# The Lottery Ticket Hypothesis for Object Recognition: Supplementary Material

Sharath Girish* sgirish@cs.umd.edu
Shishira R Maiya* shishira@umd.edu
Kamal Gupta kampta@umd.edu
Hao Chen chenh@umd.edu
Larry Davis lsd@umiacs.umd.edu
Abhinav Shrivastava abhinav@cs.umd.edu
University of Maryland, College Park

We provide additional details for some of the experiments presented in the paper. In particular, we provide comparison with a simpler ImageNet ticket transfer alternative in Section A, analyse the transfer of tickets across downstream tasks in Section B, compare the different errors made by dense and pruned models in Section C, verify the faster convergence of sparser models in Section D and finally analyze the disk space and number of compute operations in Section E.

## A. Mask Transfer Without Retraining

In Section 4.2, we analyzed the effects of transferring tickets only for the ImageNet trained backbones. While this deals with transferring the ticket mask as well as values, we further analyze whether transferring only the mask provides winning tickets for these tasks using the methodology from [3]. We use the default ImageNet weights in the ResNet-18 and ResNet-50 backbone and keep the top $p\%$ of the weights in convolutional layers while setting the rest to zeros and maintaining it throughout the training of the entire network. We refer to this method as 'Mask Transfer'. Since training the backbone on much larger ImageNet data is performed only once, 'Mask Transfer' is a much cheaper or computationally efficient way of obtaining tickets from parent task. We observe that behavior of 'Mask Transfer' is similar to the 'Transfer Ticket' obtained by method discussed in Section 4.2 where the sparse subnetwork weights are fully retrained on ImageNet. Either cases are outperformed by direct pruning on the downstream tasks. The results are summarized in Figure 1 (ResNet-18) and Table 1 (ResNet-50).

## B. Do tickets transfer from one task to another

We showed that ImageNet tickets and/or masks transfer to a limited extent to downstream tasks. In this section, we further study whether the tickets obtained from the downstream task of detection/segmentation transfer to keypoint estimation and vice-versa. We train Mask-RCNN and Keypoint-RCNN respectively for the two tasks on the COCO dataset while maintaining a sparsity level of 80%. For both the tasks we transfer all values till box head modules, after which the model structures differ. The results are shown in Table 2. We can observe that the drop is marginal for the transfer of tickets between detection-segmentation to

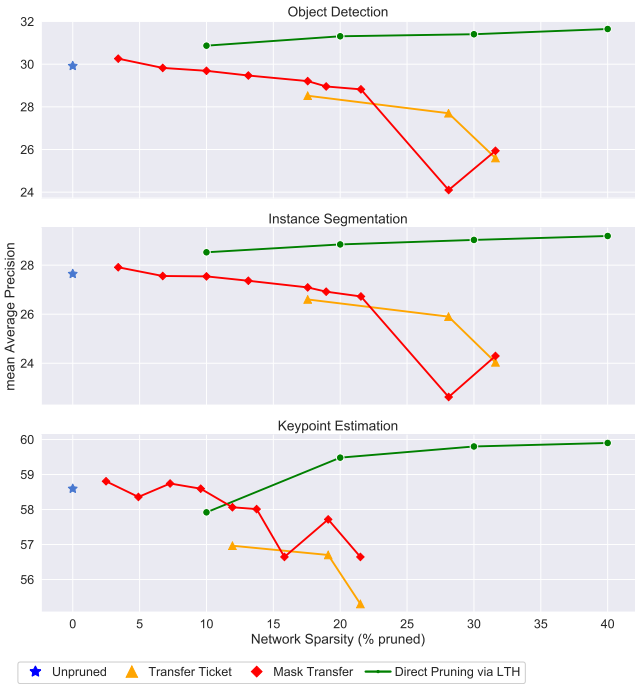



Figure 1: Transferring ImageNet backbone tickets to object recognition tasks *vs*. Direct pruning via LTH on the object recognition tasks. We experiment with two variations of transferring ImageNet backbone tickets to object recognition tasks. 'Transfer ticket' refers to the case when we transfer the lottery ticket backbone trained on ImageNet data to downstream task (also discussed in the Section 4 of the paper). 'Mask Transfer' refers to the case when ticket is transferred without retraining on ImageNet, *i.e.*, only the relevant mask from backbone is transferred keeping ImageNet weights the same. Best viewed in color.

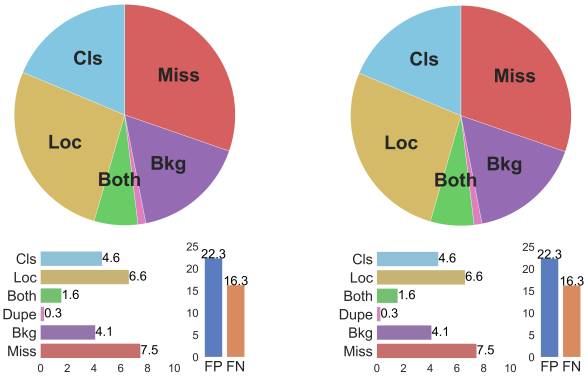
*Equal contribution

Table 1: Performance on the COCO dataset for ImageNet backbones with mask transfer tickets for ResNet-50 at various levels of pruning. The results for VOC are averaged over 5 runs with the standard deviation in parantheses.

| Prune % | COCO Detection | | | COCO segmentation | | | COCO Keypoint | | | VOC Detection | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Network sparsity | mAP | AP50 | Network sparsity | mAP | AP50 | Network sparsity | mAP | AP50 | Network sparsity | mAP |
| 90% | 41.99% | 35.46 | 56.51 | 41.99% | 32.40 | 52.89 | 31.49% | 62.27 | 84.93 | 65.37% | 61.75(±0.22) |
| 80% | 37.33% | 36.52 | 57.28 | 37.33% | 33.53 | 54.15 | 28% | 63.48 | 85.72 | 58.11% | 67.30(±0.39) |
| 50% | 24.55% | 37.99 | 58.83 | 24.55% | 34.76 | 55.91 | 19.71% | 64.21 | 86.32 | 36.32% | 70.32(±0.23) |
| 0% | 0% | 38.5 | 59.29 | 0% | 35.13 | 56.39 | 0% | 64.59 | 86.48 | 0% | 71.21(±0.32) |

Table 2: Effect of ticket transfer across tasks. Transferred tickets do worse than direct training as expected, but still do not result in drastic drops in the mAP or AP50 metrics. Here we do task transfer using the 80% pruned model.

| Target task | Source task | Network sparsity | mAP | AP50 |
|---|---|---|---|---|
| Det | Det/Seg | 78.4% | 30.04 | 49.40 |
| | Keypoint | 50.11% | 23.94 | 41.08 |
| Seg | Det/Seg | 78.4% | 27.90 | 46.68 |
| | Keypoint | 50.11% | 23.02 | 39.01 |
| Keypoint | Det/Seg | 76.98% | 58.31 | 81.53 |
| | Keypoint | 79.4% | 59.34 | 82.36 |




(a) Unpruned model (b) Pruned model

Figure 2: Error analysis of unpruned *vs*. pruned on object detection. The error types of unpruned and pruned models are nearly the same.

keypoint task, as compared with the reverse case which registers a significant drop. This might be because the ticket is obtained on the keypoint task which is trained only on 'human' class and it fails to transfer well for the detection task which uses the entire COCO dataset.

## C. Error analysis on downstream tasks

The mAP score provides us a good way to summarize the performance of an object recognition model with a single number. But it hides a lot of information regarding what kind of mistakes the model is making. Do the sparse subnetworks obtained by LTH make same mistakes as the dense models? In order to answer this question, we consider a dense Mask R-CNN model with ResNet-50 backbone and a sparse Mask R-CNN model with 20% of the parameters obtained via LTH. Both the models achieve same performance on downstream tasks as also discussed in Section 4.2 of the paper.

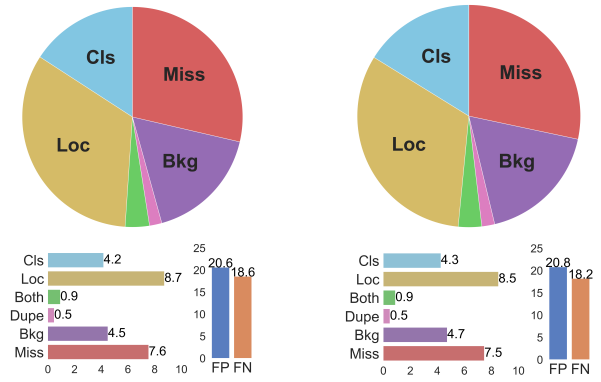
### C.1. Object Detection and Instance Segmentation

We resort to a toolbox from [1] to analyze object detection and instance segmentation errors. We consider 5 main sources of errors in object detection. (i) 'Cls' refers to an error corresponding to miss-classification of a bounding box by a model, (ii) 'Loc' refers to the case when bounding box is classified properly but not localized properly, (iii) 'Dupe' corresponds to the errors when model makes multiple predictions at the same location, (iv) 'Bkgd' are the cases when background portion of the image (with no objects) are tagged as an object, and finally (v) 'Missed' cases when the objects are not detected by the model.

Figures 2 and 3 summarize the analysis of detection and segmentation errors obtained for dense model as compared to a sparse model (with only 20% of the weights). While in the case of object detection, the performance of both the models is identical, subtle differences emerge in case of segmentation where sparse model makes fewer localization errors but higher background errors.

### C.2. Keypoint Estimation

We use [4] to perform a similar analyses for sparse and dense models on the task of keypoint estimation. In case of keypoints, we compute the Precision Recall Curve of the model while removing the impact of individual errors of following kinds — (i) 'Miss' - large localization errors, (ii) 'Swap' - confusion between same keypoint of two different persons, (iii) 'Inversion' - confusion between two different keypoints of the same person, (iv) 'Jitter' - small localization error, and (v) 'FP' - background false positives. Fig. 5 summarizes the results. As in the previous case, it appears




(a) Unpruned model (b) Pruned model

Figure 3: Error analysis of unpruned *vs*. pruned on instance segmentation. The error types of unpruned and pruned models are quite similar.

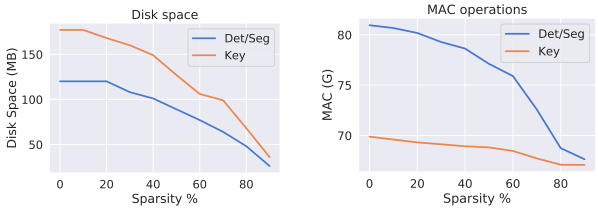



Figure 4: Disk space and MAC operations of pruned models with ResNet18 backbone for the various tasks on the COCO dataset.

that both the dense and sparse models make similar mistakes.

## D. Sparse subnetworks converge faster

The LTH paper [2] claimed that sparse subnetworks obtained by pruning, often converge faster than their dense counterparts. In this section we verify the claims of the paper on object recognition tasks and found them to hold true. We plot the validation loss during training for the dense unpruned model, and the sparse subnetwork obtained by keeping only 20% of the weights of dense model. Both the models achieve a similar mAP after convergence. Fig. 6 shows the task loss against the number of epochs during training. The comparisons confirm that the sparse subnetwork initialized from the winning ticket weights converge much faster. This observation is consistent for heterogeneous tasks, *e.g.*, object detection, instance segmentation, and keypoint estimation.

## E. Disk space and compute operations

Finally, we analyze the disk space and MAC operations of pruned models in Figure 4. We store the index and values of only the non zero weights, when above a threshold sparsity, while we store the full weight values for denser sparsity levels. As expected, we observe significant reductions in disk space for higher levels of sparsity. However, number of operations decreases at a much slower rate. This can possibly be improved further with dedicated hardware for sparse operations.

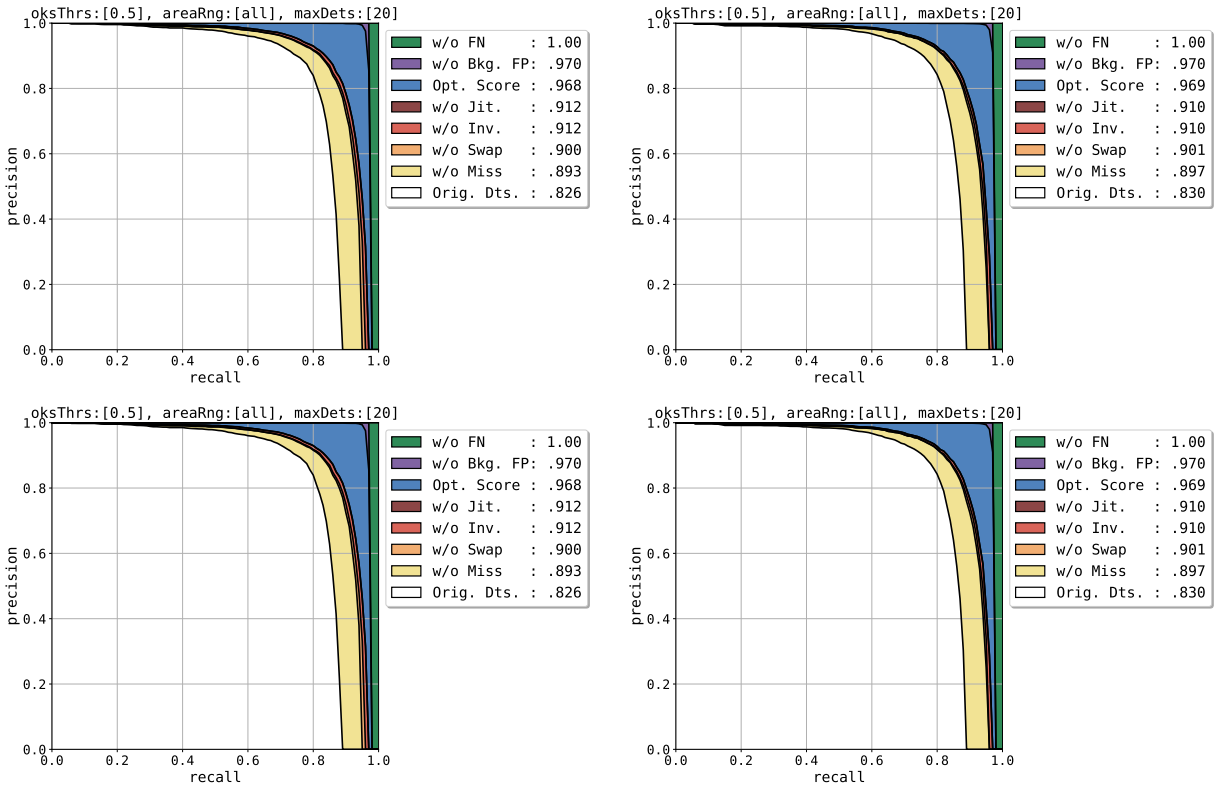
## References

[1] Daniel Bolya, Sean Foley, James Hays, and Judy Hoffman. Tide: A general toolbox for identifying object detection errors. In *ECCV*, 2020.

[2] J Frankle and M Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019.

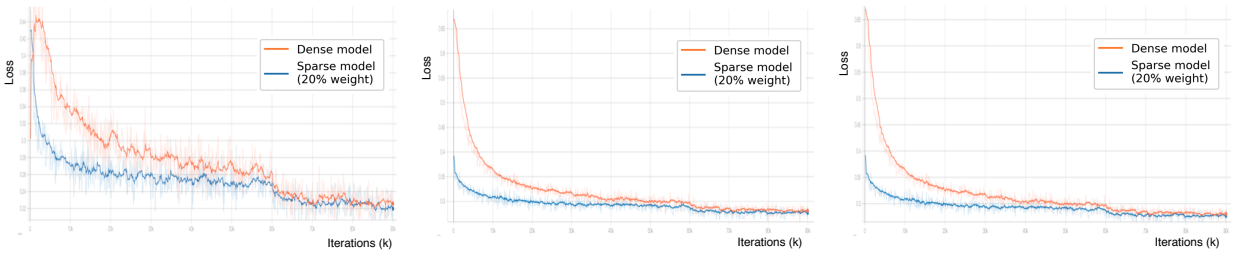
[3] R Mehta. Sparse transfer learning via winning lottery tickets. In *NeurIPS*, 2020.

[4] Matteo Ruggero Ronchi and Pietro Perona. Benchmarking and error diagnosis in multi-instance pose estimation. In *ICCV*, 2017.




(a)Unpruned model (b) Pruned model

Figure 5: Error analysis of unpruned *vs*. pruned on kyepoint estimation. The error types of unpruned and pruned models are quite similar while the unpruned one has slightly better performance




(a) Object detection (b) Instance segmentation (c) Keypoint estimation

Figure 6: Training curves of dense model *vs*. sparse model