WASSERSTEIN BARYCENTER-BASED MODEL FUSION AND LINEAR MODE CONNECTIVITY OF NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Based on the concepts of Wasserstein barycenter (WB) and Gromov-Wasserstein barycenter (GWB), we propose a unified mathematical framework for neural network (NN) model fusion and utilize it to reveal new insights about the linear mode connectivity of SGD solutions. In our framework, the fusion occurs in a layer-wise manner and builds on an interpretation of a node in a network as a function of the layer preceding it. The versatility of our mathematical framework allows us to talk about model fusion and linear mode connectivity for a broad class of NNs, including fully connected NN, CNN, ResNet, RNN, and LSTM, in each case exploiting the specific structure of the network architecture. We present extensive numerical experiments to: 1) illustrate the strengths of our approach in relation to other model fusion methodologies and 2) from a certain perspective, provide new empirical evidence for recent conjectures which say that two local minima found by gradient-based methods end up lying on the same basin of the loss landscape after a proper permutation of weights is applied to one of the models.

1 INTRODUCTION

The increasing use of edge devices like mobile phones, tablets, and vehicles, along with the sophistication in sensors present in them (e.g. cameras, GPS, and accelerometers), has led to the generation of an enormous amount of data. However, data privacy concerns, communication costs, bandwidth limits, and time sensitivity prevent the gathering of local data from edge devices into one single centralized location. These obstacles have motivated the design and development of federated learning strategies which are aimed at pooling information from locally trained neural networks (NNs) with the objective of building strong centralized models without relying on the collection of local data (McMahan et al., 2017; Kairouz et al., 2019). Due to these considerations, the problem of NN fusion–i.e. combining multiple models which were trained differently into a single model–is a fundamental task in federated learning.

A standard fusion method for aggregating models with the same architecture is FedAvg (McMahan et al., 2017), which involves element-wise averaging of the parameters of local models. This is also known as vanilla averaging (Singh & Jaggi, 2019). Although easily implementable, vanilla averaging performs poorly when fusing models whose weights do not have a one-to-one correspondence. This happens because even when models are trained on the same dataset it is possible to obtain models that differ only by a permutation of weights (Wang et al., 2020; Yurochkin et al., 2019); this feature is known as *permutation invariance property* of neural networks. Moreover, vanilla averaging is not naturally designed to work when using local models with different architectures (e.g., different widths). In order to address these challenges, (Singh & Jaggi, 2019) proposed to first find the best alignment between the neurons (weights) of different networks by using optimal transport (OT) (Villani, 2008; Santambrogio, 2015; Peyré & Cuturi, 2018) and then carrying out a vanilla averaging step. In (Liu et al., 2022), the authors formulate the model fusion as a graph matching problem, which utilizes the second-order similarity of model weights to align neurons. Other approaches, like those proposed in (Yurochkin et al., 2019; Wang et al., 2020), interpret nodes of local models as random permutations of latent "global nodes" modeled according to a Beta-Bernoulli process prior (Thibaux & Jordan, 2007). By using "global nodes", nodes from different input NNs can be embedded into a common space where comparisons and aggregation are meaningful. Most

works in the literature discussing the fusion problem have mainly focused on the aggregation of fully connected (FC) neural networks and CNNs, but have not, for the most part, explored other kinds of architectures like RNNs and LSTMs. One exception to this general state of the art is the work (Wang et al., 2020), which considers the fusion of RNNs by ignoring hidden-to-hidden weights during the neurons' matching, thus discarding some useful information in the pre-trained RNNs. For more references on the fusion problem see in the Appendix A.1.

A different line of research that has attracted considerable attention in the past few years is the quest for a comprehensive understanding of the loss landscape of deep neural networks, a fundamental component in studying the optimization and generalization properties of NNs (Li et al., 2018; Mei et al., 2018; Neyshabur et al., 2017; Nguyen et al., 2018; Izmailov et al., 2018). Due to over-parameterization, scale, and permutation invariance properties of neural networks, the loss landscapes of DNNs have many local minima (Keskar et al., 2016; Zhang et al., 2021). Different works have asked and answered affirmatively the question of whether there exist paths of small-increasing loss connecting different local minima found by SGD (Garipov et al., 2018; Draxler et al., 2018). This phenomenon is often referred to as mode connectivity (Garipov et al., 2018) and the loss increase along paths between two models is often referred to as (energy) barrier (Draxler et al., 2018). It has been observed that low-barrier paths are non-linear, i.e., linear interpolation of two different models will not usually produce a neural network with small loss. These observations suggest that, from the perspective of local structure properties of loss landscapes, different SGD solutions belong to different (well-separated) basins (Neyshabur et al., 2020). However, recent work (Entezari et al., 2021) has conjectured that local minima found by SGD do end up lying on the same basin of the loss landscape *after* a proper permutation of weights is applied to one of the models. The question of how to find these desired permutations remains in general elusive.

The purpose of this paper is twofold. On one hand, we present a large family of barycenter-based fusion algorithms that can be used to aggregate models within the families of fully connected NNs, CNNs, ResNets, RNNs and LSTMs. The most general family of fusion algorithms that we introduce relies on the concept of Gromov-Wasserstein barycenter (GWB), which allows us to use the information in hidden-to-hidden layers in RNNs and LSTMs in contrast to previous approaches in the literature like that proposed in (Wang et al., 2020). In order to motivate the GWB based fusion algorithm for RNNs and LSTMs, we first discuss a Wasserstein barycenter (WB) based fusion algorithm for fully connected, CNN, and ResNet models which follows closely the OT fusion algorithm from (Singh & Jaggi, 2019). By creating a link between the NN model fusion problem and the problem of computing Wasserstein (or Gromov-Wasserstein) barycenters, our aim is to exploit the many tools that have been developed in the last decade for the computation of WB (or GWB) —see the Appendix A.2 for references— and to leverage the mathematical structure of OT problems. Using our framework, we are able to fuse models with different architectures and build target models with arbitrary specified dimensions (at least in terms of width). On the other hand, through several numerical experiments in a variety of settings (architectures and datasets), we provide new evidence backing certain aspects of the conjecture put forward in (Entezari et al., 2021) about the local structure of NNs' loss landscapes. Indeed, we find out that there exist sparse couplings between different models that can map different local minima found by SGD into basins that are only separated by low energy barriers. These sparse couplings, which can be thought of as approximations to actual permutations, are obtained using our fusion algorithms, which, surprisingly, only use training data to set the values of some hyperparameters. We explore this conjecture in imaging and natural language processing (NLP) tasks and provide visualizations of our findings. Consider, for example, Figure 1 (left), which is the visualization of fusing two FC NNs independently trained on the MNIST dataset. We can observe that the basins where model 1 and permuted model 2 (i.e. model 2 *after* multiplying its weights by the coupling obtained by our fusion algorithm) land are close to each other and are only separated by low energy barriers.

Our **main contributions** can then be summarized as follows: **(a)** we formulate the network model fusion problem as a series of Wasserstein (Gromov-Wasserstein) barycenter problems, bridging in this way the NN fusion problem with computational OT; **(b)** we empirically demonstrate that our framework is highly effective at fusing different types of networks, including RNNs and LSTMs. **(c)** we visualize the result of our fusion algorithm when aggregating two neural networks in a 2D-plane. By doing this we not only provide some illustrations on how our fusion algorithms perform, but also present empirical evidence for the conjecture made in (Entezari et al., 2021), casting light over the loss landscape of a variety of neural networks.

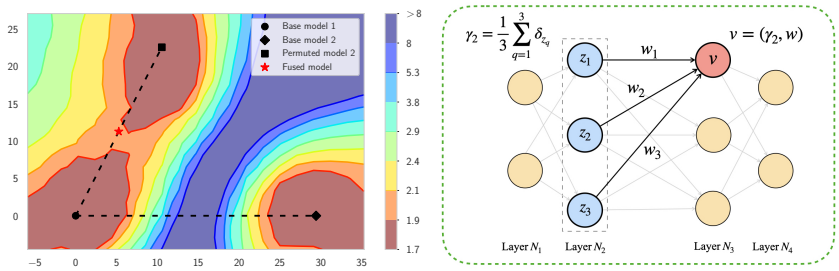

Figure 1: **Left:** The test error surface of FC NNs trained on MNIST. The permuted model 2 is model 2 *after* multiplying its weights by the coupling obtained by our fusion algorithm. **Right:** The illustration of our interpretations of FC NNs. Following our definitions, node $v := (\gamma_2, w)$, where γ_2 is a probability measure on layer N_2 and $w : N_2 \to \mathbb{R}$ is the weight function corresponding to node v. For example, the scalar $w(z_2)$ is the weight between nodes v and z_2, and we use w_2 as the shorthand notation of $w(z_2)$.

At the time of completing this work, we became aware of two very recent preprints which also explore the conjecture made in (Entezari et al., 2021) empirically. In particular, (Ainsworth et al., 2022) demonstrates that there is zero-barrier LMC (after permutation) between two independently trained NNs (including ResNet) provided the width of layers is large enough. In (Benzing et al., 2022), the conjecture is explored for FC NNs, finding that the average of two randomly initialized models using the permutation revealed through training gives a non-trivial NN. Compared to our work, none of these two works explored this conjecture for recurrent NNs; we highlight that our GWB fusion method is of particular relevance for this aim. To the best of our knowledge, we thus provide the first-ever exploration of the conjecture posited in (Entezari et al., 2021) for NLP tasks.

1.1 Notation

We first introduce some basic notation and briefly review a few relevant concepts from OT. A simplex of histograms with n bins is denoted by $\Sigma_n := \{a \in \mathbb{R}^n_+ : \sum_i a_i = 1\}$. The set of couplings between histograms $a \in \Sigma_{n_1}$ and $b \in \Sigma_{n_2}$ is denoted by $\Gamma(a, b) := \{\Pi \in \mathbb{R}^{n_1 \times n_2}_+ : \Pi \mathbb{1}_{n_2} = a, \Pi^T \mathbb{1}_{n_1} = b\}$, where $\mathbb{1}_n := (1, \dots, 1)^T \in \mathbb{R}^n$. For any 4-way tensor $\mathcal{L} = \left[\mathcal{L}_{ijkl}\right]_{i,j,k,l}$ and matrix $\Pi = \left[\pi_{ij}\right]_{i,j}$, we define the tensor-matrix multiplication of $\mathcal{L}$ and Π as the matrix $\mathcal{L} \otimes \Pi := \left[\sum_{k,l} \mathcal{L}_{ijkl}\pi_{kl}\right]_{i,j}$.

1.2 Optimal transport and Wasserstein barycenters

Let $\mathcal{X}$ be an arbitrary topological space and let $c : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ be a cost function assumed to satisfy $c(x, x) = 0$ for every x. We denote by $\mathcal{M}^+_1(\mathcal{X})$ the space of (Borel) probability measures on $\mathcal{X}$. For $\{x_i\}_{i=1}^{n_1}, \{y_j\}_{j=1}^{n_2} \in \mathcal{X}$, define discrete measures $\mu = \sum_{i=1}^{n_1} a_i \delta_{x_i}$ and $\nu = \sum_{j=1}^{n_2} b_j \delta_{y_j}$ in $\mathcal{M}^+_1(\mathcal{X})$, where $a \in \Sigma_{n_1}$, $b \in \Sigma_{n_2}$, and δ_x denotes the Dirac delta measure at $x \in \mathcal{X}$. The Wasserstein "distance" between μ and ν, relative to the cost c, is defined as

$$W(\mu, \nu) := \inf_{\Pi \in \Gamma(\mu,\nu)} \langle C, \Pi \rangle, \tag{1}$$

where $C := \left[c(x_i, y_j)\right]_{i,j}$ is the "cost" matrix between $\{x_i\}_i, \{y_j\}_j \in \mathcal{X}$, $\Pi := \left[\pi_{ij}\right]_{i,j} \in \Gamma(\mu, \nu)$ is the coupling matrix between μ and ν, and $\langle A, B \rangle := \mathrm{tr}(A^T B)$ is the Frobenius inner product.

Let $\{\gamma^i\}_{i=1}^n \in \mathcal{M}^+_1(\mathcal{X})$ be a collection of discrete probability measures. The Wasserstein barycenter problem (WBP) (Agueh & Carlier, 2011) associated with these measures reads

$$\min_{\gamma \in \mathcal{M}^+_1(\mathcal{X})} \frac{1}{n} \sum_{i=1}^{n} W(\gamma, \gamma^i). \tag{2}$$

A minimizer of this problem is called a Wasserstein barycenter (WB) of the measures $\{\gamma^i\}_{i=1}^n$ and can be understood as an average of the input measures. In the sequel we will use the concept of WB to define fusion algorithms for FC NN, CNN, and ResNet. For RNN and LSTM the fusion reduces

to solving a series of Gromov-Wasserstein barycenter-like problems (see the reviews of GWBP in the Appendix).

2 WASSERSTEIN BARYCENTER BASED FUSION

In this section, we discuss our layer-wise fusion algorithm based on the concept of WB. First we introduce the necessary interpretations of nodes and layers of NNs in Section 2.1. Next in Section 2.2, we describe how to compare layers and nodes across different NNs so as to make sense of aggregating models through WB. Finally we present our fusion algorithm in Section 2.3.

2.1 NESTED DEFINITION OF FULLY CONNECTED NN

For a *fully connected* network N, we use v to index the nodes in its l-th layer N_l. Let γ_l denote a probability measure on the l-th layer defined as the weighted sum of Dirac delta measure over the nodes in that layer, i.e.,

$$\gamma_l := \frac{1}{|N_l|} \sum_{v \in N_l} \delta_v \in \mathcal{M}_1^+(N_l). \tag{3}$$

We interpret a node v from the l-th layer as an element in N_l that couples a function on the domain N_{l-1} (previous layer) with a probability measure. In particular, the node v is interpreted as $v := (\gamma_{l-1}, w)$, where γ_{l-1} is a measure on the previous layer N_{l-1} and w represents the weights between the node v and the nodes in previous layer N_{l-1}. These weights can be interpreted as a function $w : N_{l-1} \to \mathbb{R}$ and we use the notation w_q to denote the value of function w evaluated at the q-th node in the previous layer N_{l-1}. For the first layer i.e. $l = 1$, the nodes simply represent placeholders for the input features. The above interpretation is illustrated in Figure 1 (right). This interpretation of associating nodes with a function of previous layer allows us to later define "distance" between nodes in different NNs (see Section 2.2) and is motivated from TL^p spaces and distance (García Trillos & Slepčev, 2015; Thorpe et al., 2017) which is designed for comparing signals with different domains (see more details in Appendix C.1).

2.2 COST FUNCTIONS FOR COMPARING LAYERS AND NODES

Having introduced our interpretations of NNs, we now define the cost functions for comparing layers and nodes which will be used to aggregate models through WB. Consider the l-th layers N_l and N_l' of two NNs N and N' respectively. We use Wasserstein distance between the measures γ_l and γ_l' over N_l and N_l' respectively to define distance between the layers:

$$d_\mu(\gamma_l, \gamma_l') := W(\gamma_l, \gamma_l') = \inf_{\Pi_l \in \Gamma(\gamma_l, \gamma_l')} \langle C_l, \Pi_l \rangle \tag{4}$$

where matrix $\Pi_l = [\pi_{l,jg}]_{j,g}$ is a *coupling* between the measures γ_l and γ_l'; and C_l is the cost matrix give by $C_l := \left[c_l(v, v')\right]_{v,v'}$, where c_l is a cost function between nodes on the l-th layers.

Following our inductive interpretation of NNs, the cost function c_l can also be defined inductively. Consider nodes v and v' from l-th layer of NNs N and N' respectively. For the first layer $l = 1$, we pick a natural candidate for cost function, namely $c_1(v, v') := \mathbb{1}_{v \neq v'}$, a reasonable choice given that all networks have the same input layer. For $l \geq 2$, recall our interpretation of nodes $v = (\gamma_{l-1}, w), v' = (\gamma_{l-1}', w')$, where γ_{l-1} and γ_{l-1}' denotes the respective measures associated with previous layer $l-1$ and w, w' denotes the respective weight functions for nodes v and v'. Since the domains of the weight functions w and w' are layers in different NNs, it is not clear how to compare them directly. However in TL^p interpretation, after finding a suitable coupling between the support measures γ_{l-1} and γ_{l-1}', one can couple the functions w and w' and use a direct L2-comparison. Motivated by computational and methodological considerations, we use a slight modification of the TL^p distance and decouple the problem for the measures from the weights. Specifically, we define $c_l(v, v') := d_\mu(\gamma_{l-1}, \gamma_{l-1}') + d_W(w, w')$; where d_μ is the Wasserstein distance (as defined in equation 4) between the measures γ_{l-1} and γ_{l-1}' from layers $l-1$. And d_W is defined using the *optimal coupling* of weight functions' support measures, i.e.,

$$d_W(w, w') := \sum_{q,s} \left(w_q - w_s'\right)^2 (\pi_{l-1,qs})^* =: \langle L(w, w'), (\Pi_{l-1})^* \rangle, \tag{5}$$

where $L(w, w') := \left[(w_q - w'_s)^2\right]_{q,s}$ and $(\Pi_{l-1})^* = \left[(\pi_{l-1,qs})^*\right]_{q,s}$ is the optimal coupling between γ_{l-1} and γ'_{l-1}. Note that $d_\mu(\gamma_{l-1}, \gamma'_{l-1})$ is a fixed constant when comparing any two nodes on the l-th layers N_l and N'_l. For simplicity, we let $c_l(v, v') = d_W(W, W')$ in what follows, and the information of support measures γ_{l-1} and γ'_{l-1} is implicitly included in their optimal coupling $(\Pi_{l-1})^*$. Here we have omitted bias terms to ease the exposition of our framework, but a natural implementation that accounts for bias terms can be obtained by simply concatenating them with the weight matrix.

We set $(\Pi_1)^*$ equal to the identity matrix normalized by the size of input layer given that this is a solution to equation 4 when the cost c_1 is defined as $c_1(v, \tilde{v}) := \mathbb{1}_{v \neq \tilde{v}}$. Other choices of cost function c_l are possible, e.g. the activation-based cost function proposed in (Singh & Jaggi, 2019).

2.3 Fusion algorithm

In the following we consider n input FC NNs $N^1, \ldots, N^n$. We use N_l^i to denote the l-th layer of the i-th network N^i and k_l^i to denote the number of nodes in that layer, i.e. $k_l^i = |N_l^i|$. Let γ_l^i to be the probability measure on layer N_l^i similar to definition in equation 3 with the support points being nodes in that layer. We denote the target model (i.e. the desired fusion output) by N^{new} and use $k_2, \ldots, k_m$ to denote the sizes of its layers $N_2^{\text{new}}, \ldots, N_m^{\text{new}}$, respectively. We assume that all networks, including the target model, have the same input layer and the same number of layers m.

Based on the discussion in Sections 2.1 and 2.2, we now describe an inductive construction of the layers of the target network N^{new} by fusing all n input NNs. First, N_1^{new} is set to be equal to N_1^1: this is the base case of the inductive construction and simply means that we set the input layer of N^{new} to be the same as that of the other models; we also set $\gamma_1 := \gamma_1^1$. Next, assuming that the fusion has been completed for the layers 1 to $l-1$ ($l \geq 2$), we consider the fusion of the l-th layer. For the simplicity of notations, we drop the index l while referring to nodes and their corresponding weights in this layer. In particular, we use v_g^i and w_g^i to denote the nodes in layer N_l^i and their corresponding weights. To carry out fusion of the l-th layer of the input models, we aggregate their corresponding measures through finding WB which provides us with a sensible "average" l-th layer for the target model. Hence, we consider the following WBP over $\gamma_l^1, \ldots, \gamma_l^n$:

$$\min_{\gamma_l, \{\Pi_l^i\}_i} \quad \frac{1}{n}\sum_{i=1}^n W(\gamma_l, \gamma_l^i) := \frac{1}{n}\sum_{i=1}^n \langle C_l^i, \Pi_l^i \rangle \qquad \text{s.t. } \gamma_l = \frac{1}{k_l}\sum_{j=1}^{k_l} \delta_{v_j},\ v_j = (\gamma_{l-1}, w_j). \tag{6}$$

Here the measure γ_l is the candidate l-th layer "average" of the input models and is forced to take a specific form (notice that we have fixed the size of its support and the masses assigned to its support points). Nodes v_j in the support of γ_l are set to take the form $v_j = (\gamma_{l-1}, w_j)$, i.e. the measure γ_{l-1} obtained when fusing the $(l-1)$-th layers is the first coordinate in all the v_j. This plugs the current layer of the target model with its previous layer. As done for the input models, w_j is interpreted as a function from the $(l-1)$-th layer into the reals, and represents the actual weight vector from the $(l-1)$-layer to the j-th node in the l-th layer of the new model. $C_l^i := \left[c_l(v_j, v_g^i)\right]_{j,g}$ are the cost matrices corresponding to WBP in equation 6, where c_l is a cost function between nodes on the l-th layers (see in Section 2.2). Let W_l and W_l^i to be the weight function matrices of the l-th layer of target models N^{new} and input model N^i respectively (e.g. $W_l := (w_1, \ldots, w_{k_l})^T$) and define $\mathcal{L}(W_l, W_l^i) := \left[\left(w_{jq} - w_{gs}^i\right)^2\right]_{j,g,q,s}$, where w_{jq} denotes the function w_j evaluated at the q-th node in layer $l-1$ and similarly for w_{gs}^i. The cost matrices C_l^i can now be rewritten as

$$C_l^i := \left[c_l(v_j, v_g^i)\right]_{j,g} = \left[d_W(w_j, w_g^i)\right]_{j,g} = \mathcal{L}(W_l, W_l^i) \otimes (\Pi_{l-1}^i)^*, \tag{7}$$

where $(\Pi_{l-1}^i)^*$ is the *optimal coupling* between measures γ_{l-1} and γ_{l-1}^i. Combining equation 7 with equation 6 gives us the following optimization problem which we solve to obtain the fused layer:

$$\min_{W_l, \{\Pi_l^i\}_i} B(W_l; \{\Pi_l^i\}_i) := \frac{1}{n}\sum_{i=1}^n \langle \mathcal{L}(W_l, W_l^i) \otimes (\Pi_{l-1}^i)^*, \Pi_l^i \rangle. \tag{8}$$

In order to solve the minimization problem 8, we can follow a strategy discussed in (Cuturi & Doucet, 2014; Anderes et al., 2016; Claici et al., 2018), i.e., alternatingly update weights W_l and

couplings $\{\Pi_l^i\}_i$ (remember that the $(\Pi_{l-1}^i)^*$ are computed once and for all and are fixed in equation 8. In particular, after initializing weight matrices, we alternate between two steps until some stopping criterion is reached:

Step 1: For fixed W_l, we update the couplings $\{\Pi_l^i\}_i$. Note that the minimization of $B(W_l; \{\Pi_l^i\}_i)$ over the couplings $\{\Pi_l^i\}_i$ splits into n OT problems, each of which can be solved using any of the algorithms used in computational OT (e.g. Sinkhorn's algorithm (Cuturi, 2013)).

Step 2: For fixed couplings $\{\Pi_l^i\}_i$, we update the weights W_l. Note that for fixed couplings the objective $B(W_l; \{\Pi_l^i\}_i)$ is quadratic in W_l and hence we obtain the following update formula:

$$W_l \leftarrow k_l k_{l-1} \frac{1}{\mathbb{1}_{k_{l-1}} \mathbb{1}_{k_l}^T} \frac{1}{n} \sum_{i=1}^{n} \Pi_l^i W_l^i (\Pi_{l-1}^i)^{*T}, \tag{9}$$

where $\frac{\cdot}{\cdot}$ is elementwise division. We refer to the above fusion algorithm as *Wasserstein barycenter-based fusion* (WB fusion). The pseudo-code for this algorithm and corresponding computational complexity can be found in the Appendix C, where we also provide some details on how to adapt our fusion method to handle convolutional layers and skip-connections.

3 Gromov-Wasserstein barycenter-based fusion

In this section we discuss extension of our fusion framework to cover RNNs and LSTMs. Compared to FC networks, RNNs contain "self-loops" in each layer (hidden-to-hidden recurrent connections) which allows information to be passed from one step of the neural network to the next. Similar to our interpretation of neurons in the FC case, a node v_g^i on the l-th layer will be represented as $v_g^i := \left[(\gamma_{l-1}^i, w_g^i); (\gamma_l^i, h_g^i)\right]$, where w_g^i is the weight function between inputs of the preceding layer and hidden states, and h_g^i is the weight function between hidden states; γ_{l-1}^i and γ_l^i are the probability measures corresponding to layer $l-1$ and layer l respectively. This definition comes from the observation that hidden-to-hidden weight functions h_g^i are supported on the l-th layer itself, whereas w_g^i is supported on the $(l-1)$-th layer.

Having carried out the fusion of the first $l-1$ layers we consider the following problem to fuse the l-th layers:

$$\min_{W_l, H_l, \{\Pi_l^i\}_i} B(W_l, H_l; \{\Pi_l^i\}_i) := \frac{1}{n} \sum_{i=1}^{n} \langle \mathcal{L}(W_l, W_l^i) \otimes (\Pi_{l-1}^i)^* + \alpha_H \mathcal{L}(H_l, H_l^i) \otimes \Pi_l^i, \Pi_l^i \rangle, \tag{10}$$

where α_H is a hyperparameter that balances the importance of input-to-hidden weights and hidden-to-hidden weights during the fusion; we'll use $(\Pi_l^i)^*$ to denote an optimal Π_l^i. We use H_l and H_l^i to denote the hidden-to-hidden weight function matrices of layer N_l^{new} and N_l^i respectively, and we let $\mathcal{L}(H_l, H_l^i) := \left[\left(h_{jq} - h_{gs}^i\right)^2\right]_{j,g,q,s}$. $\mathcal{L}(W_l, W_l^i)$ is defined the same as in the fully connected case. Notice that this is a GW-like barycenter problem.

We provide more detailed explanation on how to derive optimization problem 10 and adapt the GWB fusion for RNNs discussed in this section to the LSTM case in Appendix E. In Section 4.3 we show that the models obtained when setting $\alpha_H > 0$ in equation 10 greatly outperform the models obtained when setting $\alpha_H = 0$, justifying in this way the use of GWBs.

4 Experiments

Overview: We present an empirical study of our proposed WB and GWB based fusion algorithms to assess its performance in comparison to other state of the art fusion methodologies and reveal new insights about the loss landscapes for different types of network architectures and datasets. We first consider the fusion of models trained on heterogeneous data distributions. Next we present results for WB fusion of FC NNs and deep CNNs, and draw connections between workings of WB fusion and LMC of SGD solutions. Finally, we consider GWB fusion and present results on RNNs, LSTMs and extend the conjecture made in Entezari et al. (2021) for recurrent NNs.

Baselines: For baselines, we consider vanilla averaging and the state-of-the-art fusion methodologies like OT fusion Singh & Jaggi (2019) and FedMA Wang et al. (2020). For a fair comparison

under the experimental settings of one-shot fusion we consider FedMA without the model retraining step and restrict its global model to not outgrow the base models. We refer to this as "one-shot FedMA". For RNNs and LSTMs, our baselines additionally include slightly modified versions of WB based fusion and OT fusion where we ignore the hidden-to-hidden connections. Other methods which require extensive training are not applicable in one-shot model aggregation settings.

Base models & General-setup: For our experiments on FC NNs, we use MLPNET introduced in Singh & Jaggi (2019), which consists of 3 hidden layers of sizes $\{400, 200, 100\}$. Additionally, we introduce MLPLARGE and MLPSMALL with hidden layers of size $\{800, 400, 200\}$ and $\{200, 100, 50\}$ respectively. For deep CNNs, we use VGG11 Simonyan & Zisserman (2014) and RESNET18 He et al. (2016). For recurrent NNs, we work with RNNs and LSTMs with one hidden layer of size 256 and 4×256 respectively. Hyperparameters are chosen using a validation set and final results are reported on a held out test set. More training details are provided in the Appendix.

Visualization methodology: We visualize the result of fusing two pre-trained models on a two-dimensional subspace of NNs' loss landscape by using the method proposed in Garipov et al. (2018). In particular, each plane is formed by all affine combinations of three weight vectors corresponding to the parameters of base model 1, base model 2 and permuted model 2 (i.e. base model 2 *after* multiplying its weights by the coupling obtained by our fusion algorithm) respectively.

4.1 WB FUSION UNDER HETEROGENEOUS DATA DISTRIBUTIONS

Setup: We first apply WB fusion in aggregating models trained on heterogeneous data distributions which is a setting often found in federated learning where the clients have local data generated from different distributions and privacy concerns prevent data sharing among them. Here we follow the setup described in Singh & Jaggi (2019). To simulate heterogeneous data-split on MNIST digit classification one of the models (named A) is trained with a special skill to recognize one of the digits (eg. digit 4) that is not known to the other model, named B. Model B is trained on 90% of the training data for remaining digits while model A uses the other 10% data. Under this data split, we consider two settings. For the first setting, the base models are fused into a target model of the same architecture (MLPNET). For the second setting, we consider the fusion of two small base models (MLPSMALL) into a large target model (MLPNET). This simulates the setting where clients in federated learning are constrained by memory resources to train smaller models. In both cases we use model fusion to aggregate knowledge learned from the base models into a single model, a more memory-efficient strategy than its ensemble-based counterparts.

Quantitative results: Figure 2 shows the results of single shot fusion when different proportions of the base models are considered. We find that (a) WB fusion consistently outperforms the baselines, (b) for certain combinations WB produces fused models with accuracy even better than the base model and demonstrates successful one shot knowledge aggregation. Note that for each proportion of model aggregation (x-axis), the results are reported over multiple runs where one of the base models is randomly chosen to initialize the target model in the fusion algorithm. We find that WB fusion is more stable against initialization as indicated by the lower variance in Figure 2. For fusion into different architectures vanilla averaging is not applicable, and we do not include "one-shot FedMA" for comparison here since it is not clear how to assign different proportions to base models in FedMA, or to specify a target architecture different from the base models.

4.2 WB FUSION UNDER HOMOGENEOUS DATA DISTRIBUTIONS AND CONNECTIONS TO LMC

Setup: In this section we perform WB fusion for various models and architectures, and provide loss landscape visualizations which reveal workings of the fusion algorithm and shed light on linear model connectivity (LMC) of SGD solutions after applying appropriate permutations. We first consider fusion of FC NNs on the MNIST dataset Deng (2012) and train MLPNET following Singh & Jaggi (2019). For this we consider two different settings. In the first setting, the target model has the same architecture as the base models. For the second one, we fuse the base models into a larger model MLPLARGE. As noted before, the latter scenario is relevant for federated learning, given the limitations of memory and computational resources on edge devices. Next, we consider fusion of deep CNNs like VGG11, RESNET18 trained on CIFAR10 dataset Krizhevsky et al. (2009). For all these cases, we fuse 2 trained models initialized differently. For the skip-connection and fusion into different architectures, FedMA is not directly applicable and hence not considered for comparisons.

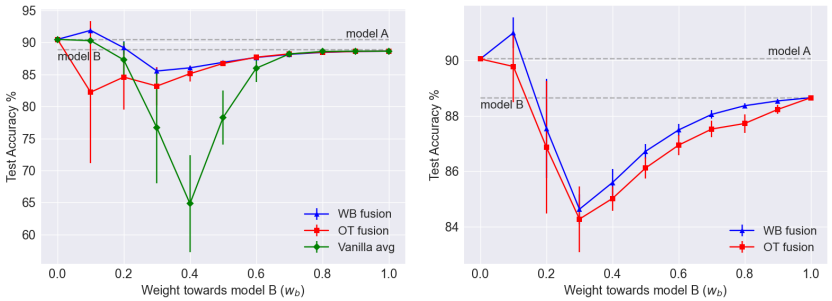

Figure 2: **Left / Right:** Test accuracy % for fused models when base models are trained on heterogeneous data distributions and combined with various proportions into a target model of **same / different** architecture. Some models obtained by WB fusion outperform even the base models.

Quantitative results: Table 1 contains the results of fusion for FC NNs and deep CNNs. We find that (a) WB fusion produces models at par or outperforms other fusion methods for all considered model types and datasets, (b) for fusion into different architectures and ResNets, we find that WB fusion is more effective and robust.

Table 1: Performance comparison (Test accuracy $\pm$ standard deviation %) of different fusion algorithms under various network architectures and datasets. "BASE" means initializing target model with one of the base models. For each case, the target model obtained by WB fusion gets the highest test accuracy and smallest standard deviation.

	MNIST		CIFAR10	
	MLPNET/BASE	MLPLARGE	VGG11/BASE	RESNET18/BASE
BASE MODEL AVG	98.31 ± 0.02	-	90.14 ± 0.19	91.56 ± 0.34
VANILLA AVG	86.50 ± 4.60	-	30.82 ± 4.49	20.56 ± 3.90
ONE-SHOT FEDMA	97.89 ± 0.10	-	$\mathbf{85.42 \pm 1.01}$	-
OT	97.84 ± 0.12	91.53 ± 2.64	85.39 ± 0.93	71.37 ± 6.53
WB	97.92 ± 0.12	$\mathbf{94.93 \pm 1.18}$	85.39 ± 0.93	$\mathbf{73.75 \pm 4.39}$

Visualizations: Figure 1 (left) contains the visualization of fusing two MLPNET trained on MNIST dataset under WB framework and Figure 3 (left) contains the fusion result of WB fusion of two VGG11 models trained on CIFAR10. We find that (a) the couplings obtained in WB fusion (refer to equation 8) between the layers of target model and base models are sparse, i.e. they are almost permutations; (b) the basins of the permuted model 2 (obtained by multiplying the weights of base model 2 by the found couplings) and base model 1 lie close to each other and are separated by a low energy barrier. These visualizations thus provide new empirical evidence in support of the conjecture made in Entezari et al. (2021). They also shed light on the workings on WB fusion algorithm. In particular, equation 9 can be interpreted as coordinate-wise averaging of the permuted models. Since permuted models land in basins that are separated by low energy barriers, their linear interpolation gives a good fused model.

4.3 GWB FUSION FOR RECURRENT NEURAL NETWORKS

Setup: In this section, we consider the fusion of NNs like RNNs and LSTMs on sequence based tasks. We use 4 different datasets for this setting: i) MNIST Deng (2012): Images of 28×28 dimensions are interpreted as 28 length sequences of vectors $\in \mathbb{R}^{28}$; ii) SST-2 Socher et al. (2013): Binary classification task of predicting positive and negative phrases; iii) AGNEWS Zhang et al. (2015): Corpus of news articles from 4 classes; and iv) DBpedia Zhang et al. (2015): Ontology classification dataset containing 14 non-overlapping classes. For the NLP tasks, we use pre-trained GloVe embeddings Pennington et al. (2014) of dimensions 100 and 50 for RNNs and LSTMs respectively. The embedding layer is not updated during the model training. We set the target model to have the same architecture as the base models.

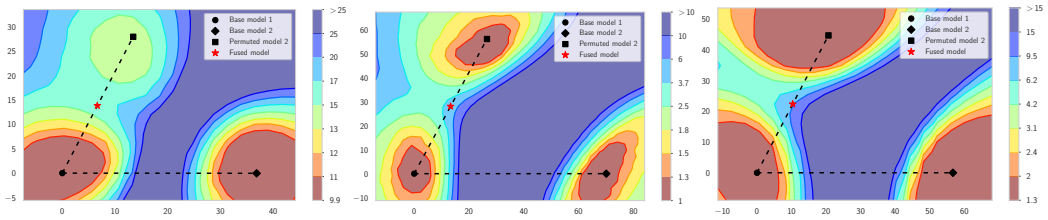

Figure 3: Visualizations of the fusion results on the test error surface, which is a function of network weights in a two-dimensional subspace, for different models and datasets. **Left:** Fusion of two VGG11 models trained on CIFAR10 dataset using WB framework. **Middle:** Fusion of two LSTM models trained on MNIST dataset . **Right:** Fusion of two LSTM models trained on DBpedia dataset. We can observe that in all these cases the basins of permuted model 2 (obtained by multiplying the weights of base model 2 by the found coupling) and base model 1 lie close to each other and are separated by a low energy barrier.

Quantitative results: Table 2 contains the result of fusion for various datasets and model architectures. We find that (a) our GWB framework outperforms other fusion algorithms for each combination of model type and dataset, which highlights the importance of using hidden-to-hidden connections for the fusion of recurrent NNs; (b) the accuracy gains for GWB over WB is different for different tasks, which indicates that relative importance of hidden-to-hidden connections is task dependent; (c) the accuracy of fused model is higher for LSTMs in comparison to RNNs, which we attribute to the fact that LSTMs have four hidden states and thus four input-to-hidden and hidden-to-hidden weight matrices. More information for each hidden node allows the algorithm to uncover better couplings. Our results in (a) and (b) show the usefulness of hyperparameter α_H (set between $[1, 20]$) from equation 10 in balancing the relative importance of hidden-to-hidden weights.

Table 2: Performance comparison (Test accuracy $\pm$ standard deviation %) of different fusion algorithms under various network architectures and datasets. For each case, target model obtained by GWB fusion reaches the highest test accuracy and small standard deviation.

	MNIST		AGNEWS		DBPEDIA		SST-2	
	RNN	LSTM	RNN	LSTM	RNN	LSTM	RNN	LSTM
BASE MODEL AVG	96.68 ± 0.29	98.99 ± 0.09	88.68 ± 0.12	92.38 ± 0.17	97.12 ± 0.21	98.62 ± 0.11	87.32 ± 1.03	90.31 ± 0.27
VANILLA AVG	28.54 ± 10.70	31.92 ± 4.86	40.77 ± 4.94	74.01 ± 3.89	30.95 ± 4.53	50.93 ± 2.17	73.91 ± 2.73	74.25 ± 1.92
OT	36.78 ± 14.13	68.33 ± 7.07	53.05 ± 4.30	86.19 ± 2.14	37.91 ± 4.86	77.95 ± 3.20	78.92 ± 2.97	82.13 ± 0.60
ONE-SHOT FEDMA	34.16 ± 7.26	66.98 ± 5.17	55.78 ± 3.64	86.30 ± 2.40	42.16 ± 6.24	81.81 ± 3.29	79.17 ± 2.27	82.53 ± 1.01
WB	29.41 ± 7.05	67.66 ± 6.27	55.63 ± 4.18	86.25 ± 2.37	42.52 ± 6.26	82.57 ± 3.55	79.57 ± 2.36	82.87 ± 1.09
GWB	$\mathbf{81.39 \pm 2.97}$	$\mathbf{93.27 \pm 1.86}$	$\mathbf{61.01 \pm 3.87}$	$\mathbf{87.96 \pm 0.91}$	$\mathbf{55.15 \pm 5.97}$	$\mathbf{87.50 \pm 2.89}$	$\mathbf{82.60 \pm 1.05}$	$\mathbf{84.04 \pm 0.77}$

Visualizations: Figure 3 (middle, right) contains visualization of fusing LSTM models under the GWB framework. As noted for the FC NNs and deep CNNs visualizations, we find that (a) the couplings found by GWB fusion algorithm are sparse, and (b) these couplings map different local minima into neighboring basins that are separated by low energy barriers. This empirical evidence suggests that the original conjecture in Entezari et al. (2021) can be extended to richer network architectures and tasks (RNNs and LSTMs on NLP datasets).

5 CONCLUSION

In this paper we have proposed neural network fusion algorithms that are based on the concept of Wasserstein/Gromov-Wasserstein barycenter. Our fusion algorithms allow us to aggregate models within a variety of NN architectures, including RNN and LSTM. Through extensive experimentation we: 1) illustrated the strengths of our algorithms 2) provided new empirical evidence backing recent conjectures about the linear mode connectivity of different neural networks with architectures such as RNN or LSTM and for different imaging and NLP datasets.

Limitations and future work: NNs with ReLU activation are also scale-invariant across the layers which is currently not handled in our cost functions. Although the empirical evidence in (Du et al.,

2018; Entezari et al., 2021) suggests that the models trained on same datasets using SGD converges to solutions with more balanced weights, it might be the case that for certain heterogeneous settings the weights across models become less balanced. For future work we would like to explore fusion using scale-invariant cost functions and apply WB/GWB fusion algorithms to federated learning.

REFERENCES

Martial Agueh and Guillaume Carlier. Barycenters in the wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924, 2011.

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

Ethan Anderes, Steffen Borgwardt, and Jacob Miller. Discrete wasserstein barycenters: Optimal transport for discrete data. *Mathematical Methods of Operations Research*, 84(2):389–409, 2016.

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015.

Frederik Benzing, Simon Schug, Robert Meier, Johannes von Oswald, Yassir Akram, Nicolas Zucchet, Laurence Aitchison, and Angelika Steger. Random initialisations performing above chance and how to find them. *arXiv preprint arXiv:2209.07509*, 2022.

Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974*, 2020.

Sebastian Claici, Edward Chien, and Justin Solomon. Stochastic wasserstein barycenters. *arXiv preprint arXiv:1802.05757*, 2018.

Sebastian Claici, Mikhail Yurochkin, Soumya Ghosh, and Justin Solomon. Model fusion with kullback–leibler divergence. *arXiv preprint arXiv:2007.06168*, 2020.

N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1853–1865, 2017. doi: 10.1109/TPAMI.2016.2615921.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pp. 2292–2300, 2013.

Marco Cuturi and Arnaud Doucet. Fast computation of wasserstein barycenters. 2014.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Pierre Dognin, Igor Melnyk, Youssef Mroueh, Jerret Ross, Cicero Dos Santos, and Tom Sercu. Wasserstein barycenter model ensembling. *arXiv preprint arXiv:1902.04999*, 2019.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pp. 1309–1318. PMLR, 2018.

Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in Neural Information Processing Systems*, 31, 2018.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

Nicolás García Trillos and Dejan Slepčev. Continuum limit of total variation on point clouds. *Archive for Rational Mechanics and Analysis*, pp. 1–49, 2015. ISSN 1432-0673. doi: 10.1007/s00205-015-0929-z. URL http://dx.doi.org/10.1007/s00205-015-0929-z.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Nhat Ho, XuanLong Nguyen, Mikhail Yurochkin, Hung Hai Bui, Viet Huynh, and Dinh Phung. Multilevel clustering via wasserstein means. *arXiv preprint arXiv:1706.03883*, 2017.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Paul Knopp and Richard Sinkhorn. A note concerning simultaneous integral equations. *Canadian Journal of Mathematics*, 20:855–861, 1968.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Xin-Chun Li, Yi-Chu Xu, Shaoming Song, Bingshuai Li, Yinchuan Li, Yunfeng Shao, and De-Chuan Zhan. Federated learning with position-aware neurons. *arXiv preprint arXiv:2203.14666*, 2022.

Chang Liu, Chenfei Lou, Runzhong Wang, Alan Yuhan Xi, Li Shen, and Junchi Yan. Deep neural network fusion via graph matching with applications to model ensemble and federated learning. In *International Conference on Machine Learning*, pp. 13857–13869. PMLR, 2022.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4):417–487, 2011.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523, 2020.

Dang Nguyen, Khai Nguyen, Dinh Phung, Hung Bui, and Nhat Ho. Model fusion of heterogeneous neural networks via cross-layer alignment. *arXiv preprint arXiv:2110.15538*, 2021.

Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. On the loss landscape of a class of deep neural networks with no bad local valleys. *arXiv preprint arXiv:1809.10749*, 2018.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pp. 2664–2672. PMLR, 2016.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. 2018.

Ievgen Redko, Nicolas Courty, Rémi Flamary, and Devis Tuia. Optimal transport for multi-source domain adaptation under target shift. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 849–858, 2019.

Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55(58-63):94, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *arXiv preprint arXiv:1910.05653*, 2019.

Richard Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405, 1967.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Sanvesh Srivastava, Volkan Cevher, Quoc Dinh, and David Dunson. Wasp: Scalable bayes via barycenters of subset posteriors. In *Artificial Intelligence and Statistics*, pp. 912–920, 2015.

Sanvesh Srivastava, Cheng Li, and David B Dunson. Scalable bayes via barycenter in wasserstein space. *The Journal of Machine Learning Research*, 19(1):312–346, 2018.

Romain Thibaux and Michael I. Jordan. Hierarchical beta processes and the indian buffet process. 2007.

Matthew Thorpe, Serim Park, Soheil Kolouri, Gustavo K Rohde, and Dejan Slepčev. A transportation l^p distance for signal analysis. *Journal of mathematical imaging and vision*, 59(2):187–210, 2017.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

Hongteng Xu, Dixin Luo, and Lawrence Carin. Scalable gromov-wasserstein learning for graph partitioning and matching. *Advances in neural information processing systems*, 32:3052–3062, 2019a.

Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin Duke. Gromov-wasserstein learning for graph matching and node embedding. In *International conference on machine learning*, pp. 6932–6941. PMLR, 2019b.

Fuxun Yu, Weishan Zhang, Zhuwei Qin, Zirui Xu, Di Wang, Chenchen Liu, Zhi Tian, and Xiang Chen. Fed2: Feature-aligned federated learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2066–2074, 2021.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. *arXiv preprint arXiv:1905.12022*, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

A Related works

A.1 Other model fusion algorithms

Different from the post-processing strategies proposed in (Singh & Jaggi, 2019; Yurochkin et al., 2019; Wang et al., 2020; Liu et al., 2022), which propose aligning different models *after* training, works like (Yu et al., 2021; Li et al., 2022) solve the alignment problem *during* the training process of local models. In particular, the work (Yu et al., 2021) proposes to align features during training by separating features into different groups; (Li et al., 2022), on the other hand, proposes to break permutation invariance by adding position encodings during training. With any of these approaches coordinate-wise (vanilla) averaging becomes sensible and local models can get aligned during local updates using direct averaging. The work (Claici et al., 2020) considers a general model fusion problem (e.g. topic models, not restricted to NN) and takes a Bayesian approach. In their framework, the parameters specifying the target global model are obtained by minimizing the sum of KL distances between "posterior distributions" of global and local models. (Chen & Chao, 2020) also takes a Bayesian perspective: local models are used to estimate an ensemble of Gaussian or Dirichlet distributions and the output ensemble is used to generate the global model using knowledge distillation. In (Nguyen et al., 2021), the problem of fusing NNs with different number of layers is considered. Their idea is to first apply dynamic programming to find one-to-one cross-layer alignments. Based on these alignments, the number of layers of local NNs gets balanced, and they can then reduce their fusion problem to one where existing layer-wise fusion methods can be used.

A.2 Computation of Wasserstein barycenters

Solving Wasserstein barycenter problems (WBP) has become computationally more feasible because of recent substantial advancements in OT algorithms. This development started with the work (Cuturi, 2013) where they proposed to add an entropic regularization term in the transport problem and use the Sinkhorn algorithm (Sinkhorn, 1967; Knopp & Sinkhorn, 1968) to solve it. (Cuturi & Doucet, 2014) and (Benamou et al., 2015) extended the entropic regularization idea to the barycenter problem and proposed to solve it through sub-gradient descent and Bregman projection iterations respectively. In (Claici et al., 2018), the authors proposed to alternatively update the measure support and the weights of the barycenter by using stochastic algorithms. The work (Peyré et al., 2016) proposed to use projected gradient descent to solve the minimization problem corresponding to the entropic Gromov-Wasserstein (GW) discrepancy, which is shown to be equivalent to solve a entropy-regularized OT problem at each iteration. (Xu et al., 2019a;b) replaced the entropy regularizer with KL-divergence in the GW discrepancy and similarly solve it with proximal point methods. Both of the above methods are applied to solve GWBP. Thanks to the development of these computational techniques, Wasserstein barycenters have been applied to various machine learning problems such as clustering (Ho et al., 2017), Bayesian inference (Srivastava et al., 2015; 2018), domain adaptation (Redko et al., 2019), ensemble learning (Dognin et al., 2019), among others. Here we apply some of these computational tools for NN fusion.

B Computational optimal transport

In this section, we review the computational optimal transport methods used in our paper.

B.1 Entropic Optimal Transport

Let $p \in \Sigma_{N_1}$ and $q \in \Sigma_{N_2}$ be two histograms and let $C \in \mathbb{R}_+^{N_1 \times N_2}$ be a cost matrix, where C_{ij} represents the transportation cost between positions indexed by i and j. Define the solution of

entropically-regularized optimal transport between p and q as

$$\mathcal{T}(C, p, q) := \underset{\Pi \in \Gamma(p,q)}{\arg\min} \langle C, \Pi \rangle - \varepsilon H(\Pi), \tag{11}$$

where $H(\Pi) := -\sum_{i,j=1}^{N} \Pi_{ij}(\log \Pi_{ij} - 1)$ is the entropy of Π. It can be shown that the solution to this problem reads $\mathcal{T}(C, p, q) = \text{diag}(a) K \text{diag}(b)$, where $K := e^{-\frac{c}{\varepsilon}} \in \mathbb{R}_+^{N_1 \times N_2}$ is the so-called Gibbs kernel associated to c, and $(a, b) \in \mathbb{R}_+^{N_1} \times \mathbb{R}_+^{N_2}$ can be computed using Sinkhorn iterations (Cuturi, 2013)

$$a \leftarrow \frac{p}{Kb} \quad \text{and} \quad b \leftarrow \frac{q}{K^\top a}, \tag{12}$$

where here $\frac{\cdot}{\cdot}$ denotes elementwise division.

B.2 Gromov-Wasserstein distance and Gromov-Wasserstein barycenters

The Gromov-Wasserstein problem (Mémoli, 2011; Peyré et al., 2016) is a variant of the OT problem introduced with the purpose of comparing different spaces when each of them is endowed with a base probability distribution and a notion of similarity (or dissimilarity) between pairs of its elements. For two matrices $C \in \mathbb{R}^{n_1 \times n_1}$ and $\bar{C} \in \mathbb{R}^{n_2 \times n_2}$ representing the similarity between pairs of points in the support of $\mu = \sum_{i=1}^{n_1} a_i \delta_{x_i}$ and $\nu = \sum_{j=1}^{n_2} b_j \delta_{y_j}$ respectively, we define the Gromov-Wasserstein distance between the two measured similarity matrices $(C, a) \in \mathbb{R}^{n_1 \times n_1} \times \Sigma_{n_1}$ and $(\bar{C}, b) \in \mathbb{R}^{n_2 \times n_2} \times \Sigma_{n_2}$ as follows:

$$GW((C, a), (\bar{C}, b)) := \min_{\Pi \in \Gamma(a,b)} \langle \mathcal{L}(C, \bar{C}) \otimes \Pi, \Pi \rangle, \tag{13}$$

where $\mathcal{L}(C, \bar{C}) := \left[L(C_{ik}, \bar{C}_{jl})\right]_{i,j,k,l}$ and L is some loss function to account for the misfit between the similarity matrices. In direct analogy with the barycenter problem in optimal transport, we define the Gromov-Wasserstein barycenter problem (GWBP) for the measured similarity matrices $\{(C_i, b_i)\}_{i=1}^n$ using a Fréchet mean formulation:

$$\min_{C \in \mathbb{R}^{m \times m}, \{\Pi^i\}_i} \frac{1}{n} \sum_{i=1}^{n} GW((C, a), (C_i, b_i)) = \frac{1}{n} \sum_{i=1}^{n} \langle \mathcal{L}(C, C_i) \otimes \Pi^i, \Pi^i \rangle, \tag{14}$$

where we assume $a \in \Sigma_m$ to be known. A minimizer of this problem is called Gromov-Wasserstein barycenter (GWB) of the measured similarity matrices $\{(C_i, b_i)\}_{i=1}^n$ (for fixed a). We used the concept of GWB to define fusion algorithms for RNN and LSTM—see Section 3.

B.3 Entropic Gromov-Wasserstein distance

In order to solve the minimization problem (13), consider the following entropic approximation of the initial GW formulation:

$$GW_\varepsilon((C, p), (\bar{C}, q)) := \min_{\Pi \in \Gamma(p,q)} \mathcal{E}_{C,\bar{C}}(\Pi) - \varepsilon H(\Pi). \tag{15}$$

In (Peyré et al., 2016), the authors propose to use projected gradient descent to solve this non-convex optimization problem, where both the gradient step and the projection are computed according to the Kullback-Leibler (KL) divergence. Iterations of the corresponding algorithm are given by

$$T \leftarrow \text{Proj}^{\text{KL}}_{\Gamma(p,q)}\left(T \odot e^{-\tau(\nabla \mathcal{E}_{C,\bar{C}}(\Pi) - \varepsilon \nabla H(\Pi))}\right), \tag{16}$$

where $\tau > 0$ is a small enough step size, and the KL projector of any matrix K is defined as

$$\text{Proj}^{\text{KL}}_{\Gamma(p,q)} := \underset{\Pi' \in \Gamma(p,q)}{\arg\min} KL(\Pi' | K). \tag{17}$$

Propostition 1 (Proposition 2 in (Peyré et al., 2016)). *In the special case $\tau = \frac{1}{\varepsilon}$, iteration (16) reads*

$$\Pi \leftarrow \mathcal{T}(\mathcal{L}(C, \bar{C}) \otimes \Pi, p, q). \tag{18}$$

C DETAILS ON WB FUSION USING TL^p FORMALISM

C.1 TL^p SPACE

We first review concepts from TL^p spaces which we use to motivate our interpretations of nodes in NNs. Let $\mu \in \mathcal{M}_1^+(\mathcal{X})$ be a probability measure and let $L^p(\mu; \mathbb{R}^r) := \{f : \mathcal{X} \to \mathbb{R}^r \mid \int_{\mathcal{X}} \|f\|_p^p d\mu(x) < \infty\}$. We define the TL^p space associated to $\mathcal{X}$ (see (García Trillos & Slepčev, 2015; Thorpe et al., 2017) and references within) as

$$TL^p(\mathcal{X}) := \{(\mu, f) \,|\, \mu \in \mathcal{M}_1^+(\mathcal{X}), f \in L^p(\mu; \mathbb{R}^r)\}. \tag{19}$$

The TL^p distance for pairs $(\mu_1, f_1), (\mu_2, f_2) \in TL^p(\mathcal{X})$ is defined by

$$d_{TL^p}^p\big((\mu_1, f_1), (\mu_2, f_2)\big) := \min_{\pi \in \Gamma(\mu_1, \mu_2)} \int_{\mathcal{X} \times \mathcal{X}} c(x_1, x_2; f_1, f_2) d\pi, \tag{20}$$

where the cost function is chosen to be $c(x_1, x_2; f_1, f_2) := |x_1 - x_2|_p^p + |f_1(x_1) - f_2(x_2)|_p^p$.

In this paper we will mainly consider the case $p = 2$. The idea behind the above construction of TL^p space is to provide a common space where it is possible to compare functions that have different supports. In other words, suppose that $f_1 : \mathcal{X}_1 \to \mathbb{R}$ and $f_2 : \mathcal{X}_2 \to \mathbb{R}$ are two functions where $\mathcal{X}_1, \mathcal{X}_2$ are subsets of $\mathcal{X}$. While a direct comparison between f_1 and f_2 would be possible if $\mathcal{X}_1$ and $\mathcal{X}_2$ were the same, it is less clear how to compare them when their domains are different. In the TL^p interpretation we think of f_1 as a pair (μ_1, f_1) where μ_1 is a probability measure supported on $\mathcal{X}_1$ and f_2 is interpreted similarly. Then, after finding a suitable coupling between the measures μ_1 and μ_2, one can couple the functions f_1 and f_2 and use a direct L^2-comparison. The notation TL^p suggests the use of an L^p comparison after solving an optimal transport problem. This idea has been used for the task of domain adaptation (Courty et al., 2017).

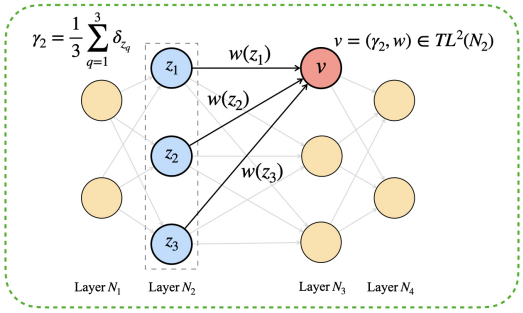

Figure 4: Following our definitions, node $v = (\gamma_2, w) \in TL^2(N_2)$, where γ_2 is a probability measure on layer N_2 and $w : N_2 \to \mathbb{R}$ is the weight function corresponding to node v. For example, $w(z_2)$ is the true parameter (weight) between nodes v and z_2.

C.2 NESTED DEFINITION OF FULLY CONNECTED NNS USING TL^p FORMALISM

We now provide detailed explanations for our interpretations of a node in a neural network based on TL^p formalism. Intuitively, we need to construct a common space that allows us to talk about the "distance" between neurons in different neural networks (at the same layer). Therefore, inspired by the idea of TL^p space, we interpret node v in the l-th layer N_l (for $l > 1$) as an element in $TL^2(N_{l-1})$, that is, as a function on the domain N_{l-1} (previous layer) coupled with a probability measure. In particular, the node v is interpreted as $v := (\gamma_{l-1}, w)$, where γ_{l-1} is a measure on the previous layer N_{l-1} and w represents the collection of weights between the node v and the nodes in previous layer N_{l-1}, which can be interpreted as a function $w : N_{l-1} \to \mathbb{R}$. An illustration of the above interpretation for v is shown in Figure 4.

Next, to construct a common space using the above interpretations, we let $\mathcal{N}_l := TL^2(\mathcal{N}_{l-1})$ to be the collection of all the neurons on the l-th layer for $l \geq 2$ (over different neural networks). A simple inductive argument shows that $N_l^i \subseteq \mathcal{N}_l$ for all i and all $l \geq 2$. We can now embed the nodes on the l-th layers of different input models into a single common space $\mathcal{N}_l$, and in turn we can define a

notion of "distance" to compare neurons in different neural networks (see subsection 2.2). For the first layer, we simply let $\mathcal{N}_1 := N_1^1$, since we assume all the base models have the same input layer.

C.3 WB FUSION ALGORITHM

The WB fusion algorithm is summarized in Algorithm 1.

Algorithm 1 WB Fusion

Input: Neural networks $N^1, \cdots, N^n$ (m layers);
Number of nodes k_l for layer N_l^{new}, for $l = 2, \cdots, m$;
Initialized weight functions $W_2, \cdots, W_m$;
Set $(\Pi_1^i)^*$ to be the identity matrix in $\mathbb{R}^{k_1}$ for all $i = 1, \ldots, n$. Set also $\gamma_1 = \gamma_1^1$;
for $l = 2, \cdots, m$ **do**
  repeat
    Step 1: for Wasserstein barycenter problem (8), fix W_l and obtain couplings Π_l^i by solving n independent OT problems;
    Step 2: fix current optimal couplings Π_l^i and update weight function W_l using formula (9);
  until W_l and the Π_l^i converge;
  Obtain measure γ_l based on γ_{l-1} and W_l;
  Obtain optimal couplings $(\Pi_l^i)^* \in \Gamma(\gamma_l, \gamma_l^i)$ for $i = 1, \cdots, n$.
end for
Output: The new NN N^{new} as specified by the measures $\gamma_1, \ldots, \gamma_m$.

Remark 1 (Total computational complexity for WB fusion)**.** *Without loss of generality, we assume that all the layers in each of the models has M number of nodes. In practice, we set the maximum number of iterations of Steps 1 and 2 as T (in the numerical experiments, we observe convergence within $T = 10$). From Proposition 1 in (Peyré et al., 2016), one can compute $\mathcal{L}(W_l, W_l^i) \otimes (\Pi_{l-1}^i)^*$ in $O(M^3)$ operations, and by using Sinkhorn algorithm (Cuturi, 2013), the time complexity of solving an OT problem is roughly $O(M^2)$. Therefore, minimization with respect to $\{\Pi_l^i\}_i$ needs $O(nM^3 + nM^2)$ operations. On the other hand, minimization w.r.t. W_l reduces to a simple matrix multiplication, which can be computed in $O(M^3)$ operations. Therefore, the complexity of updating $\{\Pi_l^i\}_i$ and W_l for one round of Steps 1 and 2 is $O(nM^3)$. Then the total computational complexity of WB fusion algorithm is $O(nmM^3)$, where m is the number of layers for the input models. Note that for modern neural networks, the number of nodes on each layer is not large, i.e., M is relatively small. Therefore, our algorithm is quite fast especially when compared to training a NN from scratch.*

C.4 EXTENSION TO CONVOLUTIONAL NEURAL NETWORKS

For CNNs, we substitute "nodes" for "channels" in the discussion in Sections 2.1 and 2.2, and instead of defining probability measures and functions on sets of nodes, we define them on sets of channels . To be more concrete, let us consider a convolutional layer N_l with weights having dimensions $\mathbb{R}^{C^{\text{in}} \times k \times k \times C^{\text{out}}}$, where C^{in}, C^{out} are the number of input and output channels and $k \times k$ is the size of filters. Then, in formula (3), v must be interpreted as a channel on layer N_l, and its corresponding weight w must be interpreted as a function mapping N_{l-1} into $\mathbb{R}^{k \times k}$ (illustrated in Figure 5). Since now $w(z_q)$ is a $\mathbb{R}^{k \times k}$ matrix, we can redefine the cost function d_W in (5) using the Frobenius norm.

C.5 EXTENSION TO RESNETS

ResNet models are neural networks that contain skip connections that "jump" over some layers to avoid the problem of vanishing gradients. Figure 6 shows a typical building block of a ResNet model. For a layer N_l we let $\mathcal{C}_l = \{N_\alpha\}_{\alpha \in \Lambda}$ be the collection of all the previous layers that connect with it. Since the outputs of layers in $\mathcal{C}_l$ are added up before feeding them into layer N_l, skip connections constrain the layers in $\mathcal{C}_l$ to share the same coupling, i.e $\Pi_\alpha = \Pi^* \ \forall \alpha$. This constraint allows us to use any of the previous layers in $\mathcal{C}_l$ as the support for N_l since they share the same coupling Π^* and hence produce the same cost to be used in optimization equation 8. Our convention

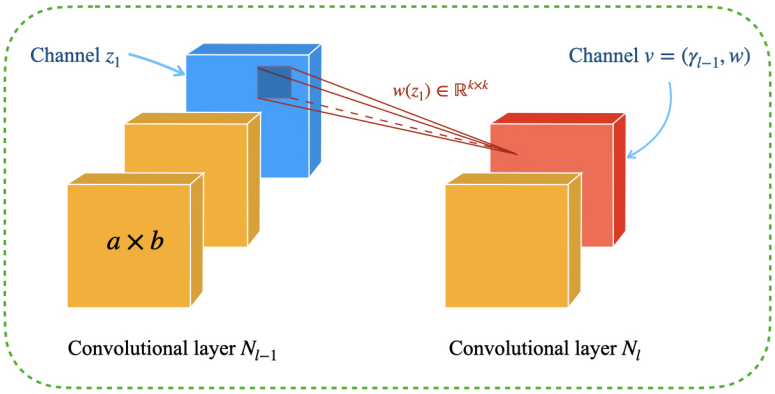

Figure 5: A channel v in the convolutional layer N_l is interpreted as a node in our framework, with weight function w mapping previous layer elements (channels) to $\mathbb{R}^{k \times k}$.

is to find a coupling for the earliest layer in $\mathcal{C}_l$ first and then use the same coupling when we fuse the other layers in $\mathcal{C}_l$.

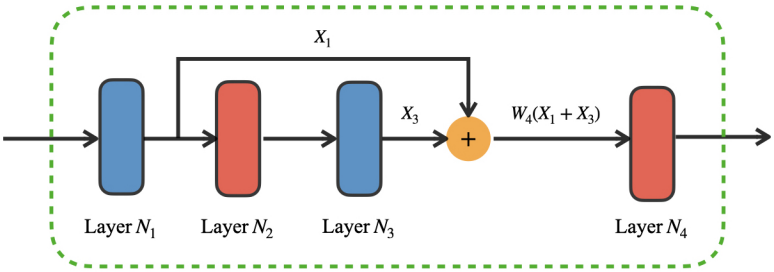

Figure 6: A building block of ResNet. Due to skip connection, outputs X_1 and X_3 of layers N_1 and N_3 are added up before feeding them to layer N_4.

C.6 Theoretical Analysis of WB Fusion

Theorem C.1. *Let $f_{v_1,U_1}(x) = v_1\sigma(U_1x)$, $f_{v_2,U_2}(x) = v_2\sigma(U_2x)$ be two one-hidden layer neural networks where $\sigma(\cdot)$ is the activation function, $v_1, v_2 \in \mathbb{R}^{1\times h}$ and $U_1, U_2 \in \mathbb{R}^{h\times d}$ are the parameters, and $x \in \mathbb{R}^d$ is the input data. Let $\mathcal{W}_2, \mathcal{W}_3$ denote the Wasserstein barycenter problem's (WBP) objective on the hidden and last layers respectively:*

$$\mathcal{W}_2 = \inf_{\gamma_2} \frac{1}{2}W(\gamma_2, \gamma_2^1) + \frac{1}{2}W(\gamma_2, \gamma_2^2) \tag{21}$$

$$\mathcal{W}_3 = \inf_{\gamma_3} \frac{1}{2}W(\gamma_3, \gamma_3^1) + \frac{1}{2}W(\gamma_3, \gamma_3^2) \tag{22}$$

Let Π_1, Π_2 be the permutation matrices corresponding to solutions of WBP for f_{v_1,U_1}, f_{v_2,V_2} yielding permuted models $f_{v_1',U_1'} = f_{v_1\Pi_1^T,\Pi_1U_1}$ and $f_{v_2',U_2'} = f_{v_2\Pi_2^T,\Pi_2U_2}$ respectively. If the WBP minimizations are bounded by $\mathcal{W}_2 \leq \varepsilon^2$ and $\mathcal{W}_3 \leq \eta^2$, then $\forall ||x||_2 \leq \sqrt{d}$ and $\alpha \in [0, 1]$:

$$|f_{\alpha v_1'+(1-\alpha)v_2',\alpha U_1'+(1-\alpha)U_2'}(x) - \alpha f_{v_1,U_1}(x) - (1-\alpha)f_{v_2,U_2}(x)| \leq \alpha(1-\alpha)C(\varepsilon, \eta, \sqrt{d}), \tag{23}$$

where (i) $C(\varepsilon, \eta, \sqrt{d}) = 4\varepsilon\eta\sqrt{d}$ for $\sigma(x) = x$, and (ii) $C(\varepsilon, \eta, \sqrt{d}) = \varepsilon(4\eta + 2||v_2||_2)\sqrt{d}$ for $\sigma(x) = ReLU(x)$.

Proof. Let's first consider the WBP for the hidden layer. Since the optimal couplings for the first layer are identity matrices (we assume the input layers are the same), using the notations in equa-

tion 6 and equation 8, the WBP problem can be written as

$$\begin{aligned} &\min_{\gamma_2,\Pi_1,\Pi_2} \frac{1}{2}W(\gamma_2,\gamma_2^1)+\frac{1}{2}W(\gamma_2,\gamma_2^2) \\ \iff \quad &\min_{U,\Pi_1,\Pi_2} \frac{1}{2}\langle \mathcal{L}(U,U_1)\otimes I,\Pi_1\rangle+\frac{1}{2}\langle \mathcal{L}(U,U_1)\otimes I,\Pi_2\rangle \end{aligned} \tag{24}$$

Assume Π_1,Π_2 and $U := \frac{1}{2}\Pi_1 U_1 + \frac{1}{2}\Pi_2 U_2$ (obtained by using equation 9 and the constants are absorbed in the matrices Π_1,Π_2 to ensure they are permutation matrices) are the solutions to the WBP problem. Then

$$\begin{aligned} \varepsilon^2 &\geq \min_{U,\Pi_1,\Pi_2} \frac{1}{2}\langle \mathcal{L}(U,U_1)\otimes I,\Pi_1\rangle+\frac{1}{2}\langle \mathcal{L}(U,U_1)\otimes I,\Pi_2\rangle \\ &= \frac{1}{2}\sum_{i,j}\|u_i-u_{1,j}\|_2^2\Pi_{1,ij}+\frac{1}{2}\sum_{i,j}\|u_i-u_{2,j}\|_2^2\Pi_{2,ij} \\ &\qquad (\text{where } u_i \text{ is the } i\text{-th column of matrix } U \text{ and similarly for } u_{1,j} \text{ and } u_{2,j}.) \\ &= \frac{1}{2}\|U-\Pi_1U_1\|_F^2+\frac{1}{2}\|U-\Pi_2U_2\|_F^2 \quad (\text{Since } \Pi_1 \text{ and } \Pi_2 \text{ are permutation matrices.}) \\ &= \frac{1}{2}\|\frac{1}{2}\Pi_1U_1+\frac{1}{2}\Pi_2U_2-\Pi_1U_1\|_F^2+\frac{1}{2}\|\frac{1}{2}\Pi_1U_1+\frac{1}{2}\Pi_2U_2-\Pi_2U_2\|_F^2 \\ &= \frac{1}{4}\|\Pi_1U_1-\Pi_2U_2\|_F^2 \end{aligned} \tag{25}$$

Also note that $v := \alpha v\Pi_1^T + (1-\alpha)v_2\Pi_2^T$ is the solution of the last layer's WBP problem. Then similar to above derivation,

$$\eta^2 \geq \frac{1}{4}\|v_1\Pi_1^T - v_2\Pi_2^T\|_2^2 \tag{26}$$

Then one can compute

$$\begin{aligned} &|f_{\alpha v_1'+(1-\alpha)v_2',\alpha U_1'+(1-\alpha)U_2'}(x)-\alpha f_{v_1,U_1}(x)-(1-\alpha)f_{v_2,U_2}(x)| \\ &= |(\alpha v_1\Pi_1^T+(1-\alpha)v_2\Pi_2^T)\sigma(\alpha\Pi_1U_1x+(1-\alpha)\Pi_2U_2x)-\alpha v_1\Pi_1^T\sigma(\Pi_1U_1x)-(1-\alpha)v_2\Pi_2^T\sigma(\Pi_2U_2x)| \\ &\qquad (\text{since } \Pi_1 \text{ and } \Pi_2 \text{ are permutation matrices}) \\ &= |(\alpha v_1+(1-\alpha)v_2)\sigma(\alpha U_1x+(1-\alpha)U_2x)-\alpha v_1\sigma(U_1x)-(1-\alpha)v_2\sigma(U_2x)| \\ &\qquad (\textbf{for notation simplicity, we use } v_i,U_i \textbf{ to represent } v_1\Pi_i^T,\Pi_iU_i \textbf{ for } i=1,2 \textbf{ in all later derivations}) \\ &= |\alpha v_1\big[\sigma(\alpha U_1x+(1-\alpha)U_2x)-\sigma(U_1x)\big]+(1-\alpha)v_2\big[\sigma(\alpha U_1x+(1-\alpha)U_2x)-\sigma(U_2x)\big]| \\ &=: |\alpha v_1K_1+(1-\alpha)v_2K_2|, \\ &\qquad \big(\text{where we denote } K_1 := \sigma(\alpha U_1x+(1-\alpha)U_2x)-\sigma(U_1x), K_2 := \sigma(\alpha U_1x+(1-\alpha)U_2x)-\sigma(U_2x)\big) \\ &= |\alpha^2v_1K_1+\alpha(1-\alpha)v_1K_1+(1-\alpha)^2v_2K_2+\alpha(1-\alpha)v_2K_2| \\ &= |\alpha^2v_1K_1-\alpha^2v_2K_1+\alpha(1-\alpha)v_1K_1-\alpha(1-\alpha)v_2K_1+\alpha(1-\alpha)v_2K_1+(1-\alpha)^2v_2K_2+\alpha(1-\alpha)v_2K_2| \\ &= |\alpha^2(v_1-v_2)K_1+\alpha(1-\alpha)(v_1-v_2)K_1+(1-\alpha)v_2(\alpha K_1+(1-\alpha)K_2)+\alpha v_2(\alpha K_1+(1-\alpha)K_2)| \\ &= |\alpha(v_1-v_2)K_1+v_2(\alpha K_1+(1-\alpha)K_2)| \\ &\leq \alpha\|v_1-v_2\|_2\|K_1\|_2+\|v_2\|_2\|\alpha K_1+(1-\alpha)K_2\|_2 \end{aligned} \tag{27}$$

Case 1: If σ is a linear activation function, i.e., $\sigma(x)=x$, one can compute

$$\begin{aligned} \|K_1\|_2 &= \|\sigma(\alpha U_1x+(1-\alpha)U_2x)-\sigma(U_1x)\|_2 \\ &= \|\alpha U_1x+(1-\alpha)U_2x-U_1x\| \\ &= (1-\alpha)\|(U_1-U_2)x\|_2 \\ &\leq (1-\alpha)\|U_1-U_2\|_F\|x\|_2 \end{aligned} \tag{28}$$

and

$$\|\alpha K_1+(1-\alpha)K_2\|_2 = \|\sigma(\alpha U_1x+(1-\alpha)U_2x)-\alpha\sigma(U_1x)-(1-\alpha)\sigma(U_2x)\|_2 = 0 \tag{29}$$

Then

$$\begin{aligned}
&|f_{\alpha v_1'+(1-\alpha)v_2',\alpha U_1'+(1-\alpha)U_2'}(x) - \alpha f_{v_1,U_1}(x) - (1-\alpha) f_{v_2,U_2}(x)| \\
&\quad \le \alpha\|v_1 - v_2\|_2 \|K_1\|_2 + \|v_2\|_2 \|\alpha K_1 + (1-\alpha)K_2\|_2 \\
&\quad = \alpha(1-\alpha)\|v_1 - v_2\|_2 \|U_1 - U_2\|_F \|x\|_2 \\
&\quad \le 4\alpha(1-\alpha)\varepsilon\eta\|x\|_2 \qquad \text{(use equation 25 and equation 26)}
\end{aligned} \tag{30}$$

Case 2: If σ is ReLU activation, i.e., $\sigma(x) = \max\{x, 0\}$. One can compute

$$\begin{aligned}
\|K_1\|_2 &= \|\sigma(\alpha U_1 x + (1-\alpha)U_2 x) - \sigma(U_1 x)\|_2 \\
&\le \|\alpha U_1 x + (1-\alpha)U_2 x - U_1 x\|_2 \qquad (|\sigma(y) - \sigma(z)| \le |y - z|) \\
&\le (1-\alpha)\|U_1 - U_2\|_F \|x\|_2
\end{aligned} \tag{31}$$

and

$$\begin{aligned}
\|\alpha K_1 + (1-\alpha)K_2\|_2 &= \|\sigma(\alpha U_1 x + (1-\alpha)U_2 x) - \alpha\sigma(U_1 x) - (1-\alpha)\sigma(U_2 x)\|_2 \\
&\le \alpha\|\sigma(\alpha U_1 x + (1-\alpha)U_2 x) - \sigma(U_1 x)\|_2 + (1-\alpha)\|\sigma(\alpha U_1 x + (1-\alpha)U_2 x) - \sigma(U_2 x)\|_2 \\
&\le \alpha\|\alpha U_1 x + (1-\alpha)U_2 x - U_1 x\|_2 + (1-\alpha)\|\alpha U_1 x + (1-\alpha)U_2 x - U_2 x\|_2 \\
&\le 2\alpha(1-\alpha)\|U_1 - U_2\|_F \|x\|_2
\end{aligned} \tag{32}$$

Then

$$\begin{aligned}
&|f_{\alpha v_1'+(1-\alpha)v_2',\alpha U_1'+(1-\alpha)U_2'}(x) - \alpha f_{v_1,U_1}(x) - (1-\alpha) f_{v_2,U_2}(x)| \\
&\quad \le \alpha\|v_1 - v_2\|_2 \|K_1\|_2 + \|v_2\|_2 \|\alpha K_1 + (1-\alpha)K_2\|_2 \\
&\quad \le \alpha(1-\alpha)\|v_1 - v_2\|_2 \|U_1 - U_2\|_F \|x\|_2 + \|v_2\|_2 2\alpha(1-\alpha)\|U_1 - U_2\|_F \|x\|_2 \\
&\quad \le \alpha(1-\alpha)\varepsilon(4\eta + 2\|v_2\|_2)\|x\|_2 \qquad \text{(use equation 25 and equation 26)} \\
&\quad \le \alpha(1-\alpha)\varepsilon(4\eta + 2)\|x\|_2
\end{aligned} \tag{33}$$

□

D COMPARISON BETWEEN WB FRAMEWORK AND OT FRAMEWORK

In this section, we explain in detail how our Wasserstein barycenter-based method is different from OT fusion proposed in (Singh & Jaggi, 2019). Assume we have n pre-trained models $N^1, \ldots, N^n$ to aggregate. As before, denote the weights of the l-th layer in neural network N^i and target model N^{new} as W_l^i and W_l respectively. When doing the fusion on the l-th layer, OT fusion algorithm first aligns the incoming edge for the current layer l by post-multiplying with the previous layer transport matrix $(\Pi_{l-1}^i)^*$ and normalizing via the inverse of the probability measure over the l-th layer in the target model, i.e.,

$$\widehat{W}_l^i \leftarrow k_{l-1} \frac{1}{\mathbb{1}_{k_l - 1}} W_l^i (\Pi_{l-1}^i)^{*T} \tag{34}$$

Then the neurons in layer l of the pre-trained models N_l^i are aligned with respect to the target model N^{new} by considering optimal transport problems with cost matrices:

$$C_{l,\text{OT}}^i := \left[\|w_j - \widehat{w}_g^i\|_2^2\right]_{j,g}, \tag{35}$$

where w_j is the j-th row vector of weight matrix W_l and $\widehat{w}_g^i$ is the g-th row vector of $\widehat{W}_l^i$. On the other hand, the cost function in our WB framework reads:

$$C_{l,\text{WB}}^i := \left[c_l(v_j, v_g^i)\right]_{j,g} = \left[d_W(w_j, w_g^i)\right]_{j,g} = \left[(w_{jq} - w_{gs}^i)^2\right]_{j,g,q,s} \otimes (\Pi_{l-1}^i)^* \tag{36}$$

We can see that the cost functions used in our algorithm equation 36 are different from the ones in OT fusion equation 35 . We highlight that our cost function equation 36 was derived from a first principle, in this case a bona fide Wasserstein barycenter problem. In contrast, the aligning step equation 34 in OT fusion is introduced without a similar derivation.

Since our target problem is from the beginning a minimization problem, it is clear that one should iterate the proposed steps until convergence, rather than stopping after one iteration as in OT fusion. As illustrated in the numerical experiments (Section 4.1, 4.2), iterating until convergence is important.

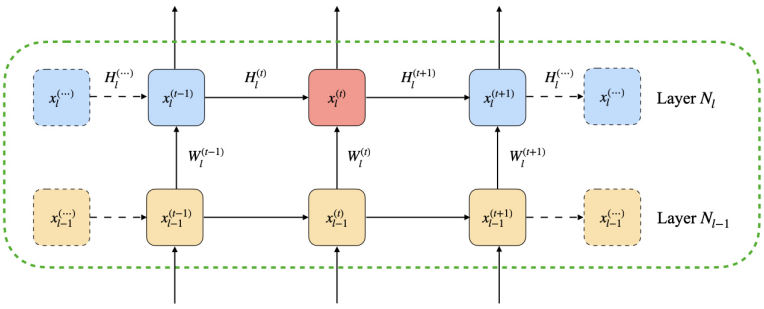

Figure 7: A building block of unfolded RNN motivating problem equation 10. $W_l^{(t)}$ and $H_l^{(t)}$ are collections of input-to-hidden and hidden-to-hidden weights functions at time step t respectively (for actual RNNs, W_l and H_l don't depend on t). $x_l^{(t)}$ denotes the output of layer N_l at time step t; t changes horizontally and indexes the unfolded units.

E Details on GWB fusion

Analogous to the fusion of FC networks, we can think of RNN fusion on the l-th layer following the "layerwise" fashion with respect to the "time step". Specifically, consider the unfolded RNN shown in Figure 7 and the fusion for layer l at time step t. We assume that the fusion before time step t has finished. Denote $v_j^{(t)} = [(\gamma_{l-1}, w_j^{(t)}); (\gamma_l^{(t-1)}, h_j^{(t)})]$ and $v_g^i = [(\gamma_{l-1}^i, w_g^i); (\gamma_l^i, h_g^i)]$ as the j-th node in layer N_l at time t and the g-th node in layer N_l^i respectively. Then the cost function between nodes $v_j^{(t)}$ and v_g^i at time t could be defined as

$$c_l(v_j^{(t)}, v_g^i) := \underbrace{d_\mu(\gamma_{l-1}, \gamma_{l-1}^i) + d_W(w_j^{(t)}, w_g^i)}_{TL^p \text{ cost for input-to-hidden weights}} + \underbrace{d_\mu(\gamma_l^{(t-1)}, \gamma_l^i) + d_H(h_j^{(t)}, h_g^i)}_{TL^p \text{ cost for hidden-to-hidden weights}}, \tag{37}$$

where $d_\mu(\cdot,\cdot)$, $d_W(\cdot,\cdot)$ are defined the same as equation 4 and equation 5 respectively. Denote

$$\Pi_l^{i,(t-1)} := \arg\min_{\Pi_l^i} d_\mu(\gamma_l^{(t-1)}, \gamma_l^i) = \arg\min_{\Pi_l^i} \langle C_l^{i,(t-1)}, \Pi_l^i \rangle, \tag{38}$$

where $C_l^{i,(t-1)} := [c_l(v_j^{(t-1)}, v_g^i)]_{j,g}$ is the cost matrix for l-th layer at time $t-1$. We use $H_l^{(t)} := (h_1^{(t)}, h_2^{(t)}, \ldots, h_{k_l}^{(t)})^T$ to denote the hidden-to-hidden weight function matrix of layer N_l^{new} at time step t. Since $H_l^{(t)}$ is supported on the l-th layer (at time step $t-1$), we define

$$d_H(H_l^{(t)}, H_l^i) := \mathcal{L}(H_l^{(t)}, H_l^i) \otimes \Pi_l^{i,(t-1)}, \tag{39}$$

where $\mathcal{L}(H_l^{(t)}, H_l^i) := \left[\left(h_{jq}^{(t)} - h_{gs}^i\right)^2\right]_{j,g,q,s}$ is a 4-way tensor. Now consider the WBP (6) again (plugging-in the cost function equation 37). Note that $\gamma_l^{(t-1)}$ is given, then $d_\mu(\gamma_l^{(t-1)}, \gamma_l^i)$ becomes a constant and does not involve any optimization variable in (6) like $d_\mu(\gamma_{l-1}, \gamma_{l-1}^i)$. Therefore, the barycenter problem on the l-th layers at time step t is to minimize the objective function

$$B(W_l^{(t)}, H_l^{(t)}; \{\Pi_l^{i,(t)}\}_i) := \frac{1}{n}\sum_{i=1}^n \langle \mathcal{L}(W_l^{(t)}, W_l^i) \otimes (\Pi_{l-1}^i)^* + \alpha_H \mathcal{L}(H_l^{(t)}, H_l^i) \otimes \Pi_l^{i,(t-1)}, \Pi_l^{i,(t)} \rangle, \tag{40}$$

with respect to $W_l^{(t)}$, $H_l^{(t)}$ and $\{\Pi_l^{i,(t)}\}_i$. However, since RNNs presuppose that all weights should be the same for all unfolded units (i.e. for all t along a fixed l), it is natural to consider invariants (i.e. independent of t) of the ensemble of problems indexed by t. One such invariant can be obtained by formally taking the limit $t \to \infty$ in the above problem: this gives rise to the following problem

$$\min_{W_l, H_l, \{\Pi_l^i\}_i} B(W_l, H_l; \{\Pi_l^i\}_i) := \frac{1}{n}\sum_{i=1}^n \langle \mathcal{L}(W_l, W_l^i) \otimes (\Pi_{l-1}^i)^* + \alpha_H \mathcal{L}(H_l, H_l^i) \otimes \Pi_l^i, \Pi_l^i \rangle \tag{41}$$

E.1 GWB FUSION ALGORITHM

Notice that equation 10 is a GW-like barycenter problem. Therefore, following the algorithm proposed in (Peyré et al., 2016), we solve it using a block coordinate relaxation, i.e., alternatively minimizing with respect to the couplings $(\Pi_l^i)_i$ and weight functions W_l and H_l.

Minimization with respect to $\{\Pi_l^i\}_i$. The optimization (10) over $(\Pi_l^i)_i$ alone decouples as n independent GW-like optimization problems. For $i = 1, \cdots, n$

$$\min_{\Pi_l^i \in \Gamma(\gamma_l, \gamma_l^i)} \langle \mathcal{L}(W_l, W_l^i) \otimes (\Pi_{l-1}^i)^* + \mathcal{L}(H_l, H_l^i) \otimes \Pi_l^i,\, \Pi_l^i \rangle. \tag{42}$$

As proposed in (Peyré et al., 2016), a stationary point of this optimization problem can be reached following the iterations:

$$\Pi_l^i \leftarrow \mathcal{T}\big(\mathcal{L}(W_l, W_l^i) \otimes (\Pi_{l-1}^i)^* + \mathcal{L}(H_l, H_l^i) \otimes \Pi_l^i, \gamma_l, \gamma_l^i\big). \tag{43}$$

Minimization with respect to W_l and H_l. For given $\{\Pi_l^i\}_i$, the minimization with respect to W_l and H_l reads

$$\min_{W_l, H_l} \frac{1}{n} \sum_{i=1}^{n} \langle \mathcal{L}(W_l, W_l^i) \otimes (\Pi_{l-1}^i)^* + \mathcal{L}(H_l, H_l^i) \otimes \Pi_l^i,\, \Pi_l^i \rangle. \tag{44}$$

By first-order optimality conditions, we have the update formulas,

$$W_l \leftarrow k_l k_{l-1} \frac{1}{\mathbb{1}_{k_{l-1}} \mathbb{1}_{k_l}^T} \frac{1}{n} \sum_{i=1}^{n} \Pi_l^i W_l^i (\Pi_{l-1}^i)^{*T}, \tag{45}$$

$$H_l \leftarrow k_l^2 \frac{1}{\mathbb{1}_{k_l} \mathbb{1}_{k_l}^T} \frac{1}{n} \sum_{i=1}^{n} \Pi_l^i H_l^i (\Pi_l^i)^T \tag{46}$$

The GWB fusion algorithm is summarized in Algorithm 2.

Algorithm 2 GWB Fusion

Input: Neural networks $N^1, \cdots, N^n$ (m layers);
Number of nodes k_l for layer N_l^{new}, for $l = 2, \cdots, m$;
Initialized weight functions $\{W_l\}_{l=2}^n$ and $\{H_l\}_{l=2}^n$;
Set $(\Pi_1^i)^*$ to be the identity matrix in $\mathbb{R}^{k_1}$ for all $i = 1, \ldots, n$. Set also $\gamma_1 = \gamma_1^1$;
for $l = 2, \cdots, m$ **do**
  repeat
    for $i = 1, \cdots, n$ **do**
      Initialize Π_l^i;
      repeat
        Compute $C_l^i := \mathcal{L}(W_l, W_l^i) \otimes (\Pi_{l-1}^i)^* + \mathcal{L}(H_l, H_l^i) \otimes \Pi_l^i$;
        Updating Π_l^i by solving entropy-regularized optimal transport problem equation 43;
      until Π_l^i converges;
    end for
    Updating W_l and H_l using (45) and (46);
  until W_l, H_l and $\{\Pi_l^i\}_i$ converge;
  Obtain measure γ_l based on γ_{l-1}, W_l and H_l;
  Obtain optimal couplings $(\Pi_l^i)^* \in \Gamma(\gamma_l, \gamma_l^i)$ for $i = 1, \cdots, n$;
end for
Output: The new NN N^{new} as specified by the measures $\gamma_1, \ldots, \gamma_m$.

Remark 2 (Total computational complexity for GWB fusion). *Similar to the analysis of WB fusion, we assume each layer has M number of neurons and we set the maximum number of times we run the outer repeat loop in Algorithm 2 to be T. Also, we let $\tilde{T}$ be maximum number of times we run the inner repeat loop in Algorithm 2 (in our numerical experiments, we observe convergence within $\tilde{T} = 10$). From Proposition 1 in (Peyré et al., 2016), one can compute the cost matrix $C_l^i = \mathcal{L}(W_l, W_l^i) \otimes$*

$(\Pi_{l-1}^i)^* + \mathcal{L}(H_l, H_l^i) \otimes \Pi_l^i$ *in* $O(M^3)$ *operations. The time complexity of solving entropy-regularized OT problem equation 43 using Sinkhorn algorithm is* $O(M^2)$*. Therefore, minimizing with respect to* $\{\Pi_l^i\}_i$ *needs* $O(n\tilde{T}M^3 + n\tilde{T}M^2)$ *operations. On the other hand, minimizing w.r.t* W_l *and* H_l *can be computed in* $O(M^3)$ *operations. Therefore, the complexity of updating* $\{\Pi_l^i\}_i$*,* W_l *and* H_l *for one iteration is* $O(nM^3)$*. Then, the total computational complexity of GWB fusion algorithm is* $O(nmM^3)$.

E.2 Extension to LSTM

Our GWB framework for RNN models can be easily extended to the case of LSTMs. LSTMs are more sophisticated than RNNs. For the purpose of fusion the major difference is that LSTMs have 4 cell states (hidden states) in comparison to RNNs which have one hidden state. We consider each cell of LSTM as an individual unit during the fusion process, i.e., treating each of the cells as a basic RNN unit. However, since these cells share the same input from the previous layer, the alignments of cells should also be the same, and thus we add the constraint that the coupling matrices corresponding to each cell should be the same. To reflect this constraint, the cost function for each hidden layer in LSTM is updated to be the sum of cost functions defined for each cell unit.

F Experiment Details

In this section, we provide more details on model training and related hyperparameters for our experiments.

F.1 Model training

MLPNet training details. For the fusion and distillation experiments, MLPNet, MLPSmall and MLPLarge models are trained using SGD optimizer at a constant learning rate of 0.05 and momentum of 0.5 for 20 epochs. In the fusion experiments for heterogeneous data distributions, we follow (Singh & Jaggi, 2019) and training the MLPSmall models using SGD optimizer with learning rate of 0.01, momentum of 0.5 for 10 epochs.

VGG11 training details. We follow (Singh & Jaggi, 2019), and train the VGG11 model using SGD optimizer for 300 epochs. The initial learning rate of 0.05 decays after every 30 epochs by a multiplicative factor of 0.5. We use SGD with momentum of 0.9 and weight decay of 0.0005. The batch size used in training is 128. The model with best validation accuracy is selected for fusion.

ResNet18 training details. The models are trained using SGD optimizer for 300 epochs with an initial learning rate of 0.1 which gets decayed by a multiplicative factor of 0.1 at epoch 150. We use momentum of 0.9 and weight decay of 0.0001.

We skip the batch normalization layer in ResNet18 models for the current work. However, our framework can be extended to handle batch normalization parameters by appropriately including them as a part of the node and using the coupling associated with the previous layers. We leave this extension for the future work.

RNN training details. For all our experiments the RNN model has one hidden layer of dimension 256. In general, the RNN base models are trained using Adam (Kingma & Ba, 2014) optimizer. We use momentum of 0.9 and weight decay of 1×10^{-4}. For MNIST dataset, the images of 28×28 dimensions are interpreted as 28 length sequences of vectors $\in \mathbb{R}^{28}$. For the NLP tasks, we use pre-trained GloVe embeddings (Pennington et al., 2014) of dimension 100. The pre-trained embedding layer does not get updated during the training phase. The dataset specific training details are as follows: i) MNIST: Models are trained for 20 epoch with a constant learning rate of 0.001 and a batch size of 64; ii) SST-2: Models are trained for 20 epochs with a constant learning rate of 0.001 and batch size of 256. The maximum sequence length for the dataset is set to 56; iii) AGNEWS: Models are trained for 10 epochs. The constant learning rate is 0.0001 and the batch size used in training is 128. The maximum sequence length for each training data is set to 60 for stable training of RNN; and iv) DBpedia: Models are trained for 30 epochs with a constant learning rate 0.0001, batch size 256 and a maximum sequence length of 60.

LSTM training details. Similar to the RNN case, for all our experiments the LSTM model has one hidden layer of hidden dimension 256. Since LSTMs have 4 hidden states, this corresponds to a hidden layer of total 4×256 dimensions. In general, the LSTM base models are trained using Adam (Kingma & Ba, 2014) optimizer with a constant learning rate 0.001. We use momentum of 0.9 and weight decay of 1×10^{-4}. For the NLP tasks, we use pre-trained GloVe embeddings (Pennington et al., 2014) of dimension 50 and the pre-trained embedding layer does not get updated during the training phase. As for the RNNs, the images of 28×28 dimensions in MNIST dataset are interpreted as 28 length sequences of vectors $\in \mathbb{R}^{28}$. The dataset specific training details are as follows: i) MNIST: Models are trained for 50 epochs with batch size of 64; ii) SST-2: Models are trained for 20 epochs. The batch size used for training is 256. The maximum sequence length is set to 56; iii) AGNEWS: Models are trained for 10 epochs with batch size 128 and a maximum sequence length of 160; and iv) DBpedia: Models are trained for 5 epochs and the batch size used for training is 256. The maximum sequence length for this dataset is set to 100.

F.2 Hyperparameter selection

The hyperparameters are selected using a separate validation split. For solving the optimal transport problem in our proposed framework, we use Sinkhorn algorithm (Cuturi, 2013) with regularization hyperparameters $0.01, 0.005, 0.001, 0.0005$. The hyperparameter α_H in equation 10 is chosen in the interval $[1, 20]$. Note that in our implementation, we use the variable $alpha\_h$ to capture α_H and set it between $[100, 2000]$ since we simplify the implementations and don't normalize the coupling matrices Π_1^i by the size of layers.

G Additional experiments and results

G.1 Fusion of multiple model counts

Setup: In this experiment, we consider fusion of 2, 4 and 6 base models into a target model of different architecture. We use MLPNet trained on MNIST dataset as the base models. The target model is set to be MLPLarge which has twice the width of the base models. Since the base and target models have different architectures, we initialize the target model randomly before starting fusion.

Quantitative results: Table 3 shows the results for fusion of multiple models. We find that the WB framework performs much better than OT fusion algorithm (higher accuracy and low standard deviations). These improvements in accuracy are especially higher when the number of base models being fused is large. These experiments illustrate that in comparison to OT fusion WB fusion is much more robust to randomness in initialization.

Table 3: Performance comparison (Test accuracy $\pm$ standard deviation %) of OT and WB frameworks for fusing multiple models into a different target architecture. For each case, the target model obtained by WB fusion has higher test accuracy and smaller standard deviation.

	MNIST/MLPLarge		
	# of Models=2	# of Models=4	# of Models=6
Base Model Avg	98.31 ± 0.02	98.31 ± 0.02	98.31 ± 0.02
OT	91.53 ± 2.64	52.10 ± 8.35	43.16 ± 6.17
WB	$\mathbf{94.93 \pm 1.18}$	$\mathbf{89.03 \pm 4.05}$	$\mathbf{86.40 \pm 4.29}$

G.2 FURTHER VISUALIZATIONS UNDER DIFFERENT NETWORK ARCHITECTURES AND DATASETS

In this section, we include more visualizations of the models produced by our algorithms when fusing different network architectures trained over different datasets [1]. We find that in each setting, the basins of the permuted model 2 and base model 1 lie close to each other and are separated by a relatively low energy barrier, especially when compared to the energy barriers between the basins of model 1 and model 2. However, we also observe that, for cases like RESNET18 trained on CIFAR10 dataset (Figure 9 (right)) or RNN trained on AGNEWS and DBpedia datasets (Figure 10 (left) and Figure 11 (left)), the energy barrier between the basins of model 1 and permuted model 2 is not as low as for our other experiments. This might be attributed to specific properties of the model types (deep network architectures) and datasets. We leave further exploration of this as an important future work.

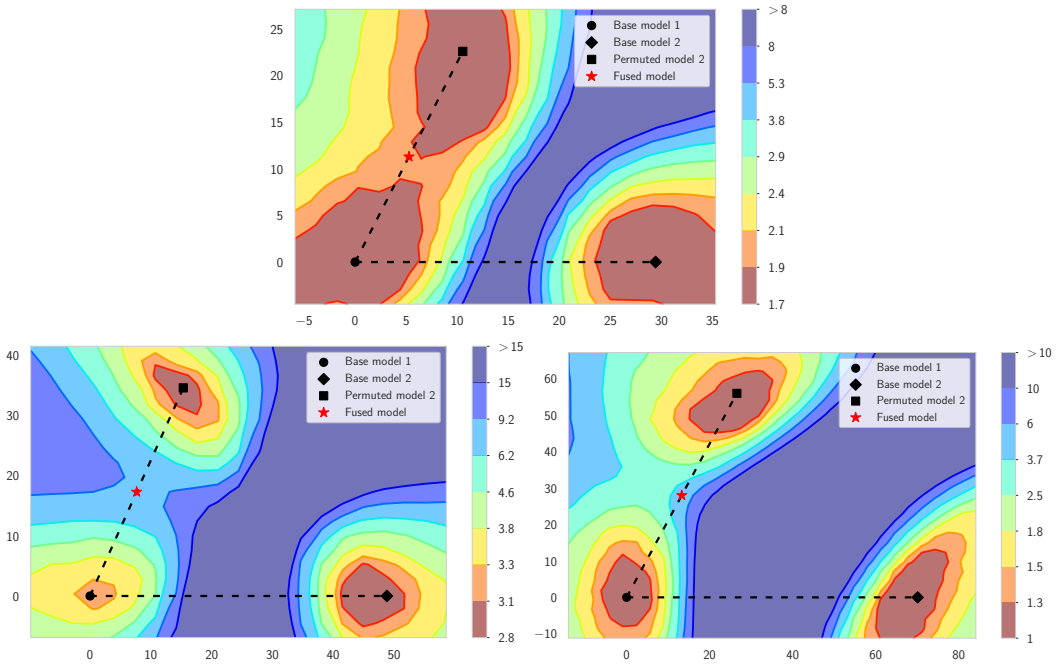

Figure 8: The test error surface of (**Top**) MLPNET trained on MNIST dataset, (**Bottom Left**) RNN trained on MNIST dataset, (**Bottom Right**) LSTM trained on MNIST dataset.

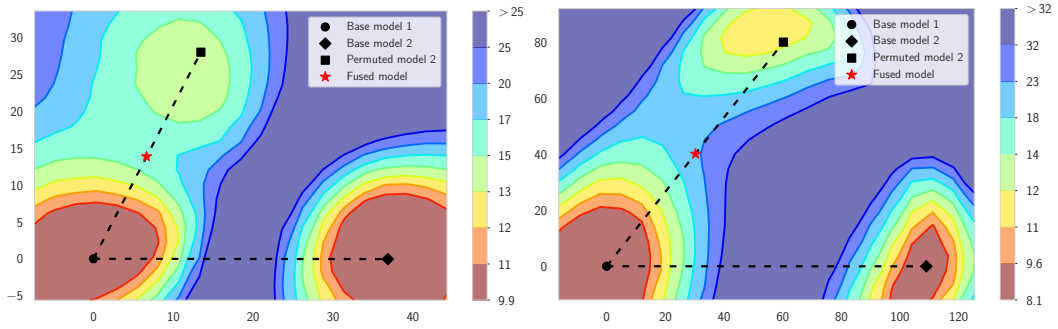

Figure 9: The test error surface of (**Left**) VGG11 trained on CIFAR10 dataset, (**Right**) RESNET18 trained on CIFAR10 dataset.

[1] We use the visualization method proposed in (Garipov et al., 2018); their code is available at https://github.com/timgaripov/dnn-mode-connectivity

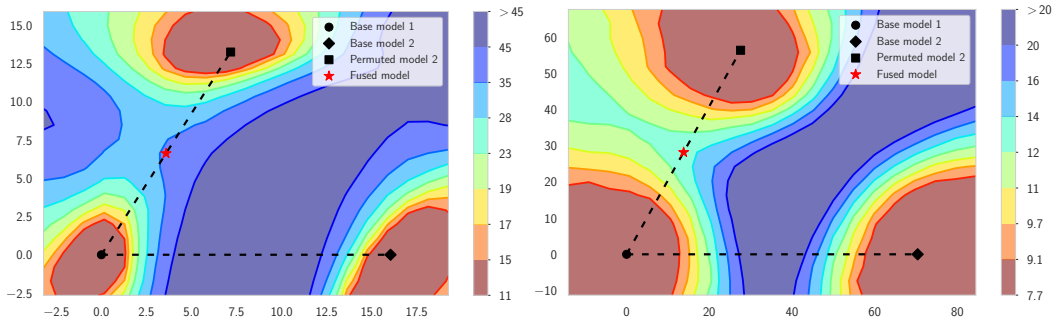

Figure 10: The test error surface of (**Left**) RNN trained on AGNEWS dataset, (**Right**) LSTM trained on AGNEWS dataset.

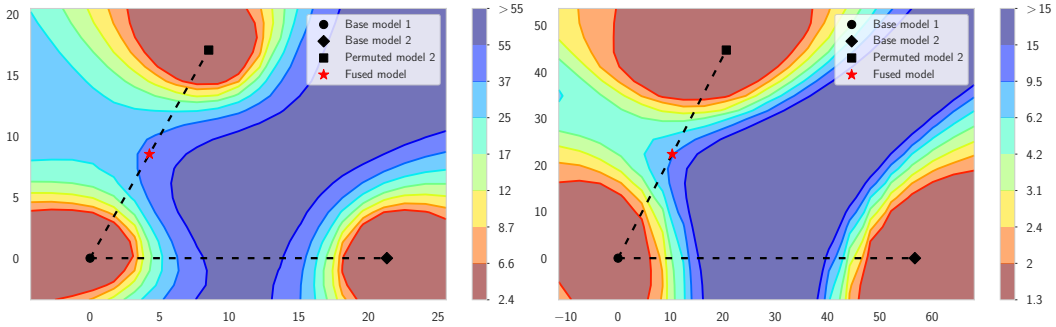

Figure 11: The test error surface of (**Left**) RNN trained on DBpedia dataset, (**Right**) LSTM trained on DBpedia dataset.

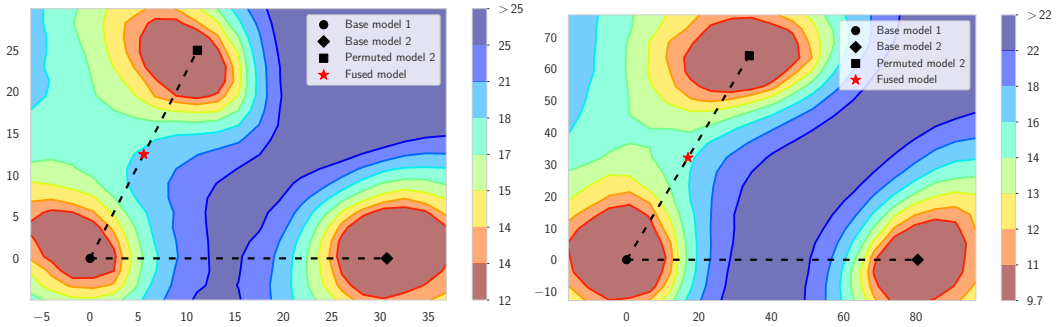

Figure 12: The test error surface of (**Left**) RNN trained on SST-2 dataset, (**Right**) LSTM trained on SST-2 dataset.