To Reviewer #1. L-OBS as competitor. Though the very deep models are widely utilized in real-world applications, they were rarely chosen by the previous pruning methods for the effectiveness validation, as it is difficult to manually tune the layer-wise pruning ratios to achieve a satisfactory compression ratio. We chose L-OBS as the competitor on ResNet-50 simply because 1) it is the only previous method which reported results on ResNet-50, to the best of our knowledge, and 2) as DNS requires some hyper-parameters, it is unpractical for us to tune the method carefully on ResNet-50 and reproduce a reasonable performance. **Pruning sensitivity.** The sensitivity is *not predicted by GSM*, but we noticed that if directly pruning a layer of the original model by a certain ratio (with all the other layers kept the same) results in a significant accuracy reduction, GSM chooses to prune it less, and vice versa. This is actually a strength of GSM and a quantitative validation is provided in Section 4.2. The sensitivity of a layer is an underlying property which can be defined via a natural proxy: the accuracy reduction caused by pruning a certain ratio of parameters. In our paper, we first evaluate such sensitivity via single-layer pruning attempts with different pruning ratios (Figure 2). E.g., for the curve labeled as "prune 90%" of LeNet-5, we first experiment on the first layer by setting 90% of the parameters with smaller magnitude to zero, and testing the model to obtain the accuracy. Then we restore the first layer, prune the second layer and test. The same procedure is applied to the third and fourth layers. After that, we vary the pruning ratio and obtain three curves in the same way. From such experiments we learn that the third layer is the least sensitive. Then we show the layer-wise sparse ratios of the GSM-pruned model. Naturally, for a sensitive layer, GSM is expected to prune it less. I.e., when using GSM, the sensitivity of a layer can be measured using another proxy: the resulting sparse ratio of the layer. As shown in Figure 2, the sensitivities measured in the two proxies are closely related, thus we conclude that GSM automatically knows which layers to prune harder. We will provide more details in the final version. **About Figures.** We use 3 values of β to show a larger β can accelerate the reduction of parameter value, and use $\beta = 0.98$ as the representative to show that the approximation is accurate enough. We will plot the relative approximation error of the 3 values in the final version, and use markers on curves to make figures readable in black and white.

To Reviewer #2. About compression ratio. DNN *connection pruning* (e.g., our method) targets at significantly reducing the number of non-zero parameters, resulting in a sparse model, which can be stored using much less space, but cannot effectively reduce the computational burdens on off-the-shelf hardware and software platforms. On the other hand, *channel pruning* (e.g., Fisher Pruning) cannot achieve a high compression ratio of the model size, but can convert a wide CNN into a narrower (but still dense) one to reduce the runtime memory space and accelerate the computation. Connection pruning and channel pruning are complementary and often used together. For connection pruning, the core trade-off is the model size v.s. accuracy. E.g., the storage space of a mobile APP is usually constrained. Therefore, with a high compression ratio, we can use a big and sparse model to achieve better accuracy than a small and dense one. We will mention this in the final version. **Fisher Pruning.** Our method differs from Fisher Pruning in two aspects. 1) Fisher Pruning falls into the category of channel pruning whereas ours is a typical connection pruning method. 2) The metrics used to evaluate a parameter are different. Fisher Pruning measures the importance of a single parameter θ using a derivation of [Molchanov et al. Pruning convolutional neural networks for resource efficient inference]: $T'(\theta) = \frac{1}{2N}\theta^2 \sum_{n=1}^{N} g_n^2$, where N is the number of data points for measurement, and g_n is the gradient of θ on the n-th data point. Fisher Pruning uses this to *greedily* remove parameters *one-by-one*. On the other hand, we select the important parameters by $T(\theta) = |g\theta|$ for each batch of data. **Model agnostic.** GSM is model agnostic because it makes no assumptions on the model structure or the form of loss function. All the information GSM needs is the value of gradient and parameter. The information of model structure is actually encoded in the gradients via back propagation. I.e., the calculation of gradients is model-related, of course, but it is model-agnostic to use them for GSM pruning. **Lottery ticket.** Thanks for your suggestion, we found out that our method can be used as a more powerful method to find the winning tickets! The major contribution of the lottery paper is an observation (rather than a new method) that the important parameters which are trained to become important (winning tickets) are actually important at the very beginning (after random initialization but before training). That paper discovered that if we 1) randomly initialize a network parameterized by $\boldsymbol{W}_0$, 2) train and obtain $\boldsymbol{W}$, 3) prune some parameters based on the properties of $\boldsymbol{W}$ resulting in a subnetwork parameterized by $\boldsymbol{W}' \subset \boldsymbol{W}$, which is referred to as the winning tickets, 4) find the parameters $\hat{\boldsymbol{W}} \subset \boldsymbol{W}_0$ in the initialized model corresponding to $\boldsymbol{W}'$, 5) train $\hat{\boldsymbol{W}}$ only, and remove the other parameters, we may obtain a comparable level of accuracy with the trained-then-pruned model $\boldsymbol{W}'$. In that paper, the third step is accomplished by simply preserving the parameters with the largest magnitude. In our experiments, we found out that GSM can find a better set of winner tickets than the original simple magnitude-based method. Concretely, we only replace the third step by a pruning process via GSM, and use the resulting non-zero parameters as the winning tickets, and all the other training settings are kept the same as introduced in our paper. On LeNet-300-100 with a compression ratio of 60$\times$, finding winning tickets by the original magnitude criterion and GSM delivers a final accuracy of 96.85% and 97.36%. On LeNet-5 with a compression ratio of 300$\times$, the accuracies of the two methods are 97.94% and 99.04%.

To Reviewer #3. GSM can be viewed as sampling and training a subnetwork of the model. As many classic works have shown, such a subnetwork can satisfactorily approximate the original model. Though GSM focuses on global pruning, we still performed layer-wise pruning experiments, and the results are better than the pruning-then-finetuning methods, which are not demonstrated due to page space limitation.