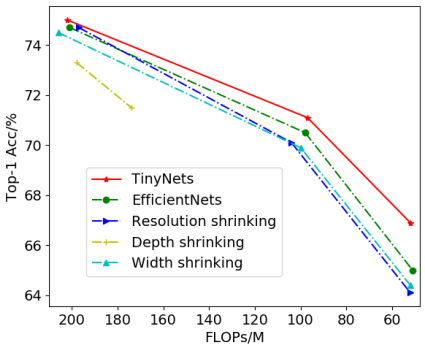
We sincerely thank all the reviewers for their helpful comments.

Response to Reviewer #1:

Q: *Relationship with EfficientNet.* **A:** Our main contributions: 1) Our method provides a general pipeline for shrinking neural networks, in contrast to EfficientNet that enlarges CNNs. 2) We found that "resolution and depth are more important than width for tiny networks" which is different from EfficientNet. 3) Different to the giant formula in EfficientNet that is handcrafted, the proposed data-driven formula twists the three dimensions based on the observation of frontier models. 4) The obtained TinyNets show outstanding performance.



Q: *Typo.* **A:** The actual accuracy is 75.8%, not 78.5%. We'll fix it.

Q: *More FLOPs in Fig.6.* **A:** We include more FLOPs as shown in Fig⇒

Response to Reviewer #2:

Q: *Why Gaussian process regression.* **A:** In Fig.3, the curves are not simple linear regression and unknown to us. We utilize Gaussian process regression to fit the curves for its effectiveness and nonparametric property.

Q: *Relationship with NAS.* **A:** The main differences: 1) Our method gives principle to obtain smaller models which can be interpretable and general, while NAS only results in several architectures which are hard to understand. 2) The formula in our method can output the configurations for any FLOPs, while NAS only search for predefined FLOPs.

Q: *Setting of RandAugment.* **A:** We set the magnitude of RandAugment to be 9 with a std as 0.5 in all networks.

Response to Reviewer #3:

Q: *Just a supplement for EfficientNet.* **A:** Thanks for this comment. In fact, resolution, depth and width are three key dimensionalities for the performance and computational cost of neural networks. This paper aims to transfer the success of EfficientNets for enlarging models to tiny models with higher performance. We found that "resolution and depth are more important than width for tiny networks" which is different from EfficientNet. Besides, we present a data-driven algorithm for finding the optimal tiny formula by twisting the three dimensions based on the observation of frontier models. The performance of TinyNets is about 0.3-3.8% higher than that of EfficientNet with similar computational complexity, which is a significant improvement on ImageNet.

Q: *Why resolution & depth > width.* **A:** Basically, The resolution contain the raw information of images, *e.g.* the average resolution of ImageNet images is 482x418 pixels, so changing too much of resolution will cause severe information loss to the objects. The depth directly controls the times of nonlinear transformation and the receptive field of CNNs, which is important for extracting hierarchical features from images. As for width, the channels contains more redundancy as shown in many pruning methods, so the width is not so important as resolution and depth. More discussions and visualizations will be included in the final version.

Q: *Evolutionary searching.* **A:** Thanks for this nice concern. Random sampling here provides a simple yet effective approach for TinyNets with higher performance and compact architectures. We also agree that the evolutionary algorithm can provide better results which can be explored as an extension of our method.

Q: *CAM in Fig.4.* **A:** We follow the EfficientNet paper and show these class activation map figures. These visualizations only mean that models with better performance can focus on the more relevant regions.

Q: *Training cost.* **A:** Different from NAS which gives several 'strange' architectures, our method provides general and interpretable formula for shrinking CNNs. We randomly sample 100 models and train them on a ImageNet subset with cost of 10.4 GPUdays, compared to 3800 GPUdays@MnasNet, 8.3 GPUdays@ProxylessNAS, 9 GPUdays@FBNet.

Q: *Hyper-parameters to train EfficientNet.* **A:** We use the thirdparty Pytorch code which reproduces the official Tensorflow code with similar performance (diff$\leq$0.1%). All the TinyNets and EfficientNets are trained under the same settings, so the results and conclusion are fairly given.

Response to Reviewer #4:

Q: *The gains are marginal.* **A:** Thanks for this nice concern. We have hilighted the accuracy gains of our TinyNets with similar FLOPs to those of EfficientNets in Table R1. Considering that the ImageNet is still a difficult benchmark for modern lightweight CNNs, these gains are exactly significant improvements. For example, MobileNetV2 is about 0.5% higher than that of ShuffleNetV1 (SOTA of that time).

Q: *A study on different tasks.* **A:** The comparsion experiments on COCO dataset as shown in Table R2 illustrate that our TinyNet can also surpass the conventional EfficientNet on the Faster RCNN framework.

Q: *Related work.* **A:** We will cite and discuss about the width scaling methods used in these recent excellent networks including ESPNetv2, ShuffleNetv1-v2, MobileNets.

Q: *Generalizability.* **A:** Thanks for this constructive comment. This paper is mainly focusing on tiny models for computer vision tasks. To apply our method to NLP models, some topics need to be explored accordingly, *e.g.* how to define 'resolution' in NLP models. More discussion on future work and broader impact will be included.

Table R1. ImageNet gains of TinyNet over EfficientNet.

FLOPs/M	339	202	97	52	24
Acc gain/%	0.5	0.3	0.6	1.9	3.8

Table R2. COCO performance of TinyNet & EfficientNet.

EfficientNet (Backbone FLOPs/mAP)	387M/30.4%	98M/18.5%
TinyNet (Backbone FLOPs/mAP)	339M/30.5%	97M/19.8%