R1: **(a)** *"blur the distributions":* As Wasserstein barycenter *adjusts the support*, blurring is more likely for Euclidean avg. **(b)** *continual learning:* Growing the barycentric network gradually & unbalanced OT is left for future work.

R2, **R4**: We present results on an additional (harder) dataset, CIFAR100, to illustrate that *our results indeed generalize!* **1.** In Table 1, we adapt the VGG11 architecture (used for CIFAR10) and train multiple copies *with different initializations*, in a similar manner for 300 epochs. Here, our focus was not to train individual models with best accuracy, *rather to investigate the efficacy of fusion*. OT fusion results in a mean test accuracy gain $\sim \{1.4\%, 1.7\%, 2\%\}$ over the best individual models, in case of $\{4, 6, 8\}-$ base models, and is *# model $\times$ more efficient* than ensembling them. Vanilla averaging, in contrast, fails to fine-tune despite trying numerous settings of optimization hyperparameters. **2.** Also, Fig 1, shows similar gains for *data-free post-processing* in case of structured pruning (as in Sec 5.2).

CIFAR100 + VGG11	Individual Models	Prediction avg.	Finetuning Vanilla	Finetuning OT
Accuracy Efficiency	[62.70, 62.57, 62.50, 62.92] 1 ×	66.32 1 ×	4.02 4 ×	**64.29± 0.26** **4 ×**
Accuracy Efficiency	[62.70, 62.57, 62.50, 62.92, 62.53, 62.70] 1 ×	66.99 1 ×	0.85 6 ×	**64.55 ± 0.30** **6 ×**
Accuracy Efficiency	[62.70, 62.57, 62.50, 62.92, 62.53, 62.70, 61.60, 63.20] 1 ×	67.28 1 ×	1.00 8 ×	**65.05± 0.53** **8 ×**

Table 1: Efficient alternative to ensembling via OT fusion on **CIFAR100** for VGG11. Vanilla average fails to retrain. Results shown are mean ± std. deviation over **5 seeds**.

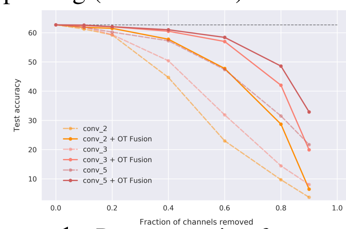

Figure 1: Post-processing for structured pruning via OT-fusion on **CIFAR100**.

R2, **R3** *"there could possibly be more competent baselines":* **1.** We compare OT fusion in the context of: ensembling (Sec 5.3), vanilla averaging (Sec 5.1, 5.3) widely used in federated learning, distillation (Sec 5.3, S12), & show a favorable *accuracy-efficiency trade-off*. **2.** Averaging parameters of neural-networks with **different widths** (Sec 5.2) *is being enabled for the first time*, to our knowledge. **3.** Greedily matching neurons performs worse than OT, as expected theoretically.

R2: (a) *"forward for each of K individual models $\cdots$ compared to "prediction average"*: The activation-based alignment (acts) does this **only once**, while prediction avg. will have to do this every time during inference. (b) *"published structured pruning methods"*: Lines 295-297, our goal here is **not to propose a new method**, rather a post-processing technique that is independent of the pruning algorithm. (c) We will surely organize the algorithm better.

R3: **(a)** *"special and general models $\cdots$ seems a bit artificial:* A similar setting was considered in the distillation paper (Hinton et.al. 2015, Section 3), and likewise, in continual learning variants of this setup (Split-MNIST) are used for benchmarking. The 'constraint' of performing this without sharing of sensitive training data arises in many applications, such as healthcare, legal, etc. **(b)** *"improvement over vanilla averaging is very marginal"*: We respectfully disagree. **1.** 2-model case: Besides the results in Table 1 please refer to other fine-tuning settings in Table S7, S8 where OT fusion also outperforms. Plus, we are fine-tuning for a significant duration (~ 100 epochs) to adequately illustrate that vanilla avg. can't recover. **2.** ≥ 2 models: Vanilla avg. fails to retrain despite trying a large set of hyperparameters (Appendix S4.2), also check the results on CIFAR100 in Table 1, reported over 5 seeds. **(c)** *"people don't average the weights":* As noted by **R2**, **R4**, and as discussed above, element-wise averaging of weights has a widespread adoption in federated learning (FedAvg, McMahan et. al. 2016). **(d)** *Miscellaneous:* **1.** For structured pruning (Fig. 3), we use weight-based variant to avoid the usage of data (Line 277). But, in general, activation-based alignment (acts) performs on par (and often slightly better), so we use it for all other results (Line 193). **2.** Fig S9 caption: it should be "all".

R3, **R4**: *"model benefit from fusion with (almost) itself?"* Due to mass conservation when doing OT between dense and pruned model layers, the (removed) filters of the dense model, which either detect similar features or whose features can be composed, get fused into the remaining filters of the smaller model. We will add the activation maps in the paper.

R4: (a) *FedMA*. **1. Flexibility**: FedMA inherently solves a hard-assignment problem to obtain a permutation, while our approach is based on the more general optimal transportation problem (OT). So, if the number of neurons being matched are different, OT can transport a distribution [1/2, 1/2] to [1/4, 1/4, 1/4, 1/4] and vice versa. This fundamental difference allows us to fuse into a smaller model (as illustrated by the two applications in Section 5.2), in a rather effortless way using OT as compared to FedMA. **2. Practicality**: FedMA is restrictive from the practical viewpoint, since it requires extensive coordination and communication. It assumes that same set of clients communicate repeatedly **for # layer many rounds**, where each round involves freezing the previously matched layers across the devices, and then matching the current layer. After which, the rest of the layers get retrained and the procedure is repeated until all the layers get matched. But in practice (Kairouz et. al., 2019), the server samples a random subset of active devices in each round. Also, straggler devices can **hinder a proper alignment** of models in FedMA, hence limiting its practical applicability. **3. Stability:** Their intermittent "freezing and retraining" process is known to suffer from convergence instabilities during retraining (see Appendix A of their paper). In contrast, our one-shot fusion of entire models via OT does not suffer from these issues. (b) *matching of layers of different size:* The mass splitting example above should better explain how the matching might behave (also see the shared point with **R3**).