



Knowledge Distillation

Daman Arora, COV884

Overview:



- Knowledge Distillation(in general)
 - Distilling the Knowledge in a Neural Network
- Knowledge Distillation(in NLP)
 - Sequence Level Knowledge Distillation
 - Lifelong Language Knowledge Distillation
 - Patient Knowledge Distillation for BERT Model Compression
- The Lottery Ticket Hypothesis
 - LTH for Pre-trained BERT Networks

What is Knowledge Distillation(KD)?



- Current DL models are too large to be deployed.
- **KD**: Transferring knowledge from a large model to a small model that is more suitable for deployment.

Distilling the Knowledge in a Neural Network

Hinton et. al, NeurIPS '14 Deep Learning Workshop

- Consider a large pre-trained model and a smaller model.
- How to train smaller model?
 - a. Train on gold labels of the training data
 - b. Train on output of the large pre-trained model.

The idea of “Dark Knowledge”



- A BMW is much closer to a garbage truck than it is to a carrot.
- Training on probabilities as “soft targets” much better:
 - More informative.
 - Less variance in gradients, leads to faster training since learning rate can be increased.

Increasing entropy in targets



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

T is a temperature parameter and
 q_i are the logits of a NN

Demonstration




- Training on MNIST:
 - Large Network (2 hidden layers, 1200 ReLU)
 - Small Network (2 hidden layers, 800 ReLU)

Setting	Number of errors
Large Model	67
Small Model	146
Small Model trained on Soft Targets of Large Model	74

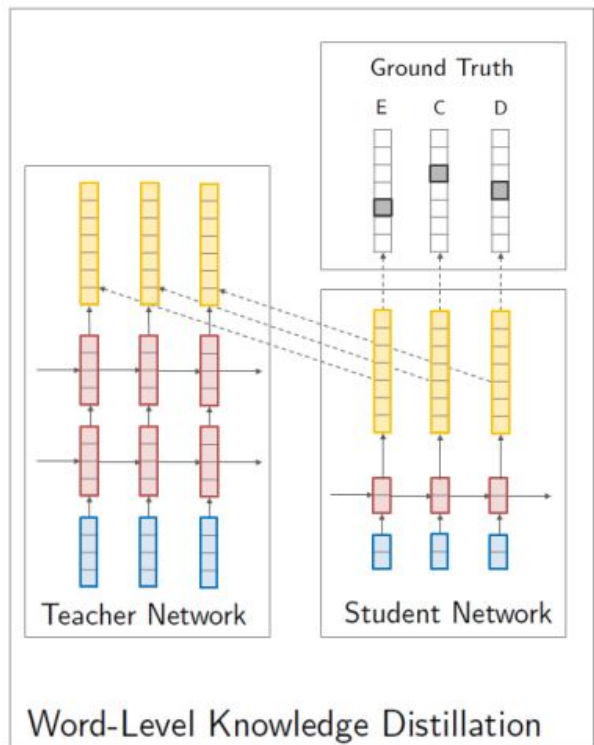
Sequence Level Knowledge Distillation

Kim et. al, EMNLP '16

- 
- Word Level KD
 - Sequence Level KD
 - Sequence Level Interpolation

*Next few slides have been borrowed from
https://nlp.seas.harvard.edu/slides/emnlp16_seqkd.pdf

Word Level Knowledge Distillation



Hard targets

$$\mathcal{L}_{NLL} = - \sum_t \sum_{k \in \mathcal{V}} \mathbb{1}\{y_t = k\} \log p(w_t = k | \mathbf{y}_{1:t-1}; \theta)$$

Soft targets

$$\mathcal{L}_{WORD-KD} = - \sum_t \sum_{k \in \mathcal{V}} q(w_t = k | \mathbf{y}_{1:t-1}; \theta_T) \log p(w_t = k | \mathbf{y}_{1:t-1}; \theta)$$

$$\mathcal{L} = \alpha \mathcal{L}_{WORD-KD} + (1 - \alpha) \mathcal{L}_{NLL}$$

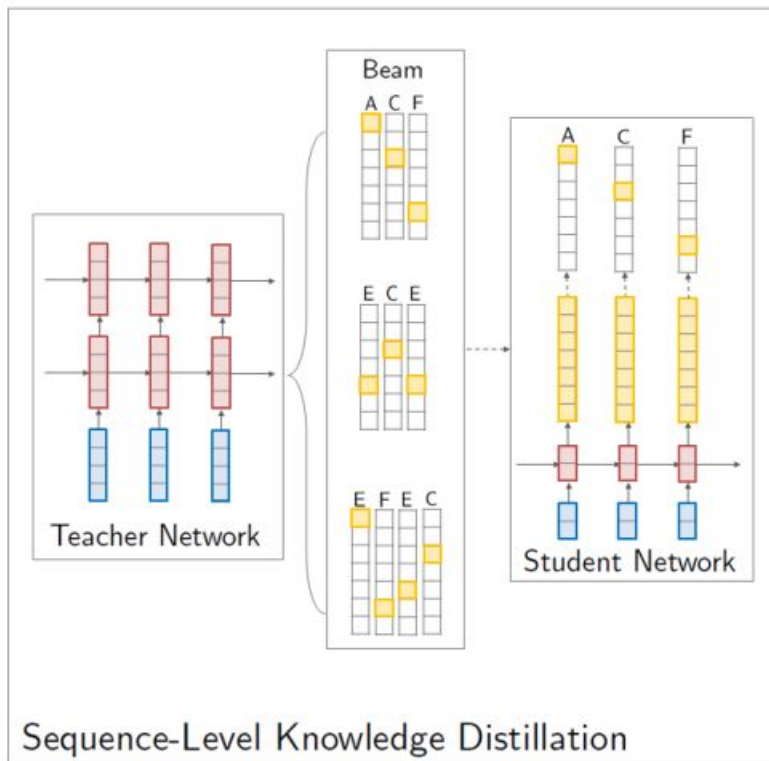
Word Level Knowledge Distillation



English → German (WMT 2014)

Model	BLEU
4 × 1000 Teacher	19.5
2 × 500 Baseline (No-KD)	17.6
2 × 500 Student (Word-KD)	17.7
2 × 300 Baseline (No-KD)	16.9
2 × 300 Student (Word-KD)	17.6

Sequence Level Knowledge Distillation

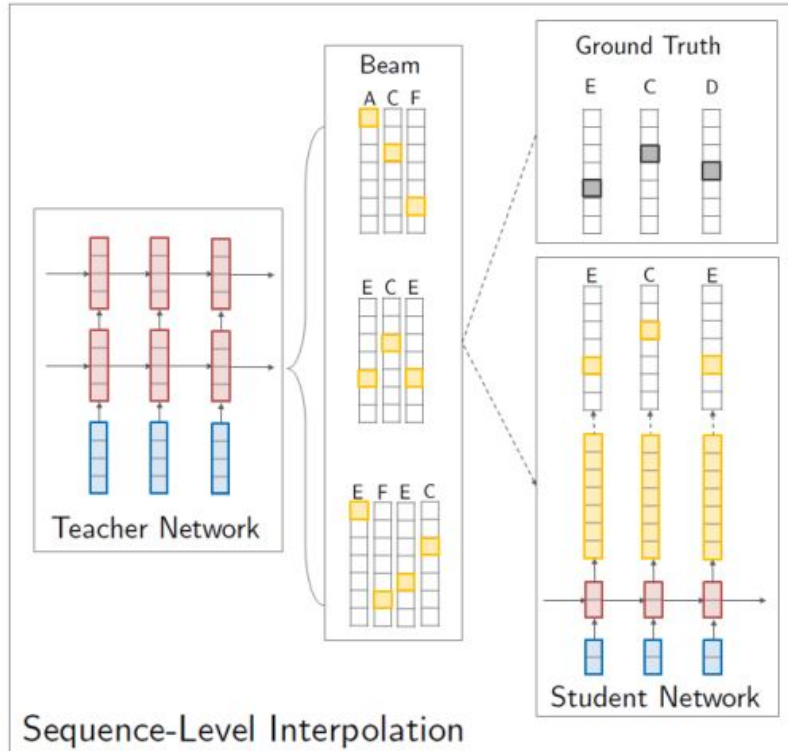


- Increase the probability of the sentence selected using Beam search.

$$\hat{\mathbf{y}} \approx \arg \max_{\mathbf{w}} q(\mathbf{w} | \mathbf{x})$$

$$\mathcal{L}_{\text{SEQ-KD}} = -\log p(t = \hat{\mathbf{y}} | s)$$

Sequence Level Interpolation



- Problem with Seq KD is that y and \hat{y} can be very different
- Solution:
 - Take sentence closest to y in the beam.

$$\tilde{y} \approx \operatorname{argmax}_{\mathbf{t} \in \mathcal{T}_K} \operatorname{sim}(\mathbf{t}, \mathbf{y})$$

- Measure similarity using smoothed sentence level BLEU

Model	$\text{BLEU}_{K=1}$	$\Delta_{K=1}$	$\text{BLEU}_{K=5}$	$\Delta_{K=5}$	PPL	$p(\hat{\mathbf{y}})$
4×1000						
Teacher	17.7	—	19.5	—	6.7	1.3%
2×500						
Student	14.7	—	17.6	—	8.2	0.9%
Word-KD	15.4	+0.7	17.7	+0.1	8.0	1.0%
Seq-KD	18.9	+4.2	19.0	+1.4	22.7	16.9%
Seq-Inter	18.9	+4.2	19.3	+1.7	15.8	7.6%

Applications:



- Multilingual NMT with KD - ICLR '19
- Attention-Guided Answer Distillation for Machine Reading Comprehension - EMNLP '19
- ...

Lifelong Language Knowledge Distillation

Chuang et. al, EMNLP '20

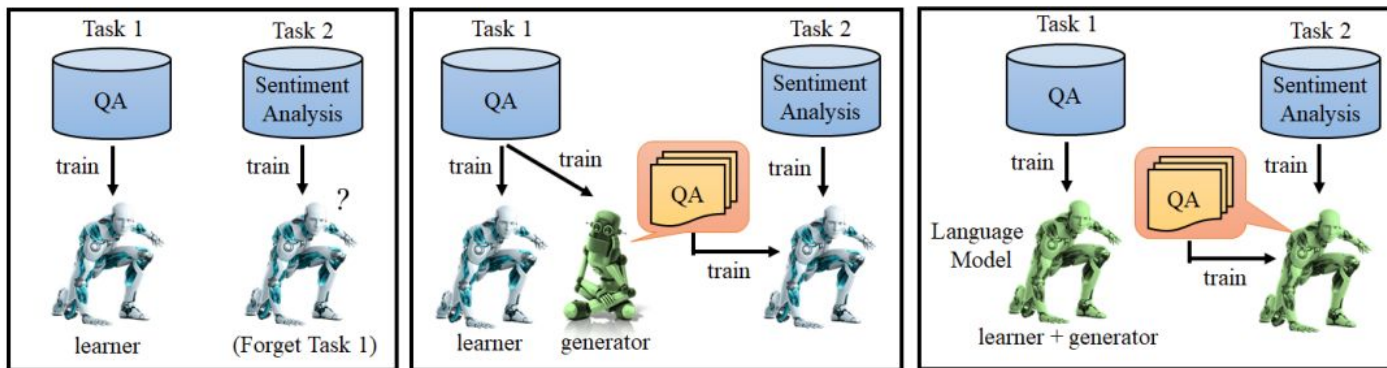



Figure 1: **Left:** After learning Task 2, the learner has already forgotten how to solve Task 1. This is “catastrophic forgetting”. **Middle:** The basic idea of the data-based LLL approach. A generator is learned to generate examples it has seen before. Using the generator, the learner also learns from examples from the previous task to prevent it from forgetting. **Right:** A language model that simultaneously takes on the roles of learner and generator.

- 
- LAMOL learns on gold labels from the dataset
 - Why don't we apply tricks from KD?

$$\mathcal{L}_{\text{Word-KD}}(x; \theta_S; \theta_T) =$$

$$\sum_{t=t_0}^T \sum_{k=1}^{|\mathcal{V}|} -P(\mathcal{V}_k | x_{<t}; \theta_T) \log P(\mathcal{V}_k | x_{<t}; \theta_S),$$

$$\mathcal{L}_{\text{Seq-KD}}(\hat{x}; \theta_S) = \sum_{t=t_0}^T -\log P(\hat{x}_t | \hat{x}_{<t}; \theta_S).$$

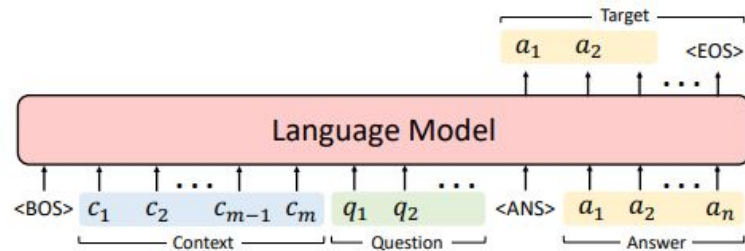
$$\mathcal{L}_{\text{Seq-KD}_{\text{soft}}}(\hat{x}; \theta_S; \theta_T) =$$

$$\sum_{t=t_0}^T \sum_{k=1}^{|\mathcal{V}|} -P(\mathcal{V}_k | \hat{x}_{<t}; \theta_T) \log P(\mathcal{V}_k | \hat{x}_{<t}; \theta_S).$$

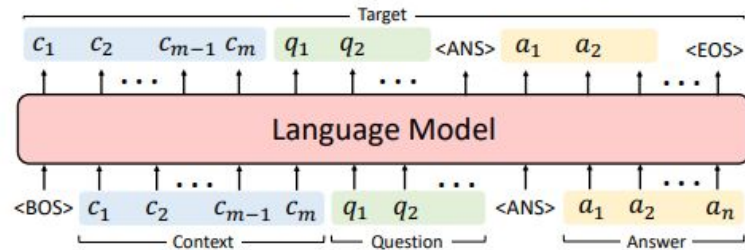
Setting



Assume that we have been trained on tasks $\{D_1, D_2, \dots, D_m\}$




(a) Learning to solve target tasks (QA).



(b) Learning to generate pseudo-data (LM).

Figure 2: Illustration of learning QA and LM in LAMOL.



Given a training sample $X_m^i = \{x_1, x_2, \dots, x_T\} \in D_m$
we minimize

$$\mathcal{L}_{\text{new}}(X_i^m; \theta_S; \theta_T^m) = \mathcal{L}_{\text{new}}^{\text{QA}} + \mathcal{L}_{\text{new}}^{\text{LM}}$$

$$\mathcal{L}_{\text{new}}^{\text{QA}} = \mathcal{L}_{\text{Word-KD}}(X_i^m; \theta_S; \theta_T^m; t_0 = a_1)$$

$$\mathcal{L}_{\text{new}}^{\text{LM}} = \mathcal{L}_{\text{Word-KD}}(X_i^m; \theta_S; \theta_T^m; t_0 = 0),$$

What to do with previous tasks?



We don't have hard labels or a teacher for the previous task.

So, just use NLL loss based on the output of the generators.

$$\begin{aligned}\mathcal{L}_{\text{prev}}(X_i^{\text{prev}}; \theta_S) &= \mathcal{L}_{\text{prev}}^{\text{QA}} + \mathcal{L}_{\text{prev}}^{\text{LM}} \\ \mathcal{L}_{\text{prev}}^{\text{QA}} &= \mathcal{L}_{\text{NLL}}(X_i^{\text{prev}}; \theta_S; t_0 = a_1) \\ \mathcal{L}_{\text{prev}}^{\text{LM}} &= \mathcal{L}_{\text{NLL}}(X_i^{\text{prev}}; \theta_S; t_0 = 0).\end{aligned}$$

Final objective:



$$\theta_S^* = \arg \min_{\theta_S} \left(\sum_{X_i^m \in D_m} \mathcal{L}_{\text{new}} + \sum_{X_i^{\text{prev}} \in D_{\text{prev}}} \mathcal{L}_{\text{prev}} \right)$$

Algorithm 1 L2KD: Lifelong Language Knowledge Distillation

Input: current task dataset D_m , teacher model with parameters θ_T , knowledge distillation loss function \mathcal{L}_{KD} , pseudo-data sample rate γ .

Output: LLL model parameters θ_S .

Optimize teacher model on D_m to get parameters θ_T .

Sample $\gamma \cdot |D_m|$ pseudo-data from θ_S to form D_{prev} .

for all training samples $\{X_i^m\}_{i=1}^n \in D_m$ **do**

for $i = 1$ **to** n **do**

 update θ_S to minimize $\mathcal{L}_{\text{KD}}(X_i^m; \theta_S; \theta_T)$

end for

 Sample $n' = \gamma n$ samples $\{X_j^{\text{prev}}\}_{j=1}^{n'}$ from D_{prev}

for $j = 1$ **to** n' **do**

 update θ_S to minimize $\mathcal{L}_{\text{NLL}}(X_j^{\text{prev}}; \theta_S)$

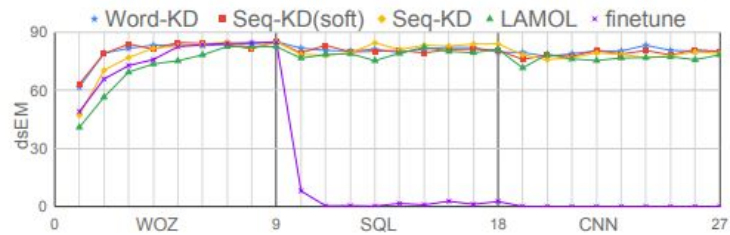
end for

end for

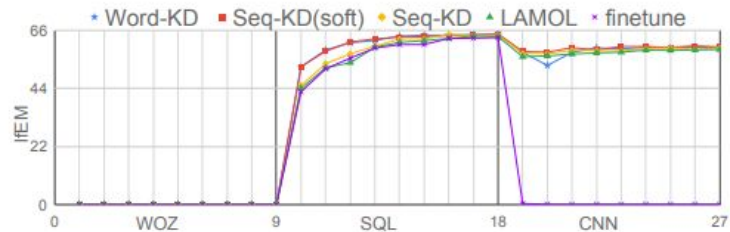
Results:

Lifelong Methods (averaged over six orders)

(a)	Finetune	28.9	19.5	21.7	23.4
(b)	LAMOL	77.7	27.0	60.0	54.9
(c)	(b) + Word-KD	81.9	27.0	61.9	57.0
(d)	(b) + Seq-KD _{soft}	82.6	26.9	61.7	57.1
(e)	(b) + Seq-KD	80.9	28.0	60.6	56.5



(a) Learning curve of WOZ.



(b) Learning curve of SQL.




(c) Learning curve of CNN.

Figure 3: The learning curves of different LLL methods in the order of WOZ \rightarrow SQL \rightarrow CNN.

Are we copying everything from the teacher?



- Split entire data into 2 parts A, and B.
- A consists of all classification tasks where teacher is correct
- B consists of all classification tasks where teacher is incorrect.
- What does the model do after training is over?



	Acc	Acc in (A)	Acc in (B)
Teacher	76.73	100.00	0.00
LAMOL	75.48	88.15	33.69
+ Word-KD	77.11	90.26 (+2.11)	33.75 (+0.06)
+ Seq-KD _{soft}	76.42	89.42 (+1.27)	33.52 (-0.17)
+ Seq-KD	76.56	89.56 (+1.41)	33.69 (+0.00)

Table 6: The accuracy in the group (A) and (B) averaged over five classification datasets. The teacher scores are from five single-task models.

- Clearly, the model isn't copying the teacher
- Can integrate knowledge from other tasks and avoid false knowledge.

Everything good about this paper...



- First attempt at using KD for LLL.
- Beats LAMOL on accuracy and std. deviation across different task permutations.
- The last experiment is very insightful. Proves that model isn't copying from the teachers.

Weaknesses of the paper

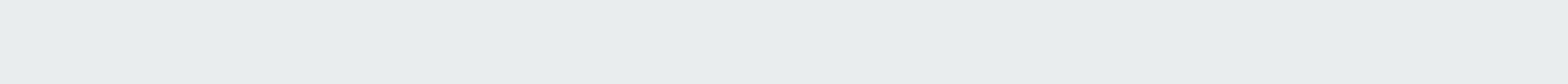


- Unable to perform at par with multi-task baseline - Rohit
- Resources need to be spent on a disposable teacher model - Rocktim, Jai, Shreya
- Not enough tasks(at most 5) - Jai, Seshank, Harman, Shreya
- Datasets are severely under-sampled which reduces noise making it easy for student to learn - Vishal [Not clear to me]
- Real world data may not have data boundaries, or sudden shifts in distribution - Harman

Extensions



- Multimodal data - Harman, Shivangi
- Multilingual data - Rocktim
- Equal weightage given to new and previous loss; Experiment with multiple loss functions - Rohit [needs clarification]
-



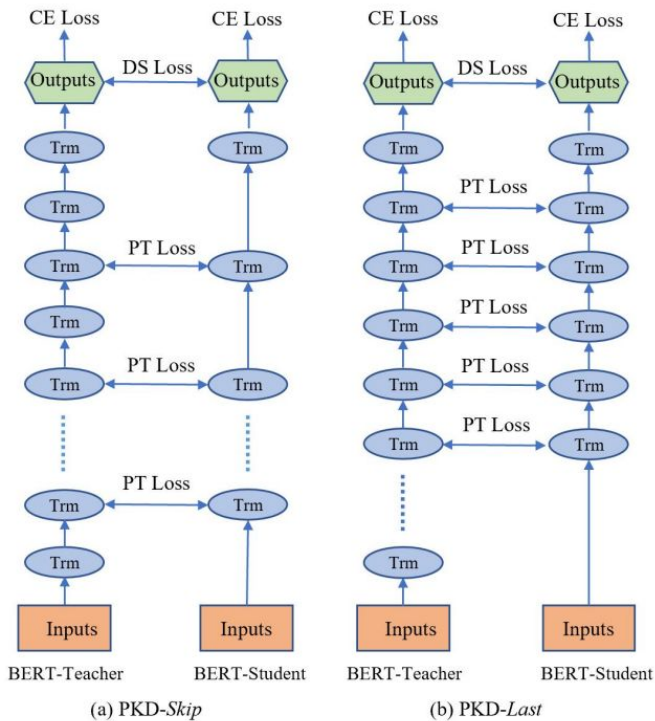
Patient Knowledge Distillation for BERT Model Compression

-Sun et. al, EMNLP '19

- Knowledge also exists in **intermediate layer** outputs of a model[CLS emb. here].
- Why just use final output predictions?
- Minimize MSE between output of intermediate layers
 - Which layers to minimize error on?
 - PKD-Skip & PKD-Last

Patient Knowledge Distillation for BERT Model Compression

-Sun et. al, EMNLP '19



$$L_{DS} = - \sum_{i \in [N]} \sum_{c \in C} \left[P^t(\mathbf{y}_i = c | \mathbf{x}_i; \hat{\theta}^t) \cdot \log P^s(\mathbf{y}_i = c | \mathbf{x}_i; \theta^s) \right]$$


$$L_{CE}^s = - \sum_{i \in [N]} \sum_{c \in C} \left[\mathbb{1}[\mathbf{y}_i = c] \cdot \log P^s(\mathbf{y}_i = c | \mathbf{x}_i; \theta^s) \right]$$

$$L_{PT} = \sum_{i=1}^N \sum_{j=1}^M \left\| \frac{\mathbf{h}_{i,j}^s}{\|\mathbf{h}_{i,j}^s\|_2} - \frac{\mathbf{h}_{i,I_{pt}(j)}^t}{\|\mathbf{h}_{i,I_{pt}(j)}^t\|_2} \right\|_2^2$$

$$L_{PKD} = (1 - \alpha)L_{CE}^s + \alpha L_{DS} + \beta L_{PT}$$

Model	SST-2 (67k)	MRPC (3.7k)	QQP (364k)	MNLI-m (393k)	MNLI-mm (393k)	QNLI (105k)	RTE (2.5k)
BERT ₁₂ (Google)	93.5	88.9/84.8	71.2/89.2	84.6	83.4	90.5	66.4
BERT ₁₂ (Teacher)	94.3	89.2/85.2	70.9/89.0	83.7	82.8	90.4	69.1
BERT ₆ -FT	90.7	85.9/80.2	69.2/88.2	80.4	79.7	86.7	63.6
BERT ₆ -KD	91.5	86.2/80.6	70.1/88.8	80.2	79.8	88.3	64.7
BERT ₆ -PKD	92.0	85.0/79.9	70.7/88.9	81.5	81.0	89.0	65.5
BERT ₃ -FT	86.4	80.5/ 72.6	65.8/86.9	74.8	74.3	84.3	55.2
BERT ₃ -KD	86.9	79.5/71.1	67.3/87.6	75.4	74.8	84.0	56.2
BERT ₃ -PKD	87.5	80.7/72.5	68.1/87.8	76.7	76.3	84.7	58.2

*PKD-Skip has been used here



Model	SST-2	MRPC	QQP	MNLI-m	MNLI-mm	QNLI	RTE
BERT ₆ (PKD-Last)	91.9	85.1/79.5	70.5/ 88.9	80.9	81.0	88.2	65.0
BERT ₆ (PKD-Skip)	92.0	85.0/ 79.9	70.7/88.9	81.5	81.0	89.0	65.5

Table: Comparing PKD-Last and PKD-Skip

- Skip probably works better since it “captures more diverse representations of richer semantics from low-level to high-level”

Pushing it even further...



- TinyBERT - Jiao et. al, EMNLP Findings '20
- TinyBERT₄ achieves 96.8% performance on GLUE, and is 7.5x smaller, and 9.4x faster.
- TinyBERT₆ performs as good as BERT_{BASE}

Some interesting questions




- Can distillation achieve performance at par with original teacher model?
- Does reduction in size, always lead to worse results?

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks

Frankle et. al, ICLR '19

- Deep Neural Networks are very over-parameterized.
- Can we reduce size without reducing performance?



“A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations. ”

How to find such subnetworks?



Iterative Magnitude Pruning(IMP)

1. Randomly initialize a neural network $f(x; \theta_0)$ (where $\theta_0 \sim \mathcal{D}_\theta$).
2. Train the network for j iterations, arriving at parameters θ_j .
3. Prune $p\%$ of the parameters in θ_j , creating a mask m .
4. Reset the remaining parameters to their values in θ_0 , creating the winning ticket $f(x; m \odot \theta_0)$.

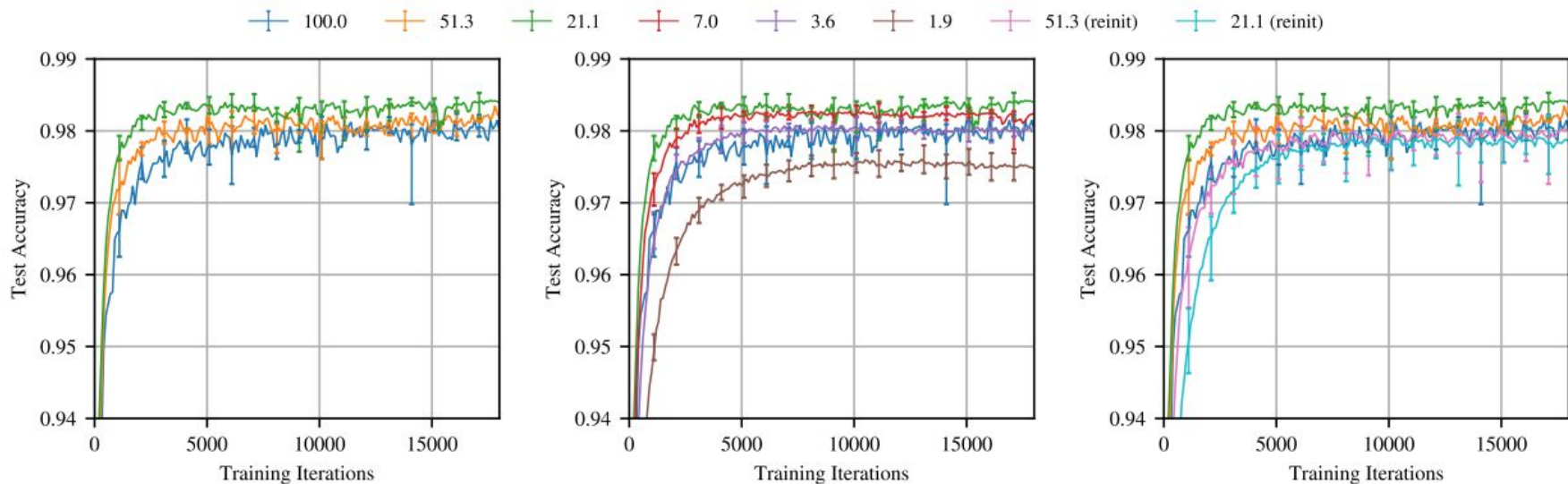



Figure 3: Test accuracy on Lenet (iterative pruning) as training proceeds. Each curve is the average of five trials. Labels are P_m —the fraction of weights remaining in the network after pruning. Error bars are the minimum and maximum of any trial.

- 
- Able to successfully find tickets which are **10-20% size** of the original network.
 - These networks have **at-least as high test accuracies** as the original networks.
 - These networks **converge in comparable number of iterations**.
 - When initialized randomly, perform far worse.
 - Therefore, **initialization is also as important**.
 - When trained on random structure, perform far worse
 - Therefore, **structure is also important**
 - Verified on ResNet-18, VGG-19, LeNet

Does BERT win the lottery?





The Lottery Ticket Hypothesis for Pre-trained BERT Networks

Chen et. al, NeurIPS '20

- Consider the BERT model which has already been pre-trained.
- This is the initialization.
- Are there winning tickets for downstream tasks?
- Can the winning ticket for task 1 be used for task 2 as well?[transferability]



Dataset	MNLI	QQP	STS-B	WNLI	QNLI	MRPC	RTE	SST-2	CoLA	SQuAD	MLM
Sparsity	70%	90%	50%	90%	70%	50%	60%	60%	50%	40%	70%
Full BERT _{BASE}	82.4 ± 0.5	90.2 ± 0.5	88.4 ± 0.3	54.9 ± 1.2	89.1 ± 1.0	85.2 ± 0.1	66.2 ± 3.6	92.1 ± 0.1	54.5 ± 0.4	88.1 ± 0.6	63.5 ± 0.1
$f(x, m_{\text{IMP}} \odot \theta_0)$	82.6 ± 0.2	90.0 ± 0.2	88.2 ± 0.2	54.9 ± 1.2	88.9 ± 0.4	84.9 ± 0.4	66.0 ± 2.4	91.9 ± 0.5	53.8 ± 0.9	87.7 ± 0.5	63.2 ± 0.3

Do winning tickets transfer?



Some important considerations:

- At what sparsity should we evaluate the subnetworks?
 - A network with 40% sparsity might outperform another with 90% just because of higher number of parameters
 - Maintain same sparsity for all settings(70% in the paper)

Setting



1. Find a subnetwork using IMP on source task S .
2. Use this subnetwork to train on another task T , using a new randomly initialized classification layer.
3. This performance is $\text{TRANSFER}(S, T)$.

Subnetworks on the Source Tasks (Sparsity %)	MNLI (70%)	0.00	-0.75	-2.57	-5.99	-1.55	-3.44	2.38	0.84	-27.70	-3.46	-5.63	2
	QQP (70%)	-1.69	0.00	-3.16	-1.76	-1.62	-3.27	1.80	-1.19	-22.39	-4.80	-5.52	1
	STS-B (70%)	-2.51	-1.69	0.00	2.47	-2.72	-2.86	-0.97	-2.07	-34.58	-5.62	-5.57	1
	WNLI (70%)	-2.86	-2.42	-20.21	0.00	-3.93	-5.67	-3.13	-2.72	-38.89	-6.06	-5.40	0
	QNLI (70%)	-1.76	-1.20	-4.18	1.06	0.00	-3.84	0.60	-0.42	-35.24	-3.93	-5.68	2
	MRPC (70%)	-2.58	-2.07	-6.09	2.47	-3.23	0.00	-3.50	-1.84	-31.41	-6.47	-5.41	1
	RTE (70%)	-2.38	-1.77	-7.84	2.00	-2.40	-4.01	0.00	-1.84	-37.34	-5.60	-5.37	1
	SST-2 (70%)	-2.42	-1.50	-9.73	0.12	-3.12	-4.90	-1.45	0.00	-31.36	-5.31	-5.39	1
	CoLA (70%)	-2.50	-1.65	-9.86	1.06	-2.59	-4.74	-2.77	-1.42	0.00	-5.36	-5.34	1
	SQuAD v1.1 (70%)	-1.66	-1.05	-3.25	0.12	0.51	-3.52	1.56	0.19	-30.86	0.00	-5.69	4
	(IMP) MLM (70%)	0.02	0.09	0.09	1.18	0.55	6.01	1.44	1.87	8.27	0.17	0.00	10
	Pruning θ_0 (70%)	-0.10	-0.33	-2.06	-0.35	0.24	-3.02	0.47	1.07	-6.68	-1.04	-6.07	3
		MNLI	QQP	STS-B	WNLI	QNLI	MRPC	RTE	SST-2	CoLA	SQuAD v1.1	MLM	
	Transfer Tasks												


Figure 3: Transfer vs. Same-Task. The performance of transferring IMP subnetworks between tasks. Each row is a source task \mathcal{S} . Each column is a target task \mathcal{T} . Each cell is $\text{TRANSFER}(\mathcal{S}, \mathcal{T}) - \text{TRANSFER}(\mathcal{T}, \mathcal{T})$: the transfer performance at 70% sparsity minus the same-task performance at 70% sparsity (averaged over three runs). **Dark cells mean the IMP subnetwork found on task \mathcal{S} performs at least as well on task \mathcal{T} as the subnetwork found on task \mathcal{T} .**

Observations:



- MLM subnetworks have the best transfer performance
- This is obvious since MLM was used for creating the initialization.
- Better transfer is seen where source task datasets are large in size

Conclusion

- 
- Knowledge Distillation
 - Standard Methods for KD in NLP
 - Applications(LLL)
 - Lottery Ticket Hypothesis
 - LTH for pre-trained BERT networks



Thank you!

References:

- Distilling the Knowledge in a Neural Network - Hinton et. al
- Multilingual Neural Machine Translation With Knowledge Distillation - Tan et. al
- Patient Knowledge Distillation for BERT Model Compression - Sun et. al
- TinyBERT: Distilling BERT for Natural Language Understanding - Jiao et. al
- Lifelong Language Knowledge Distillation - Chuang et. al
- The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks - Frankle et. al
- The Lottery Ticket Hypothesis for Pre-trained BERT Networks - Chen et. al