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
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
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 | NLVR² dev | SNLI-VE val | RefCOCO+ val | Flickr30k IR R@1 | Flickr30k IR R@1 |
|------------------|---------------|--------------|--------------|----------|-------------|--------------|-----------------|-----------------|
| #                | Sparsity      |              |              |          |             |              |                 |                 |
| 1 UNITERB (paper)| 70.75%        | 70%          | 54.94%       | 77.18%   | 78.59%      | 75.31%       | 72.52%          | 60%             |
| 2 UNITERB (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%          |              |              |          |             |              |                 |                 |
| 4 \(f(x; m_{IM} \cdot \theta_0)\) | 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 \(f(x; m_{RP} \cdot \theta_0)\) | 60.45        | 55.95       | 25.35        | 52.42    | 71.30        | 72.95        | 61.44           | 76.80           |
| 6 \(f(x; m_{IMP} \cdot \theta'_0)\) | 67.98        | 58.45       | 50.39        | 54.15    | 76.45        | 71.09        | 63.38           | 79.30           |
| 7 \(f(x; m_{IMP} \cdot \theta''_0)\) | 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; \(\theta_0\): pre-trained UNITER weights; \(\theta'_0\): pre-trained BERT weights; \(\theta''_0\): 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

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

| Dataset           | #       | Sparsity | VQA mini-dev | GQA test-dev | VCR Q→AR val | NLVR$^2$ dev | SNLI-VE val | RefCOCO+ val | Flickr30k IR R@1 | Flickr30k TR R@1 |
|-------------------|---------|----------|--------------|--------------|--------------|--------------|-------------|--------------|----------------|-----------------|
| UNITER_B (paper)  | 1       | 70%      | 70%          | 50%          | 60%          | 70%          | 60%         | 60%          | 70.75           | 75.31           |
|                   | 2       | 70.64±0.06 | 59.64±0.15 | 54.37±0.31$^\dagger$ | 76.75±0.19 | 78.47±0.10  | 74.73±0.06  | 71.25±0.11$^*$ | 84.63±1.02$^*$ | 85.90           |
|                   | 3       | ×99%     | 69.93        | 59.04        | 53.83        | 75.98        | 77.69       | 73.98        | 70.54           | 83.78           |
|                   | 4       | 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       | 60.45        | 55.95        | 25.35        | 52.42        | 71.30        | 72.95       | 61.44        | 76.80           |                |
|                   | 6       | 67.98        | 58.45        | 50.39        | 54.15        | 76.45        | 71.09       | 63.38        | 79.30           |                |
|                   | 7       | 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; $\theta_0^*$: pre-trained UNITER weights; $\theta_0^*$: pre-trained BERT weights; $\theta_0^*$: randomly shuffled pre-trained UNITER weights. ($^\dagger$) 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. ($^\ddagger$) 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             | VQA mini-dev† | GQA test-dev | VCR Q→AR val | NLVR² dev | SNLI-VE val | RefCOCO+ val | Flickr30k IR R@1 | Flickr30k TR R@1 |
|---|---------------------|----------------|--------------|---------------|-----------|-------------|--------------|------------------|------------------|
| 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     | 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 | \(f(x; m_{\text{IMP}} \cdot \theta_0)\) | 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 | \(f(x; m_{\text{RP}} \cdot \theta_0)\) | 60.45          | 55.95        | 25.35         | 52.42     | 71.30       | 72.95        | 61.44            | 76.80            |
| 6 | \(f(x; m_{\text{IMP}} \cdot \theta_0')\) | 67.98          | 58.45        | 50.39         | 54.15     | 76.45       | 71.09        | 63.38            | 79.30            |
| 7 | \(f(x; m_{\text{IMP}} \cdot \theta_0'')\) | 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; \(\theta_0\): pre-trained UNITER weights; \(\theta_0'\): pre-trained BERT weights; \(\theta_0''\): 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 mini-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?*

After obtaining the masks, instead of resetting the weights to $\theta_0$, one should rewind the weights to $\theta_i$, the weights after $i$ steps of training.

(i) VQA Rewinding
Q4: Do winning tickets found on pre-training tasks transfer?
One Ticket To Win Them All

• Q5: Do winning tickets found on downstream tasks transfer?
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 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 | Ave. Perf. Drop (%) | 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: \[
\frac{m_i \cap m_j}{m_i} \%
\] *no clear patterns* in the similarity of learned masks

| Task       | VQA  | GOA  | VCR  | NLVR2 | VE   | RetCOCO+ | ITR   | Pre-training |
|------------|------|------|------|-------|------|----------|-------|--------------|
| VQA        | 100.0| 82.50| 82.71| 90.92| 91.35| 84.32    | 84.83 | 90.54        |
| GOA        | 82.50| 100.0| 79.12| 82.25| 82.72| 78.89    | 79.16 | 82.39        |
| VCR        | 84.71| 79.12| 100.0| 84.71| 85.34| 80.73    | 81.35 | 84.81        |
| NLVR2      | 90.27| 82.25| 84.71| 100.00| 91.34| 84.35    | 84.90 | 90.36        |
| VE         | 91.35| 82.72| 85.34| 91.34| 100.00| 84.68    | 85.56 | 91.48        |
| RetCOCO+   | 84.32| 78.89| 80.73| 84.35| 84.88| 100.00   | 80.89 | 84.37        |
| ITR        | 84.83| 79.15| 81.35| 84.90| 85.58| 80.89    | 100.00| 84.90        |
| Pre-training| 90.54| 82.39| 84.81| 90.36| 91.48| 84.37    | 84.90 | 100.00       |

| Sparsity Patterns (60%) |
|-------------------------|

| Task       | VQA  | GOA  | VCR  | NLVR2 | VE   | RetCOCO+ | ITR   | Pre-training |
|------------|------|------|------|-------|------|----------|-------|--------------|
| VQA        | 100.0| 86.33| 88.13| 92.60 | 93.61| 87.91    | 88.31 | 92.39        |
| GOA        | 86.33| 100.0| 83.58| 86.07 | 86.49| 83.45    | 83.66 | 85.97        |
| VCR        | 88.13| 83.58| 100.00| 86.05| 86.61| 84.95    | 85.44 | 87.57        |
| NLVR2      | 92.60| 86.07| 88.05| 100.00| 93.45| 87.66    | 88.30 | 92.06        |
| VE         | 93.61| 86.49| 88.61| 93.45| 100.00| 88.35    | 88.89 | 93.01        |
| RetCOCO+   | 87.91| 83.45| 84.95| 87.86| 88.35| 100.00   | 85.15 | 87.60        |
| ITR        | 88.31| 83.66| 85.44| 88.30| 88.89| 85.15    | 100.00| 87.99        |
| Pre-training| 92.39| 85.97| 87.87| 92.06| 93.01| 87.60    | 87.99 | 100.00       |

| Sparsity Patterns (70%) |

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\dagger | GQA test-dev | NLVR$^2$ dev |
|--------------------|---------------------|--------------|--------------|
| LXMERT (paper)     | 69.90               | 59.80        | 74.95        |
| LXMERT (reimp.) \times99% | 69.95±0.03        | 59.91±0.07   | 74.90±0.26   |
| Lottery Tickets Random Pruning | 69.29±0.10       | 59.40±0.17   | 74.03±0.71   |
|                     | 65.22±0.05         | 47.88±0.55   | 51.38±0.45   |

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

| Dataset            | VQA (mini-dev\dagger) | NLVR$^2$ (dev) |
|--------------------|------------------------|----------------|
| ViLT (reimp.) \times99% | 70.88±0.05            | 75.82±0.20     |
| Lottery Tickets Random Pruning | 70.17                | 75.06          |
|                     | 70.51±0.11            | 75.22±0.41     |
|                     | 65.16±0.05            | 56.14±0.40     |

Table 4: The lottery ticket results of ViLT on VQA and NLVR$^2$. (\dagger) The same mini-dev set as used in ViLT.
Lottery Tickets Results of LXMERT and ViLT

• **Q6: Do different VLP models behave differently?**
Lottery Tickets with Adversarial Training

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

![Graphs showing performance of subnetworks](image)

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

| Sparisty | VQA   | GQA   | VCR   | NLVR² | 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.

Finding lottery tickets with adversarial-training-based IMP

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!