We thank all three reviewers (**R#1, R#3, R#4**) for appreciating our novelty, strong results, and broad impact to the community, and promise to release all codes upon acceptance and to improve writing clarity. Below, we first respond to general E^2-Train questions, and then specific questions on **SMD**, **SLU** and **PSG**, being grouped between lines.

Reducing accuracy loss? (R#1). We continued striving to reduce the accuracy loss after NeurIPS submission, and found a stochastic weight averaging (SWA) technique ("SWALP" paper in ICML'19) to be helpful. As requested by R#1, we report the improved new result : after applying SWA to E^2-Train on ResNet 74, we obtain top-1 93.01% on CIFAR-10 (0.56% loss) and top-1 71.63% on CIFAR-100 (no accuracy loss) with 83.40% and 81.27% energy saving, respectively.
Experiments on adapting a pre-trained model (R#1, R#3). We performed a proof-of-concept experiment for CNN fine-tuning by splitting CIFAR-10 training set into half, where each class was i.i.d. split evenly. We first pre-train ResNet-74 on the first half, then fine-tune it on the second half. During fine-tuning, we compared two energy-efficient options: (1) fine-tuning only the last FC layer using standard training; (2) fine-tuning all layers using E^2-Train. With all hyperparameters being tuned to best efforts, the two fine-tuning methods improve over the pre-trained model top-1 accuracy by [0.30%, 1.37%] respectively, while (2) saves 61.58% more energy (FPGA-measured) than (1). That shows that E^2-Train is the preferred option: higher accuracy, and saving much more energy. We will report it in camera-ready.
On different CNNs (R#3, R#4). E^2-Train is effective in even compact networks, e.g., MobileNetV2: 92.06% / 71.61% top-1 accuracy on CIFAR-10/-100 (baseline: 92.47% / 71.91%), with 87.73% / 88.17% energy savings, respectively.
Training time (R#3). E^2-Train takes roughly the same epoch numbers to converge (line 324). Similar to energy, the per-epoch time is also reduced, thanks to our three-level savings. We will report detailed measurements in camera-ready.

Why SMD works, by theory? (R#3, R#4). We are happy to confirm that SMD is not simply a heuristic - **we recently proved** that SMD can outperform SGD at certain range of mini-batch sampling ratios and epoch numbers. The proof is inspired by *"Random Shuffling Beats SGD after Finite Epochs"*, ICML'19, and the core idea relies on the finite population correction technique. We plan to report our theoretical findings and proof in another upcoming submission.
Is SMD stable/reproducible? (R#3, R#4). We repeated training ResNet74 on CIFAR-10 using SMD for 10 times (all end up saving $\frac{1}{3}$ energy), with different random initializations. The accuracy std. is only 0.132%, showing high stability.
Is SMD generalizabe to more models? (R#4). For ResNet74 on CIFAR-100, [SMD, SMB] with $\frac{1}{3}$ energy saving achieve top-1 [71.37%, 71.11%], respectively; and for ResNet-110 on CIFAR-10, [SMD, SMB] with $\frac{1}{3}$ saving achieve [93.05%, 92.75%], respectively. Various of such experiment results confirm that SMD is consistently better than SMB.
Clarifying Fig. 3b (R#4). We tried to find more baselines for solid evaluation of SMD. As SMD outperformed SMB with standard learning rates (lr), we conjectured increasing lr might accelerate SMB's convergence (reducing epochs), thus making another strong baseline. We also repeated SMD/SMB experiments and found their gap to persist (>0.2%).

SLU on non-residual CNNs (R#3, R#4). We thank both reviewers to point out. Indeed, our current experiments are based on CNNs with skip connections, partially because they dominate in SOTA CNNs. We will make it clear in camera-ready. Furthermore, we conjecture SLU can be extended to non-residual CNNs by "appending" skip connections with gates to plain backbones (i.e., creating "ResNet" versions for their training). We leave it for future work.
Error bar of SLU (R#4). 20 trials of SLU experiments to ResNet38 on CIFAR-10 show that, with 95% confidence level, the confidence interval for the mean of the top-1 accuracy and the energy saving are [92.47%, 92.58%] (baseline: 92.50%) and [39.55%, 40.52%], respectively, verifying SLU's trustworthy effectiveness.
What auxiliary RNNs learn(R#3)/ SLU training dynamics(R#4). After training, visualizations show RNNs learned (i) the class-wise discriminative selectivity of layers (ResBlocks); and (ii) input hardness-aware routing, consistent with SkipNet's observations. During training, we observe that the auxiliary RNNs converge much faster (<20 epochs) than the main backbone part, showing "layer selectivity" can be identified at early training stage. Interestingly, this coincides with observations of "critical learning periods", ICLR'19. We will include the visualizations/analysis into camera-ready.
Adjusting p_L (R#4). p_L controls SD's drop ratio, and we always tune it to make SD's drop ratio the same as SLU (line 291). Similar to SLU, dropping more layers decreases SD's performance, in exchange for more energy saving.

Clarifying PSG description (R#4). PSG is motivated by two facts: (i) reducing precision is very effective (exponentially) for reducing hardware energy cost; and (ii) MSB parts contribute exponentially larger than LSB parts for calculating outputs. Specifically, while SGD uses x and g_y to calculate g_w, PSG uses x^{msb} and g_y^{msb} to calculate g_w^{msb} for predicting the sign of g_w when $|g_w|^{\text{msb}} \geq \tau$ (this happens with high probability thanks to fact (ii), e.g., > 60% in ResNet74 on CIFAR10). Otherwise (i.e., $|g_w|^{\text{msb}} < \tau$), PSG will proceed to finish the remaining residual computation of calculating g_w and its sign. In hardware, the computation to obtain g_w^{msb} is embedded within that of g_w. Therefore, the PSG's predictors (i.e., calculating g_w^{msb} using x^{msb} and g_y^{msb}) do not incur any energy overhead.
More requested details on PSG (R#3). We use default $\beta = 0.05$ to control MSBs, while the effectiveness of PSG is observed to be insensitive to the choice of β when it is in the range of [0.05, 0.1]. If β is too small, it will result in too frequent coarse gradients and might hurt convergence. The ratio of using merely g^{msb} typically remains at least 60% in the training process. We will include these in camera-ready, along with detailed explanation of E_1 and E_2 in (3).