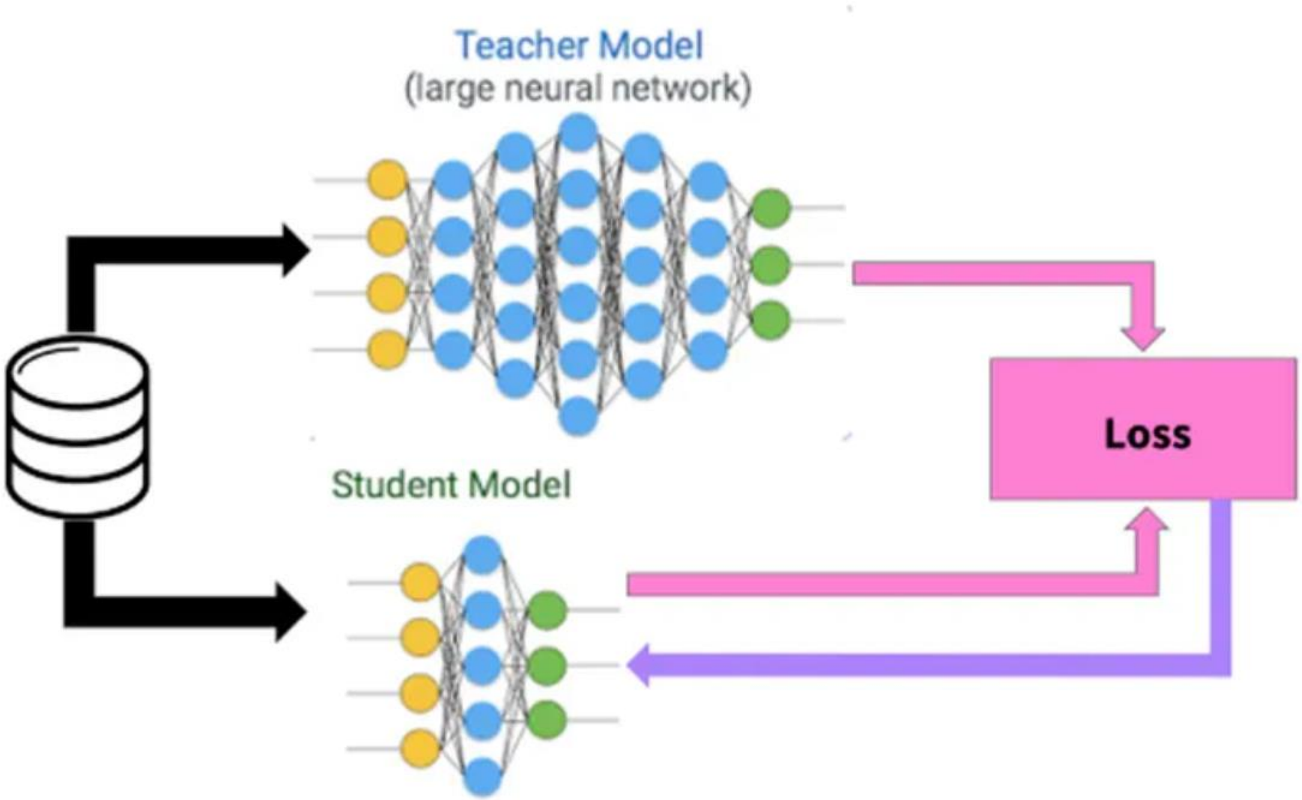


什么是知识蒸馏?

- 知识蒸馏使用“Teacher - Student”框架，其中“Teacher”是“知识”的输出者，“Student”是“知识”的接受者，分为两阶段：
  - 原始模型训练：训练“Teacher”模型（模型复杂，或由多个分别训练的模型集成）
  - 精炼模型训练：训练“Student”模型（参数量小，模型简单的单模型）
- “Teacher”和“Student”模型满足：对于输入X, 其都能输出Y, Y经过Softmax映射后输出对应类别的概率值。

知识蒸馏基本框架

- 学习能力强的“Teacher”模型将学习到的知识迁移给学习能力弱的“Student”模型，增强“Student”模型的泛化能力。
- “Teacher”模型扮演导师角色， “Student”部署上线。



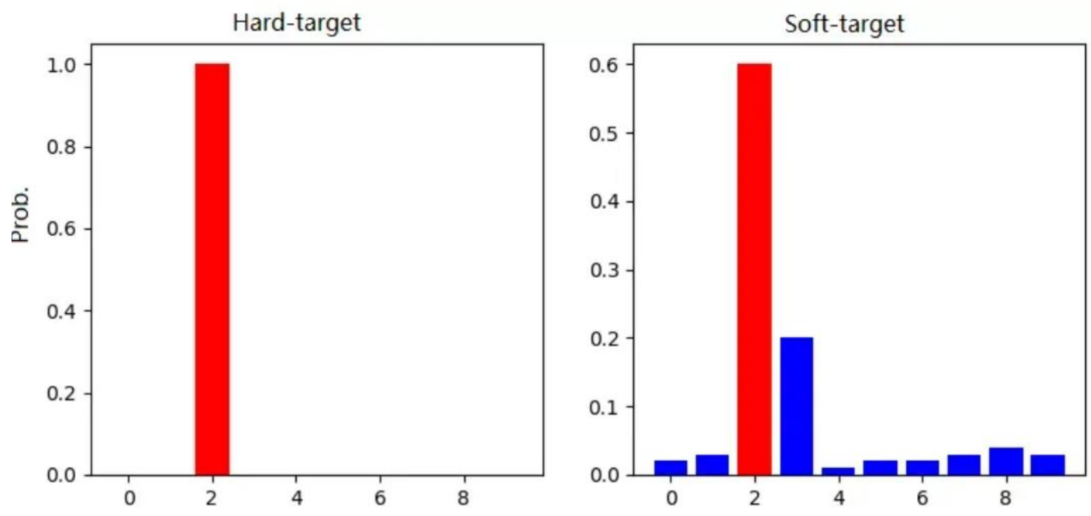
知识蒸馏大纲

- 目标蒸馏
  - KD in Network
  - Deep Mutual Learning (DML)
  - Born Again Network (BAN)

- 特征蒸馏
  - FitNets
  - Attention Transfer

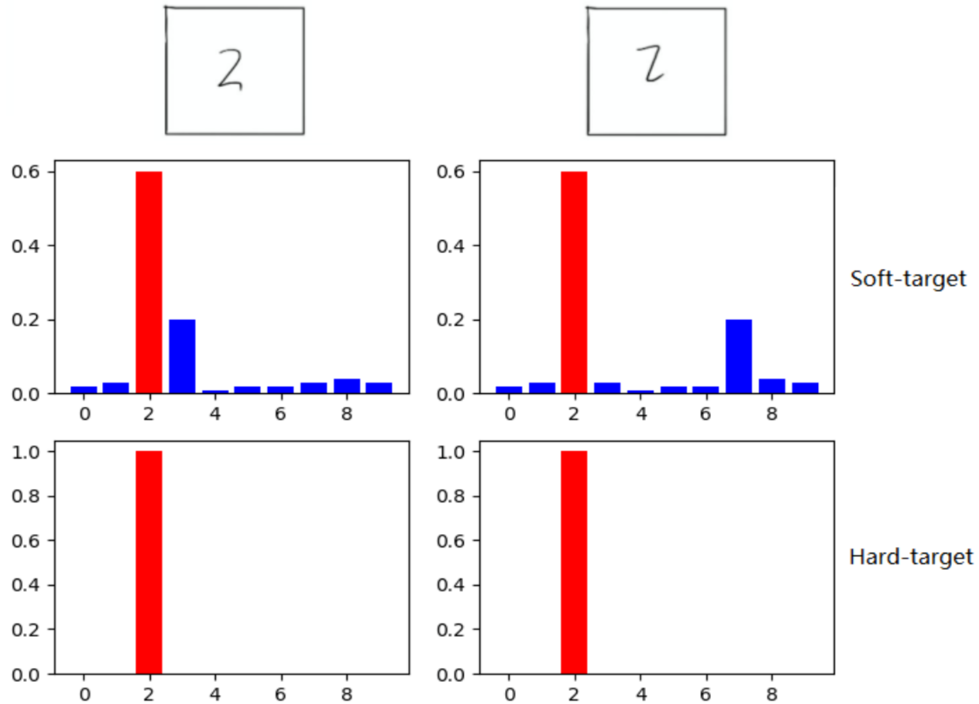
目标蒸馏

- 分类问题中模型最后有一个softmax层，其输出值对应相应类别的概率值。
- 传统训练方法通过定义一个损失函数，目标使神经网络预测值尽可能接近真实值(Hard-target)。
- 与传统训练方法不同，知识蒸馏使用“Teacher”模型的类别概率作为Soft-target来训练“Student”。



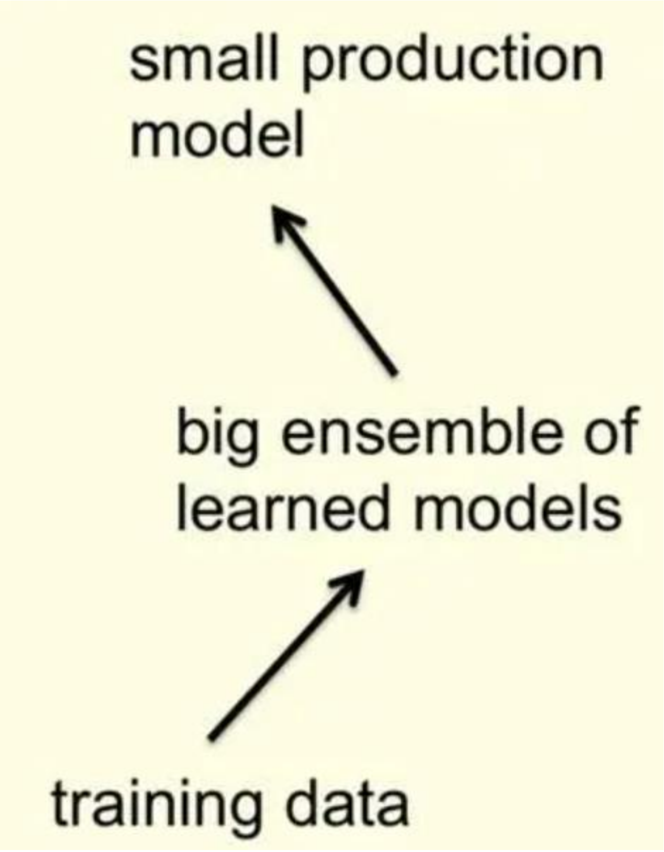
Soft-target训练优势？

- 除了正例，负标签也带有“Teacher”模型归纳推理的大量信息（某负标签对应概率大于其它负标签，代表“Teacher”模型推理时认为该样本与负标签有一定相似性）。
- 知识蒸馏使得每个样本给“Student”模型带来的信息大于Hard-target的训练方式。



暗知识（Dark Knowledge）

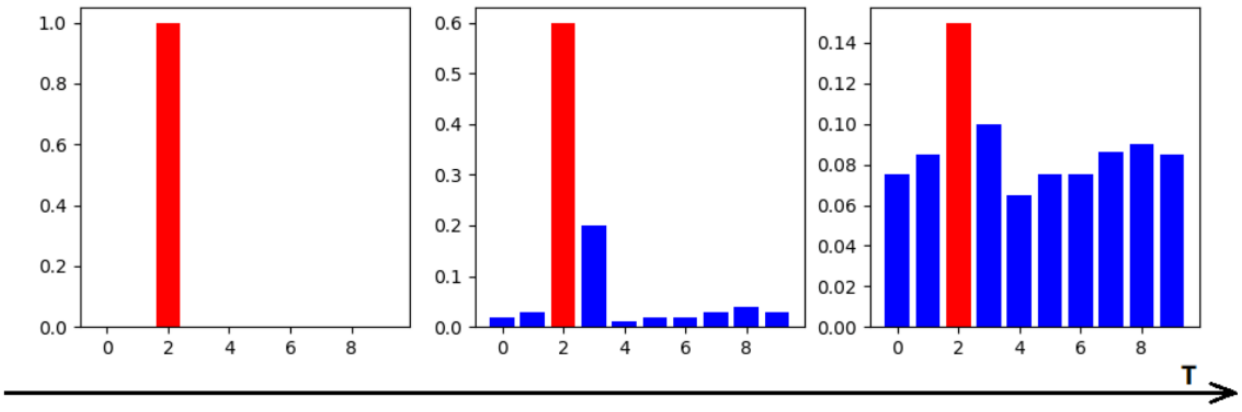
- 暗知识（Dark Knowledge）：隐藏在深度网络下的网络结构，节点之间的连接权重，以及网络的输出这些看得到的数据下的知识，如上述负标签信息（类别之间关联性的先验信息）。



蒸馏与温度？

- 温度T调高（T > 1）, Softmax的输出值分布趋向“陡峭”，接近于Hard-target, 从而减少负标签中的噪声的干扰。
- 温度T调低（T -> 0）, Softmax的输出值分布趋向“平缓”，接近于平均分布，从而实现从负标签中学习到部分信息量。
- T的选择与“Student”模型大小相关。“Student”模型参数小学习能力有限下，选择使用调低的温度。

$$q_i = \frac{exp(z_i/T)}{\sum_i exp(z_j/T)}$$



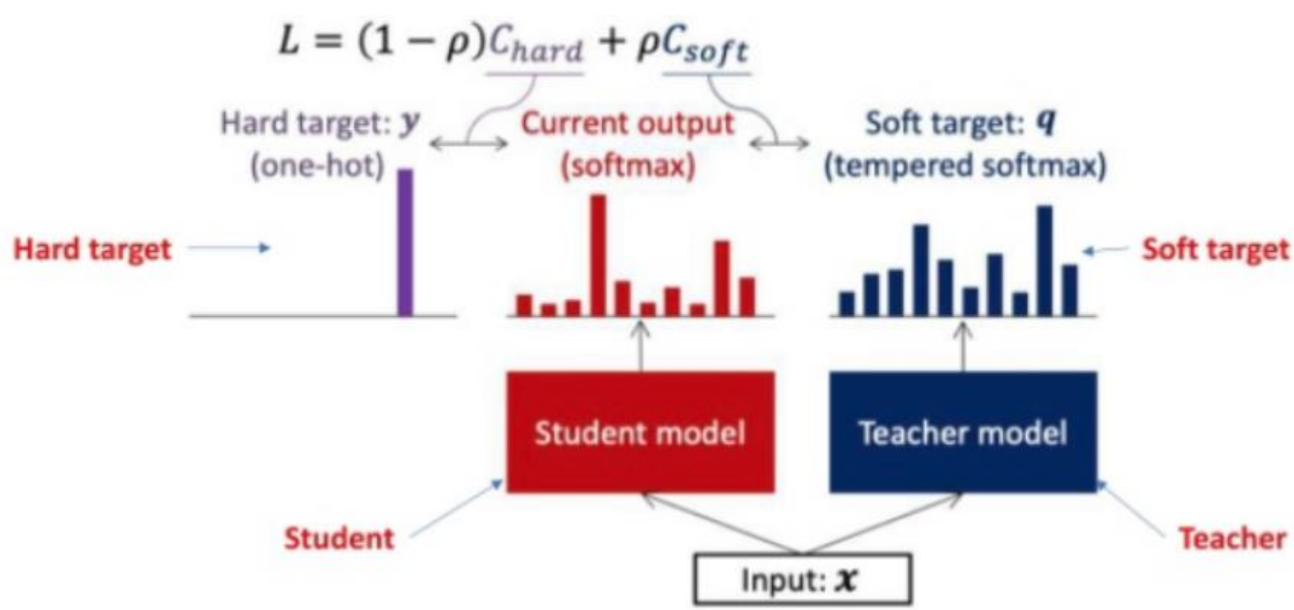
组合目标优化

- 知识蒸馏的目标函数由distill loss（对应soft-target）和student loss(对应hard-target)加权得到：

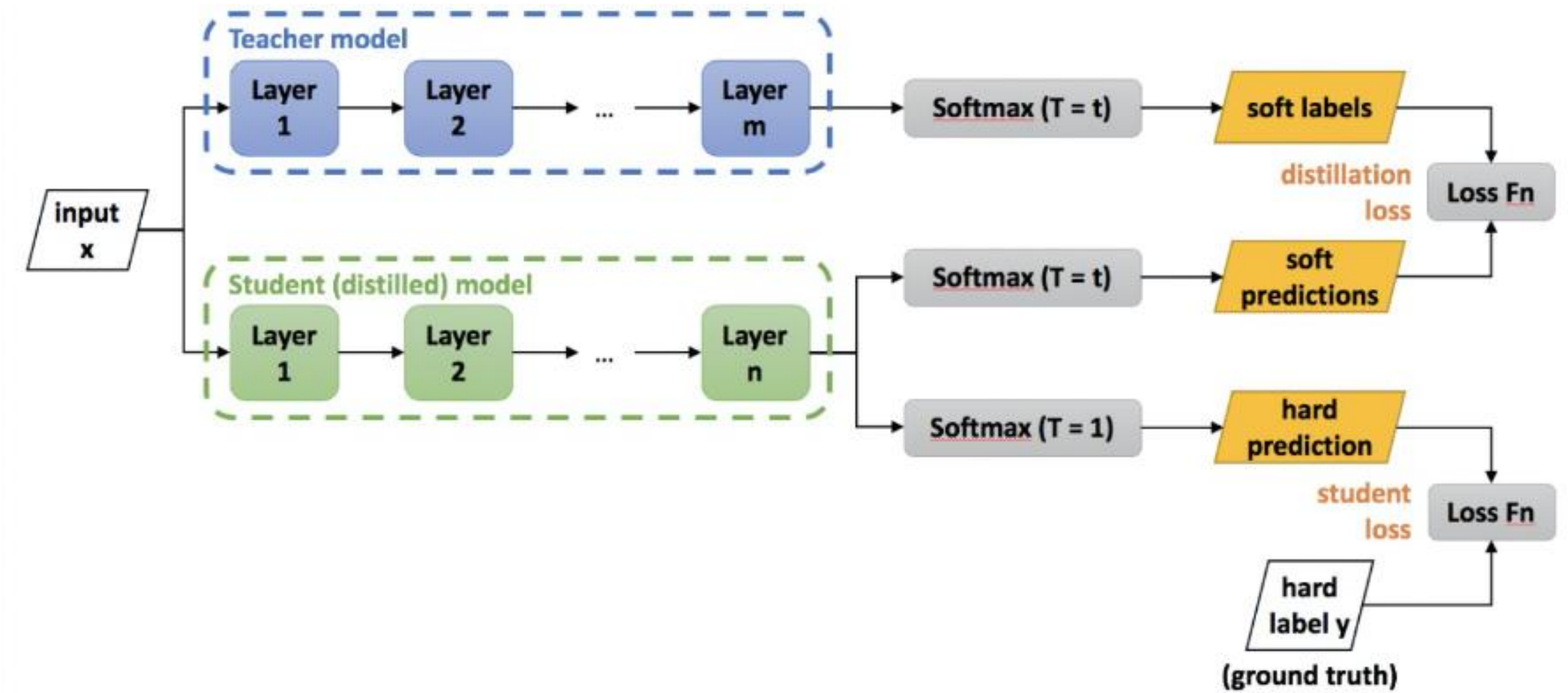
$$L = \alpha L_{soft} + \beta L_{hard}$$

$$L_{soft} = -\sum_{j}^{N} p_j^T \log(q_j^T) \quad p_i^T = \frac{\exp(v_i/T)}{\sum_k^N \exp(v_k/T)}, \quad q_i^T = \frac{\exp(z_i/T)}{\sum_k^N \exp(z_k/T)}$$

$$L_{hard} = -\sum_{j}^{N} c_j \log(q_j^1) \quad q_i^1 = \frac{\exp(z_i)}{\sum_j^N \exp(z_j)}$$



通用框架


Teacher model
Layer 1
Layer 2
...
Layer m
input x
Student (distilled) model
Layer 1
Layer 2
...
Layer n
Softmax (T = t)
Softmax (T = t)
Softmax (T = 1)
soft labels
soft predictions
hard prediction
distillation loss
Loss Fn
student loss
Loss Fn
hard label y
(ground truth)

知识蒸馏（Pytorch）

```
loss_fun = CrossEntropyLoss()
criterion  = nn.KLDivLoss()#KL散度
optimizer = torch.optim.SGD(model_student.parameters(),lr = 0.1,momentum = 0.9)

for step,batch in enumerate(dataloader):
    inputs = batch[0]
    labels = batch[1]

    #分别使用学生模型和教师模型对输入数据进行计算
    output_student = model_student(inputs)
    output_teacher = model_teacher(inputs)

    #计算学生模型预测结果和教师模型预测结果之间的KL散度
    loss_soft = criterion(output_student,output_teacher)

    #计算学生模型和真实标签之间的交叉熵损失函数值
    loss_hard = loss_fun(output_student,labels)

    loss = 0.9*loss_soft + 0.1*loss_hard
    print(loss)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

Matching Logits

- Matching Logist直接使用Softmax层的输入作为Soft-target, 通过最小化目标函数使得 “Teacher”模型和 “Student”模型之间的平方差。

$$L_{logits} = \frac{1}{2}(z_i - v_i)^2 \qquad \frac{\partial L_{logits}}{\partial z_i} = z_i - v_i$$

- 与经过Softmax层后的Soft-target关系：

$$\frac{\partial L_{soft}}{\partial z_i} = \frac{1}{T}(q_i - p_i) = \frac{1}{T}\left(\frac{exp(z_i/T)}{\sum_i exp(z_j/T)}\right)$$

$$T \rightarrow \infty \quad \frac{\partial Lsoft}{\partial z_i} \approx \frac{1}{T}\left(\frac{1+z_i/T}{N+\sum_j z_j/T} - \frac{1+v_i/T}{N+\sum_j v_j/T}\right)$$

$$\frac{\partial L_{soft}}{z_j} \approx \frac{1}{NT^2}(z_i - v_i)$$

实验结果

#	Model	SST-2	QQP	MNLI-m	MNLI-mm
		Acc	F_1/Acc	Acc	Acc
1	$\text{BERT}_{\text{LARGE}}$ (Devlin et al., 2018)	94.9	72.1/89.3	86.7	85.9
2	$\text{BERT}_{\text{BASE}}$ (Devlin et al., 2018)	93.5	71.2/89.2	84.6	83.4
3	OpenAI GPT (Radford et al., 2018)	91.3	70.3/88.5	82.1	81.4
4	BERT ELMo baseline (Devlin et al., 2018)	90.4	64.8/84.7	76.4	76.1
5	GLUE ELMo baseline (Wang et al., 2018)	90.4	63.1/84.3	74.1	74.5
6	Distilled $\text{BiLSTM}_{\text{SOFT}}$	**90.7**	**68.2/88.1**	**73.0**	**72.6**
7	BiLSTM (our implementation)	86.7	63.7/86.2	68.7	68.3
8	BiLSTM (reported by GLUE)	85.9	61.4/81.7	70.3	70.8
9	BiLSTM (reported by other papers)	$87.6^{\dagger}$	– $/82.6^{\ddagger}$	66.9^{*}	66.9^{*}

Table 1: Test results on different datasets. The BiLSTM results reported by other papers are drawn from Zhou et al. (2016),† Wang et al. (2017),‡ and Williams et al. (2017).* All of our test results are obtained from the GLUE benchmark website.

实验结果

	# of Par.	Inference Time
$\text{BERT}_{\text{LARGE}}$	335 (349×)	1060 (434×)
ELMo	93.6 (98×)	36.71 (15×)
$\text{BiLSTM}_{\text{SOFT}}$	0.96 (1×)	2.44 (1×)

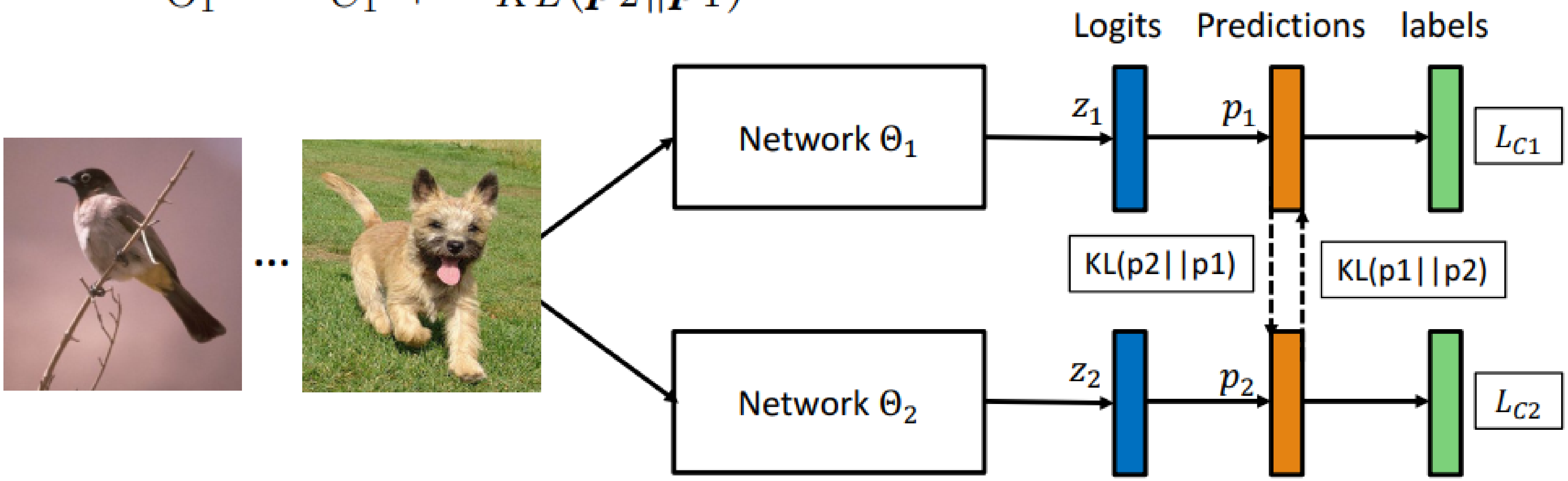
Table 2: Single-sentence model size and inference speed on SST-2. # of Par. denotes number of millions of parameters, and inference time is in seconds.

DML (互相蒸馏)

$$L_{C_1} = -\sum_{i=1}^{N}\sum_{m=1}^{M} I(y_i, m)\log(p_1^m(\boldsymbol{x}_i))$$

$$D_{KL}(\boldsymbol{p}_2\|\boldsymbol{p}_1) = \sum_{i=1}^{N}\sum_{m=1}^{M} p_2^m(\boldsymbol{x}_i)\log\frac{p_2^m(\boldsymbol{x}_i)}{p_1^m(\boldsymbol{x}_i)}$$

$$L_{\Theta_1} = L_{C_1} + D_{KL}(\boldsymbol{p}_2\|\boldsymbol{p}_1)$$



互相蒸馏实验结果

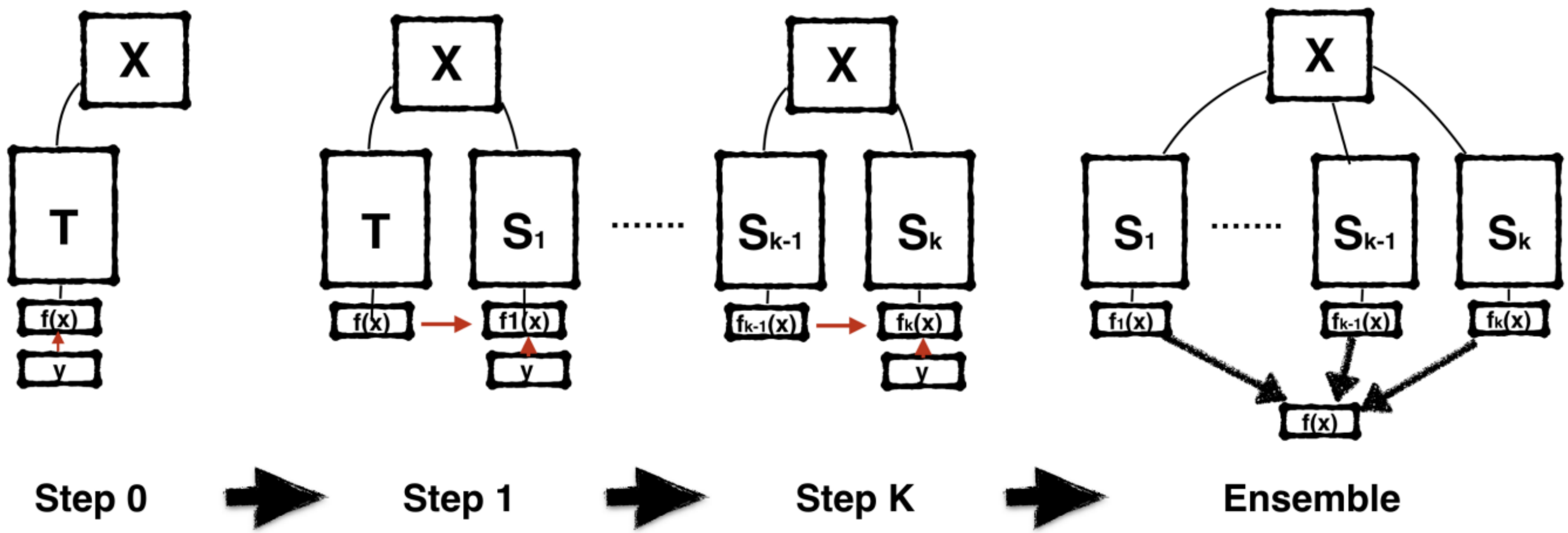
Network Types		Independent		DML		DML-Independent	
Net 1	Net 2	Net 1	Net 2	Net 1	Net 2	Net 1	Net 2
Resnet-32	Resnet-32	68.99	68.99	71.19	70.75	1.20	1.76
WRN-28-10	Resnet-32	78.69	68.99	78.96	70.73	0.27	1.74
MobileNet	Resnet-32	73.65	68.99	76.13	71.10	2.48	2.11
MobileNet	MobileNet	73.65	73.65	76.21	76.10	2.56	2.45
WRN-28-10	MobileNet	78.69	73.65	80.28	77.39	1.59	3.74
WRN-28-10	WRN-28-10	78.69	78.69	80.28	80.08	1.59	1.39

Top-1 accuracy (%) on the CIFAR-100 dataset. "DML-Independent" measures the difference in accuracy between the network learned with DML and the same network learned independently.

BAN (再生蒸馏)

$$\mathcal{L}(f(x, \arg\min_{\theta_k} \mathcal{L}(f(x, \theta_{k-1}))), f(x, \theta_k))$$

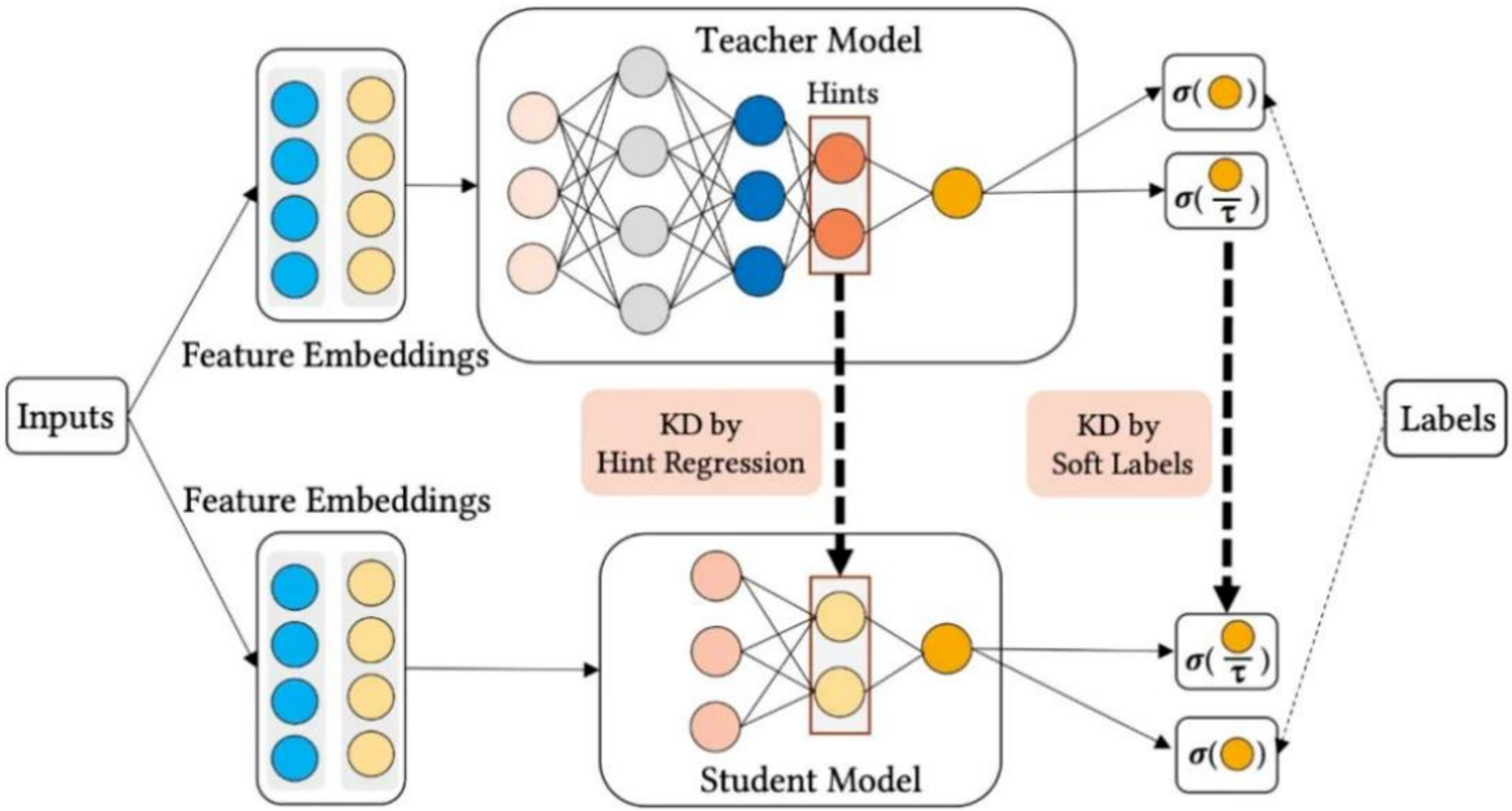
$$\hat{f}^k(x) = \sum_{i=1}^{k} f(x, \theta_i)/k$$



Graphical representation of the BAN training procedure: during the first step the teacher model T is trained from the labels Y. Then, at each consecutive step a new identical model is initialized from a different random seed and trained from the supervision of the earlier generation. At the end of the procedure additional gains can be achieved with an ensemble of multiple students generations.

特征蒸馏

- 特征蒸馏：“Teacher”模型将特征级知识迁移给“Student”模型，通过Hint Regression蒸馏损失函数。

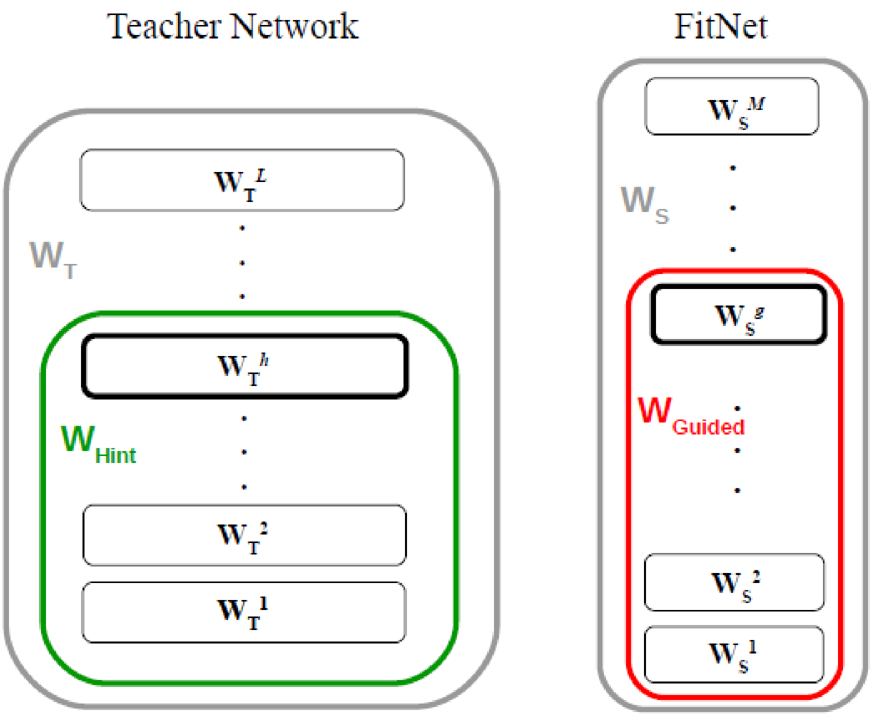


FitNets: Hints for Thin Deep Nets

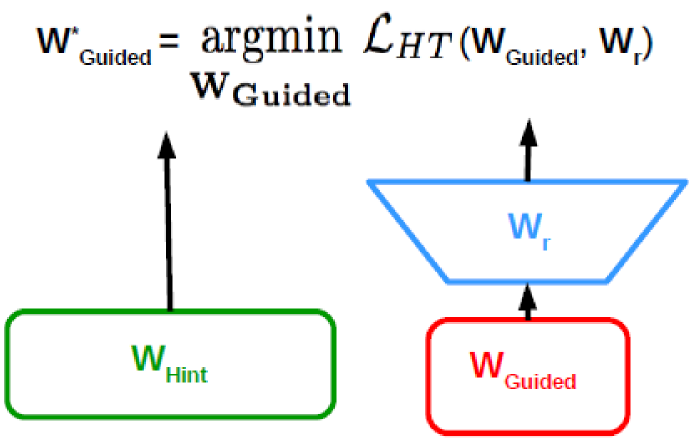
Adriana Romero[1], **Nicolas Ballas**[2], **Samira Ebrahimi Kahou**[3], **Antoine Chassang**[2], **Carlo Gatta**[4] & **Yoshua Bengio**[2]†
[1]Universitat de Barcelona, Barcelona, Spain.
[2]Université de Montréal, Montréal, Québec, Canada. †CIFAR Senior Fellow.
[3]École Polytechnique de Montréal, Montréal, Québec, Canada.
[4]Centre de Visió per Computador, Bellaterra, Spain.

Hint Regression蒸馏

$$\mathcal{L}_{HT}(\mathbf{W}_{\mathbf{Guided}}, \mathbf{W}_{\mathbf{r}}) = \frac{1}{2}||u_h(\mathbf{x}; \mathbf{W}_{\mathbf{Hint}}) - r(v_g(\mathbf{x}; \mathbf{W}_{\mathbf{Guided}}); \mathbf{W}_{\mathbf{r}})||^2$$



Teacher and Student Networks



Hints Training

FITNET训练

Algorithm 1 FitNet Stage-Wise Training.

Input: $\mathbf{W_S}, \mathbf{W_T}, g, h$
Output: $\mathbf{W_S^*}$

1: $\mathbf{W_{Hint}} \leftarrow \{\mathbf{W_T}^1, \ldots, \mathbf{W_T}^h\}$
2: $\mathbf{W_{Guided}} \leftarrow \{\mathbf{W_S}^1, \ldots, \mathbf{W_S}^g\}$
3: Intialize $\mathbf{W_r}$ to small random values
4: $\mathbf{W_{Guided}^*} \leftarrow \underset{\mathbf{W_{Guided}}}{\operatorname{argmin}} \mathcal{L}_{HT}(\mathbf{W_{Guided}}, \mathbf{W_r})$
5: $\{\mathbf{W_S}^1, \ldots, \mathbf{W_S}^g\} \leftarrow \{\mathbf{W_{Guided}}^{*1}, \ldots, \mathbf{W_{Guided}}^{*g}\}$
6: $\mathbf{W_S^*} \leftarrow \underset{\mathbf{W_S}}{\operatorname{argmin}} \mathcal{L}_{KD}(\mathbf{W_S})$

MINIST错误率

Algorithm	# params	Misclass
Compression		
Teacher	∼361K	0.55%
Standard backprop	∼30K	1.9%
KD	∼30K	0.65%
FitNet	∼30K	**0.51%**
State-of-the-art methods		
Maxout		0.45%
Network in Network		0.47%
Deeply-Supervised Networks		**0.39%**

MNIST error

CIFAR准确率

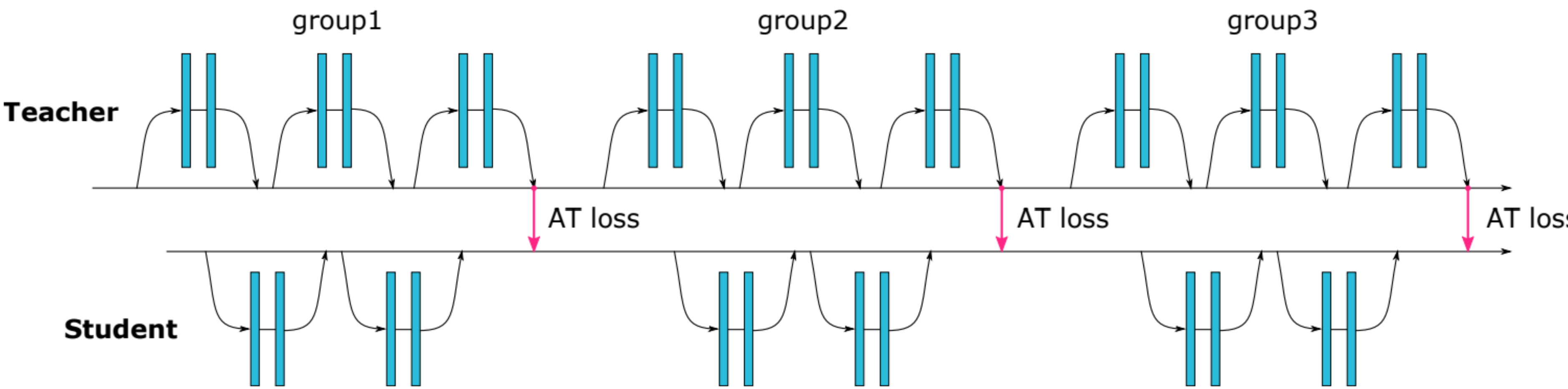
Algorithm	# params	Accuracy
Compression		
FitNet	~2.5M	**91.61%**
Teacher	~9M	90.18%
Mimic single	~54M	84.6%
Mimic single	~70M	84.9%
Mimic ensemble	~70M	85.8%
State-of-the-art methods		
Maxout		90.65%
Network in Network		91.2%
Deeply-Supervised Networks		**91.78%**
Deeply-Supervised Networks (19)		88.2%

Accuracy on CIFAR-10

Algorithm	# params	Accuracy
Compression		
FitNet	~2.5M	**64.96%**
Teacher	~9M	63.54%
State-of-the-art methods		
Maxout		61.43%
Network in Network		64.32%
Deeply-Supervised Networks		**65.43%**

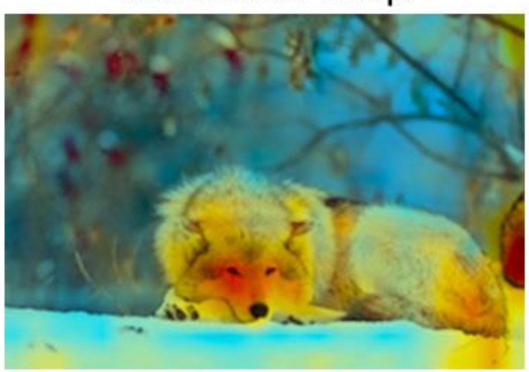
Accuracy on CIFAR-100

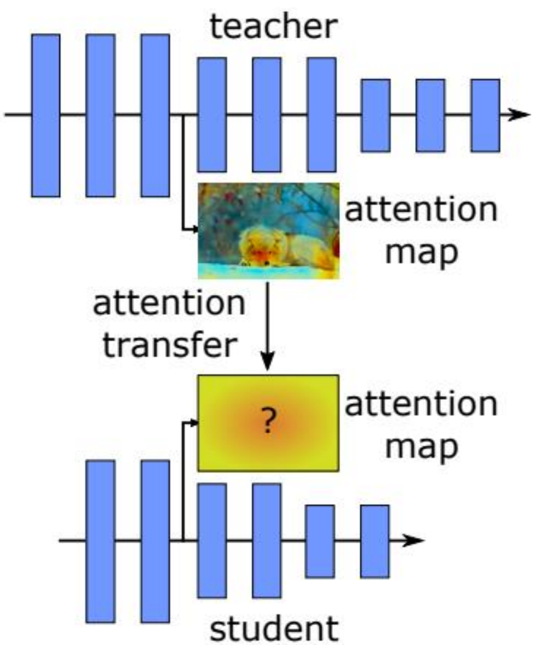
注意迁移（Attention Transfer）



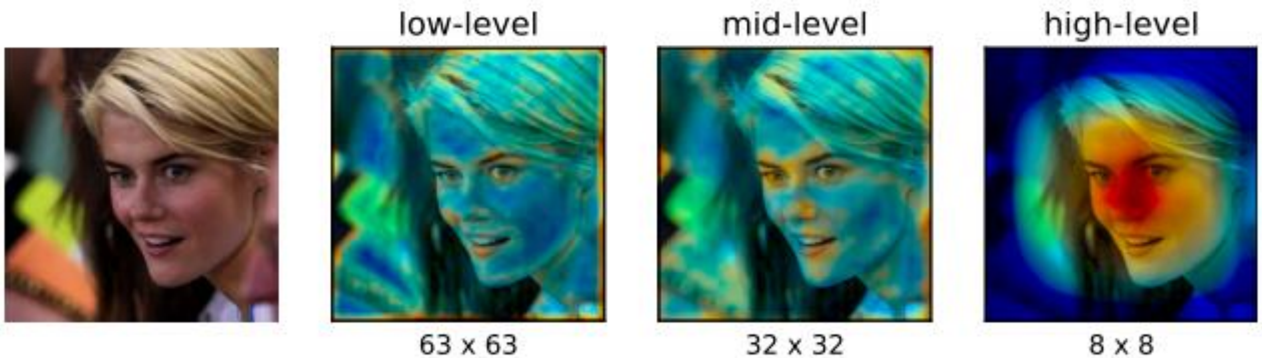
input image

attention map

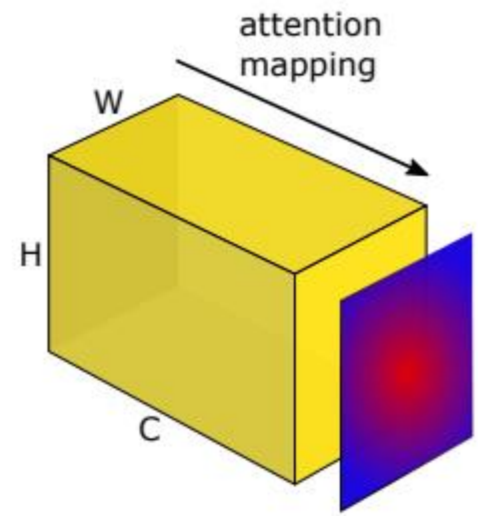




卷积网络注意力



$$\mathcal{F} : R^{C\times H\times W} \rightarrow R^{H\times W}$$



注意迁移优化目标

- 基于梯度的注意力迁移：

$$J_S = \frac{\partial}{\partial x}\mathcal{L}(\mathbf{W_S}, x), J_T = \frac{\partial}{\partial x}\mathcal{L}(\mathbf{W_T}, x)$$

$$\mathcal{L}_{AT}(\mathbf{W_S}, \mathbf{W_T}, x) = \mathcal{L}(\mathbf{W_S}, x) + \frac{\beta}{2}||J_S - J_T||_2$$

注意迁移实验结果

student	teacher	student	AT	F-ActT	KD	AT+KD	teacher
NIN-thin, 0.2M	NIN-wide, 1M	9.38	8.93	9.05	8.55	8.33	7.28
WRN-16-1, 0.2M	WRN-16-2, 0.7M	8.77	7.93	8.51	7.41	7.51	6.31
WRN-16-1, 0.2M	WRN-40-1, 0.6M	8.77	8.25	8.62	8.39	8.01	6.58
WRN-16-2, 0.7M	WRN-40-2, 2.2M	6.31	5.85	6.24	6.08	5.71	5.23

Activation-based attention transfer (AT) with various architectures on CIFAR-10. Error is computed as median of 5 runs with different seed. F-ActT means full-activation transfer

总结

- 知识蒸馏将复杂模型或者多个模型“Teacher”学习到知识迁移到一个轻量级模型“Student”，使得模型轻量级部署。
- 暗知识（Dark Knowledge）是知识蒸馏的关键要素之一。
- 目标蒸馏算法将网络的输出（Soft-target）作为知识。
- 特征蒸馏算法将网络学习到的特征作为知识。