

Model Compression

Advanced NLP: Summer 2023

Anoop Sarkar

Compressing Large Language Models

Reduce memory and compute costs

- There are many different ways to solve the compression problem:
 1. **Distillation**: train a small lightweight student model on the output of a large teacher model.
 2. **Pruning**: Use an importance criterion to prune weights, prune layers, prune attention heads, etc.
 3. **Reduce precision** of the weights: FP16, int8, etc.
 4. Low rank **factorization** of weight matrices.
 5. **Weight sharing** (ALBERT).

Distilling the Knowledge in a Neural Network

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, NIPS 2014 DL workshop

<https://arxiv.org/abs/1503.02531>

See also: Bucila, Caruana,
and Niculescu-Mizil. Model
compression. In KDD, 2006.

Large Language Models

Are they necessary?

- Scaling to larger language models has led to improved zero-shot and few-shot accuracy on many NLP tasks.
- All modern deep learning models are heavily over-parameterized compared to the dataset size they train on.
- Smaller models by themselves do not give the same accuracy.
- Deployment of LLMs is challenging from a compute cost perspective.
- "Distill" a student model by training it on the output of a "teacher" model (a LLM).

Standard setup for training a language model

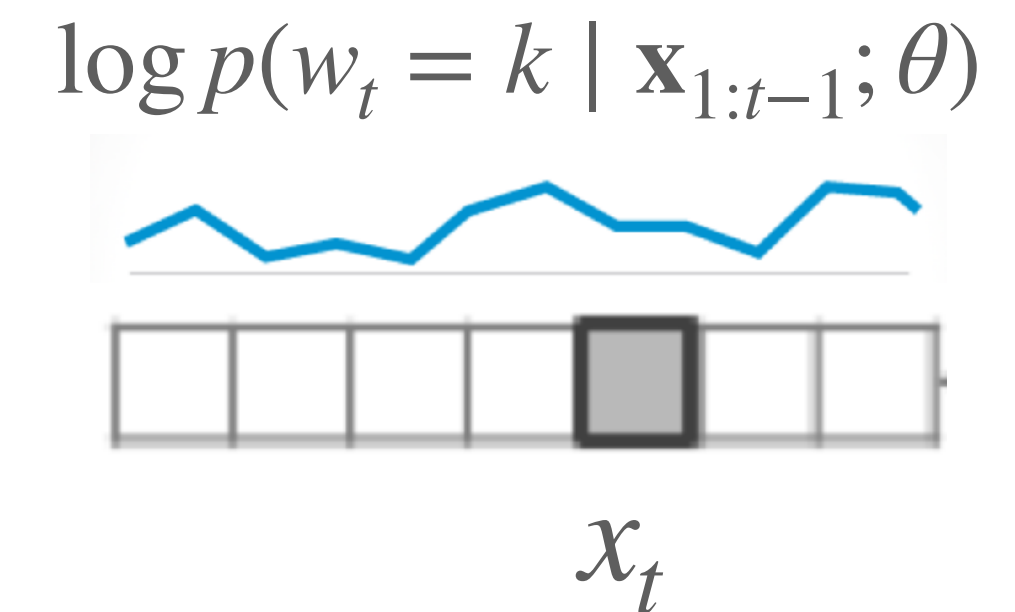
- Minimize the log likelihood loss for prediction.
- Find parameters θ to minimize loss \mathcal{L} :

$$\mathcal{L} = - \sum_t \sum_{k \in \mathcal{V}} \delta(x_t = k) \log p(w_t = k \mid \mathbf{x}_{1:t-1}; \theta)$$

w_t is the softmax over the vocabulary \mathcal{V}

x_t is the ground truth target token; \mathbf{x} is the sentence

$\delta(p) = 1$ if p is true and 0 otherwise



Knowledge Distillation

Can be used for pre-training or fine-tuning

- Train a larger teacher model (massive LLM models) to get a **teacher** distribution over outputs $q(\cdot)$ with parameters θ_T
- Train a smaller **student** model $p(\cdot)$ to mimic the teacher
- The student model has parameters $\theta \ll \theta_T$

Word level distillation for training a language model

- Teacher distribution: $q(w_t \mid \mathbf{x}_{1:t-1}; \theta_T)$

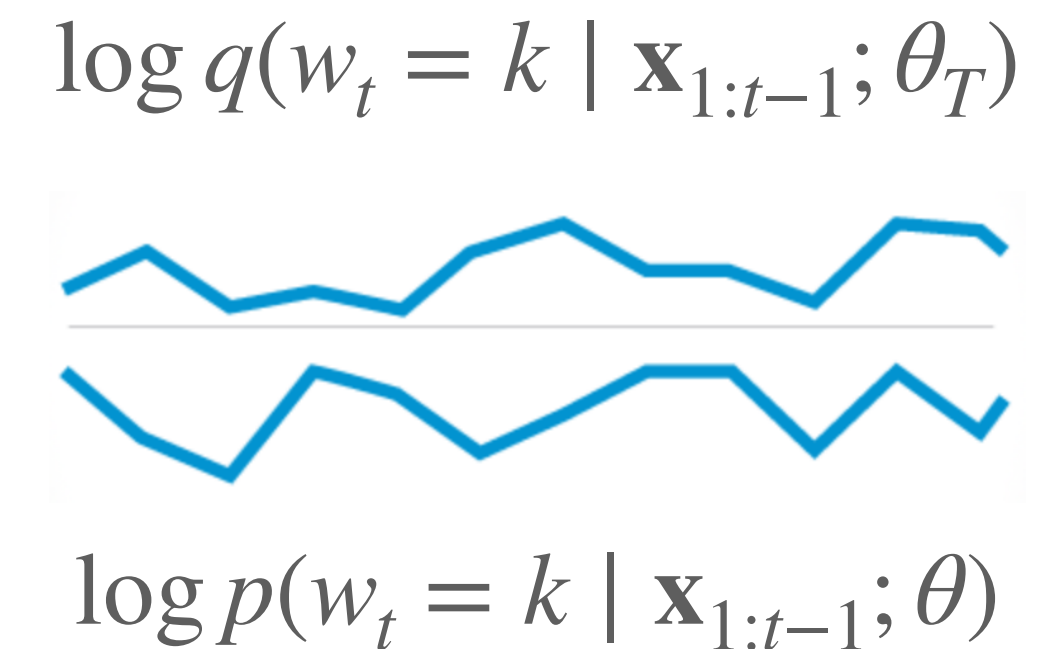
- **Standard** loss:

$$\mathcal{L} = - \sum_t \sum_{k \in \mathcal{V}} \delta(x_t = k) \log p(w_t = k \mid \mathbf{x}_{1:t-1}; \theta)$$

- **Distillation** loss (uses **cross entropy** between p and q):

$$\mathcal{L}_D = - \sum_t \sum_{k \in \mathcal{V}} q(w_t = k \mid \mathbf{x}_{1:t-1}; \theta_T) \log p(w_t = k \mid \mathbf{x}_{1:t-1}; \theta)$$

$\delta(p) = 1$ if p is true and 0 otherwise



No Knowledge Distillation

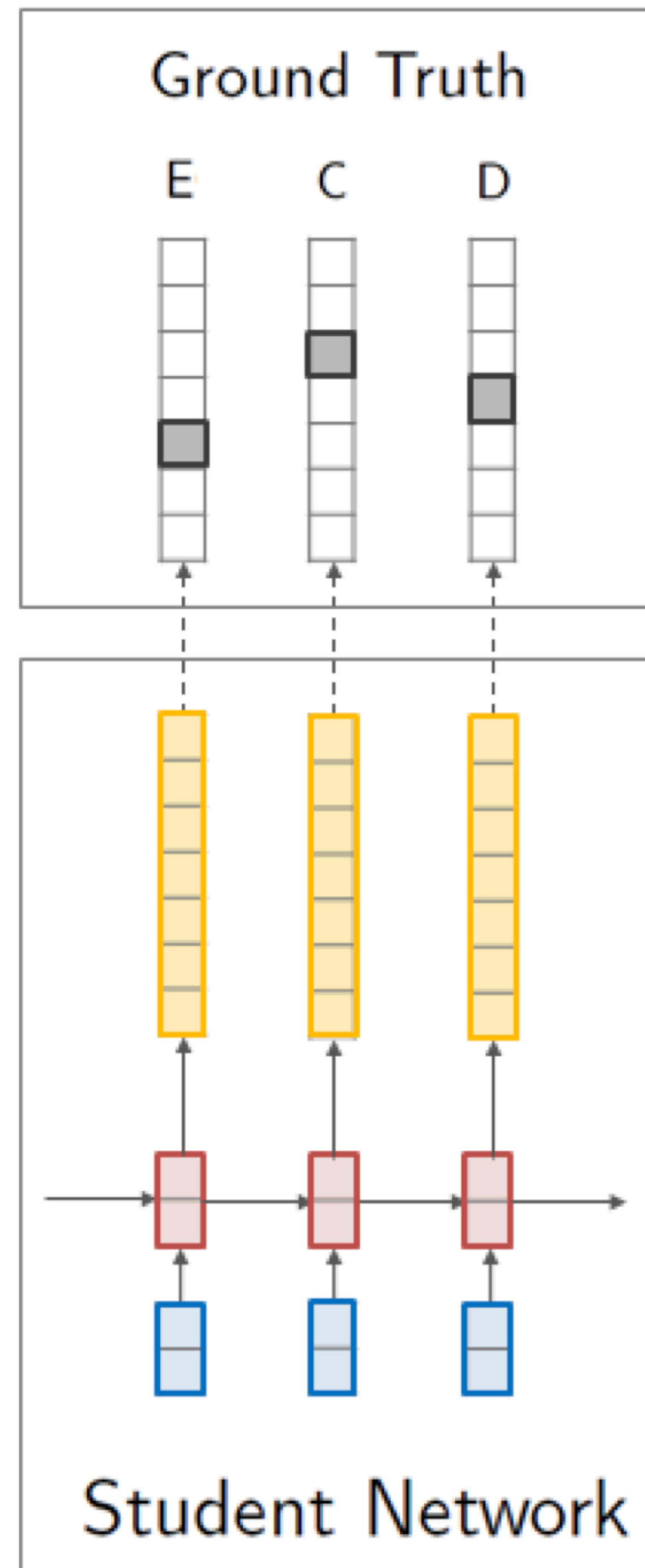


Fig from https://nlp.seas.harvard.edu/slides/emnlp16_seqkd.pdf

Word Level Knowledge Distillation

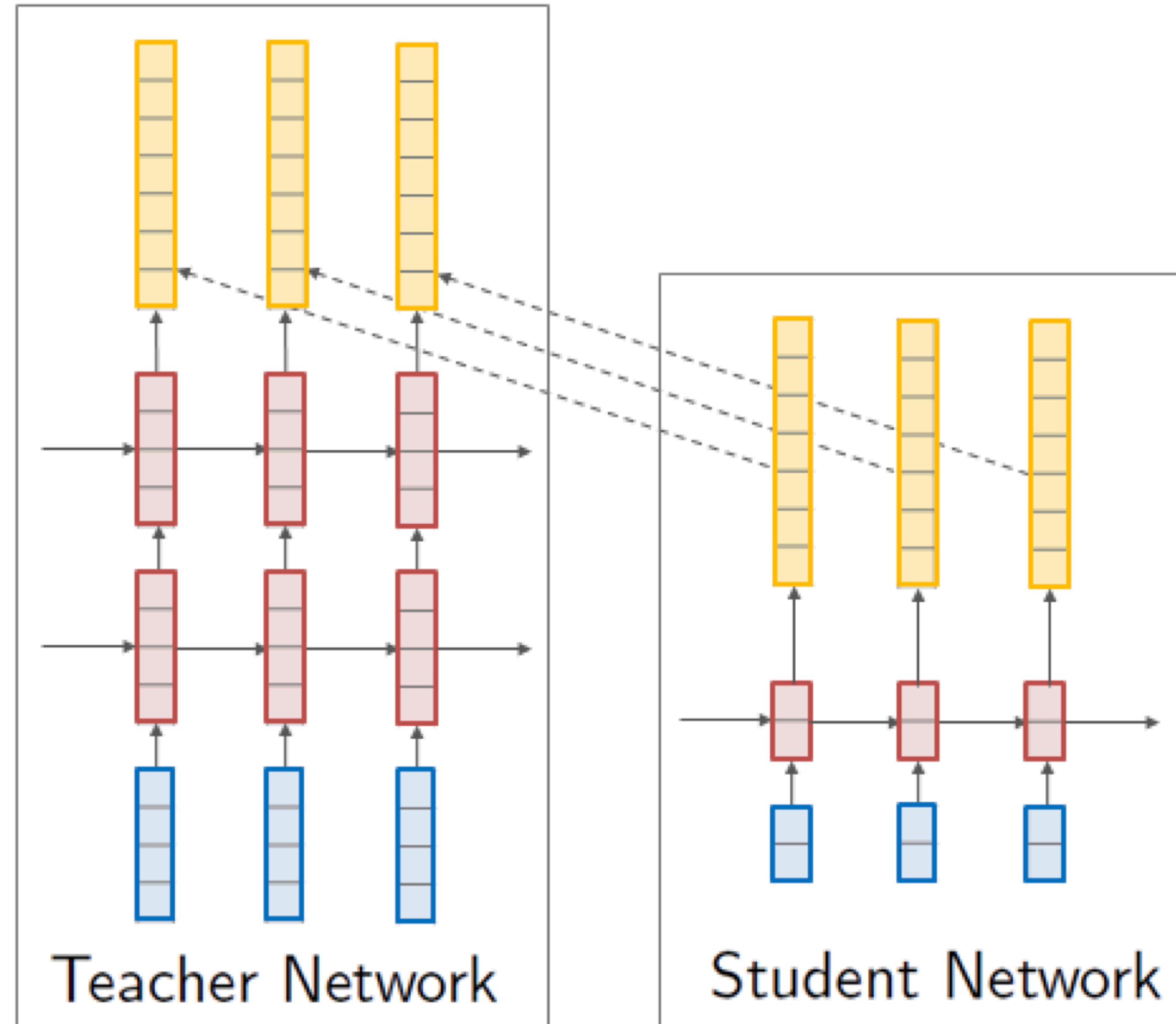


Fig from https://nlp.seas.harvard.edu/slides/emnlp16_seqkd.pdf

Combine standard loss and distillation loss

$$\mathcal{L}_C = \alpha \mathcal{L} + (1 - \alpha) \mathcal{L}_D$$

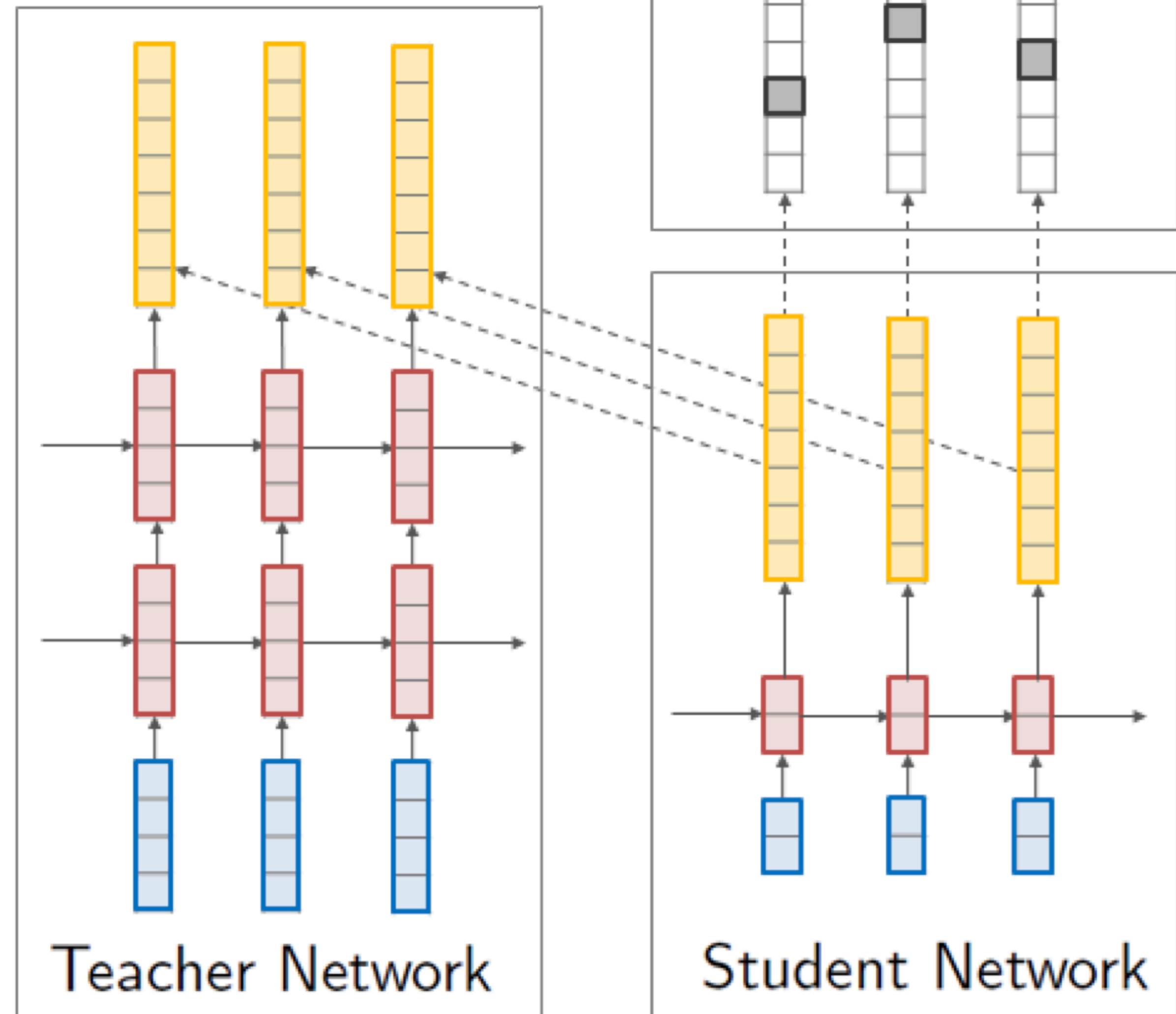


Fig from https://nlp.seas.harvard.edu/slides/emnlp16_seqkd.pdf

Soft targets

- Standard method to compute $p(w_t = k \mid \mathbf{x}_{1:t-1}; \theta)$

$$p_k = \frac{\exp(z_k)}{\sum_i \exp(z_i)}$$

z_i are the logits used to compute the softmax

- Divide the logits by a **temperature** parameter to get a softer distribution

$$p_k = \frac{\exp(\frac{z_k}{T})}{\sum_i \exp(\frac{z_i}{T})}$$

Soft targets

- Gradient wrt z_k

$$\frac{\partial \mathcal{L}_D}{\partial z_k} = \frac{1}{T}(q_k - p_k) = \frac{1}{T} \left(\frac{\exp(\frac{z_k}{T})}{\sum_i \exp(\frac{z_i}{T})} - \frac{\exp(\frac{v_k}{T})}{\sum_i \exp(\frac{v_i}{T})} \right) \approx \frac{1}{T^2}(z_k - v_k)$$

z_i, v_i are the logits used to compute the softmax for the teacher and student respectively

assuming the logits are zero-meaned, i.e. $\sum_i z_i = 0$ and $\sum_i v_i = 0$

Soft targets

Fig from Hinton's "Dark Knowledge"
talk slides

cow	dog	cat	car
0	1	0	0
cow	dog	cat	car
10^{-6}	.9	.1	10^{-9}
cow	dog	cat	car
.05	.3	.2	.005

Hard Target

Teacher distribution

Softened distribution

"Softened outputs reveal the **dark knowledge** in the teacher distribution"

Soft targets from BERT output distribution

```
Input: ['[CLS]', 'i', 'think', 'this', 'is', 'the', 'beginning', 'of', 'a', 'beautiful', '[MASK]', '.', '[SEP]']
Rank 0 - Token: day - Prob: 0.21348
Rank 1 - Token: life - Prob: 0.18380
Rank 2 - Token: future - Prob: 0.06267
Rank 3 - Token: story - Prob: 0.05854
Rank 4 - Token: world - Prob: 0.04935
Rank 5 - Token: era - Prob: 0.04555
Rank 6 - Token: time - Prob: 0.03210
Rank 7 - Token: year - Prob: 0.01722
Rank 8 - Token: history - Prob: 0.01663
Rank 9 - Token: summer - Prob: 0.01335
Rank 10 - Token: adventure - Prob: 0.01233
Rank 11 - Token: dream - Prob: 0.01209
Rank 12 - Token: moment - Prob: 0.01129
Rank 13 - Token: night - Prob: 0.01084
Rank 14 - Token: beginning - Prob: 0.00937
Rank 15 - Token: season - Prob: 0.00664
Rank 16 - Token: journey - Prob: 0.00621
Rank 17 - Token: period - Prob: 0.00553
Rank 18 - Token: relationship - Prob: 0.00517
Rank 19 - Token: thing - Prob: 0.00508
```

Fig from <https://medium.com/huggingface/distilbert-8cf3380435b5>

Distillation training step

```
1 import torch
2 import torch.nn as nn
3 import torch.nn.functional as F
4 from torch.optim import Optimizer
5
6 KD_loss = nn.KLDivLoss(reduction='batchmean')
7
8 def kd_step(teacher: nn.Module,
9             student: nn.Module,
10             temperature: float,
11             inputs: torch.tensor,
12             optimizer: Optimizer):
13     teacher.eval()
14     student.train()
15
16     with torch.no_grad():
17         logits_t = teacher(inputs=inputs)
18         logits_s = student(inputs=inputs)
19
20     loss = KD_loss(input=F.log_softmax(logits_s/temperature, dim=-1),
21                   target=F.softmax(logits_t/temperature, dim=-1))
22
23     loss.backward()
24     optimizer.step()
25     optimizer.zero_grad()
```

Fig from <https://medium.com/huggingface/distilbert-8cf3380435b5>

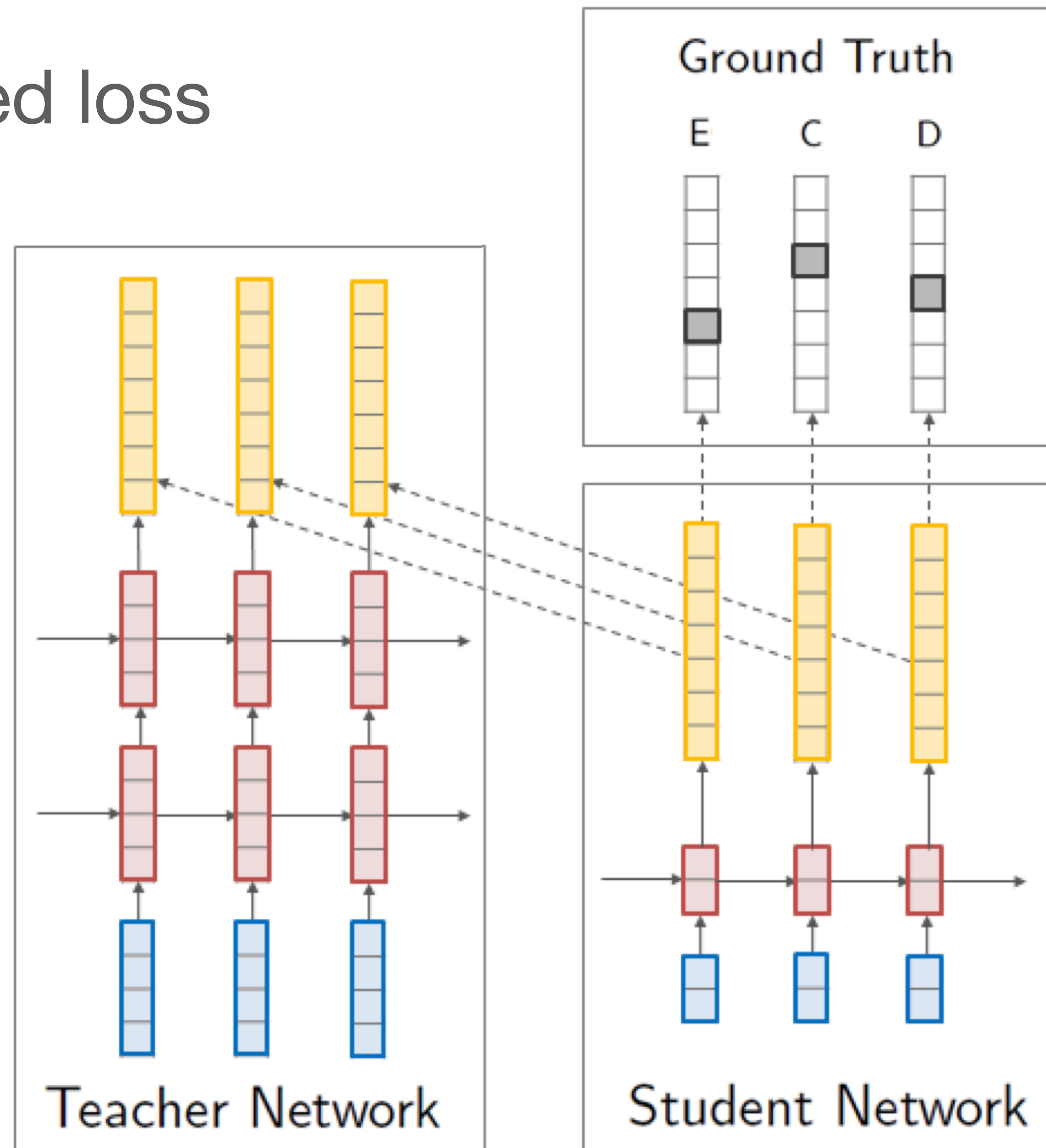
Soft targets and combined loss

$$\mathcal{L}_C = \alpha \mathcal{L} + (1 - \alpha) \mathcal{L}_D$$

Temperature is set to 1
for Ground Truth

Higher temperature
used for Teacher
Distribution

Fig from https://nlp.seas.harvard.edu/slides/emnlp16_seqkd.pdf



Distillation of BERT models

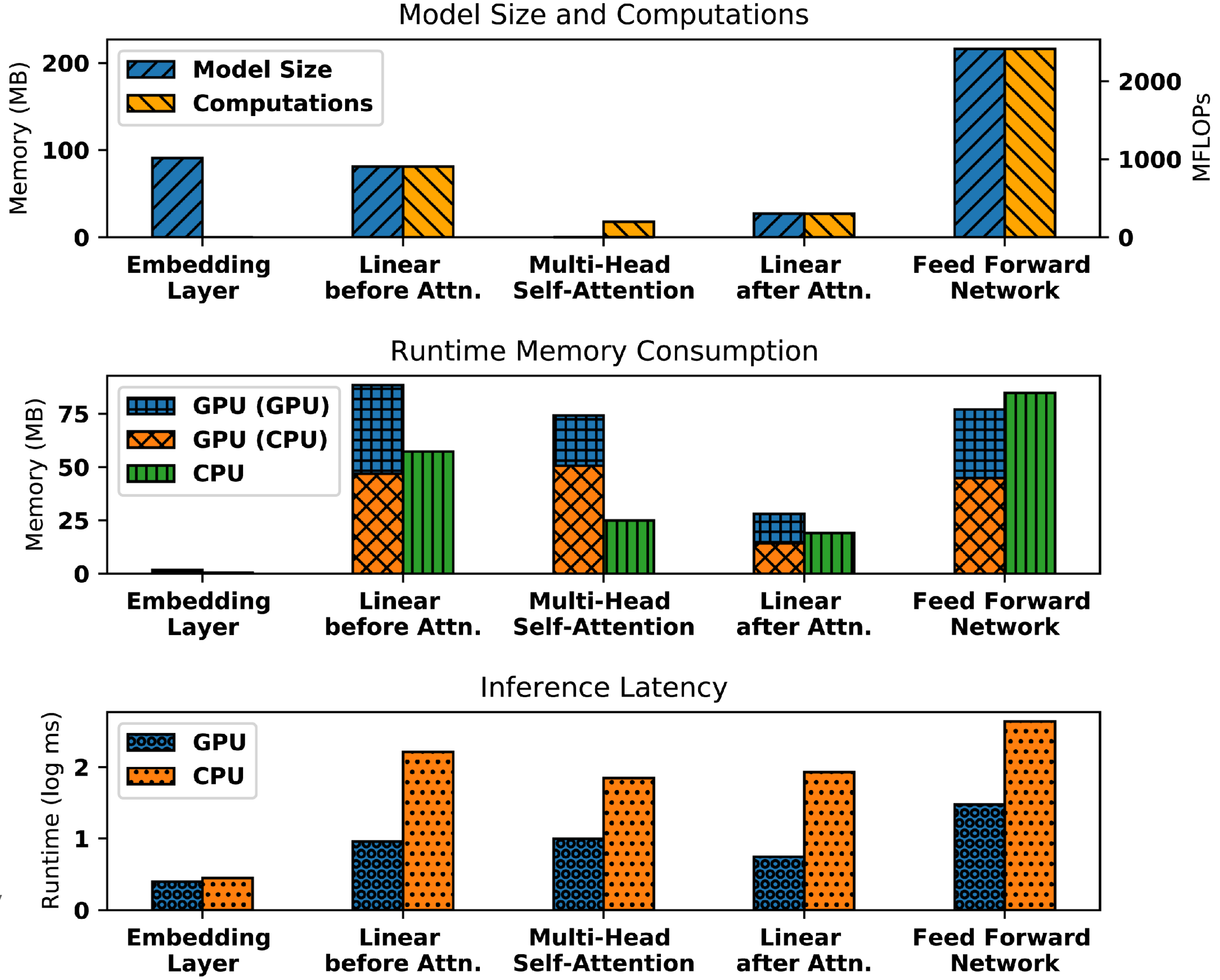
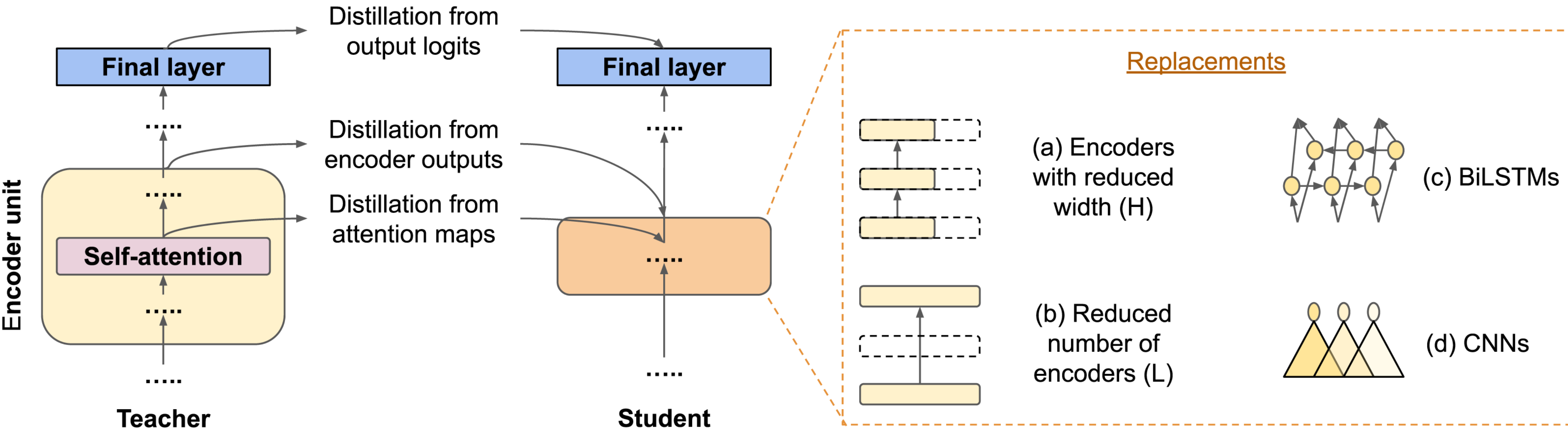


Fig from https://doi.org/10.1162/tacI_a_00413

Distillation of BERT models



DistilBERT

- *"Our student is a small version of BERT in which we removed the token-type embeddings and the pooler (used for the next sentence classification task) and kept the rest of the architecture identical while reducing the numbers of layers by a factor of two."*
- Why not reduce the hidden size as well?
 - *"In our experiments, the number of layers was the determining factor for the inference time, more than the hidden size."*
- Using L2 loss instead of cross-entropy loss?
 - *"cross-entropy loss leads to significantly better performance"*
- Initialization is important
 - *"We thus initialize our student ... by taking one layer out of two, leveraging the common hidden size between student and teacher."*

<https://medium.com/huggingface/distilbert-8cf3380435b5>

DistilBERT

	Macro Score	CoLA	MNLI	MNLI-MM	MRPC		QNLI
		mcc	acc	acc	acc	f1	acc
GLUE BASELINE (ELMo + BiLSTMs)	68.7	44.1	68.6 (avg)		70.8	82.3	71.1
BERT base	78.0	55.8	83.7	84.1	86.3	90.5	91.1
DistilBERT	75.2	42.5	81.6	81.1	82.4	88.3	85.5

QQP		RTE	SST-2	STS-B		WNLI
acc	f1	acc	acc	pearson	spearmanr	acc
88.0	84.3	53.4	91.5	70.3	70.5	56.3
90.9	87.7	68.6	92.1	89.0	88.6	43.7
90.6	87.7	60.0	92.7	84.5	85.0	55.6

	Nb of parameters (millions)	Inference Time (s)
GLUE BASELINE (ELMo + BiLSTMs)	180	895
BERT base	110	668
DistilBERT	66	410

Distillation of BERT models

Different ways to distill information from a teacher

- Distillation during fine-tuning:
 - On SQuAD 1.1 (QA task) BERT gets **88.5** F1 and DistilBERT gets **85.1**
 - Fine-tuning DistilBERT on the QA task using a fine-tuned BERT model gets **86.2** F1.
- Distillation from Output Logits
- Distillation from Encoder Outputs (distil each layer)
- Distillation from Attention Maps (attn is softmax so can be easily distilled)

THE LOTTERY TICKET HYPOTHESIS: FINDING SPARSE, TRAINABLE NEURAL NETWORKS

Jonathan Frankle

MIT CSAIL

`jfrankle@csail.mit.edu`

Michael Carbin

MIT CSAIL

`mcarbin@csail.mit.edu`