Multiple Modes for Continual Learning

Siddhartha Datta
Department of Computer Science
University of Oxford
siddhartha.datta@cs.ox.ac.uk

Nigel Shadbolt
Department of Computer Science
University of Oxford
nigel.shadbolt@cs.ox.ac.uk

Abstract

Adapting model parameters to incoming streams of data is a crucial factor to deep learning scalability. Interestingly, prior continual learning strategies in online settings inadvertently anchor their updated parameters to a local parameter subspace to remember old tasks, else drift away from the subspace and forget. From this observation, we formulate a trade-off between constructing multiple parameter modes and allocating tasks per mode. Mode-Optimized Task Allocation (MOTA), our contributed adaptation strategy, trains multiple modes in parallel, then optimizes task allocation per mode. We empirically demonstrate improvements over baseline continual learning strategies and across varying distribution shifts, namely sub-population, domain, and task shift.

1 Introduction

As the world changes, so must our models of it. The premise of continual (or incremental or lifelong) learning is to build adaptive systems that enable a model to return accurate predictions as the test-time distribution changes, such as a change in domain or task. Training sequentially on multiple different task distributions tends to result in catastrophic forgetting (McCloskey & Cohen, 1989), where parameter updates benefiting the inference of the new task may worsen that of prior tasks. Alleviating this is the motivation for our work.

To enable flexibility in adoption, we do not assume parameter adaptation w.r.t. known task boundaries at test-time, and we assume only access to model parameters alone (no conditioning inputs, query sets, rehearsal or replay buffers, N-shot metadata, or any historical data). This carries positive implications for adoption in online learning settings, and robustness towards different distribution shifts (e.g. sub-population, domain, task shifts). Interestingly. prior work in non-rehearsal methods (notably regularization and parameter isolation methods) tend to "anchor" the parameter updates with respect to a local parameter subspace. These methods begin with a model initialization, then update the model with respect to the first task, and henceforth all future parameter updates on new tasks are computed with respect to this local subspace. The key question we ask here: *what happens when we consider the global geometry of the parameter space?*

Contributions. We introduce a new rehearsal-free continual learning algorithm (Algorithm 1). We initialize pre-trained parameters, maximize the distance between the parameters on the first task, then on subsequent tasks we optimize each parameter based on the loss with respect to their joint probability distribution as well as the each parameter's drift from its prior position (and reinforce with backtracking). Evaluating forgetting per capacity, MOTA tends to outperform baseline algorithms (Table 3), and adapts parameters to sub-population, domain, and task shifts (Tables 1, 2).

Related Work. Lange et al. (2019) taxonomized continual learning algorithms into replay, regularization, and parameter isolation methods. Replay (or rehearsal) methods store previous task samples to supplement retraining with the new task, such as iCaRL (Rebuffi et al., 2017), ER (Ratcliff, 1990; Robins, 1995; Riemer et al., 2018; Chaudhry et al., 2019), and A-GEM (Chaudhry et al., 2018b).

6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans.

Parameter isolation methods allocate different models or subnetworks within a model to different tasks, such as PackNet (Mallya & Lazebnik, 2017), HAT (Serrà et al., 2018), and SupSup (Wortsman et al., 2020). Task oracles may be required to activate the task-specific parameters. Regularization methods add regularization terms to the loss function to consolidate prior task knowledge, such as EWC (Kirkpatrick et al., 2017b), SI (Zenke et al., 2017), and LwF (Li & Hoiem, 2016). These methods tend to rely on no other task-specific information or supporting data other than the model weights alone. We are amongst the first to leverage the global geometry of the loss landscape for task adaptation. Though our method leverages multiple models, we do not assume known task boundaries through a task oracle, and we use smaller models for fair capacity comparison.

2 Trade-off between Multiple Modes and Task Allocation

First we introduce the problem set-up of continual learning (with assumptions extendable to broader online learning settings. Then we share the observations that motivate our study into multiple modes. Finally we present a trade-off, which motivates our proposed learning algorithm.

Problem Setup. A base learner receives T tasks sequentially. $\mathcal{D}_t = \{x_t, y_t\}$ denotes the dataset of the t-th task. In the continual learning setting, given loss function $\mathcal{L}$, a neural network $f(\theta; x)$ optimizes its parameters θ such that it can perform well on the t-th task while minimizing performance drop on the previous $(t-1)$ tasks: $\theta^* := \arg\min_\theta \sum_{t=1}^{T} \mathcal{L}(f(\theta; x_t), y_t)$. We assume the only information available at test-time is the model(s) parameters and the new task's data points. The learner cannot access any prior data points from previous tasks, and capacity is not permitted to increase after each task. Additionally, we do not assume parameter adaptation at test-time can be conditioned on task boundaries or conditioning inputs (task index, K-shot query data, etc).

Trade-off. We denote θ^{init} as the initialization parameter, $\theta^{\text{MTL}(1,\dots,\text{T})}$ as the multi-task parameter trained on tasks $1, ..., T$, and $\theta_{i,t}$ as the parameter of mode index i updated on task t.

Many regularization-based methods are grounded on minimizing drift (change in parameters) to reduce forgetting on prior tasks. Yet in Figure 3, a multi-task learner has a higher average drift consistently between tasks than EWC, even when both begin from a shared starting point (init → task 1). When given visibility to prior tasks, a multi-task learner will depart the subspace of the previous parameter, and drift far. This contradicts with the notion of forcing parameters to reside in the subspace of the previous parameter. The regularization-based methods essentially anchor all future parameters to the first task observed.

Results in mode connectivity (Fort & Jastrzebski, 2019; Draxler et al., 2019; Garipov et al., 2018) show that a single task can have multiple parameters ("modes") that manifest functional diversity. We explored computing multiple modes with respect to task 1 to incorporate the broader geometry of the parameter space beyond the subspace of one mode. To bring performance gains and capacity efficiency, we further obtained this trade-off between the number of modes, number of tasks allocated per mode, and capacity (Theorem 1).

Theorem 1 *If the number of modes N is optimized against capacity $|\theta|$ and the set of tasks allocated per mode $|T(i) = \{t\}|$ for $i \in N$, $t \in T$, then the total task drift is lower in the multi-mode setting than single-mode setting:*

$$\Sigma_{i=1}^{N} \left[\Sigma_{t}^{T(i)} \frac{1}{|\theta|/N} \Sigma_{d=1}^{|\theta|/N} (\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 - \Sigma_{t=2}^{T} \frac{1}{|\theta|} \Sigma_{d=1}^{|\theta|} (\theta_{1,t,d} - \theta_d^{\text{MTL}})^2 \right] < 0$$

Proof. See Appendix A.3.

As a result, our proposed algorithm is motivated to train multiple modes while optimizing for tasks learnt per mode.

3 Mode-Optimized Task Allocation

Mode-Optimized Task Allocation (MOTA) Once the first task is received, we train N models in parallel such that the distance between them is maximized. Then for each subsequent task, we coordinate the parameter updates of the modes such that the drift per mode is minimized, and only the minimum number of modes needed to solve the task will be updated.

3.1 Mode initialization

We begin with a pre-trained initialization. We instantiate N models on this initialization of a fixed model architecture, and denote this set of parameters $\{\theta_{i,t}\}^N$ for $i \in N$ and $t \in T$. We train the parameters in parallel such that the distance between each other is maximized. For each batch per epoch, we randomly generate weights $\{\alpha_i\}^N$ that sum to 1 to compute an interpolated parameter $\widehat{\theta} = \Sigma_i^N \alpha_i \theta_{i,t}$. We compute the distance (cosine similarity) between the N modes $\Sigma_{j,j \neq i}^N$ `dist(`$\theta_{i,t}$, $\theta_{j,t}$ to be maximized (adjusted with a penalty coefficient β_{max}). We update each mode with respect to the input loss (evaluated with $\widehat{\theta}$) and the distance maximization term.

3.2 Mode adaptation

For each mode per epoch, we compute the loss with respect to a joint probability distribution. We also compute a distance term between each mode's parameters and its parameters at the $(t-1)$-th task `dist(`$\theta_{i,t}$, $\theta_{i,t-1}$`)` to be minimized (adjusted with a penalty coefficient β_{min}). We use the EWC regularization term for distance minimization. We update each mode with respect to the joint loss and its respective parameter drift, and checkpoint each update.

We compute the gradient update for each mode with respect to the joint probability distribution between all the modes. Specifically, we compute the average probability distribution returned at the last (softmax) layer $\ell = -1$ of each model $\rho_{\{\theta_{i,t}\}^N} = \frac{1}{N}\Sigma_i^N f(\theta_{i,t}^{\ell=-1}; x)$. If a task has a high level of certainty, then only a small subset of models would need to be updated and return a probability distribution skewed toward the target class while the other non-updated / minimally-updated models would return a random distribution, and the resulting averaged distribution would still be slightly skewed towards the target class. For a task of low certainty, then more models (of high functional diversity) would be updated to return a robust probability distribution. By the time we have jointly-accurate modes, it is likely most of the modes have been over-optimized, and thus over-drifted. Thus, we need to backtrack and find the optimal checkpoints across modes that minimize loss with respect to their checkpointed joint probability distribution and drift. We adopted the simplest backtracking algorithm: we enumerate through every combination of model checkpoint per epoch across the modes, and select the checkpoint combination that minimizes the loss with respect to the joint probability distribution for task t and minimizes total parameter drift.

4 Evaluation

We review our main results on MOTA's improvement in task adaptation here. Detailed experimental configurations and supplementary results can be found in the Appendix.

We evaluate MOTA against different types of distribution shifts (Table 1, Table 2). Evaluating on task shift in CIFAR100 and TinyImageNet, we observe improved backward and forward transfer with MOTA, indicating lower forgetting as well as improved feature transferability between tasks. As task boundaries are not necessary for parameter adaptation, we can evaluate our method on settings that do not require an assumption of task boundaries, namely sub-population and domain shift. We find that our method also outperforms on backward/forward transfer on Instance-IL CIFAR100 (sub-population shift) and Task-IL DomainNet (domain shift).

We primarily baseline MOTA against other regularization-based methods (Table 3). Though MOTA aims to find a combination of parameters that can perform close to the multi-task learning strategy, it falls shortly behind. By optimizing (i) the number of modes throughout the parameter space against (ii) optimal task allocation per parameter, MOTA can outperform other regularization and replay methods. In particular, MOTA outperforms its component baselines: EWC (where this baseline and MOTA use the EWC regularization term in minimizing mode drift), and ensembling (where this baseline and MOTA use the same distance maximization procedure from init to return modes). We show that the optimal combination of these components can yield superior performance.

Table 1: **Metrics evaluation**: Instance-IL (a) presumes the coarse labels are the same between task (5 tasks, 20 labels), thus is representative of sub-population shift. Task-IL (b) presumes unique fine labels per task (10 tasks, 10 labels), and is representative of the general continual learning setting.

(a) Split-CIFAR100 Instance-IL

Method	**Average Accuracy ↑**	**Average Task Drift ↓**	**Capacity ↓**	**Backward Transfer ↑**	**Forward Transfer ↑**	**Remembering ↑**	**Forgetting ↓**
Single-Task Learning	47.1	1.0	23,528,522	-20.2	40.8	79.8	27.0
EWC	48.8	0.360	23,528,522	-12.7	47.2	87.3	17.0
MOTA	52.3	3.13×10^{-5}	22,363,284	-11.6	55.4	88.4	13.9
Multi-Task Learning	64.1	0.910	23,528,522	0	0	1	0

(b) Split-CIFAR100 Task-IL

Method	**Average Accuracy ↑**	**Average Task Drift ↓**	**Capacity ↓**	**Backward Transfer ↑**	**Forward Transfer ↑**	**Remembering ↑**	**Forgetting ↓**
Single-Task Learning	54.8	1.0	23,528,522	-24.7	46.9	75.3	23.7
EWC	66.9	0.521	23,528,522	-8.03	62.7	92.0	6.69
MOTA	70.5	7.85×10^{-6}	22,363,284	-5.08	69.5	94.9	2.89
Multi-Task Learning	77.1	0.725	23,528,522	0	0	1	0

Table 2: **Varying Datasets:** We evaluate on a task shift dataset (a), and domain shift dataset (b).

(a) TinyImageNet Task-IL

Method	**Average Accuracy ↑**	**Backward Transfer ↑**	**Forward Transfer ↑**	**Remembering ↑**	**Forgetting ↓**
Single-Task Learning	65.2	-14.7	65.0	85.3	16.5
EWC	76.7	-4.49	74.7	95.5	3.45
MOTA	82.7	-1.70	81.2	98.3	1.57
Multi-Task Learning	90.2	0	0	1	0

(b) DomainNet Task-IL

Method	**Average Accuracy ↑**	**Backward Transfer ↑**	**Forward Transfer ↑**	**Remembering ↑**	**Forgetting ↓**
Single-Task Learning	34.8	-17.5	22.2	82.5	27.1
EWC	40.2	-9.82	31.6	90.2	8.57
MOTA	57.5	-1.62	55.9	98.4	2.43
Multi-Task Learning	70.4	0	0	1	0

Table 3: **Baseline comparison:** Evaluating on Task-IL Split-CIFAR100, we evaluate MOTA against Single/Multi-Task learning, regularization-based and replay-based methods, and ensemble ablations.

Method	**Average Accuracy ↑**	**Storage: Model parameters ↓**	**Storage: Replay buffer ↓**
Single-Task Learning	54.8	23,528,522	–
EWC (Kirkpatrick et al., 2017b)	66.9	2 × 23,528,522	–
SI (Zenke et al., 2017)	63.7	3 × 23,528,522	–
LwF (Li & Hoiem, 2016)	61.2	23,528,522	–
ER (Riemer et al., 2018)	68.2	23,528,522	10,000
A-GEM (Chaudhry et al., 2018b)	67.2	2 × 23,528,522	10,000
MOTA	70.5	2 × 11,181,642	–
Ensembles (distance max.)	60.1	2 × 11,181,642	–
Ensembles (independent seeds)	55.5	2 × 11,181,642	–
Multi-Task Learning	77.1	23,528,522	–

5 Conclusion

Driven by observations in the optimization behavior of multi-task learners, we hypothesize incorporating the broader geometry of the parameter space into a continual learner for improved adaptation between data distributions. Supported by the formulation of a trade-off between the number of modes and task allocation per mode, we demonstrate that Mode-Optimized Task Allocation (MOTA) can outperform existing baselines. It can retain a high average accuracy on current and previous data in sub-population, domain, and task shift settings.

References

Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, 2018a.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *ArXiv*, abs/1812.00420, 2018b.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning, 2019. URL `https://arxiv.org/abs/1902.10486`.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially no barriers in neural network energy landscape, 2019.

Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning, 2018. URL `https://arxiv.org/abs/1810.13166`.

Stanislav Fort and Stanislaw Jastrzebski. Large scale structure of neural network loss landscapes, 2019.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks, 2017a.

James N Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, and et. al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114 13:3521–3526, 2017b.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Matthias Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ale Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *ArXiv*, abs/1909.08383, 2019.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems*, 2018.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, pp. 614–629. Springer, 2016.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2017.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. 1989.

Adam Paszke, Sam Gross, Francisco Massa, and et. al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation, 2019.

Roger Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285–308, 1990. doi: 10.1037/0033-295X.97.2.285.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pp. 2001–2010, 2017.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995. doi: 10.1080/09540099550039318. URL `https://doi.org/10.1080/09540099550039318`.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*, 2018.

Stanford. Tiny ImageNet Challenge, CS231N Course. URL `https://tiny-imagenet.herokuapp.com/`.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition, 2020.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 3987–3995. JMLR.org, 2017.

A Appendix

A.1 Experimental configurations

Incremental Learning (IL) Settings. A *task* is a subsequent training phase with a new batch of data, pertaining to a new label set, new domain, or different output space. In instance-IL, each new task bring new instances from known classes. In class/task-IL, each new task bring instances from new classes only. Class-IL performs inference w.r.t. all observed classes. Task-IL performs inference w.r.t. the label set of the task. We evaluate on task-IL unless otherwise specified.

Datasets. Task-IL Split-CIFAR100 (Krizhevsky, 2009) is constructed by dividing 100 fine labels into 10 tasks (10 fine labels per task). Instance-IL Split-CIFAR100 (Krizhevsky, 2009) is constructed by dividing 100 fine labels (mapped to 5 coarse labels) into 5 tasks (20 coarse labels per task). Instance-IL DomainNet (Peng et al., 2019) is composed of 6 domains with 345 labels. Task-IL TinyImageNet (Stanford) is constructed by dividing 200 labels into 10 tasks (20 labels per task).

Architectures. We utilize ResNets-{18, 50, 152} (He et al., 2015), loading weights from PyTorch (Paszke et al., 2019) pretrained on ImageNet, with { $11,181,642$ || $23,528,522$ || $58,164,298$ } trainable parameters respectively. To retain comparable capacity, ResNet-50 (-18) is the default model for baselines (MOTA). We loaded pre-trained ImageNet weights for each ResNet architecture with PyTorch (Paszke et al., 2019). We trained for 200 epochs, with batch size 512, using AdamW optimizer (learning rate 0.1 with 1cycle learning rate policy) train/val/test split of 70/10/20%, We train and evaluate (including when averaging the joint probability distribution) using a cross-entropy loss function. We used seed 3407 through all operations; for those requiring multiple unique random values (e.g. multiple randomly initialized models), the seed is the index of the object (1, 2, ...).

MOTA. The distance maximization coefficient β_{max} is 100.0. For the distance minimization procedure in subsequent epochs, we retain the elastic weights consolidation procedure of computing the Fisher information matrix and computing its corresponding regularization term. We retain EWC's lambda $\beta_{\text{min}} = \lambda = 1,000$.

Baselines & Ablations. We do not assume task boundaries at test-time for parameter adaptation, and do not use task index to recompute task-specific parameters. We evaluate against regularization methods (Elastic Weight Consolidation (Kirkpatrick et al., 2017b), Synaptic Intelligence (Zenke et al., 2017), Learning without Forgetting (Li & Hoiem, 2016)). The regularization strength for weight penalty for EWC and SI is $1,000$ and 100 respectively, with SI dampening term 0.1, and LwF's temperature for distillation loss 2.0. We also compare against two replay baselines (Experience Replay (Rolnick et al., 2018), Averaged Gradient of Episodic Memory (Chaudhry et al., 2018b)), though they require a task replay buffer for parameter adaptation. The memory buffer's budget per class is 100, with A-GEM epsilon (parameter to ensure numerical stability of A-GEM) 10^{-7}. We introduce two ablations: Ensemble (distance max.) which is an ensemble of modes obtained using MOTA's distance maximization procedure and trained on all tasks, and Ensemble (independent seeds) which is an ensemble of modes trained on all tasks but from independent random initializations. We used the same number of models in ensemble as the number of modes of the comparable MOTA ($N = 2$ for Table 3). We retain the same distance maximization coefficients as MOTA, and use unique seeds (1,2,...) for each model's random initialization.

Metrics. Single-Task Learning trains on each task independently. Multi-Task Learning trains on all seen tasks simultaneously. Primarily baselined against regularization methods, capacity is the number of trainable model parameters. When considering replay methods as well, we distinguish capacity w.r.t. model parameters from repay buffer, where the replay budget is 100 per class. The average drift distance between tasks is the distance between the updated parameters and previous parameters, averaged for each task update instance. For multiple model parameters, we take the cumulative distance. We compute this from $\frac{1}{T-1}\Sigma_{t=2}^{T}\Sigma_{i}^{N}\mathtt{dist}(\theta_{i,t},\theta_{i,t-1})$. Given $\mathsf{Acc}(\theta, x_t)$ as the validation accuracy on the t-th task, average accuracy is the average validation accuracy across all seen tasks w.r.t. the parameters updated at the t-th task. We compute this from $\frac{1}{t}\sum_{v=1}^{t}\mathsf{Acc}(\theta_t, x_v)$. Backward Transfer (Lopez-Paz & Ranzato, 2017) measures the influence that learning a task has on the performance on previous tasks. We compute this from $\frac{1}{t-1}\sum_{v=1}^{t-1}\mathsf{Acc}(\theta_t, x_v) - \mathsf{Acc}(\theta_v, x_v)$. Forward Transfer (Lopez-Paz & Ranzato, 2017) measures the influence that learning a task has on the performance of future tasks. We compute this from $\frac{1}{t-1}\sum_{v=2}^{t}\mathsf{Acc}(\theta_{v-1}, x_v) - \mathsf{Acc}(\theta^{\text{init}}, x_v)$. Remembering (Díaz-Rodríguez et al., 2018) computes the forgetting part of Backward Transfer. We compute this from $1 - |\min(0, \frac{1}{t-1} \sum_{v=1}^{t-1} \texttt{Acc}(\theta_t, x_v) - \texttt{Acc}(\theta_v, x_v))|$. Forgetting (Chaudhry et al., 2018a) is calculated by the difference of the peak accuracy and ending accuracy of each task. We compute this from $\frac{1}{T-1} \sum_{v=1}^{T-1} \max_{t \in \{1,...,T-1\}} (\texttt{Acc}(\theta_t, x_v) - \texttt{Acc}(\theta_T, x_v))$.

A.2 Distance regularization term

Distance maximization term. For this distance maximization procedure, we used the average cosine similarity between each layer ℓ between every pair of models $\texttt{dist} = \frac{1}{\frac{M}{2}(N^2-N)} \Sigma_{\ell}^{M} \Sigma_{i=1}^{N-1} \Sigma_{j=i+1}^{N} \frac{\theta_i^\ell \cdot \theta_j^\ell}{||\theta_i^\ell||\ ||\theta_j^\ell||}$, out of M and N layers and models respectively.

Distance minimization term. Importance of each parameter is computed for each task by the parameter's corresponding diagonal element from its Fisher Information matrix F. Given the index of the parameters i (i^{th} element of θ_t, i^{th} diagonal element of F), importance of the previous task compared to the next task λ, we can compute the EWC regularization term (Kirkpatrick et al., 2017a): $\sum_{i=1}^{|\theta_{t-1}|} \frac{\lambda}{2} F_i (\theta_t - \theta_{t-1})^2$.

A.3 Analysis on Multiple Modes vs Task Allocation Trade-off

First we denote θ^{init} as the initialization parameter, $\theta^{\text{MTL(1,...,T)}}$ as the multi-task parameter trained on tasks $1, ..., T$, and $\theta_{i,t}$ as the parameter of mode index i updated on task t.

Lemma 1 *Iterating through each task t, for a reference multi-task parameter θ^{MTL}, the cumulative distance between an updated $\theta_{i,t}$ and previous $\theta_{i,t-1}$ parameter with respect to θ^{MTL} will exceed the drift between $\theta_{i,t}$ and $\theta_{i,t-1}$:*

$$\Sigma_{t=2}^{T}(\theta_{i,t} - \theta^{\text{MTL}}) > \Sigma_{t=2}^{T}(\theta_{i,t} - \theta_{i,t-1})$$

Proof. By the triangle inequality, the sum of the distances between the previous and the updated parameter with respect to θ^{MTL} will exceed the drift between the previous and the updated parameter:

$$(\theta_{i,t} - \theta^{\text{MTL}}) + (\theta_{i,t-1} - \theta^{\text{MTL}}) > (\theta_{i,t} - \theta_{i,t-1}) \tag{1}$$

Thus, we can show that that the cumulative distance between an updated parameter with respect to θ^{MTL} will exceed the drift between the updated and previous parameters. Hence, this cumulative distance also measures the task drift.

$$\Sigma_{t=2}^{T}(\theta_{i,t} - \theta^{\text{MTL}}) > \Sigma_{t=2}^{T}(\theta_{i,t} - \theta_{i,t-1}) \tag{2}$$

□

Definition 1 (Task Allocation) *Task Allocation is defined as a procedure that allocates a set of tasks $T(i) = \{t\}$ to be learnt by a parameter mode $\theta_{i,t}$ of index i.*

> **Definition 1.1** (Optimal Task Allocation) *Optimal Task Allocation is defined as a procedure that maximizes the number of tasks $|\{t\}|$ allocated per mode of index i, while minimizing the total drift between parameter updates $\Sigma_{i=1}^{N} \Sigma_{t}^{T(i)} (\theta_{i,t} - \theta_{i,t-1})$.*
>
> **Corollary 1** *We can approximate Optimal Task Allocation by optimizing the number of tasks $|\{t\}|$ allocated to mode i against the cumulative distance between an updated $\theta_{i,t}$ and previous $\theta_{i,t-1}$ parameter with respect to θ^{MTL} (Lemma 1). This results in:*
>
> $$T(i) := \min_{|\{t\}| \geq 1} \sum_{i=1}^{N} \sum_{t}^{T(i)} (\theta_{i,t} - \theta^{\text{MTL}}) \tag{3}$$
>
> *In other words, $T(i) \propto \frac{1}{\theta_{i,t} - \theta^{\text{MTL}}}$. At least one task must be allocated per mode $|T(i)| \geq 1$.*
>
> **Corollary 2** *Given $\mathcal{L}(\not{f}(\theta^{\text{MTL(1,...,T)}}; x^{T(i)}), y^{T(i)}) \approx \mathcal{L}(\not{f}(\theta^{T(i)}; x^{T(i)}), y^{T(i)})$, we use $\theta^{\text{MTL}} = \theta^{\text{MTL(1,...,T)}}$ as the reference multi-task parameter.*

Theorem 1 *If the number of modes N is optimized against capacity $|\theta|$ and the set of tasks allocated per mode $|T(i) = \{t\}|$ for $i \in N$, $t \in T$, then the total task drift is lower in the multi-mode setting than single-mode setting:*

$$\Sigma_{i=1}^{N} \left[\Sigma_{t}^{T(i)} \frac{1}{|\theta|/N} \Sigma_{d=1}^{|\theta|/N} (\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 - \Sigma_{t=2}^{T} \frac{1}{|\theta|} \Sigma_{d=1}^{|\theta|} (\theta_{1,t,d} - \theta_d^{\text{MTL}})^2 \right] < 0$$

Proof. Based on Lemma 1, given N modes and optimal task allocation $T(i)$ with respect to the distance between each θ_i and θ^{MTL}, we can compute the total drift with respect to θ^{MTL} as $\Sigma_{i=1}^{N}\Sigma_t^{T(i)}(\theta_{i,t} - \theta^{\text{MTL}})$.

Note that the capacity of an evaluated mode changes between the multi-mode and single-mode setting. We can compute the total drift normalized by capacity (specifically the number of parameter values) with the squared Euclidean distance averaged by number of dimensions $\frac{1}{|\theta^{\text{MTL}}|}\Sigma_{d=1}^{|\theta^{\text{MTL}}|}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2$, given $|\theta^{\text{MTL}}| \equiv |\theta_{i,t}|$.

From Corollary 1, $|T(i)|$ is larger when θ_i is closer to θ^{MTL}. Thus for a threshold $\mathcal{T}$, we can decompose the total drift into:

$$\begin{aligned}&\Sigma_{i=1}^{N}\Sigma_t^{T(i)}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2\\ =&\Sigma_{i=1}^{N}\left[\Sigma_t^{T(i)\big||T(i)|>\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 + \Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2\right]\end{aligned} \tag{4}$$

Consequently, taking the difference in total drift (Eqt 4) between multiple-mode against single-mode settings result in the follow trade-off function:

$$\begin{aligned}\min \pi &= \Sigma_{i=1}^{N}\left[\Sigma_t^{T(i)}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 - \Sigma_{t=2}^{T}\tfrac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2\right]\\ &= \Sigma_{i=1}^{N}\left[\Sigma_t^{T(i)\big||T(i)|>\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 + \Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2\right]\\ &- \Sigma_{i=1}^{N}\left[\Sigma_t^{T(i)\big||T(i)|>\mathcal{T}}\tfrac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2 + \Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2\right]\end{aligned} \tag{5}$$

Notably, for small $\mathcal{T}$ (e.g. $\mathcal{T} = 1$), the $\Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\frac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2$ term only learns a few tasks per mode, lowers the capacity available per mode $|\theta|/N$, and thus these capacity-inefficient modes are *redundant*. Furthermore, as $\mathcal{T}$ decreases, the functional diversity of a mode is less important, and any random mode can generalize the set of tasks $T(i)\big||T(i)| \leq \mathcal{T}$. Hence,

$$\Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 \approx \Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2 \tag{6}$$

If $N = 1$, then $\pi = 0$, given:

$$\begin{aligned}&\Sigma_t^{T(i)\big||T(i)|>\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 + \Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2\\ \equiv&\Sigma_t^{T(i)\big||T(i)|>\mathcal{T}}\tfrac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2 + \Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2\end{aligned} \tag{7}$$

Performance would be identical to the single-mode sequential learning case.

If $N \to \infty$ (and redundant modes dominate), then $\pi > 0$, given:

$$\begin{aligned}&\Sigma_t^{T(i)\big||T(i)|>\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 + \Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2\\ >&\Sigma_t^{T(i)\big||T(i)|>\mathcal{T}}\tfrac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2 + \Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2\end{aligned} \tag{8}$$

Though the terms where $|T(i)| > \mathcal{T}$ may reduce the cumulative distance compared to a single-mode setting, an extremely large number of modes will result in excess modes only storing one/few tasks. These excess terms will increase, and the cumulative distance from θ^{MTL} will be greater in the multi-mode setting than the single-mode setting.

If $0 < N < \infty$ is optimized, then $\pi < 0$, given:

$$\begin{aligned}&\Sigma_t^{T(i)\big||T(i)|>\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 + \Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2\\ <&\Sigma_t^{T(i)\big||T(i)|>\mathcal{T}}\tfrac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2 + \Sigma_t^{T(i)\big||T(i)|\leq\mathcal{T}}\tfrac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2\end{aligned} \tag{9}$$

For $|T(i)| \leq \mathcal{T}$, any sampled mode will be similarly distant from θ^{MTL}, thus we can cancel this term on both sides.

$$\Sigma_t^{T(i)\big||T(i)|>\mathcal{T}} \frac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 < \Sigma_t^{T(i)\big||T(i)|>\mathcal{T}} \frac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_{1,t,d} - \theta_d^{\text{MTL}})^2 \tag{10}$$

This result shows that, compared to single-mode sequential learning, if we optimize the number of modes N, then we can minimize the cumulative distance with respect to θ^{MTL}, and thus minimize the total task drift.

In other words, we conclude that optimizing the number of modes N against capacity $|\theta|$ and tasks allocated per parameter $|T(i)|$ can outperform training on a single mode. If we increase N, then we can minimize the total task drift. If N is too large, however, then the number of tasks allocated per parameter $|T(i)|$ decreases, and thus increases the number of redundant mode terms (and total task drift).

$$\Sigma_{i=1}^{N}\left[\Sigma_t^{T(i)} \frac{1}{|\theta|/N}\Sigma_{d=1}^{|\theta|/N}(\theta_{i,t,d} - \theta_d^{\text{MTL}})^2 - \Sigma_{t=2}^{T} \frac{1}{|\theta|}\Sigma_{d=1}^{|\theta|}(\theta_d^{t(1)} - \theta_d^{\text{MTL}})^2\right] < 0 \tag{11}$$

□

A.4 Tradeoff between accuracy and capacity

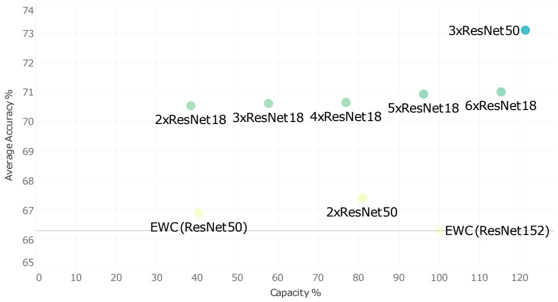

Figure 1: **Varying modes count:** Compared to EWC (ResNet50/152), we evaluate the trade-off between accuracy against capacity (number of modes), given a constant number of tasks.

We learn from ensemble performance in Table 3 that utilizing global geometry alone is not sufficient to improve average accuracy, and that we need to optimally allocate tasks per mode. Similarly in Figure 1, we observe a similar inclination for a balance between capturing global geometry and optimizing tasks per mode. We would expect that, given a constant number of tasks, an increasing number of modes would result in improved average accuracy. Instead, the average accuracy gain between 2-6 ResNet18s is minimal. We also observe a trade-off between the number of modes and optimal task allocation per mode. Considering EWC (ResNet50; i.e. $1\times$ ResNet50) and $2-3\times$ ResNet50, an increase in the number of modes results in an increase in average accuracy. Considering constant capacity, $4\times$ ResNet18 outperforms $2\times$ ResNet50; however, $3\times$ ResNet50 outperforms $6\times$ ResNet18.

A.5 Changes to the geometry of the parameter space

From Table 1, the average task drift (drift distance between the next and previous task's parameters) tends to be lower for MOTA than EWC, Single-Task, and Multi-Task Learning. This can be visually observed in the trajectory of the parameters in Figure 2.

We also observe from Figure 2 that the loss landscape changes drastically between tasks. A region considered to be low-loss by a parameter at task t becomes a high-loss region with respect to the next task $t+1$. As each task is added, the sharpness of the basin upon which the EWC parameter exists tends to increase. This change in sharpness tends to be much smaller for the regions in which MOTA modes are located, where the basin still retains a similar level of flatness.

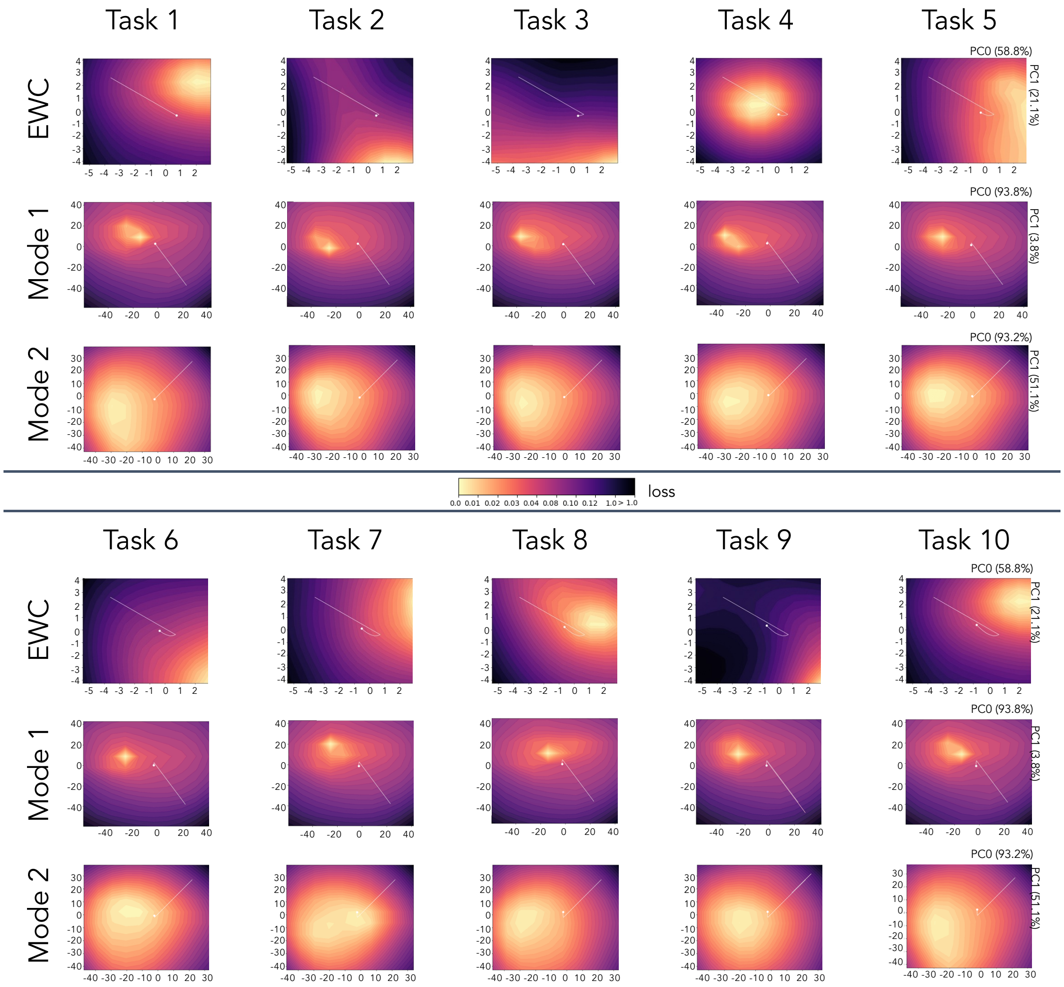

Figure 2: **Loss Landscape:** In-line with Li et al. (2018), we visualize the loss landscape by storing the model parameters along the optimization trajectory per epoch from Task 1-10 (including the last parameter θ^*), identify the top two components/directions δ, η with PCA, and with respect to each task's dataset x_t, y_t we plot the loss function $\mathcal{L}(\theta^* + \alpha_\delta \delta + \alpha_\eta \eta)$ with varying interpolation coefficients α_δ, α_η. We plot each set per method across the tasks to show the relative change in flatness/sharpness between tasks. We normalize the loss values of all plots jointly between 0 and 1. The trajectory (white line) is the position of the parameter in the parameter space at the t-th task. Note that the loss values are not necessarily synchronized for each parameter between tasks (e.g. the init parameter) as the loss for the same parameter may be different for different tasks.

Algorithm 1 `update_parameters`

```
procedure update_parameters($\mathcal{D}_t$, $\{\theta_{i,t}\}_{i=1}^N$)    ▷ Pass a new task $\mathcal{D}_t$ to our current parameters $\{\theta_{i,t}\}_{i=1}^N$
    if $t$=1 then    ▷ Check if initializing parameters for the first time
        $\{\theta_{i,t}\}_{i=1}^N \leftarrow$ initialize_task1($\mathcal{D}_1$, $\{\theta_{i,t}\}_{i=1}^N$)
    else
        $\{\theta_{i,t-1}\}_{i=1}^N \leftarrow \{\theta_{i,t}\}_{i=1}^N$    ▷ Retain a copy of the last task's parameters
        for $e$ in epochs do
            for $\theta_{i,t}$ in $\{\theta_{i,t}\}_{i=1}^N$ do
                for $(x_t, y_t)$ in $\mathcal{D}_t$ do
                    $\rho_{\{\theta_{i,t}\}_{i=1}^N}$ = joint_inference($x_t$, $\{\theta_{i,t}\}_{i=1}^N$)
                    $\mathcal{L}_t = \mathcal{L}(\rho_{\{\theta_{i,t}\}^N}, y_t) + \beta_{\min}\,\mathrm{dist}(\theta_{i,t}, \theta_{i,t-1})$    ▷ Compute loss w.r.t. joint probability and drift
                    $\theta_{i,t} := \theta_{i,t} - \frac{\partial \mathcal{L}_t}{\partial \theta_{i,t}}$    ▷ Update each parameter independently
        $\{t^*, e^*\}^N := \arg\min_{\{t,e\}^N} \mathcal{L}(\rho_{\{\theta_{i,t,e}\}^N}, y_t)$    ▷ Backtracking: Enumerate through parameter combinations
                    $+ \Sigma_{i=1}^N \texttt{dist}(\theta_{i,t,e}, \theta_{i,t-1})$ for $\{\theta_{i,t,e}\}_{i=1}^N \sim \{\theta_{i,t,e}\}_{i=1}^{N\times \texttt{epochs}}$
        $\{\theta_{i,t}\}_{i=1}^N \leftarrow \{\theta_{i,t^*,e^*}\}_{i=1}^N$
    return $\{\theta_{i,t}\}_{i=1}^N$
```

Algorithm 2 `initialize_task1`

```
procedure initialize_task1($\mathcal{D}_1$, $\{\theta_{i,t}\}_{i=1}^N$)    ▷ Initialize parameters on task $\mathcal{D}_1$ and empty parameters set $\{\theta_{i,t}\}_{i=1}^N$
    $\{\theta_{i,t}\}_{i=1}^N \leftarrow \{\theta^{\text{init}}\}_{i=1}^N$
    for $e$ in epochs do
        for $(x_t, y_t)$ in $\mathcal{D}_1$ do
            $\alpha_i \sim [0,1]\ \forall i \in N$ s.t. $\Sigma_{i=1}^N \alpha_i \equiv 1$
            $\widehat{\theta} = \Sigma_{i=1}^N \alpha_i \theta_{i,t}$    ▷ Sample interpolated parameter $\widehat{\theta}$
            $\mathcal{L}_t = \mathcal{L}(f(\widehat{\theta}); x_t, y_t) + \beta_{\max}\ \Sigma_{j=1,j\neq i}^N\ \mathrm{dist}(\theta_{i,t}, \theta_{j,t})$    ▷ Compute loss and distance term
            $\theta_{i,t} := \theta_{i,t} - \frac{\partial \mathcal{L}_t}{\partial \theta_{i,t}}$    ▷ Update each parameter independently
    return $\{\theta_{i,t}\}_{i=1}^N$
```

Algorithm 3 `joint_inference`

```
procedure joint_inference($x$, $\{\theta_{i,t}\}_{i=1}^N$)    ▷ Inference using the set of parameters $\{\theta_{i,t}\}_{i=1}^N$
    $\rho_{\{\theta_{i,t,e}\}^N} = \frac{1}{N}\Sigma_{i=1}^N f(\theta_{i,t}^{\ell=-1}; x)$    ▷ Taking average of the joint probability distribution returned at the softmax layer $\ell = -1$
    return $\rho_{\{\theta_{i,t,e}\}^N}$
```