Model Fusion *via Optimal Transport*

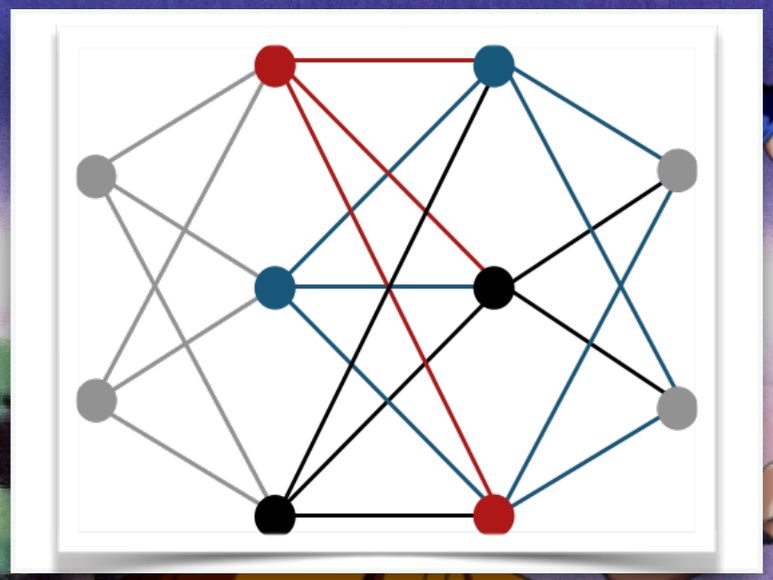
Singh & Jaggi, NeurIPS 2020

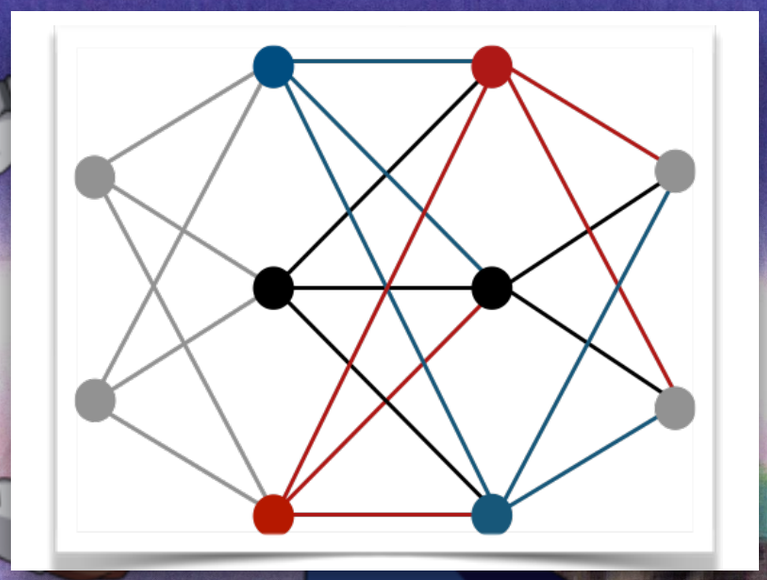
DLCT Reading Group: 25th September, 2020

EPFL

Sidak Pal Singh

Fusion





Fusion

Problem Setup

- Given K independent neural nets (models), with each trained on a possibly different training set, i.e., $(M_1, S_1), \cdots, (M_K, S_K)$.

- Aim: Fuse their capabilities in one single model $\mathscr{M}$.

- Question: How to obtain the fusion operator?

$$\mathscr{M} \leftarrow f(M_1, \cdots, M_K)$$

$$\theta_{\mathscr{M}} \leftarrow f(\theta_1, \cdots, \theta_K)$$

- (Personal opinion: fundamental yet relatively unexplored)

How hard is it?

- As a start, let's assume, same architecture and size!
- Highly non-convex parameter space.
- Neural networks possess inherent symmetries.
- No apriori guarantee that a suitable solution exists.
- Additional constraints to care about (data free; no stacking of params)

Why is it important?

- Make predictions more robust: Ensemble
- Combine models learned on different datasets/tasks:
  - Federated learning
  - Continual learning
  - .....
- "Skill/Knowledge" transfer to a smaller model

Things to care about

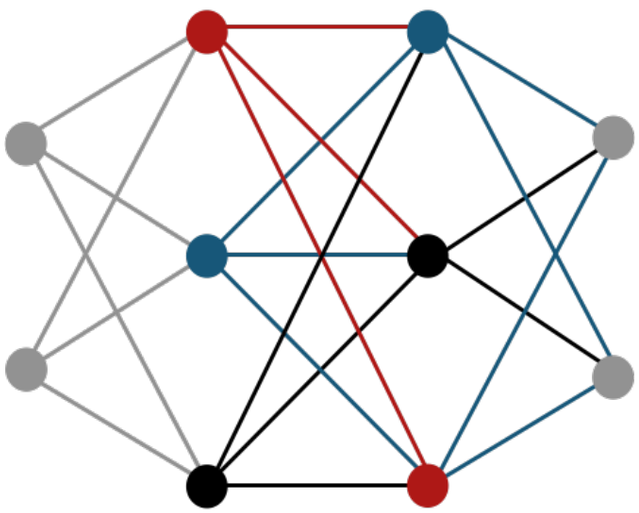
- **Resource costs** (fusion procedure, storage)
- **Inference costs** (communication, latency)
- **Privacy costs** (sharing training data)

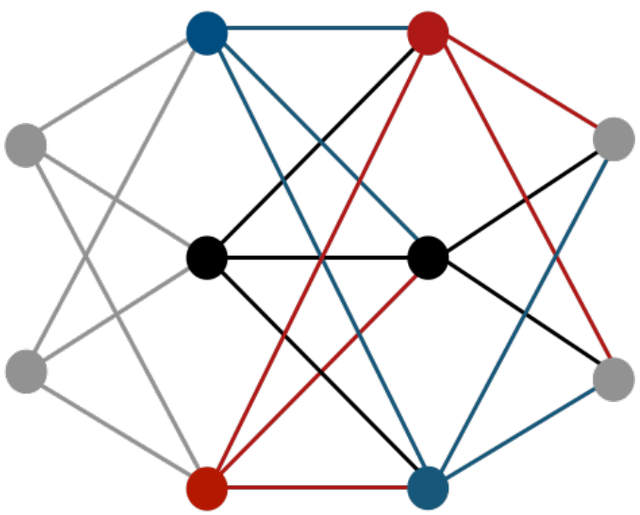
Related Work

	Resource costs	Inference costs	Privacy costs
Ensembling	☹️	☹️	😄
Distillation	😐	😄	☹️
Model Fusion	😄	😄	😄

Umm, what if we average parameters one-to-one?

Input Models





Vanilla Averaging

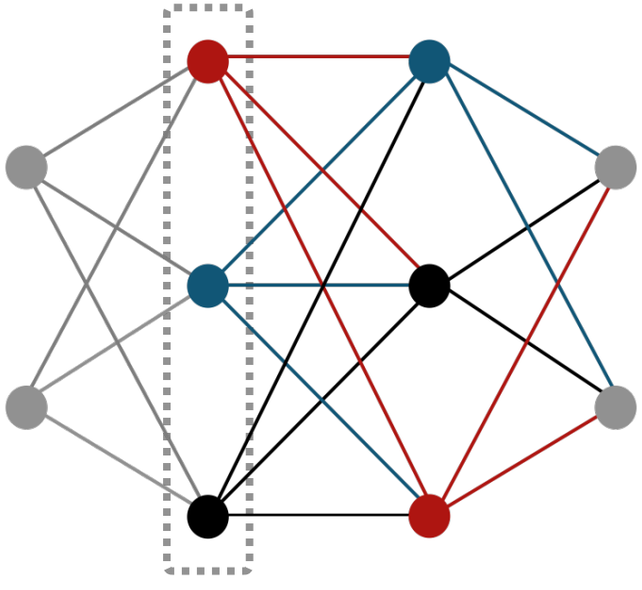
- No direct correspondence between the neurons across two models.
- The models could have been initialized differently or trained on separate data.
- Does not really work in general.

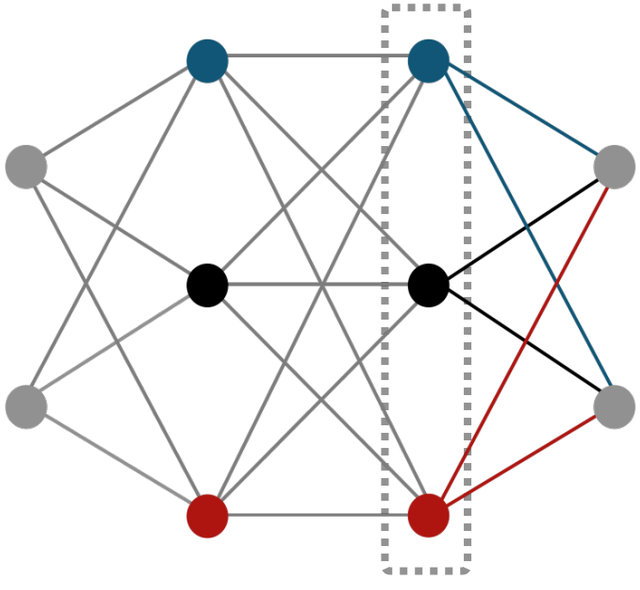
Layer-wise alignment of neurons & then averaging!

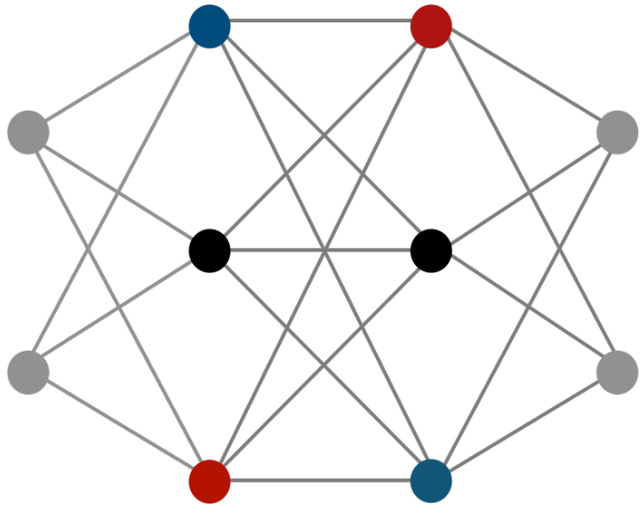
(to simplify discussion, let's consider only two models A and B)

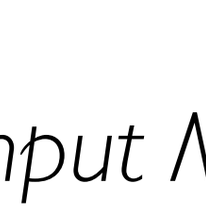
Input Models

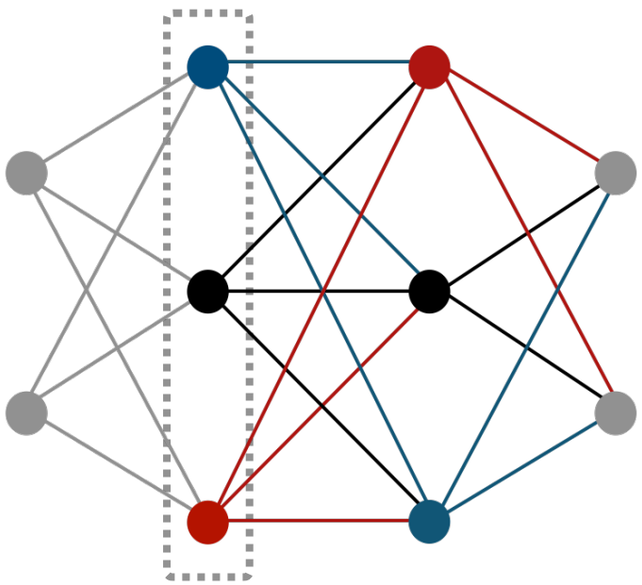
Aligned Models

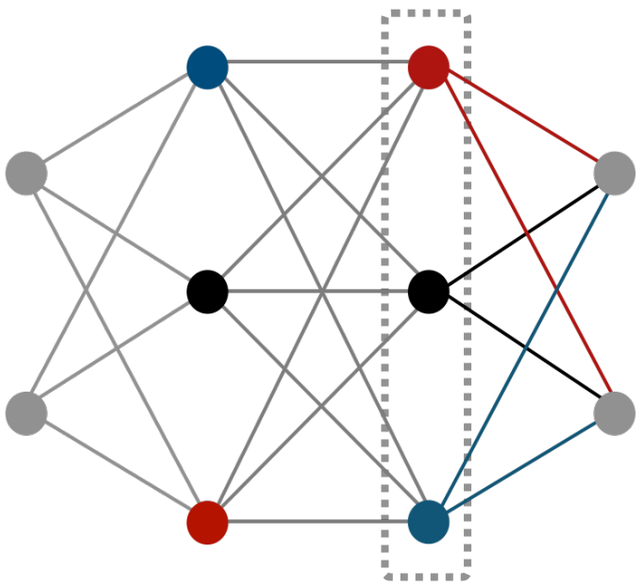


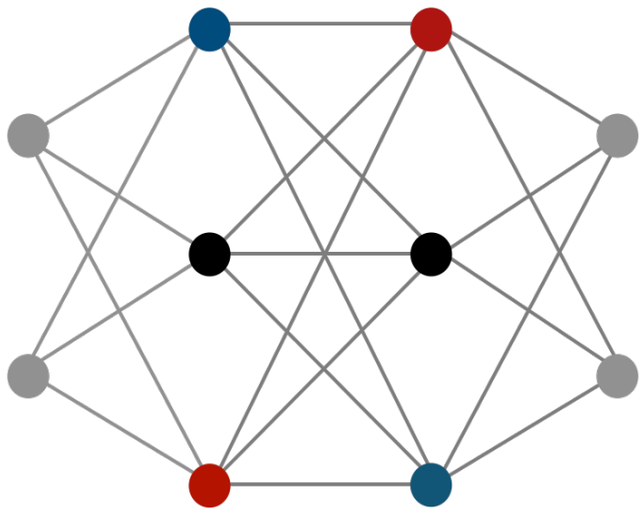




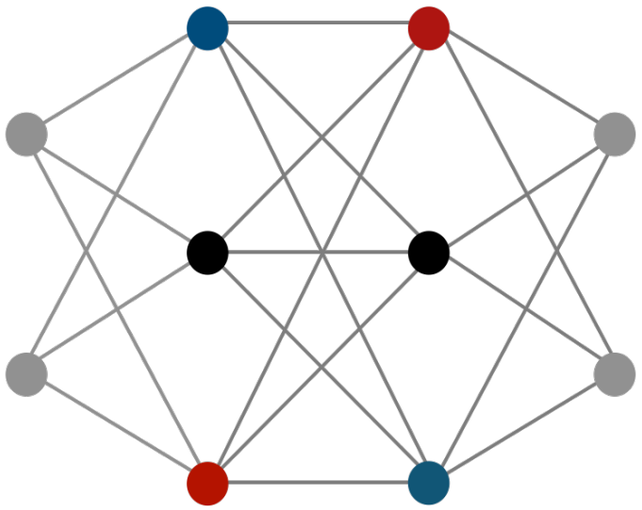


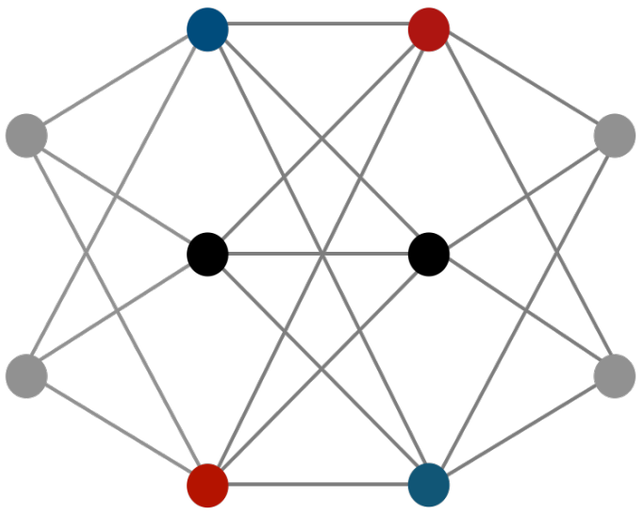




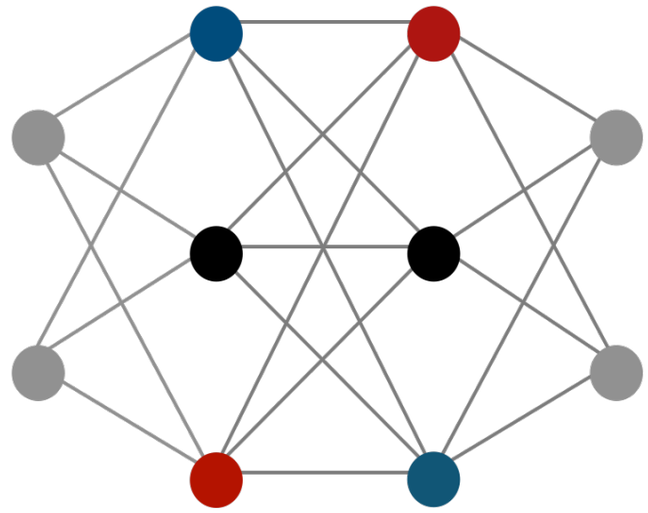


Aligned Models



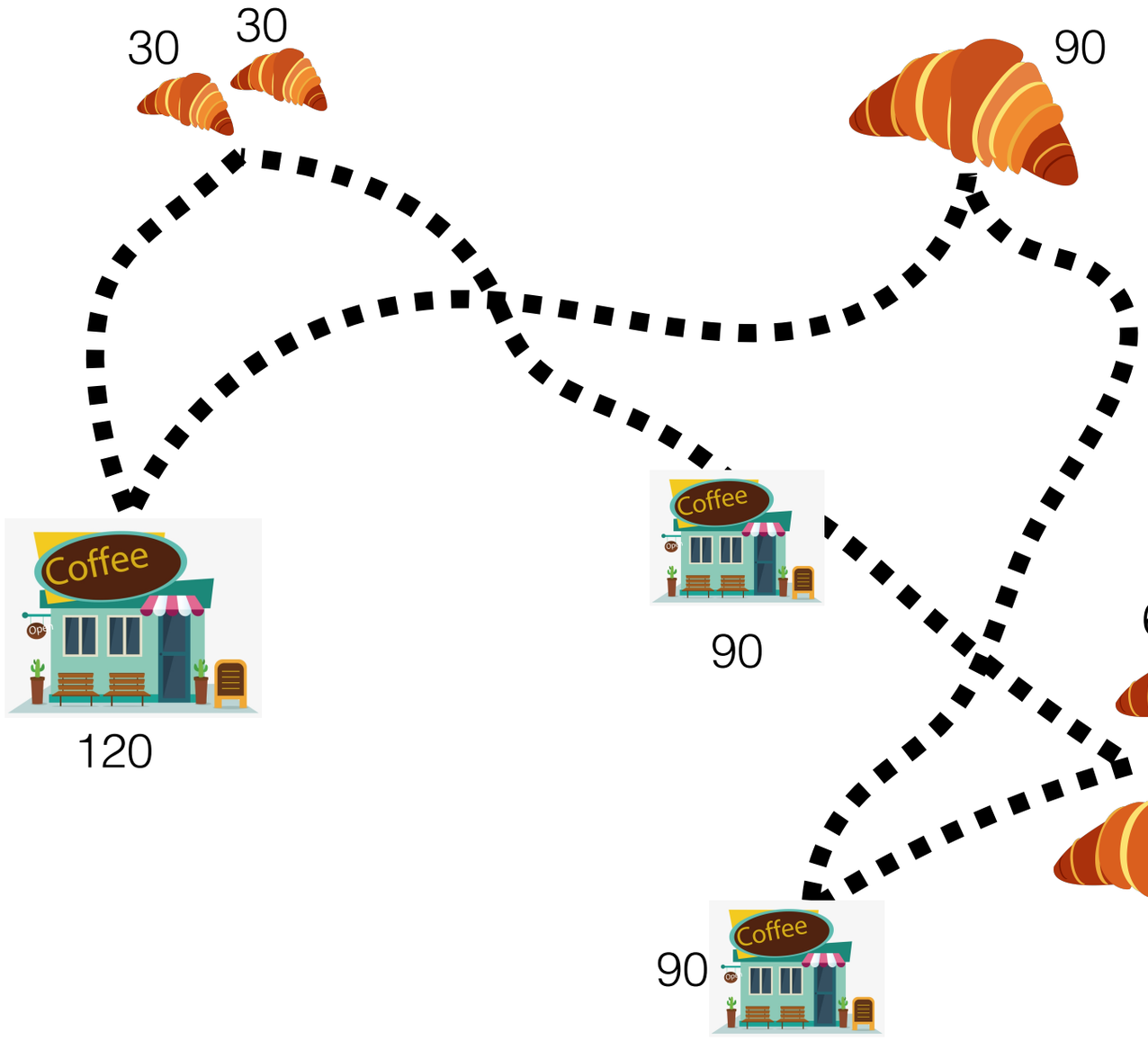


Output Model



Opening the black-box

Optimal Transport

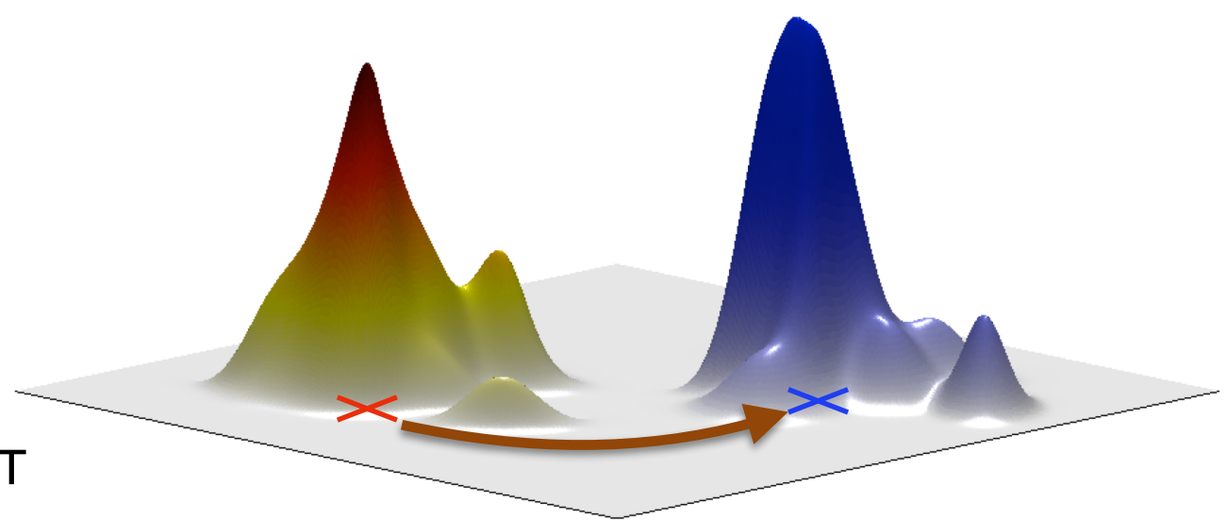


$$\mathrm{OT}\left(\mu_A, \mu_B; \boldsymbol{C}\right) := \min_{\boldsymbol{T} \in \mathbb{R}_A^{(n_A \times n_B)}} \langle \boldsymbol{T}, \boldsymbol{C} \rangle$$

$$\text{s.t.} \quad \boldsymbol{T}\mathbf{1}_{n_B} = \boldsymbol{\alpha}, \boldsymbol{T}^\top \mathbf{1}_{n_A} = \boldsymbol{\beta}$$

$$\mathscr{W}_p\left(\mu_A, \mu_B\right) := \mathrm{OT}\left(\mu_A, \mu_B; D_{\mathcal{S}}^p\right)^{1/p}$$

$$\text{where,} \quad C_{ij} = D_{\mathcal{S}}\left(\boldsymbol{s}^{(i)}, \boldsymbol{s}^{(j)}\right)^p$$



Ref: Cuturi & Solomon, NeurIPS 2017 Tutorial on OT

Algorithm: Layerwise alignment & average!

For each layer

1. $\widehat{\boldsymbol{W}}_A^{(\ell,\ell-1)} \leftarrow \boldsymbol{W}_A^{(\ell,\ell-1)} \boldsymbol{T}^{(\ell-1)} \operatorname{diag}\left(\frac{1}{\boldsymbol{\beta}^{(\ell-1)}}\right)$ // Align incoming edges for M_A weights
2. $\boldsymbol{\alpha}^{(\ell)}, \quad \boldsymbol{\beta}^{(\ell)} \leftarrow \mathbf{1}_{n_A^{(l)}}/n_A^{(\ell)}, \quad \mathbf{1}_{n_B^{(\ell)}}/n_B^{(\ell)}$ // Initialise probability mass values for neurons
3. $\mu_A^{(\ell)}, \ \mu_B^{(\ell)} \leftarrow \left(\boldsymbol{\alpha}^{(\ell)}, \boldsymbol{S}_A[\ell]\right), \quad \left(\boldsymbol{\beta}^{(\ell)}, \boldsymbol{S}_B[\ell]\right)$ // Write as empirical measures
4. $D_{\mathcal{S}}^{(\ell)} \leftarrow \left\|\boldsymbol{S}_A[\ell]^\top - \boldsymbol{S}_B[\ell]^\top\right\|_2$ // Form ground metric
5. $\boldsymbol{T}^{(\ell)}, \mathcal{W}_2^{(\ell)} \leftarrow \mathrm{OT}\left(\mu_A^{(\ell)}, \mu_B^{(\ell)}, D_{\mathcal{S}}^{(\ell)}\right)$ // Compute OT map and distance
6. $\widetilde{\boldsymbol{W}}_A^{(\ell,\ell-1)} \leftarrow \operatorname{diag}\left(\frac{1}{\beta^{(\ell)}}\right) \boldsymbol{T}^{(\ell)^\top} \widehat{\boldsymbol{W}}_A^{(\ell,\ell-1)}$ // Align model A's neurons
7. $\boldsymbol{W}^{(\ell,\ell-1)} \leftarrow \frac{1}{2}\left(\widetilde{\boldsymbol{W}}_A^{(\ell,\ell-1)} + \boldsymbol{W}_B^{(\ell,\ell-1)}\right)$ // Average model weights

Convex combination of neurons in A

End for

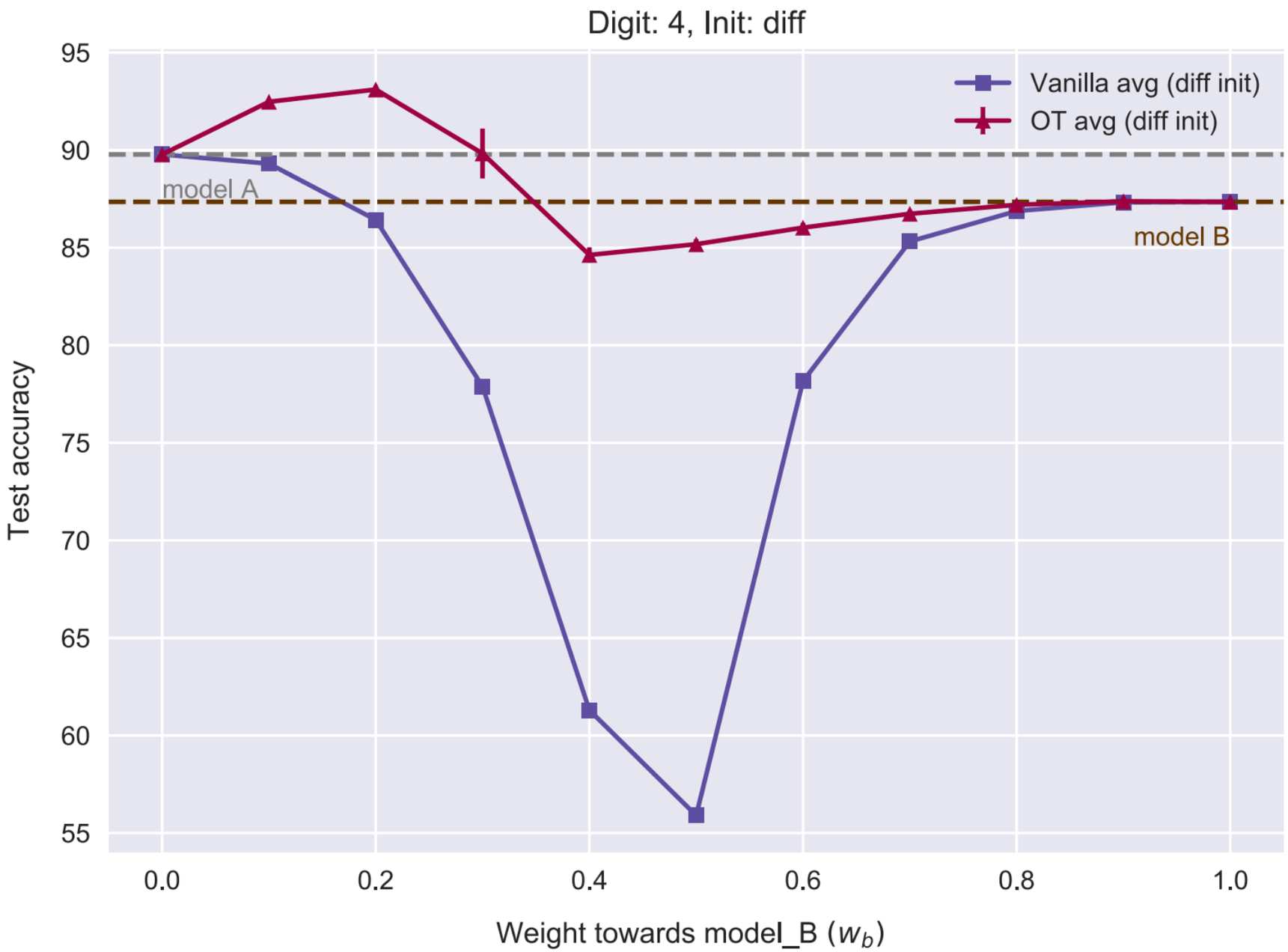
Similar extension for multiple models, pick one model to align against!

Discussion

- How to design the ground metric (i.e., compare neurons)?
  - Activation-based alignment
  - Weight-based alignment
- We still have a greedy procedure:
  - Ideal procedure is combinatorially hard
- Runtime Efficiency (even with exact OT):
  - 15 seconds to fuse 6 VGG11 models 1 Nvidia V100 GPU

Fusion for heterogeneous data and tasks

Applications: Skill Transfer, Federated Learning

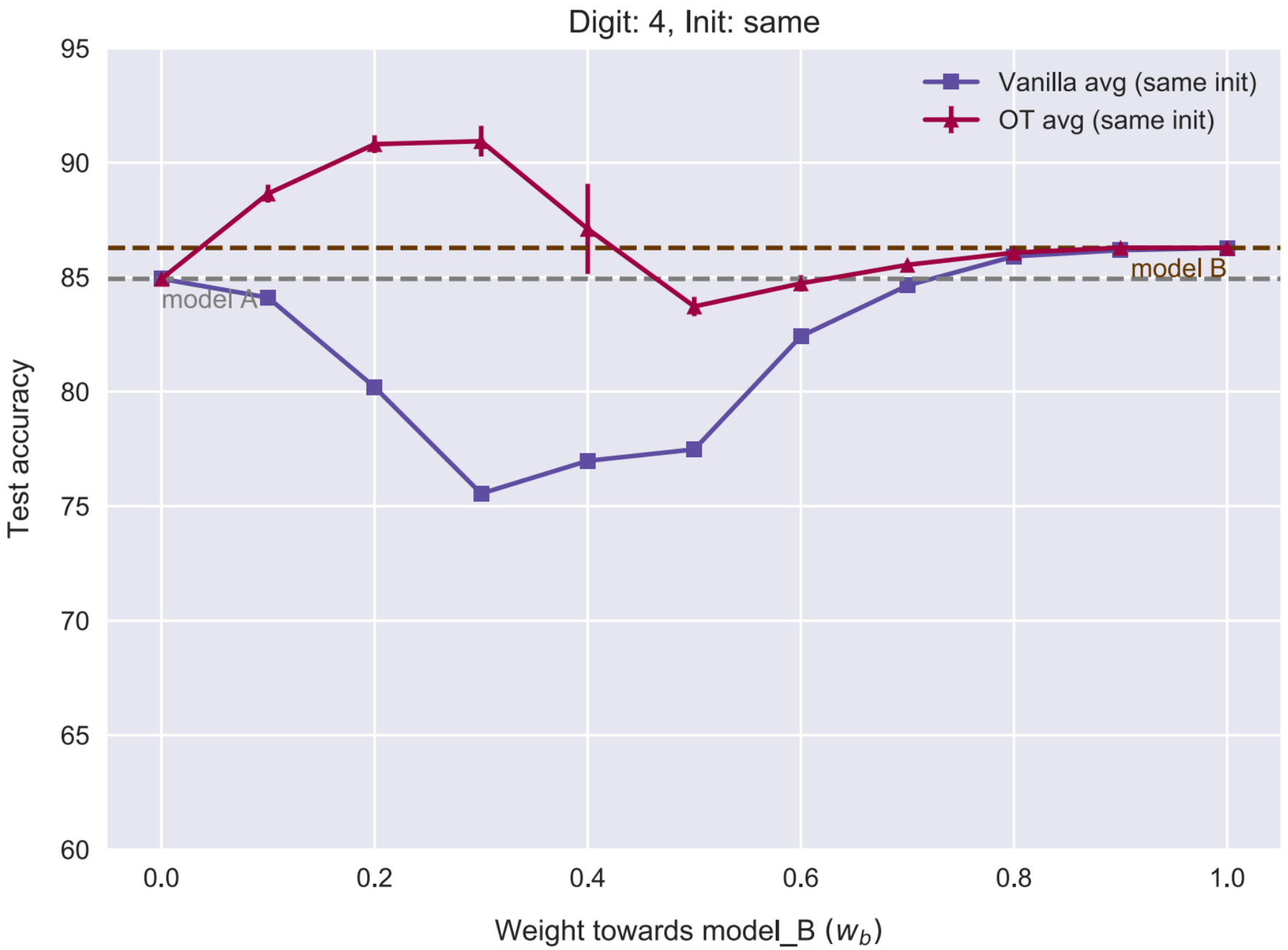


Different Initialization

Model A is a "specialist",
Model B is a "generalist".
How to get the best?

Over here, it is just one-shot averaging!

Applications: Skill Transfer, Federated Learning



Same Initialization

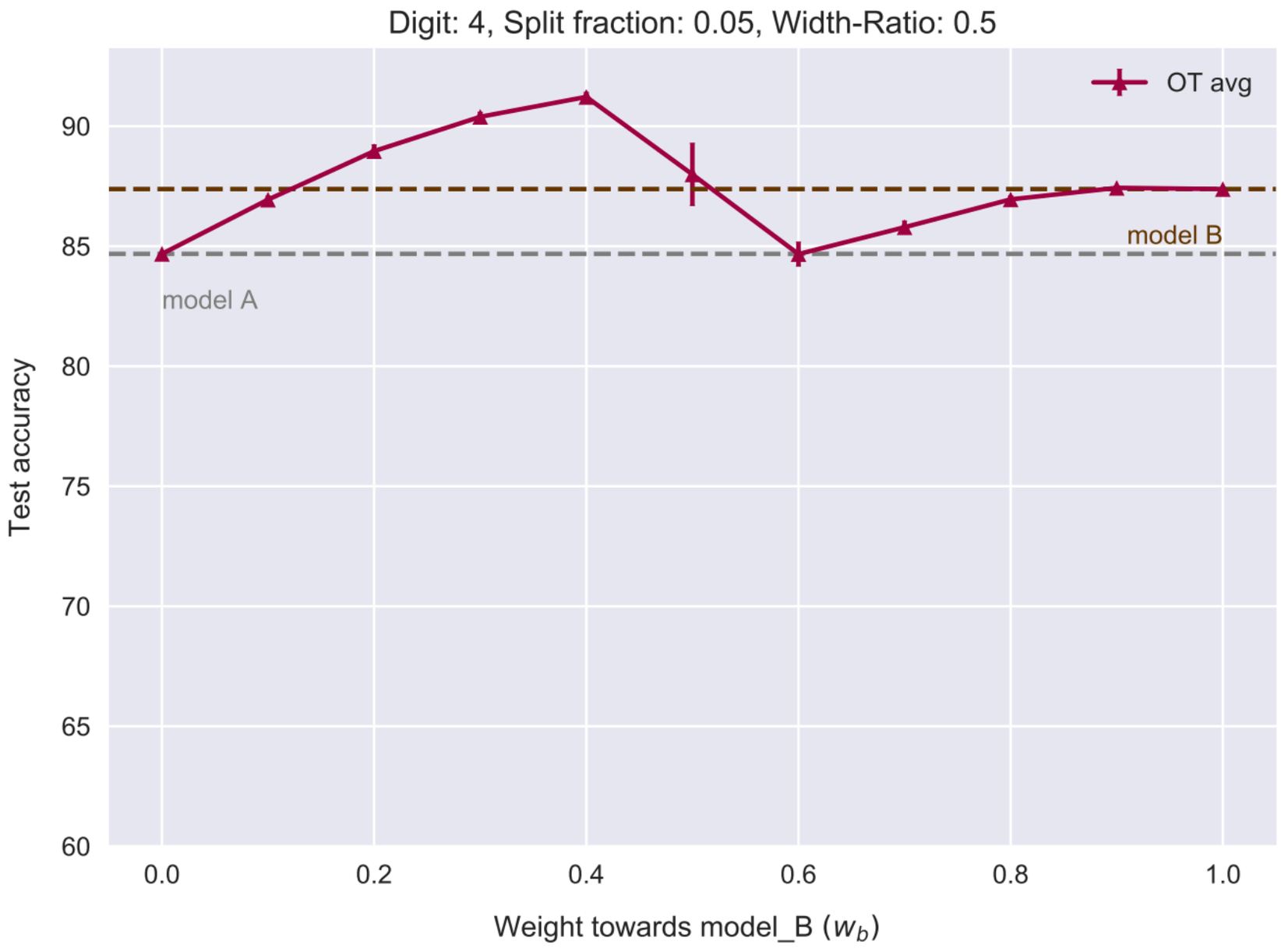
Model A is a "specialist",
Model B is a "generalist".
How to get the best?

Over here, it is just one-shot averaging!

Fusion for different sized models

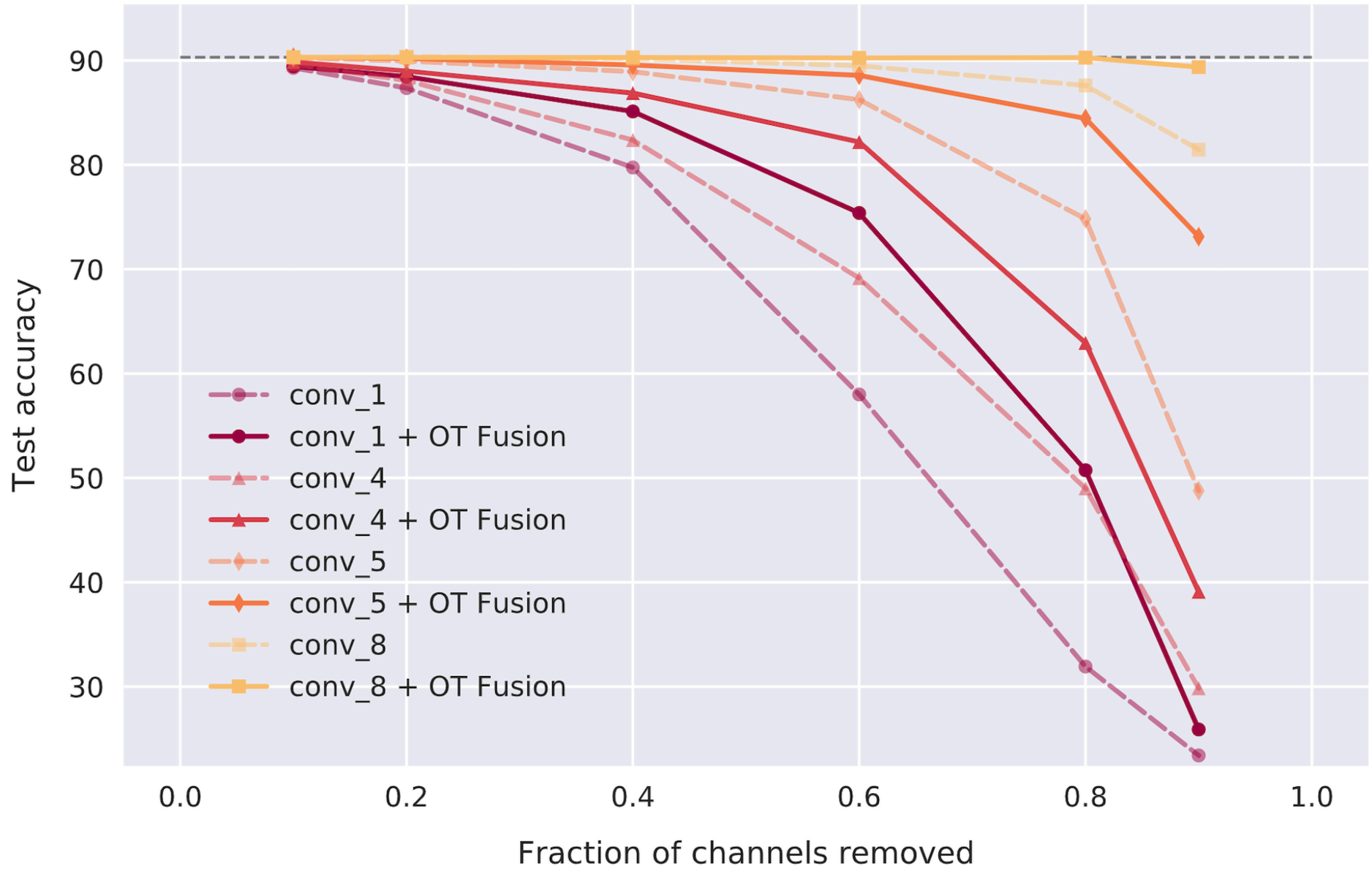
(OT can also provide soft-maps)

Applications: Adapting sizes across client-server



Model size at client != Model size at server

Applications: Post-processing tool for pruning



Data free!

VGG11, CIFAR10 (similar for CIFAR100)

Fusion for efficient ensembling

Applications: Efficient ensemble

Dataset + Model	M_A	M_B	Prediction avg.	Vanilla avg.	OT avg.	Finetuning Vanilla	Finetuning OT
CIFAR10 + VGG11	90.31	90.50	91.34	17.02	85.98	90.39	90.73
	1 ×		1 ×	2 ×	2 ×	2 ×	2 ×
CIFAR10 + ResNet18	93.11	93.20	93.89	18.49	77.00	93.49	93.78
	1 ×		1 ×	2 ×	2 ×	2 ×	2 ×

Two model case: ***CIFAR10***

Applications: Efficient ensemble

CIFAR10+ VGG11	Individual Models	Prediction avg.	Vanilla avg.	OT avg.	Finetuning Vanilla	Finetuning OT
Accuracy	[90.31, 90.50, 90.43, 90.51]	91.77	10.00	73.31	12.40	90.91
Efficiency	1 ×	1 ×	4 ×	4 ×	4 ×	4 ×
Accuracy	[90.31, 90.50, 90.43, 90.51, 90.49, 90.40]	91.85	10.00	72.16	11.01	91.06
Efficiency	1 ×	1 ×	6 ×	6 ×	6 ×	6 ×

Multiple models case: ***CIFAR10***

Applications: Efficient ensemble

CIFAR100 + VGG11	INDIVIDUAL MODELS	PREDICTION AVG.	FINETUNING VANILLA	FINETUNING OT
Accuracy	[62.70, 62.57, 62.50, 62.92]	66.32	4.02	**64.29± 0.26**
Efficiency	1 ×	1 ×	4 ×	**4** ×
Accuracy	[62.70, 62.57, 62.50, 62.92, 62.53, 62.70]	66.99	0.85	**64.55 ± 0.30**
Efficiency	1 ×	1 ×	6 ×	**6** ×
Accuracy	[62.70, 62.57, 62.50, 62.92, 62.53, 62.70, 61.60, 63.20]	67.28	1.00	**65.05± 0.53**
Efficiency	1 ×	1 ×	8 ×	**8** ×

Table 3: 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**.

Multiple models case: ***CIFAR100***

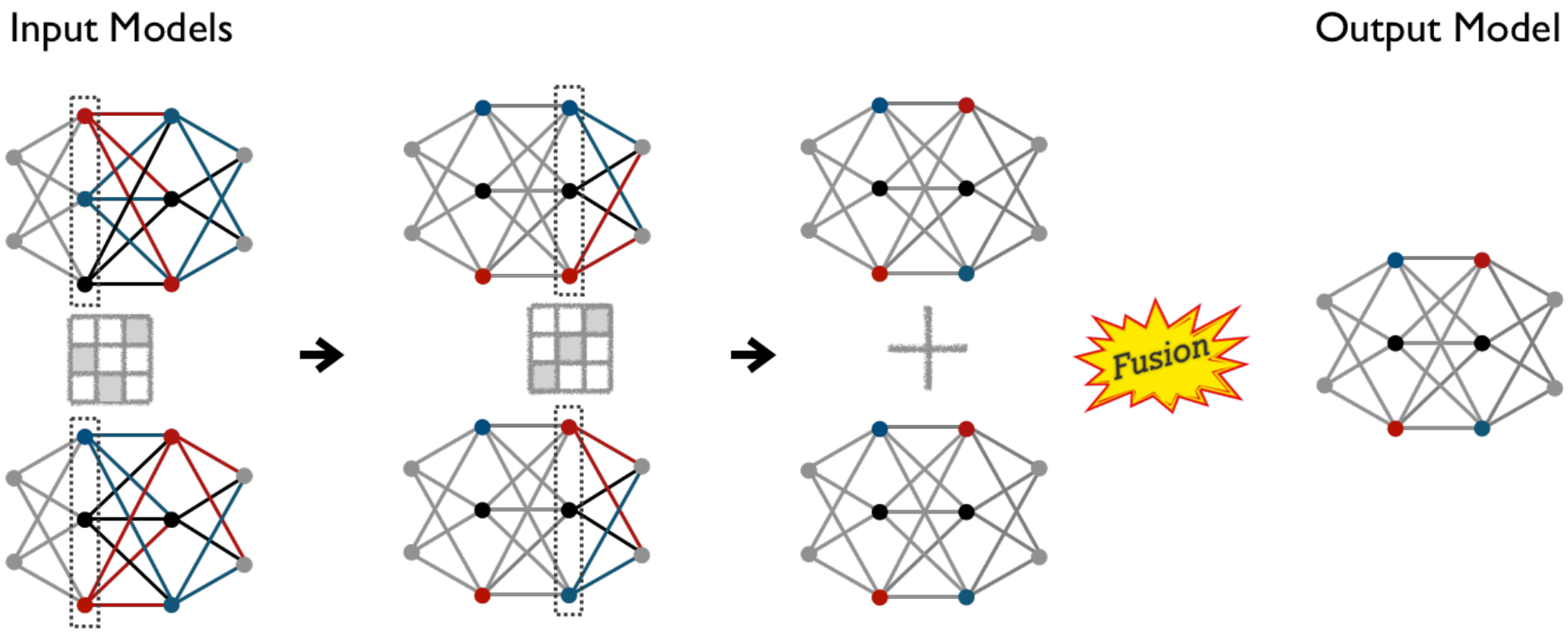
Conclusion

- Fuse multiple models (of possibly different widths):
  - Layerwise soft alignment of neurons via OT & then average (+ mild fine-tuning)
- Significantly outperforms vanilla averaging (accuracy) and ensembling (efficiency)
- Can be used alongside Distillation (as an initialization)
- Overall, this results in
  (a) Successful one-shot transfer of knowledge without sharing data
  (b) Data free and algorithm independent post-processing
  (c) Combine parameters of different sized models
  (d) Keep one model rather than an ensemble of models

Future work

- Open problem: How to fuse models with different depths?
- Adjust the size of the fused model
- Fusing generative models: GANs
- Continual learning
- Toolkit to compare & interpolate between models

Thank you!



sidakpal.com

https://github.com/modelfusion/otfusion