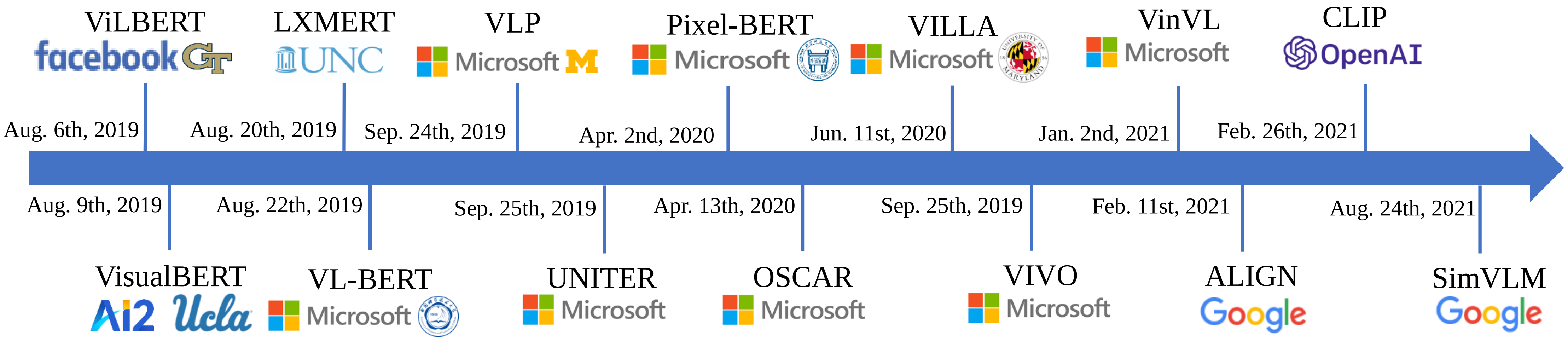
Playing Lottery Tickets with Vision and Language

Zhe Gan, Yen-Chun Chen, Linjie Li, Tianlong Chen,
Yu Cheng, Shuohang Wang, Jingjing Liu, Lijuan Wang, Zicheng Liu

Microsoft
TEXAS The University of Texas at Austin
清華大學 Tsinghua University

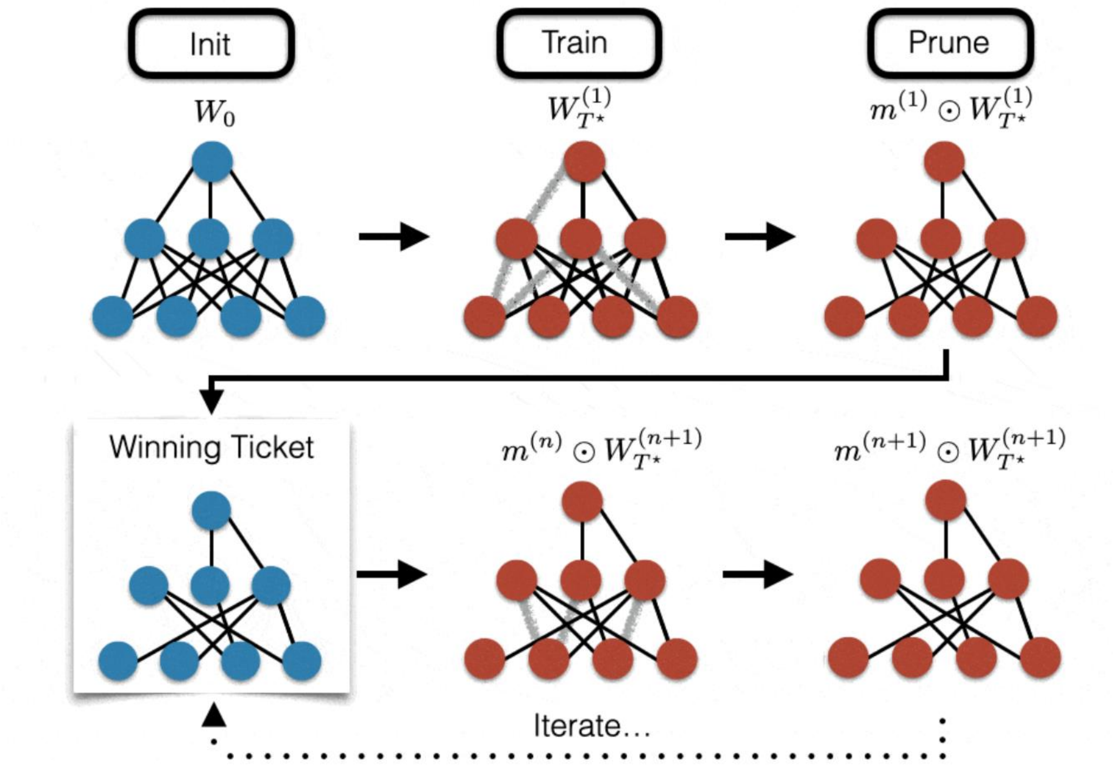
Vision-Language Pre-training (VLP)

- VLP has achieved great success; however, the large number of parameters in such models hinder their application in practice
- *Model efficiency:* Can we prune a large pre-trained VL model while preserving its performance and transferability?



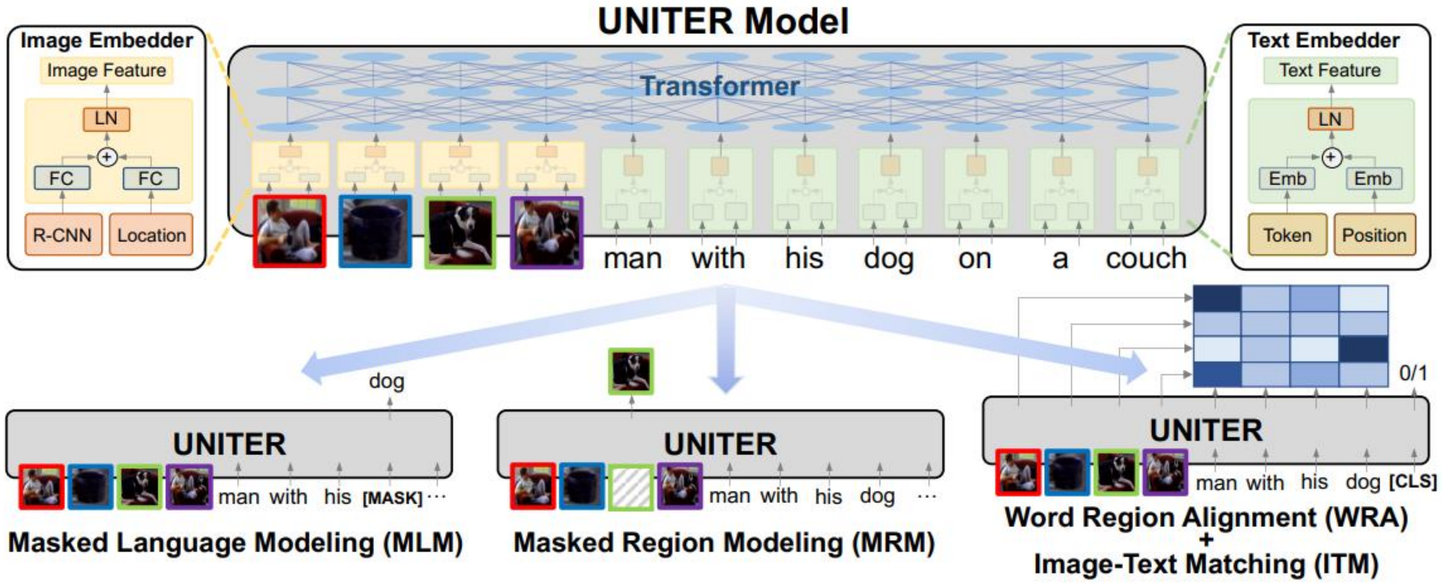
Lottery Ticket Hypothesis (LTH)

- We aim to answer this question via the lens of lottery ticket hypothesis, which states that deep neural networks contain small matching subnetworks that can achieve on par or even better performance than the dense networks when trained in isolation
- Winning tickets are typically found via unstructured Iterative Magnitude Pruning (IMP)
- LTH has been extensively studied for image classification, and recently been introduced to NLP, GAN, GNN etc.



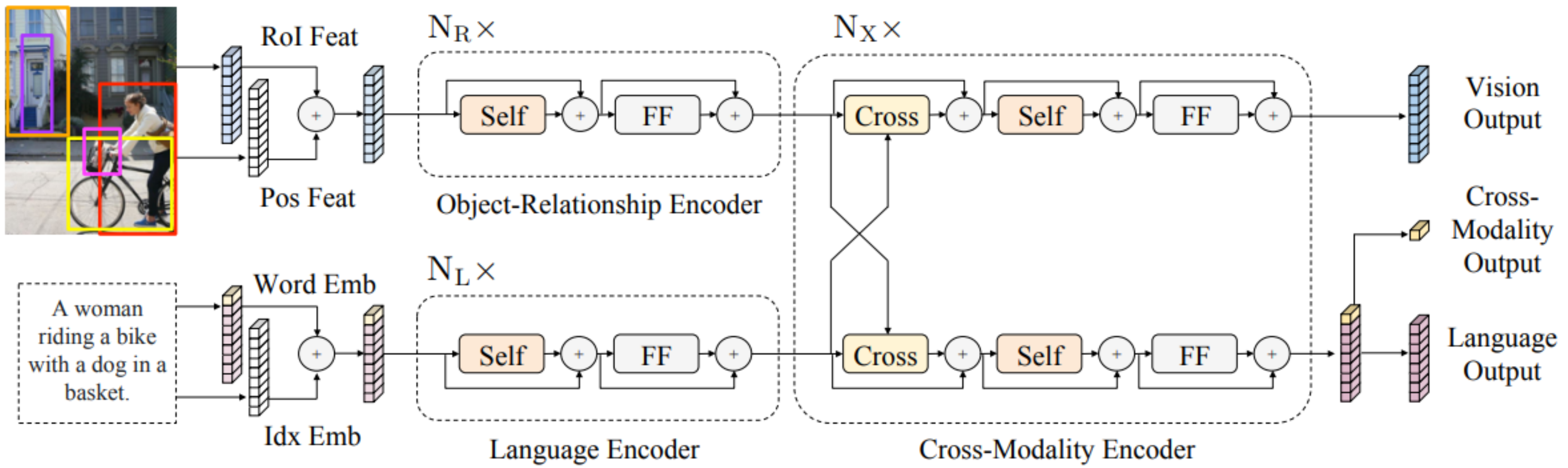
Playing Lottery Tickets with Vision and Language

- LTH has not been introduced to VL tasks yet, it could be a powerful tool to understand the parameter redundancy in the current prevailing VLP models
- To start, we focus on UNITER, and then extend our analysis to LXMERT and ViLT

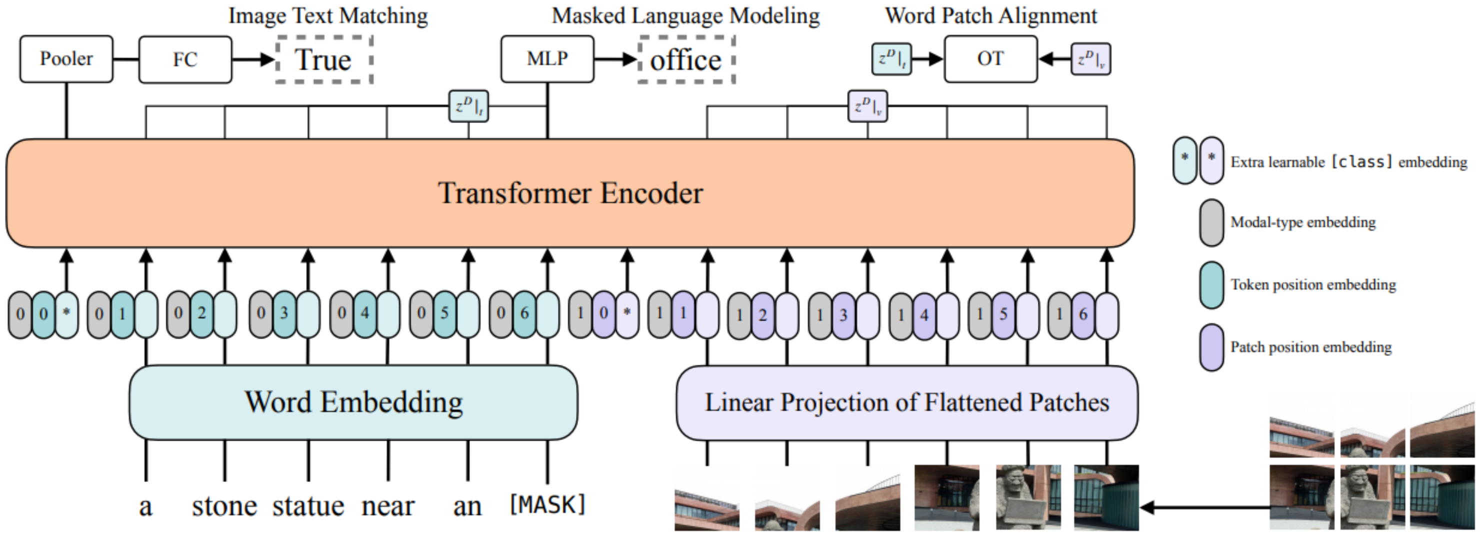


Playing Lottery Tickets with Vision and Language

LXMERT

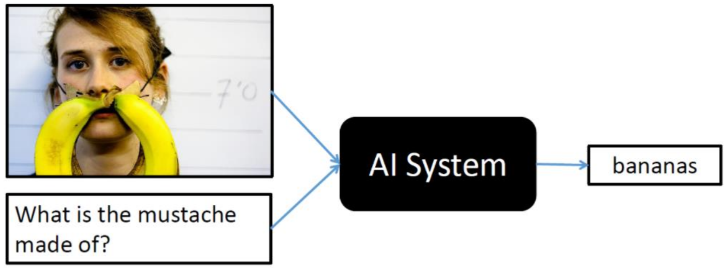


ViLT



Playing Lottery Tickets with Vision and Language

- Downstream tasks: VQA, VCR, GQA, NLVR2, visual entailment, referring expression comprehension, and image-text retrieval

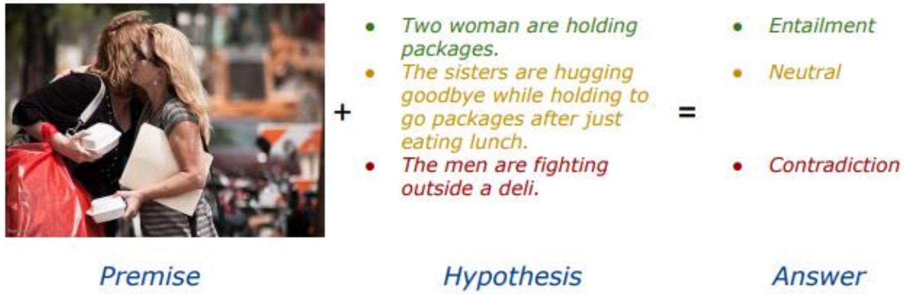


VCR
VISUAL COMMONSENSE REASONING

GQA

Referring Expressions


The left image contains twice the number of dogs as the right image, and at least two dogs in total are standing.



Caption-Based Image Retrieval

Questions We Aim to Answer

- *Existence*: Can we draw winning tickets successfully for various VL tasks?
  - Use pre-trained weights as model initialization for task-specific finetuning
  - Use IMP to draw tickets for each VL task
- *Transferability*: Can we find tickets that transfer universally to all VL tasks?
  - Perform IMP on the pre-training tasks using the pre-training data
  - Analyze the transfer behavior among all the tasks
- *Compatibility*: Do the LTH observations on UNITER still hold when switching to different backbones (e.g., LXMERT, ViLT), and training strategies (e.g., adversarial training)?

Playing Lottery Tickets with Vision and Language

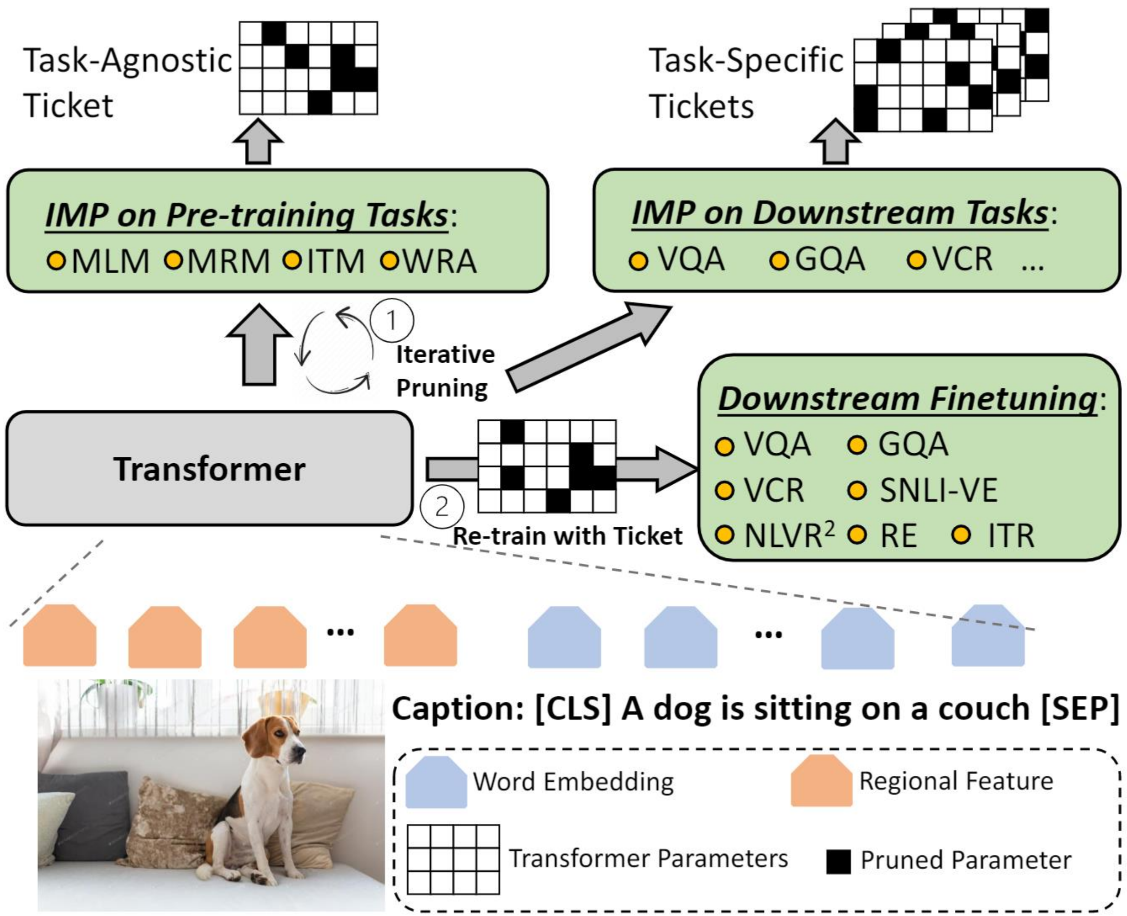


Figure 1: Overview of our training paradigm for *playing lottery tickets with vision and language*. Matching subnetworks (or winning tickets) can be found by Iterative Magnitude-based Pruning (IMP). We then re-train the found ticket with the original parameter initialization to verify the downstream performance. Not only *task-specific* winning tickets can be found when running IMP on each downstream task separately, a *task-agnostic* winning ticket is also discovered via IMP on joint pre-training. The task-agnostic ticket results in *universally transferable* subnetworks at 60%/70% sparsity that matches 98%/96% of the full accuracy averaged over all the tasks considered.

Our Empirical Findings

- *VLP can play lottery tickets too*: “Relaxed” winning tickets that match 99% of the full accuracy can be found at 50%-70% sparsity across all the VL tasks
- *One ticket to win them all*: Matching subnetworks found via IMP on pre-training tasks transfer universally. Interestingly, matching subnetworks found on each downstream task also transfer to other tasks well
- *Different VLP models behave differently*: The highest sparsity we can achieve for ViLT is far lower than LXMERT and UNITER (30% vs. 70%)
- *Playing lottery tickets adversarially*: Sparse winning tickets can also be identified with adversarial training, with enhanced performance

VLP Can Play Lottery Tickets Too

- *Q1: Are there winning tickets in UNITER?*

#	Dataset	VQA mini-dev[†]	GQA test-dev	VCR Q→AR val	NLVR[2] dev	SNLI-VE val	RefCOCO+ val[d]	Flickr30k IR R@1	Flickr30k TR R@1
#	Sparsity	70%	70%	50%	60%	60%	70%	60%	60%
1	UNITER_B (paper)	70.75	–	54.94	77.18	78.59	75.31	72.52	85.90
2	UNITER_B (reimp.)	$70.64\pm_{0.06}$	$59.64\pm_{0.15}$	$54.37\pm_{0.31}{}^{\ddagger}$	$76.75\pm_{0.19}$	$78.47\pm_{0.10}$	$74.73\pm_{0.06}$	$71.25\pm_{0.11}{}^{\star}$	$84.63\pm_{1.02}{}^{\star}$
3	$\times$99%	69.93	59.04	53.83	75.98	77.69	73.98	70.54	83.78
4	$f(\boldsymbol{x}; \boldsymbol{m}_{\text{IMP}} \cdot \boldsymbol{\theta}_0)$	$69.98\pm_{0.05}$	$59.26\pm_{0.09}$	$53.15\pm_{1.02}$	$76.32\pm_{0.41}$	$77.69\pm_{0.07}$	$74.06\pm_{0.27}$	$70.15\pm_{0.71}$	$83.77\pm_{0.76}$
5	$f(\boldsymbol{x}; \boldsymbol{m}_{\text{RP}} \cdot \boldsymbol{\theta}_0)$	60.45	55.95	25.35	52.42	71.30	72.95	61.44	76.80
6	$f(\boldsymbol{x}; \boldsymbol{m}_{\text{IMP}} \cdot \boldsymbol{\theta}'_0)$	67.98	58.45	50.39	54.15	76.45	71.09	63.38	79.30
7	$f(\boldsymbol{x}; \boldsymbol{m}_{\text{IMP}} \cdot \boldsymbol{\theta}''_0)$	60.46	47.49	6.25	51.52	69.32	67.34	38.94	48.00

Table 1: Performance of subnetworks at the highest sparsity for which IMP finds "relaxed" winning tickets that maintains 99% of the full accuracy on each task. Entries with $\pm$ are the average across three runs. IMP: Iterative Magnitude Pruning; RP: Random Pruning; $\boldsymbol{\theta}_0$: pre-trained UNITER weights; $\boldsymbol{\theta}'_0$: pre-trained BERT weights; $\boldsymbol{\theta}''_0$: randomly shuffled pre-trained UNITER weights. (†) To avoid submitting results to the VQA test server too frequently, instead of reporting results on test-dev/-std sets, we use a mini-dev set for comparison. The same min-dev set was also used in UNITER. (‡) For fair comparison on transfer learning, we did not perform 2-nd stage pre-training for VCR task as in UNITER. ($\star$) To rule out other factors that may influence results besides pruning, we did not use hard negative mining as in UNITER.

VLP Can Play Lottery Tickets Too

- *Q1: Are there winning tickets in UNITER?*

#	Dataset	VQA mini-dev[†]	GQA test-dev	VCR Q→AR val	NLVR[2] dev	SNLI-VE val	RefCOCO+ val[d]	Flickr30k IR R@1	Flickr30k TR R@1
	Sparsity	70%	70%	50%	60%	60%	70%	60%	60%
1	UNITER$_B$ (paper)	70.75	–	54.94	77.18	78.59	75.31	72.52	85.90
2	UNITER$_B$ (reimp.)	70.64±0.06	59.64±0.15	54.37±0.31[‡]	76.75±0.19	78.47±0.10	74.73±0.06	71.25±0.11[⋆]	84.63±1.02[⋆]
3	×99%	69.93	59.04	53.83	75.98	77.69	73.98	70.54	83.78
4	$f(\boldsymbol{x}; \boldsymbol{m}_{\text{IMP}} \cdot \boldsymbol{\theta}_0)$	69.98±0.05	59.26±0.09	53.15±1.02	76.32±0.41	77.69±0.07	74.06±0.27	70.15±0.71	83.77±0.76
5	$f(\boldsymbol{x}; \boldsymbol{m}_{\text{RP}} \cdot \boldsymbol{\theta}_0)$	60.45	55.95	25.35	52.42	71.30	72.95	61.44	76.80
6	$f(\boldsymbol{x}; \boldsymbol{m}_{\text{IMP}} \cdot \boldsymbol{\theta}'_0)$	67.98	58.45	50.39	54.15	76.45	71.09	63.38	79.30
7	$f(\boldsymbol{x}; \boldsymbol{m}_{\text{IMP}} \cdot \boldsymbol{\theta}''_0)$	60.46	47.49	6.25	51.52	69.32	67.34	38.94	48.00

Table 1: Performance of subnetworks at the highest sparsity for which IMP finds "relaxed" winning tickets that maintains 99% of the full accuracy on each task. Entries with ± are the average across three runs. IMP: Iterative Magnitude Pruning; RP: Random Pruning; $\boldsymbol{\theta}_0$: pre-trained UNITER weights; $\boldsymbol{\theta}'_0$: pre-trained BERT weights; $\boldsymbol{\theta}''_0$: randomly shuffled pre-trained UNITER weights. (†) To avoid submitting results to the VQA test server too frequently, instead of reporting results on test-dev/-std sets, we use a mini-dev set for comparison. The same min-dev set was also used in UNITER. (‡) For fair comparison on transfer learning, we did not perform 2-nd stage pre-training for VCR task as in UNITER. (⋆) To rule out other factors that may influence results besides pruning, we did not use hard negative mining as in UNITER.

VLP Can Play Lottery Tickets Too

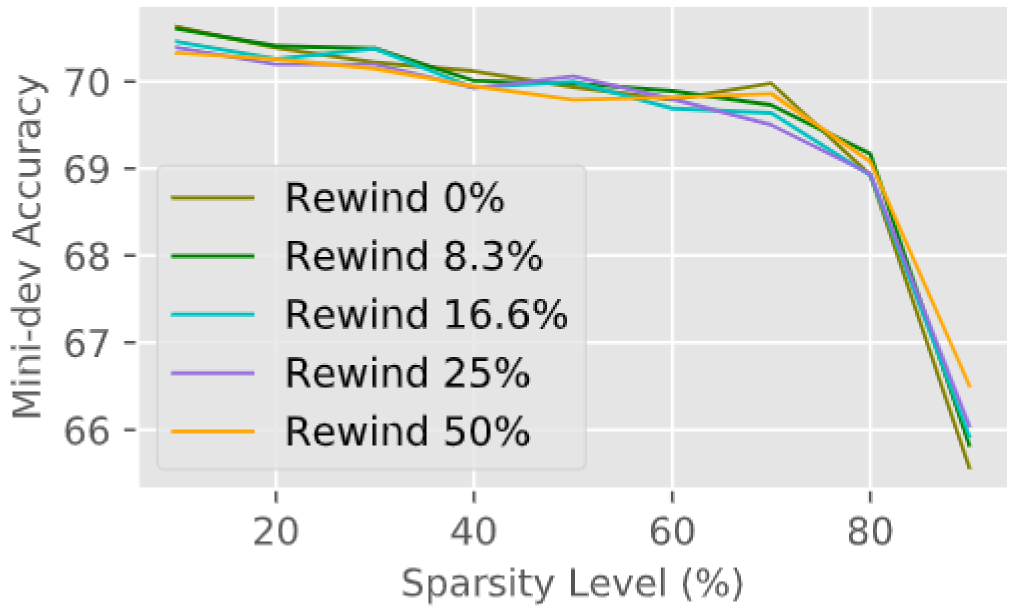
- *Q2: Are winning tickets sparser than randomly pruned or initialized subnetworks?*

#	Dataset / Sparsity	VQA mini-dev†	GQA test-dev	VCR Q→AR val	NLVR² dev	SNLI-VE val	RefCOCO+ val^d	Flickr30k IR R@1	Flickr30k TR R@1
#	Sparsity	70%	70%	50%	60%	60%	70%	60%	60%
1	UNITER$_B$ (paper)	70.75	–	54.94	77.18	78.59	75.31	72.52	85.90
2	UNITER$_B$ (reimp.)	70.64±0.06	59.64±0.15	54.37±0.31‡	76.75±0.19	78.47±0.10	74.73±0.06	71.25±0.11⋆	84.63±1.02⋆
3	×99%	69.93	59.04	53.83	75.98	77.69	73.98	70.54	83.78
4	$f(\boldsymbol{x}; \boldsymbol{m}_{\mathrm{IMP}} \cdot \boldsymbol{\theta}_0)$	69.98±0.05	59.26±0.09	53.15±1.02	76.32±0.41	77.69±0.07	74.06±0.27	70.15±0.71	83.77±0.76
5	$f(\boldsymbol{x}; \boldsymbol{m}_{\mathrm{RP}} \cdot \boldsymbol{\theta}_0)$	60.45	55.95	25.35	52.42	71.30	72.95	61.44	76.80
6	$f(\boldsymbol{x}; \boldsymbol{m}_{\mathrm{IMP}} \cdot \boldsymbol{\theta}'_0)$	67.98	58.45	50.39	54.15	76.45	71.09	63.38	79.30
7	$f(\boldsymbol{x}; \boldsymbol{m}_{\mathrm{IMP}} \cdot \boldsymbol{\theta}''_0)$	60.46	47.49	6.25	51.52	69.32	67.34	38.94	48.00

Table 1: Performance of subnetworks at the highest sparsity for which IMP finds "relaxed" winning tickets that maintains 99% of the full accuracy on each task. Entries with ± are the average across three runs. IMP: Iterative Magnitude Pruning; RP: Random Pruning; $\boldsymbol{\theta}_0$: pre-trained UNITER weights; $\boldsymbol{\theta}'_0$: pre-trained BERT weights; $\boldsymbol{\theta}''_0$: randomly shuffled pre-trained UNITER weights. (†) To avoid submitting results to the VQA test server too frequently, instead of reporting results on test-dev/-std sets, we use a mini-dev set for comparison. The same min-dev set was also used in UNITER. (‡) For fair comparison on transfer learning, we did not perform 2-nd stage pre-training for VCR task as in UNITER. (⋆) To rule out other factors that may influence results besides pruning, we did not use hard negative mining as in UNITER.

VLP Can Play Lottery Tickets Too

- *Q3: Does rewinding improve performance?*

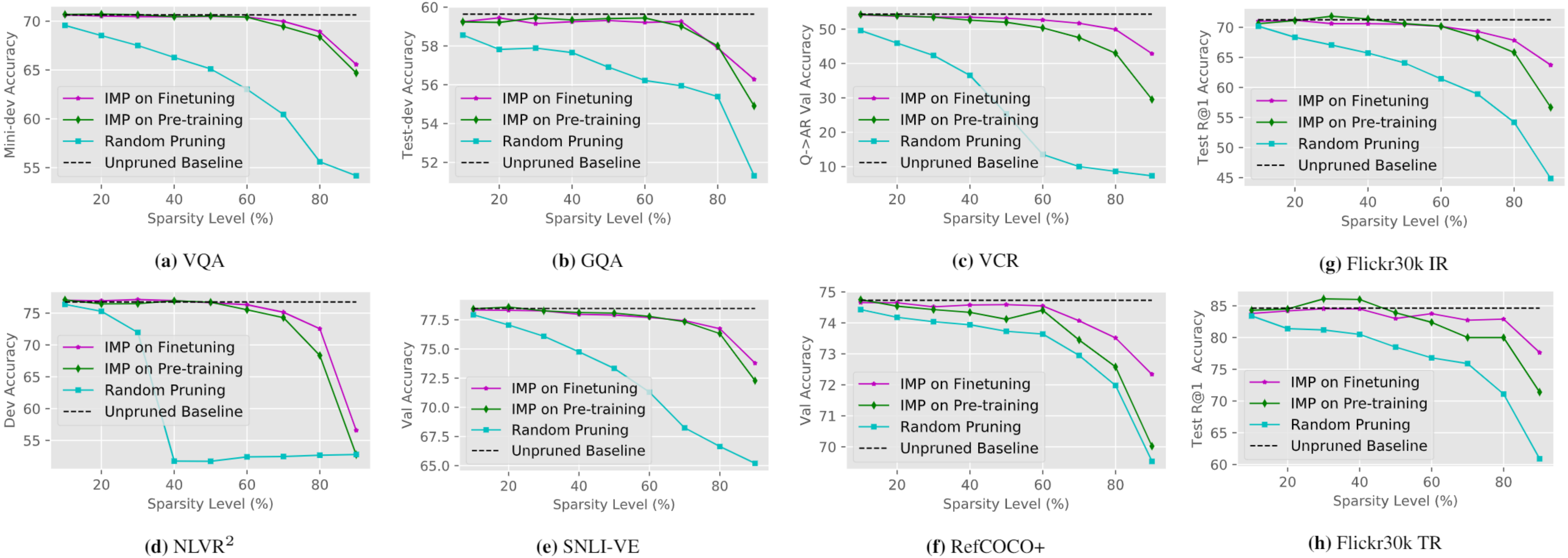


(i) VQA Rewinding

After obtaining the masks, instead of resetting the weights to θ_0, one should rewind the weights to θ_i, the weights after i steps of training

One Ticket To Win Them All

- *Q4: Do winning tickets found on pre-training tasks transfer?*

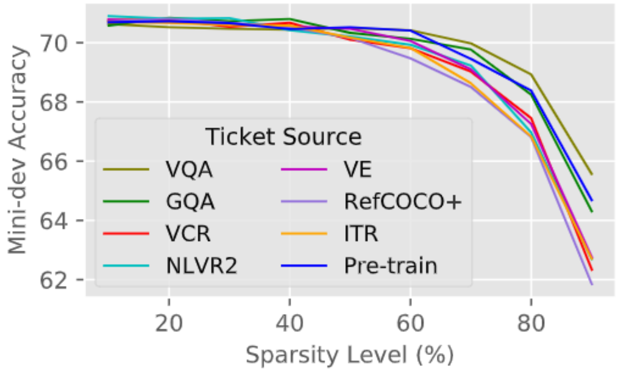


(a) VQA **(b)** GQA **(c)** VCR **(g)** Flickr30k IR

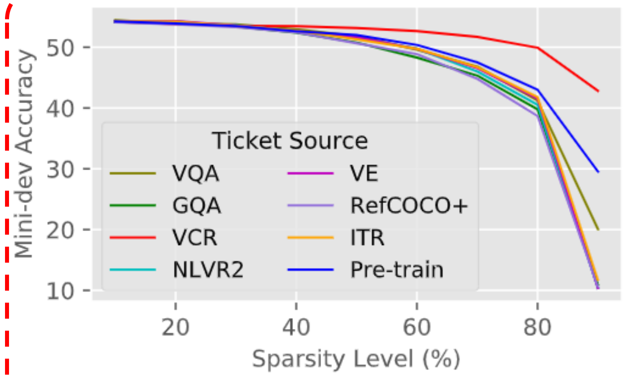
(d) NLVR^2 **(e)** SNLI-VE **(f)** RefCOCO+ **(h)** Flickr30k TR

One Ticket To Win Them All

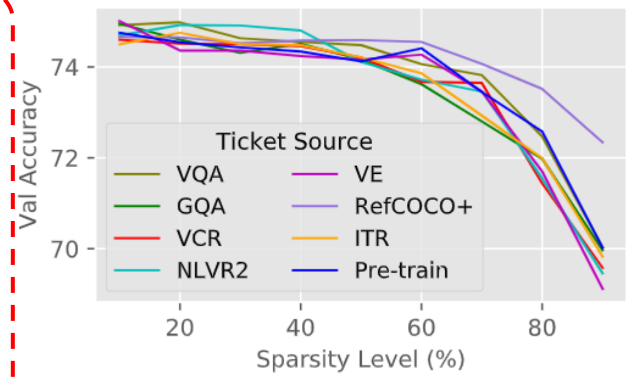
- *Q5: Do winning tickets found on downstream tasks transfer?*



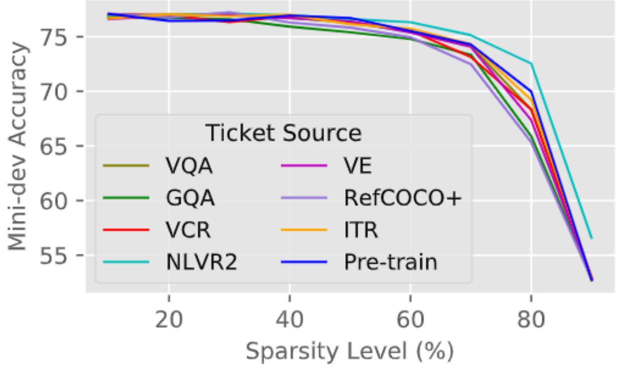
(a) VQA



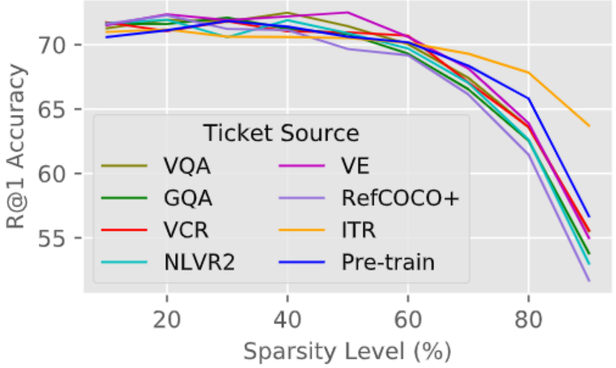
(b) VCR



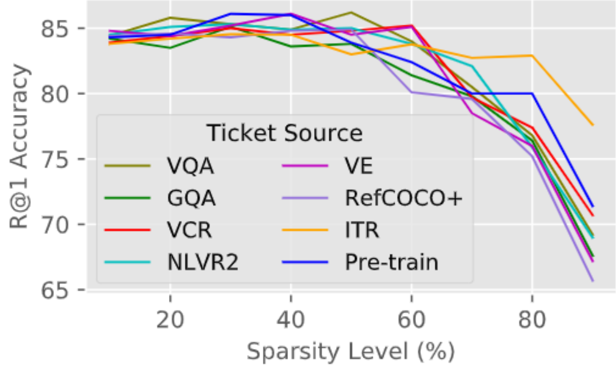
(c) RefCOCO+



(d) NLVR^2



(e) Flickr30k IR



(f) Flickr30k TR

One Ticket To Win Them All

- The universal subnetwork at 60%/70% sparsity matches 98%/96% of the full accuracy over all the tasks, effectively serving as a task-agnostic compressed UNITER model.
- This number changes to 99%/97% if the VCR task is not counted in.

Sparisty	VQA	GQA	VCR	NLVR^2	SNLI-VE	RefCOCO+	Flickr30k IR	Flickr30k TR	Ave. Perf. Drop (%)	
	mini-dev	test-dev	Q→AR val	dev	val	val^d	R@1	R@1	All	w/o VCR
0%	70.64	59.64	54.37	76.75	78.47	74.73	71.25	84.63	–	–
50%	70.52	59.41	52.01	76.71	78.08	74.12	70.62	83.90	1.00	0.52
60%	70.41	59.44	50.37	75.52	77.79	74.41	70.18	82.40	1.88	1.10
70%	69.45	59.02	47.52	74.29	77.34	73.45	68.36	80.00	3.90	2.66
80%	68.38	58.01	42.99	69.98	76.32	72.58	65.82	80.00	6.80	4.78

Table 2: Performance of the universal transferable subnetwork found on pre-training at specified sparsities.

Intriguing Properties of the Found Masks

- Mask similarity: $\frac{m_i \bigcap m_j}{m_i \bigcup m_j}\%$ *no clear patterns* in the similarity of learned masks

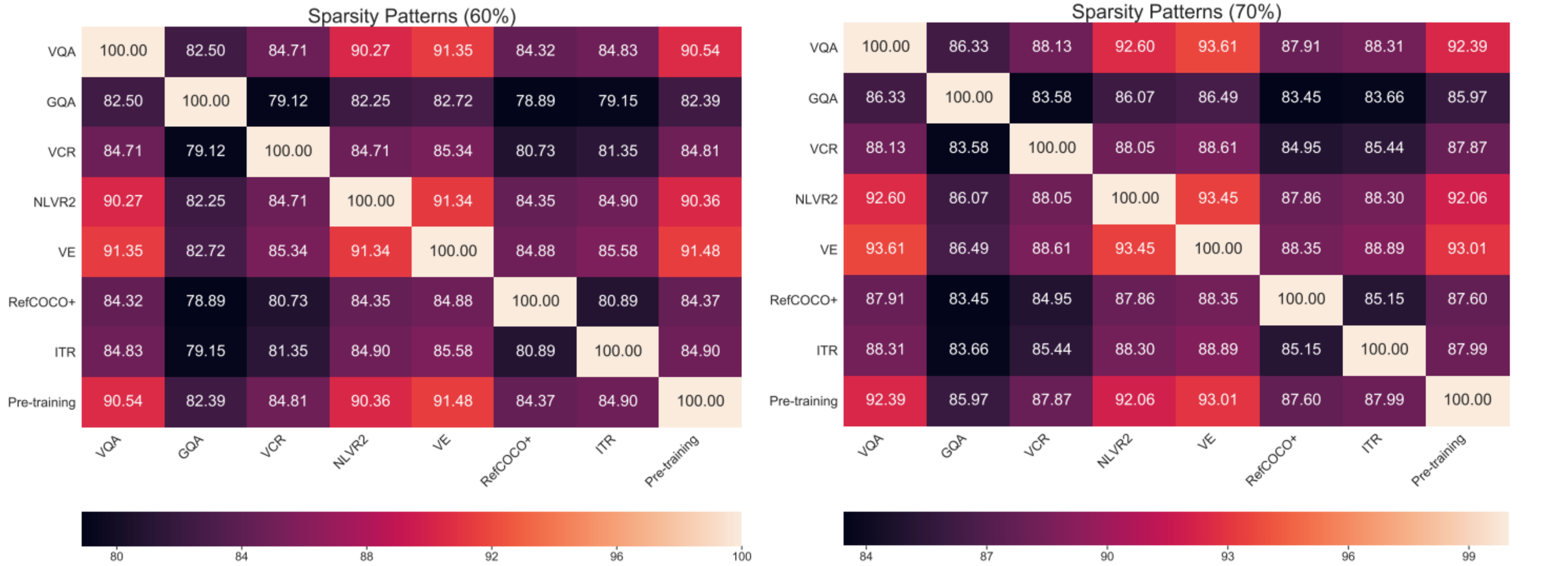


Figure 8: The overlap in sparsity patterns found on each downstream task and pre-training tasks with with sparsity 60% and 70%, respectively.

Lottery Tickets Results of LXMERT and ViLT

- *Q6: Do different VLP models behave differently?*
  - The highest sparsity we can achieve for ViLT is much lower (30% vs. 70%)

Dataset	VQA mini-dev†	GQA test-dev	NLVR^2 dev
Sparsity	70%	70%	70%
LXMERT (paper)	69.90	59.80	74.95
LXMERT (reimp.)	69.95±0.03	59.91±0.07	74.90±0.26
×99%	69.25	59.31	74.15
Lottery Tickets	69.29±0.10	59.40±0.17	74.03±0.71
Random Pruning	65.22±0.05	47.88±0.55	51.38±0.45

Table 3: The LTH results of LXMERT on VQA, GQA, and NLVR^2. (†) The same mini-dev set as used in LXMERT.

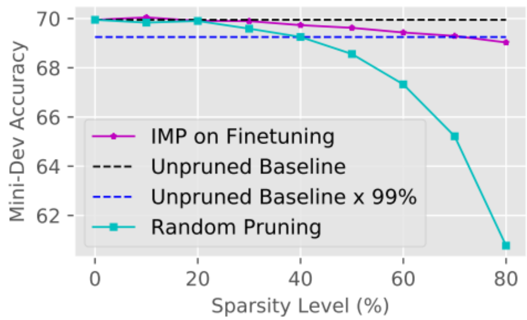
Dataset	VQA (mini-dev†)	NLVR^2 (dev)
Sparsity	30%	30%
ViLT (reimp.)	70.88±0.05	75.82±0.20
×99%	70.17	75.06
Lottery Tickets	70.51±0.11	75.22±0.41
Random Pruning	65.16±0.05	56.14±0.40

Table 4: The lottery ticket results of ViLT on VQA and NLVR^2. (†) The same mini-dev set as used in ViLT.

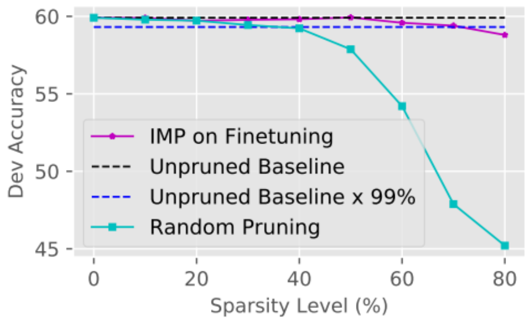
Lottery Tickets Results of LXMERT and ViLT

- *Q6: Do different VLP models behave differently?*

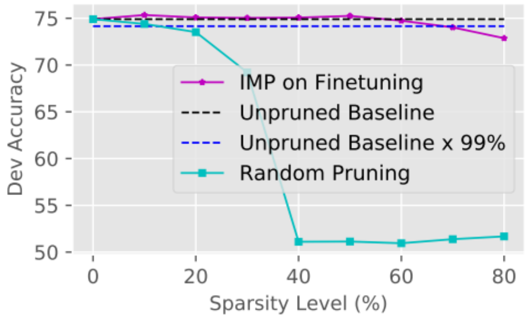
LXMERT



(a) VQA

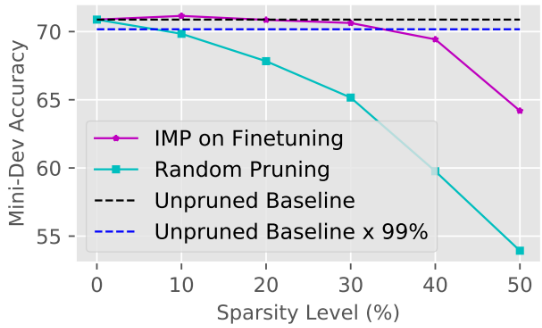


(b) GQA

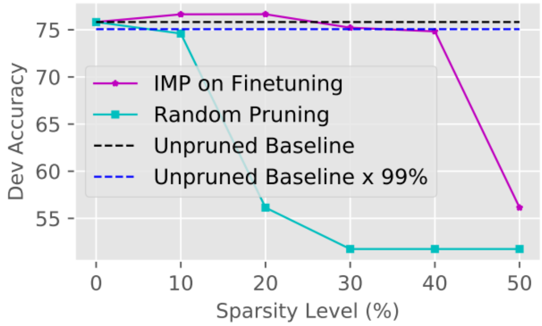


(c) NLVR^2

ViLT



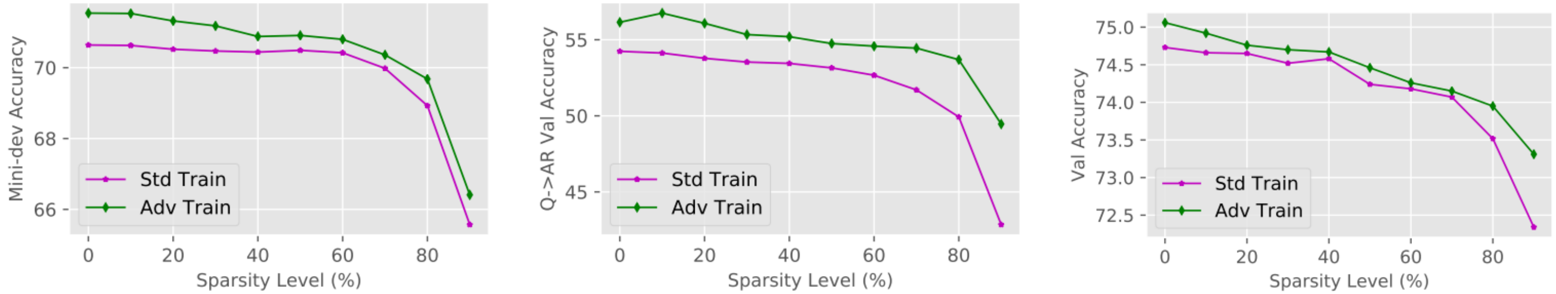
(a) VQA



(b) NLVR^2

Lottery Tickets with Adversarial Training

- *Q7: Can VLP models play lottery tickets adversarially?*



(a) VQA (b) VCR (c) RefCOCO+

Figure 5: Performance of subnetworks that are found by adversarial training on the tasks of VQA, VCR and RefCOCO+.

Finding lottery tickets with adversarial-training-based IMP

Sparisty	VQA	GQA	VCR	NLVR^2	VE	RefCOCO+
60% (Std.)	70.41	59.44	50.37	75.52	77.79	74.41
60% (Adv.)	70.80	59.85	51.07	76.70	77.99	74.74
70% (Std.)	69.45	59.02	47.52	74.29	77.34	73.45
70% (Adv.)	69.79	59.37	48.50	75.29	77.51	74.08

Table 5: Performance of adversarial training on the universal subnetworks at 60% and 70% sparsities. Std.: standard cross-entropy training; Adv.: adversarial training.

Enhancing lottery tickets with adversarial training

Limitations of This Study

- *Efficiency:* We mainly focused on the scientific study of LTH. For future work, we plan to investigate the real speedup results on a hardware platform that is friendly to unstructured pruning

- *Object Detection*: For UNITER/LXMERT, we studied the LTH for multimodal fusion, while keeping the object detection module untouched. In terms of end-to-end VLP, we focused on ViLT. For future work, we plan to study the LTH of object detection and other end-to-end VLP models.

Future Directions

- *Early-bird lottery tickets*: Identifying structured sparsity patterns early in the training, rather than repeating the train-prune-retrain cycle with unstructured pruning for real speedup
- *Data-free pruning*: Obtain trainable sparse neural networks at initialization before the main training process based on some salience criteria.
- *Dynamic sparse training*: Sticking to a fixed small parameter budget, grow and prune subnetworks on the fly throughout the entire training process

Collaborators

Yen-Chun Chen

Linjie Li

Tianlong Chen

Yu Cheng

Shuohang Wang

Jingjing Liu

Lijuan Wang

Zicheng Liu

Thank you!