A Details of the experimental setup

A.1 Datasets

We performed experiments using the following datasets.

1. Rotated-MNIST (Lopez-Paz and Ranzato, 2017) uses the MNIST dataset to generate 5 different 10-way classification tasks. Each task involves using the entire MNIST dataset rotated by 0, 10, 20, 30, and 40 degrees, respectively.
2. Permuted-MNIST (Kirkpatrick et al., 2017) involves 5 different 10-way classification tasks with each task being a different permutation of the input pixels. The first task is the original MNIST task as is convention. All other tasks are distinct random permutations of MNIST images.
3. Split-MNIST (Zenke et al., 2017) has 5 tasks with each task consisting of 2 consecutive labels (0-1, 2-3, 4-5, 6-7, 8-9) of MNIST.
4. Split-CIFAR10 (Zenke et al., 2017) has 5 tasks with each task consisting of 2 consecutive labels (airplane-automobile, bird-cat, deer-dog, frog-horse, ship-truck) of CIFAR10.
5. Split-CIFAR100 (Zenke et al., 2017) has 20 tasks with each task consisting of 5 consecutive labels of CIFAR100. See the original paper for the exact constitution of each task.
6. Coarse-CIFAR100 (Rosenbaum et al., 2017) has 20 tasks with each task consisting of 5 labels. The tasks are based on an existing categorization of classes into super-classes (https://www.cs.toronto.edu/ kriz/cifar.html).
7. Split-miniImagenet (Vinyals et al., 2016) is a ariant introduced in Chaudhry et al. (2019b), consisting of 20 tasks, with each task consisting of 10 consecutive labels. We merge the meta-train and meta-test categories to obtain a continual learning problem with 20 tasks. Each task containing 10 consecutive labels and 20% of the samples are used as the validation set.

The CIFAR10 and CIFAR100-based datasets consist of RGB images of size 32×32 while MNIST-based datasets consist of images of size 28×28. The Mini-imagenet dataset consists of RGB images of size 84×84.

A.2 Architecture

We use the Wide-Resnet (Zagoruyko and Komodakis, 2016) architecture for some of our experiments. The final pooling layer is replaced with an adaptive pooling layer in order to handle input images of different sizes. Convolutional layers are initialized using the Kaiming-Normal initialization. The bias parameter in batch normalization is set to zero with the affine scaling term set to one. The bias of the final classification layer is also set to zero; this helps keep the logits of the different tasks on a similar scale.

To ensure that the number of weights is similar to those in other methods, we also consider a smaller convolution neural network consisting of 3 convolution layers, with batch-normalization, ReLU and max-pooling present between each layer.

A.3 Training setup

Optimization. All models are trained in mixed-precision (32-bit weights, 16-bit gradients) using Stochastic Gradient Descent (SGD) with Nesterov's acceleration with momentum coefficient set to 0.9 and cosine annealing of the learning rate schedule for 200 epochs. Training of any model with multiple tasks involves mini-batches that contain samples from all tasks.

Hyper-parameter optimization. We used Ray Tune (Liaw et al., 2018) for hyper-parameter optimization. The Async Successive Halving Algorithm (ASHA) scheduler (Li et al., 2018) was used to prune hyper-parameter choices with the search space determined by Nevergrad (Rapin and Teytaud, 2018). The mini-batch size was varied over [8, 16, 32, 64]; the logarithm (base 10) of the learning rate was sampled from a uniform distribution on $[-4, -2]$; dropout probability was sampled from a uniform distribution on $[0.1, 0.5]$; logarithm of the weight decay coefficient was sampled from $[-6, -2]$. We used a set of experiments for continual learning on the Coarse-CIFAR100 dataset with different samples/class (100 and 500) to perform hyper-parameter tuning.

The final values of training hyper-parameters that were chosen are, learning-rate of 0.01, mini-batch size of 16, dropout probability of 0.2 and weight-decay of 10^{-5}.

Model Zoo uses $\mathscr{b} = \min(k, 5)$ at each round of continual learning where n is the number of tasks; for tasks with only 5 tasks (MNIST-variants) we use $\mathscr{b} = 2$. We did not tune these two hyper-parameters using Ray because it is quite cumbersome to do so. We selected these values manually across a few experiments; changing them may result in improved accuracy for Model Zoo.

All hyper-parameters are kept fixed for all datasets, architectures, and experimental settings. . We are interested in characterizing the performance of Model Zoo and its variants across a broad spectrum of problems and datasets. While we believe we can get even better numerical accuracy, by tuning hyper-parameters specially for each problem, we do not so for the sake of simplicity. As the main paper discusses, we outperform existing methods quite convincingly across the board in both multi-task and continual learning.

Data augmentation. MNIST and CIFAR10/100 datasets use padding (4 pixels) with random cropping to an image of size 28×28 or 32×32 respectively for data augmentation. CIFAR10/100 images additionally have random left/right flips for data augmentation. Images are finally normalized to have mean 0.5 and standard deviation 0.25. Split-miniImagenet uses the same augmentation as CIFAR-10 and CIFAR-100. We use augmentation even in the single epoch setting.

A.4 Model Zoo with Limited Replay

As discussed in §4.2, this work considers Model Zoo (10%) which stores only 10% of the data from the past tasks, in order to compare to other methods that make use of limited replay. When the task (say task A) is first seen, Model Zoo is allowed to use all available data. For all future episodes, if Model Zoo picks a past task to retrain with, such a retraining uses only a fixed subset of the tasks' data (10% of the samples are selected at random for this purpose). We sample each mini-batch to contain an equal number of samples from all past and current task. At inference time, the member of Model Zoo that is trained on all data of task A (this is the model that was fitted when task A was first shown to the continual learner) is assigned a proportionately larger weight in Eq. (7). For 10% replay, this will amount to 10 $\times$ larger weight than other models which used 10% data from task A. Mathematically, both of these training and inference modifications are equivalent to using coefficients that scale up the loss of the past task depending upon the number of samples that it has.

A.5 Evaluating Training and Inference Times

In this section, we describe the methodology used to estimate training and inference times reported in Table 2.

Inference time. The column titled inference time corresponds to per-sample prediction latency in milli-seconds. All entries for inference time in Table 2 were computed by us on an Nvidia V100 GPU and therefore they can be compared directly with each other. Note that inference times can be computed using only the architecture built by each method at the end of all continual learning episodes. We obtained the architectures used in each method from open-source implementations of the original authors (https://github.com/facebookresearch/agem and https://github.com/imirzadeh/stable-continual-learning). Inference time is computed by processing 50 mini-batches from CIFAR-100, each of batch-size 16. The inference time is computed by normalizing the total computation time by (size of mini-batch $\times$ number of mini-batches), which gives the average inference-time per sample. For Model-Zoo, we assume that inference time is approximately $\mathscr{b} = 5$ times the inference time of Isolated, where $\mathscr{b}$ is number of tasks sampled in every round).

Training time. corresponds to the time (in minutes) required to train all episodes of the Split-CIFAR100 dataset. Establishing an accurate comparison is difficult because different papers used different hardware but we have strived to be fair. The training time for EWC, Prog-NN, GEM and A-GEM are obtained from Chaudhry et al. (2019a) (we divide the numbers by 5 since this paper reports the sum of training times of 5 different runs). Chaudhry et al. (2019a) also report the training time for naive fine-tuning (21 mins) which in theory, should be very similar to the training time of our Isolated learner (the training time for us is 20.76 mins on one V100 GPU). Since the two numbers are quite similar, we can estimate training time of the other continual learning methods using their computational cost relative to naive fine-tuning. Therefore, the estimate of the training times that we have reported in Table 2 can be compared to each other.

B Additional Experiments

B.1 Understanding task competition

To understand which tasks aid each other's learning and which compete for capacity and may thereby deteriorate performance, we investigated the Coarse-CIFAR100 dataset extensively. We first computed the pairwise task competition by comparing the relative gain/drop in classification accuracy of each pair of tasks when the row task is trained in isolated versus training the row and column tasks together using a simple multi-task learner (Multi-Head). Fig. A1 discusses the results.

Fig. A2, is the extended version of Fig. 2. It shows the validation accuracy of each task (along a single row) as more tasks are added to Multi-Head. Each column is a single Multi-Head model trained on a subset of tasks from scratch. As more tasks are added, the accuracy of most tasks increases.

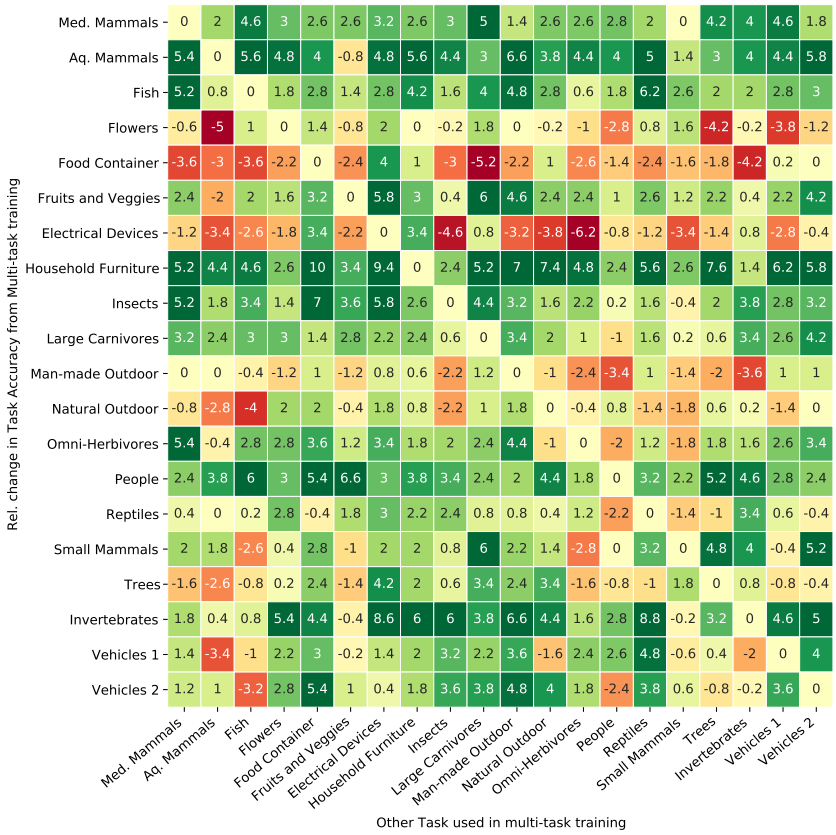

Figure A1: Pairwise task competition matrix. Cells are colored by the gain(green)/loss(warm) of accuracy of pairwise Multi-Head training as compared to training the row-task in isolation; this is a good proxy for the transfer coefficient ρ_{ij} in (5). Although most pairs benefit each other (green), certain tasks, e.g., "Food Container" are best trained in isolation while others such as "Aquatic Mammals" are typically detrimental to most other tasks. One can study this matrix and identify many more such properties. In summary, whether tasks aid or hurt each other is quite nuanced even for CIFAR100.

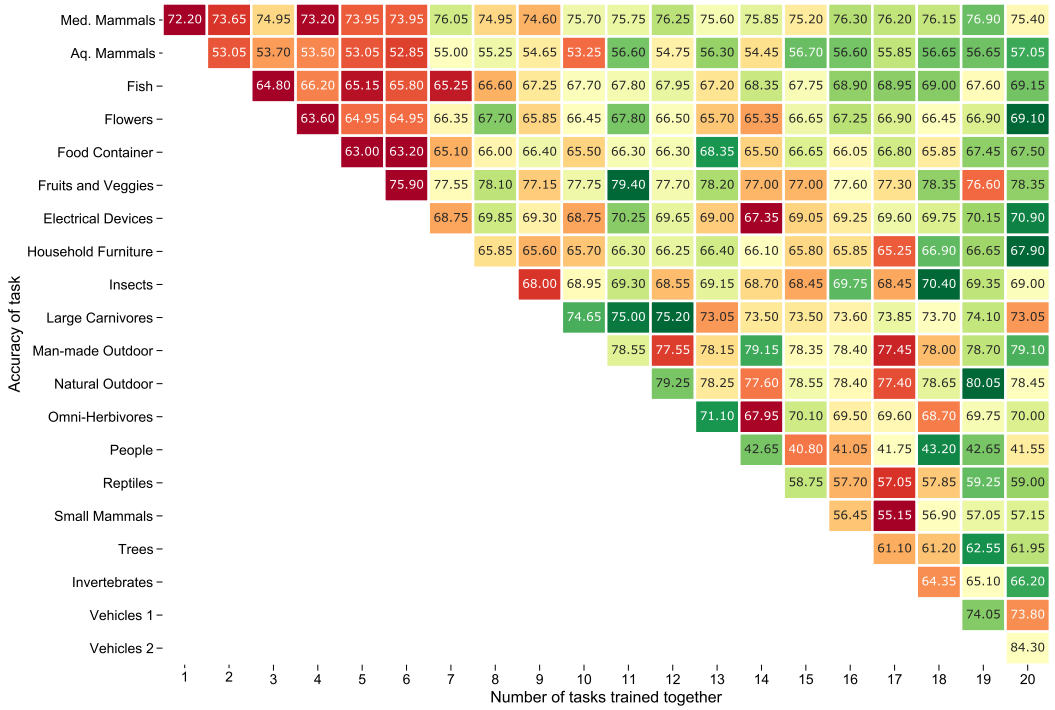

Figure A2: In order to demonstrate how some tasks help and some tasks hurt each other, we run Multi-Head for a varying number of tasks (X-axis) and track the accuracy on a few tasks from Coarse-CIFAR100. The order of tasks is the same for rows (top to bottom) and the columns (left to right). In other words, the first cell (the diagonal) indicates the accuracy of the task trained by itself in isolation (Isolated). Cells are colored warm if accuracy is worse than the median accuracy of that row. For instance, multi-task training with 11 tasks is beneficial for "Man-made Outdoor" but accuracy drops drastically upon introducing task #12, it improves upon introducing #14, while task #17 again leads to a drop. One may study the other rows to reach a similar conclusion: there is non-trivial competition between tasks, even in commonly used datasets. Tackling this issue effectively is the key to obtaining good performance on multi-task learning problems

However, the increase is not monotonic with each added task, and if one follows a particular row, there are non-trivial patterns wherein adding a particular task may deteriorate the performance on the row task and adding some other task later may recover the lost accuracy. This is a direct demonstration of the tussle between the task competition term (first) and the concentration term (third) in Theorem 2. This indicates that training on the appropriate set of tasks is crucial to learn from multiple tasks.

B.2 Competition between tasks of typical benchmark datasets

Next, we investigated such task competition on other continual learning datasets, namely, Permuted-MNIST, Rot-MNIST, Split-CIFAR10, and Split-MNIST. It is clear from Fig. A3 that there is very little competition in this case. Either the tasks are quite different from each other (like the case of Permuted-MNIST), or they are synergistic (most cells are green), or they do not hurt each other's performance, i.e., they may correspond to the model in §2.2. Note that Rotated-MNIST exactly corresponds to the multi-view setting discussed in §2.2 were different input images are simple transformations of each other.

B.3 Visualizing successive iterations of Model Zoo

In order to understand how the accuracy of Model Zoo evolves on all tasks as a function of the episodes, we created Fig. A4. This is a very insightful picture and we can draw the following conclusions from it.

(i) The accuracy along the diagonal of most tasks increases along the row, i.e., across episodes. Only for a few tasks like Food Container the accuracy drops in later episodes. Note that we also see from Fig. A1 that Food Container is a task that is best trained in isolation because it leads to deterioration of accuracy when trained with essentially any other task.

(ii) The is strong backward transfer throughout the dataset, i.e., the accuracy of a task shown in earlier rounds increases, sometimes a large number, as later synergistic tasks are shown to the learner.

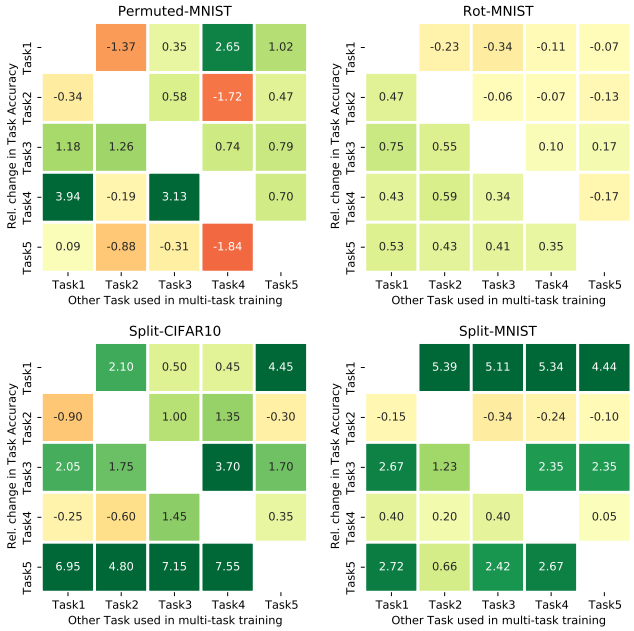

Figure A3: Each row is the relative increase/decrease (green/red) in accuracy of a two task Multi-Head learner compared to Isolated trained on the task corresponding to the particular row; all entries are computed using 100 samples/class. Cells are colored green for accuracy gained, and warm for accuracy dropped; the entries in this matrix are a good proxy for the transfer coefficient ρ_{ij} in (5). A similar plot for Coarse-CIFAR100 tasks is shown in the right panel of Fig. 2. Split-CIFAR10 and Split-MNIST indicate that most tasks mutually benefit each other. This is also true, but to a lesser extent, for Rotated-MNIST. Permuted-MNIST is a qualitatively different problem than these, perhaps because there is no obvious relationship between the tasks and there exist some tasks that lead to a large deterioration of accuracy.

(iii) We also see strong forward transfer. Roughly speaking, in the second half of the rows, the initial accuracy of most tasks does not improve much with successive episodes. This suggests that these tasks already have a good initial accuracy, i.e., there is good forward transfer in the learner.

We advocate that such plots should be made for different continual learning algorithms to obtain a precise picture of the amount of forward and backward transfer.

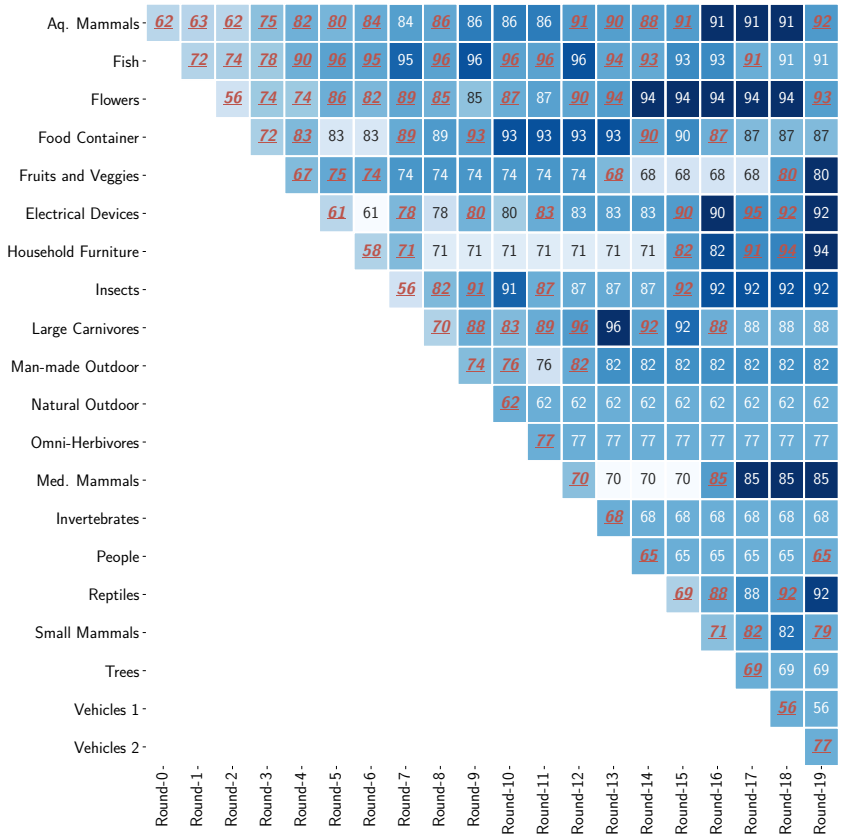

Figure A4: The iterations of Model Zoo are visualized for the Split-miniImagenet dataset for 20 rounds, with 5 tasks selected in every iteration of Model Zoo. Red elements are tasks that were selected by boosting in that particular round. We observe that the accuracy of most tasks improves over the rounds, which indicates the utility of Model Zoo-like training scheme This plot also indicates that Model Zoo can improve the per-task accuracy on nearly all tasks. The model is trained for only a single-epch per boosting round.

B.4 BASELINE PERFORMANCE OF ISOLATED TRAINING ON COARSE-CIFAR100

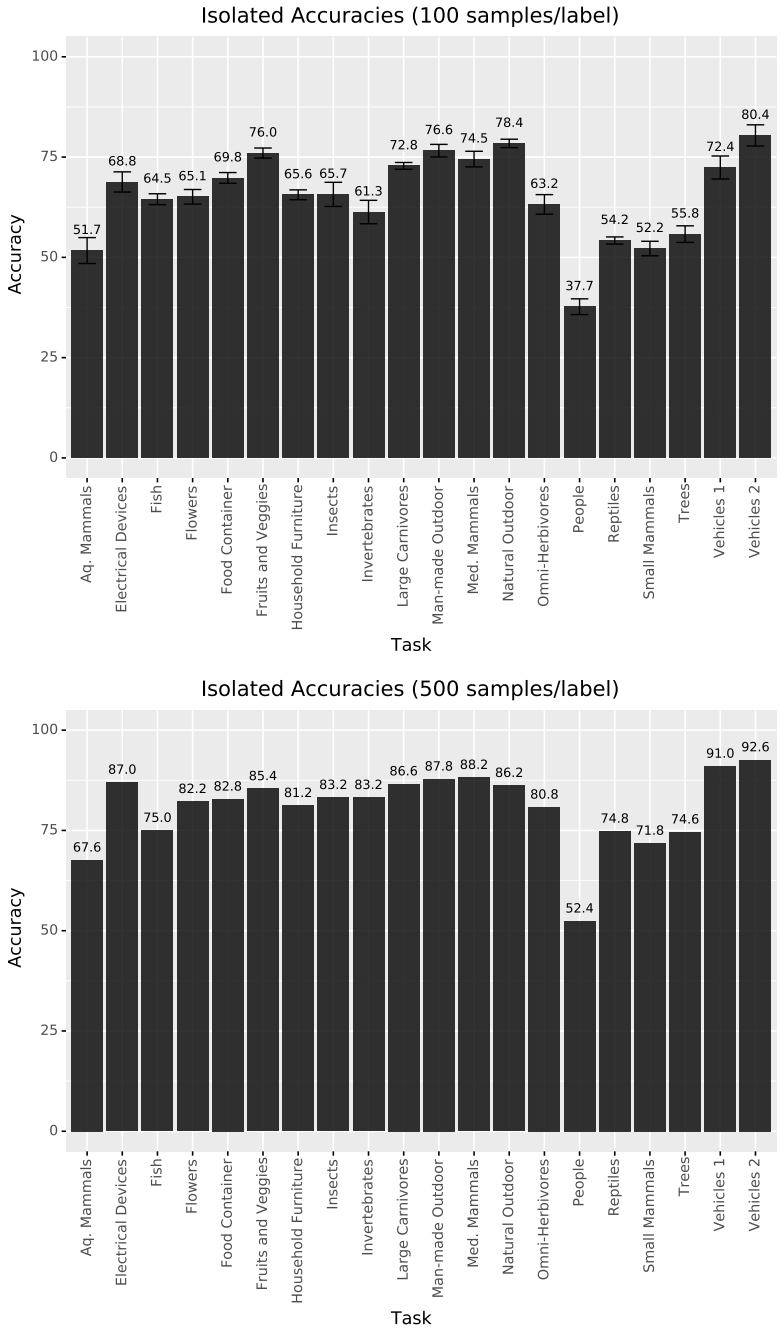

Figure A5: Per-task accuracies of Isolated on the Coarse-CIFAR100 dataset for two cases, one with 100 samples/class (top) and another with all 500 samples/class (bottom). Two points are very important to note here. First, there is a large improvement in the two accuracies for all tasks when the learner has access to more samples. Second, different tasks have very different accuracies when trained in isolation (using the same WRN-16-4 model). This indicates that different tasks are very different in terms how hard they are, for some tasks such as People, the base accuracy of the model is quite low and one must have lots of samples in order to perform well. A lot of other multi-task learning datasets, e.g., derivatives of MNIST (or even CIFAR10 to an extent) are unlike CIFAR100 in this respect.

B.5 ADDITIONAL EXPERIMENTS

Table A1 is a more detailed version of Table 1 in the main paper.

Method	Replay	Single Epoch	Rotated-MNIST	Permuted-MNIST	Split-MNIST	Split-CIFAR10	Split-CIFAR100	Coarse-CIFAR100	Split-MiniImagenet
GEM (Lopez-Paz and Ranzato, 2017)	✓	✓	86.07	82.60	-	-	67.8*	-	51.86
A-GEM (Chaudhry et al., 2019a)	✓	✓	-	89.1	-	-	62.3*	-	61.13
ER-Reservoir (Chaudhry et al., 2019b)	✓	✓	-	79.8	-	-	68.5*	-	64.03
MC-SGD (Mirzadeh et al., 2020a)	✓	✓	82.63	85.3	-	-	63.30	-	-
MEGA-II (Guo et al., 2020a)	✓	✓	-	91.20	-	-	66.12	-	-
OGD (Farajtabar et al., 2020)	✗	✓	88.32	86.44	98.84	-	-	-	-
Stable-SGD (Mirzadeh et al., 2020b)	✗	✓	70.8	80.1	-	-	59.9*	-	57.79
TAG (Malviya et al., 2021)	✗	✓	-	-	-	-	62.79	-	57.2
VCL (Nguyen et al., 2017)	✓	✗	-	95.5	98.4	-	-	-	-
FRCL (Titsias et al., 2020)	✓	✗	-	94.3	97.8	-	-	-	-
FROMP (Pan et al., 2020)	✓	✗	-	94.9	99.0	-	-	-	-
EWC (Kirkpatrick et al., 2017)	✗	✗	•84	•96.9	-	-	•42.40	-	-
Prog-Nets (Rusu et al., 2016)	✗	✗	-	•93.5	-	-	•59.2	-	-
SI (Zenke et al., 2017)	✗	✗	-	•97.1	•98.9	-	-	-	-
HAT(Serra et al., 2018)	✗	✗	-	98.6	99.0	-	-	-	-
APD (Yoon et al., 2019)	✗	✗	-	-	-	-	-	56.81	-
FedWeIT (Yoon et al., 2021)	✗	✗	-	-	-	-	-	55.16	-
RMN (Kaushik et al., 2021)	✗	✗	-	97.73	99.5	-	80.01	-	-
Our methods									
Isolated-small	✗	✗	-	-	-	96.88	90.18	69.07	82.48
Model Zoo-small	✓	✗	-	-	-	96.85	92.06	73.72	94.27
Model Zoo-small (10% replay)	✓	✗	-	-	-	96.58	89.76	77.18	84.6
Isolated-Resnet	✗	✗	-	-	-	-	88.95	-	-
Model Zoo-Resnet	✓	✗	-	-	-	-	93.15	-	-
Isolated	✗	✗	99.64	98.03	99.98	97.46	91.90	80.72	86.28
Model Zoo	✓	✗	99.66	97.71	99.97	98.68	94.99	84.27	96.84
Multi-Head (multi-task)			99.66	98.16	99.98	98.11	95.38	83.19	90.83

Table A1: Average per-task accuracy (%) for continual learning at the end of all episodes. MNIST, Permuted-MNIST and Rotated-MNIST are not informative benchmarks for judging forward and backward transfer because even Isolated achieves 99%+ accuracies. Model Zoo outperforms, by significant margins, all existing continual learning methods; in fact their accuracy is worse than Isolated which suggests little to no forward or backward transfer. **Note:** * indicates that the evaluation was on Split-CIFAR100 with each task containing randomly sampled labels and is hence it is not directly comparable to other methods. All numbers without a marker are from the paper cited in the first column. • denotes that the accuracy is not from the original paper but from one of (Nguyen et al., 2017; Serra et al., 2018; Chaudhry et al., 2019a). Numbers for other methods on Split-MiniImagenet were computed by us using open-source implementations of the original authors.

B.6 Single Epoch Metrics

We obtain metrics from publicly available implementations of a few different continual learning algorithms, which are shown in Tables A2 and A3. We see that Model Zoo and its variants uniformly have essentially no forgetting and good forward transfer. The average per-task accuracy is also higher than existing methods on these datasets. These tables show results for single-epoch training (to be consistent with the implementation of these existing methods).

Method	Avg. Accuracy	Forgetting	Forward
SGD	34.52	19.88	53.30
EWC	34.71	18.60	52.19
AGEM	37.23	16.96	52.72
ER	41.36	14.29	54.87
Stable-SGD	37.27	12.07	48.43
TAG	43.33	12.39	55.1
Isolated-small	58.719	0.0	58.71
Model Zoo-small	60.3	0.370	59.13
Isolated-large	41.28	0.0	41.28
Model Zoo-large	46.98	0.38	44.43

Table A2: Single Epoch continual learning metrics on Coarse-CIFAR100

Method	Avg. Accuracy	Forgetting	Forward
SGD	46.69	16.653	62.35
EWC	47.93	14.26	61.34
AGEM	51.86	10.102	61.13
ER	55.41	9.52	64.03
Stable-SGD	49.28	9.76	57.79
TAG	58.38	5.15	63.00
Isolated-small	65.8	0.0	65.8
Model Zoo-small	81.049	1.278	66.57
Isolated-large	40.2	0.0	40.25
Model Zoo-large	64.12	0.27	48.34

Table A3: Single Epoch continual learning metrics on Split-MinImagenet

B.7 Tracking Individual Task Accuracies

We next study how the individual per-task accuracy evolves on different datasets. The following figures are extended versions of the right panel of Fig. 1. We see that the accuracy of all tasks increases with successive episodes. This is quite uncommon for continual learning methods and indicates that Model Zoo essentially does not suffer from catastrophic forgetting. We have also juxtaposed the corresponding curves of the single-epoch setting with the multi-epoch training in Model Zoo; we would like to demonstrate the dramatic gap in the accuracy of these problem settings. Even if single-epoch variant of Model Zoo also does not forget (its accuracy is much better than existing continual learning methods), the multi-epoch variant has much higher accuracy for every task. This indicates that continual learning algorithms should also focus on per-task accuracy in addition to mitigating forgetting, if they are to be performant. The performance of Model Zoo is evidence that we can build effective continual learning methods that do not forget.

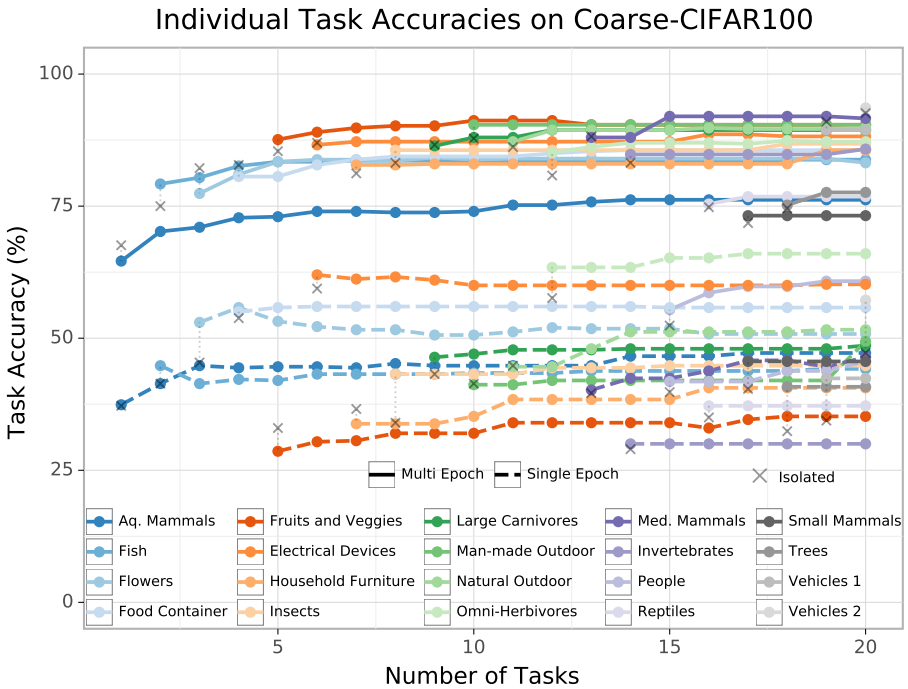

Figure A6: Evolution of task accuracy on Coarse-CIFAR100

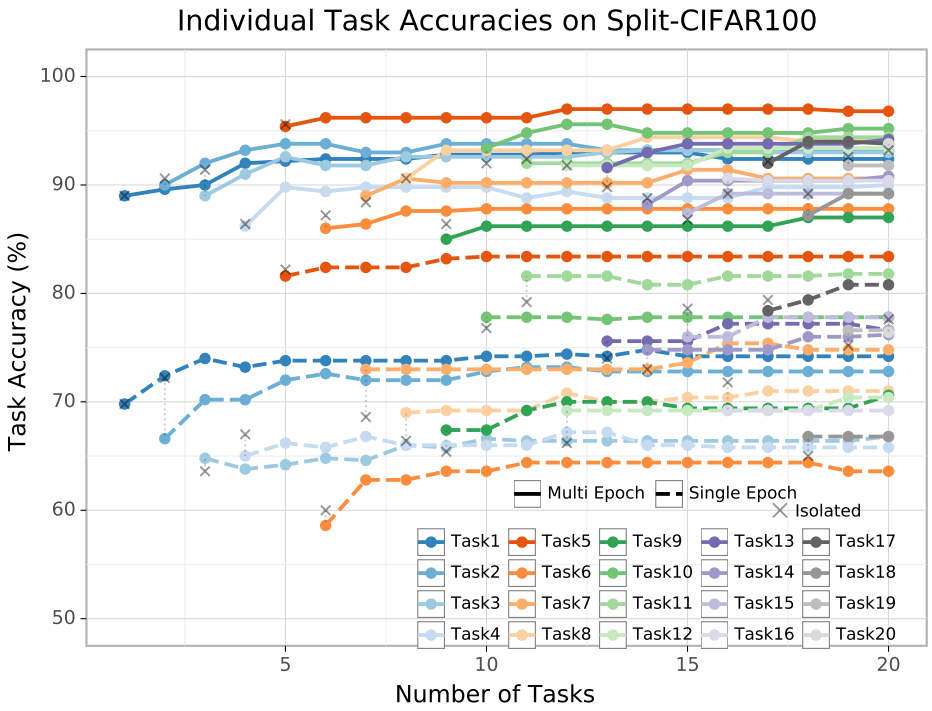

Figure A7: Evolution of task accuracy on Split-CIFAR100

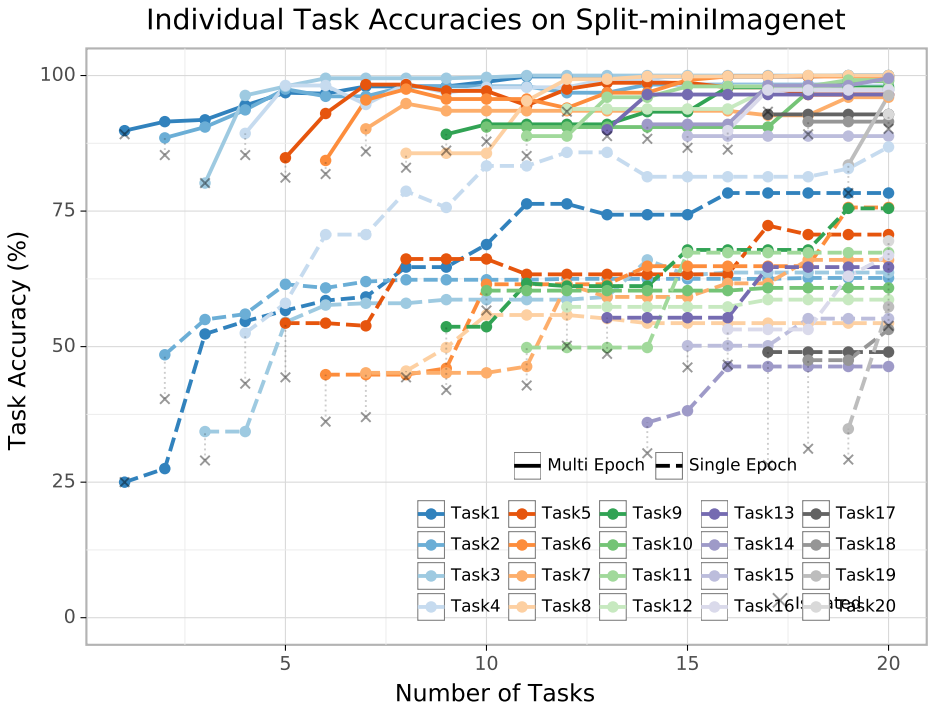

Figure A8: Evolution of task accuracy on Split-miniImagenet

B.8 Comparison To Existing Methods

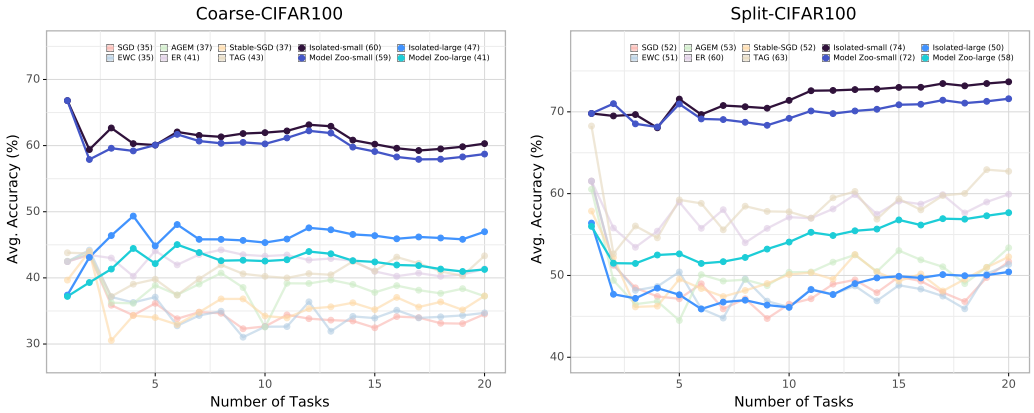

Figure A9: This figure compares Model Zoo to existing continual learning methods on the Coarse-CIFAR100 and Split-CIFAR100 datasets with respect to average task accuracy. Model Zoo and its variants are in bold, similar to the left panel of Fig. 1 (which is for Split-miniImagenet). Isolated-small and Model Zoo-small significantly outperform existing methods. All methods in the figure are run in the single-epoch setting.

B.9 Additional Continual Learning Experiments on 100 samples/label

We also performed continual learning experiments with 100 samples/class in Table A4. We find that Model Zoo-continual obtains an accuracy that lies in between those of Isolated and the approximate upper bound given by Multi-Head (multi-task learning). Note that we have shown that matching or improving upon the performance of Isolated (which trains a model independently for each task) for continual learning is quite difficult because it necessitates effective forward-backward transfer. Doing so indicates strong ability of the learner for *both* forward and backward transfer. In some cases, the continual learner even outperforms Multi-Head trained on all tasks together. This table indicates that Model Zoo can be used as a continual learning and demonstrate nontrivial forward and backward transfer even with few samples from each class.

Dataset	Isolated	Multi-Head (multi-task)	Model Zoo-Continual
Rotated-MNIST	98.17 ± 0.24	98.47 ± 0.18	98.44 ± 0.17
Split-MNIST	97.11 ± 1.21	99.47 ± 0.08	98.98 ± 0.51
Permuted-MNIST	84.59 ± 1.65	86.36 ± 1.15	86.04 ± 1.68
Split-CIFAR10	82.09 ± 0.76	85.73 ± 0.60	84.17 ± 0.60
Split-CIFAR100	80.04 ± 0.44	87.93 ± 0.50	86.27 ± 0.19
Coarse-CIFAR100	65.34 ± 0.41	69.05 ± 0.38	66.80 ± 6.27

Table A4: Average per-task accuracy (%) for continual learning at the end of all episodes using 100 samples/class, bootstrapped across 5 datasets (mean $\pm$ std. dev.). Model Zoo-continual performs better than Isolated on all problems even if tasks are shown sequentially.

We next visualize the evolution of the per-task test accuracy for various datasets. This is a qualitative way to investigate forward and backward transfer in the learner. Forward transfer is positive if the accuracy of a newly introduced task in a particular episode is higher than what it would be if the task were trained in isolation. Backward transfer is positive if successive episodes and tasks result in an increase in the accuracy of tasks that were introduced earlier in continual learning. Both Appendix B.7 and Fig. A10 consistently show non-trivial forward and backward transfer.

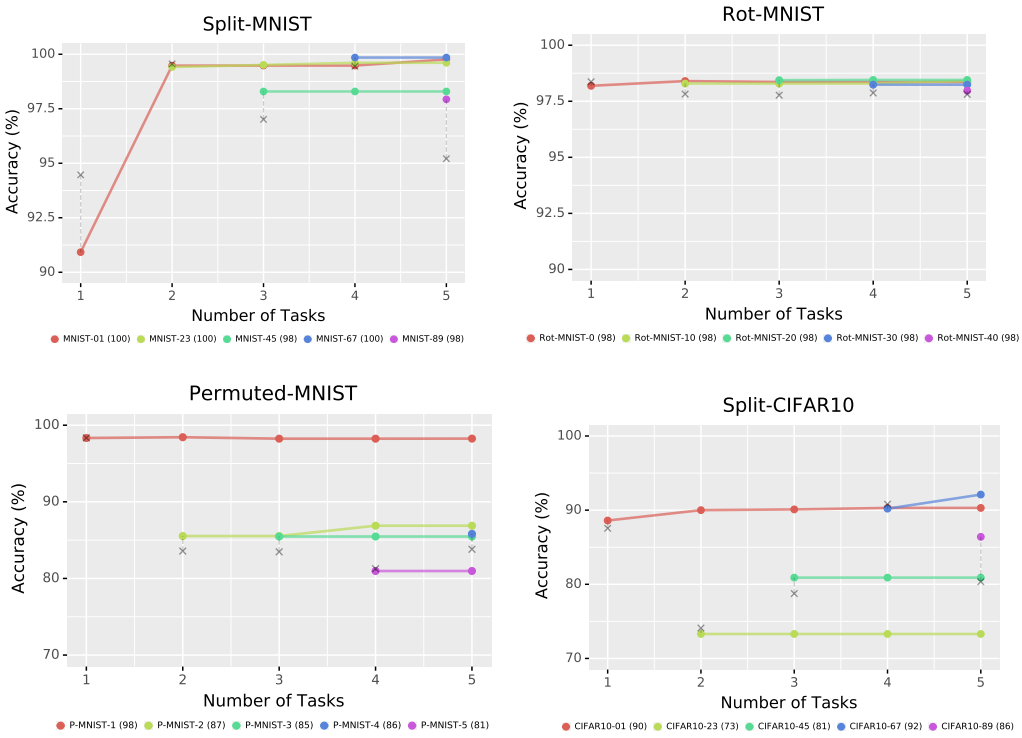

Figure A10: Per-task validation accuracy as a function of the number of episodes of continual learning for problems using variants of CIFAR10 and MNIST datasets using Model Zoo-continual. Each task has 100 samples/class. X-markers denote accuracy of Isolated on the new task. We see both forward transfer (Model Zoo often starts with a higher accuracy than Isolated) and backward transfer (accuracy of some past tasks improves in later episodes). For problems like Permuted-MNIST and Rotated-MNIST, there is little forward or backward transfer.

B.10 Model Zoo with Uniform sampling

At each round of boosting, Model Zoo samples tasks according to equation (8) i.e., tasks with high loss under the current ensemble have a higher probability of being selected in the next round. To study the importance of this heuristic, we compare Model Zoo to a variant called Model Zoo (uniform). Model Zoo (uniform) samples uniformly over all seen tasks for each round, as opposed to using equation (8).

Table A5 compares the accuracy of Model Zoo and Model Zoo (uniform) on the Coarse-CIFAR100 dataset. Model Zoo is marginally better than Model Zoo (uniform) indicating that using the training loss is a cheap proxy for splitting the capacity amongst related tasks. At the same time, this also indicates that a better measure of task-distances can further improve performance.

Method	Avg. Accuracy
Model Zoo	84.27
Model Zoo (uniform)	83.60

Table A5: Comparison of accuracies on the Coarse-CIFAR100 dataset

C Proofs

Proof of Theorem 2. From the definition of ρ_{ij} relatedness for tasks, we have

$$\begin{aligned} c\,\mathcal{E}_{P_i}^{1/\rho_{i1}}(h) &\geq \mathcal{E}_{P_1}(h, h_i^*) \\ &= \mathcal{E}_{P_1}(h) - \mathcal{E}_{P_1}(h_i^*, h_1^*). \end{aligned}$$

for any $i, j \leq n$ and $h \in H$. Let us denote $\rho_{(i)} = \rho_{i1}$. We can sum over $i \in \{1, \ldots, k\}$ and divide by k to get

$$\mathcal{E}_{P_1}(h) \leq \frac{1}{k}\sum_{i=1}^{k} \mathcal{E}_{P_1}(h_{(i)}^*) + \frac{c}{k}\sum_{i=1}^{k} \mathcal{E}_{P_{(i)}}^{1/\rho_{(i)}}(h).$$

The first term is a discrepancy term that measures how distinct different tasks are as measured by the probability of the disagreement of their individual hypotheses $h^*_{(i)}$ with that of h^*_1 under samples drawn from task P_1. We need to bound the second term on the right-hand side to prove Theorem 2. We have

$$\begin{aligned}\frac{1}{k}\sum_{i=1}^{k}\mathcal{E}^{1/\rho_{(i)}}_{P_{(i)}}(h) &\leq \frac{1}{k}\sum_{i=1}^{k}\mathcal{E}^{1/\rho_{\max}}_{P_{(i)}}(h)\\ &= \frac{1}{k}\sum_{i=1}^{k}\left(e_{P_i}(h) - e_{P_i}(h^*_i)\right)^{1/\rho_{\max}}\\ &\leq \frac{1}{k}\sum_{i=1}^{k}e^{1/\rho_{\max}}_{P_i}(h) \leq e^{1/\rho_{\max}}_{\bar{P}}(h).\end{aligned}$$

where the final step involves Jensen's inequality and $\bar{P} = 1/k\sum_{i=1}^{k}P_{(i)}$. This is the population risk of a hypothesis h on the mixture distribution $\bar{P}$ and by uniform convergence, we can bound it as

$$e^{1/\rho_{\max}}_{\bar{P}}(h) \leq \left(e_{\bar{S}}(h) + c'\left(\frac{D-\log\delta}{km}\right)^{1/2}\right)^{1/\rho_{\max}}$$

for any $h \in H$, in particular $\hat{h}^k$, with probability $1-\delta$. Putting it all together we have:

$$\begin{aligned}\mathcal{E}_{P_1}(h) &\leq \frac{1}{k}\sum_{i=1}^{k}\mathcal{E}_{P_1}(h^*_{(i)}) + \frac{c}{k}\sum_{i=1}^{k}\mathcal{E}^{1/\rho_{(i)}}_{P_{(i)}}(h)\\ &\leq \frac{1}{k}\sum_{i=1}^{k}\mathcal{E}_{P_1}(h^*_{(i)}) + \frac{c}{k}\left(e_{\bar{S}}(h) + c'\left(\frac{D-\log\delta}{km}\right)^{1/2}\right)^{1/\rho_{\max}}\end{aligned}$$

□

D Frequently asked questions (FAQs)

1. **Why do you consider the setting with unlimited replay?**
   As mentioned in §6, we would like to ground the practice of continual learning. Our investigation is inspired by the existing work on continual learning and with this paper we seek to encourage future works to focus their investigations on key desiderata of continual learning, namely per-task accuracy and forward-backward transfer.

   With this goal, we are motivated by our results in Theorem 2 that fitting a single model on a set of tasks is fundamentally limiting in performance due to competition between tasks, this problem is only exacerbated by introducing the tasks sequentially. We have developed a general method named Model Zoo that, although designed for unlimited replay, can be executed in any of the standard continual learning settings. Our experiments show that Model Zoo significantly outperforms existing methods in all of these settings, including problem settings with no replay.

   We allow Model Zoo to revisit past data and grow its capacity iteratively in order to get to the heart of the problem of learning multiple tasks sequentially. In our view, if we can demonstrate effective continual learning without forgetting at least in this setting, it will provide a good foundation to build methods that conform to the stricter problem formulations.

   We believe that such a foundation is needed today if we are to advance the practice of continual learning. Let us explain why with an example. The simplest "baseline" algorithm named Isolated in our work, surprisingly outperforms all existing continual learning methods, without performing any data replay, or leveraging data from multiple tasks. An upper bound for performance of a continual learner is the accuracy obtained by a multi-task learner that has access to all tasks before training. We argue that a good continual learner's performance should lie in between the above two: it should be—at least–comparable to training the task in isolation, and as close to the performance of the multi-task learner as possible. The fact that existing methods perform much poorly than even Isolated indicates that we need to thoroughly investigate the tradeoffs that these methods make, e.g., while the single epoch setting helps mitigate forgetting, it has quite poor accuracy.

In short, we would like to argue that before we design new sophisticated methods for continual learning, we should take a step back and evaluate what simple methods can do and ascertain some level of baseline performance, so that we have a sound benchmark to compare the sophisticated method against. This is our rationale for considering the problem setting with unlimited replay. **We would also like to emphasize that Model Zoo is a legitimate continual learner because it gets access to each task sequentially, and has a fixed computational budget at each episode.** For a multi-task learner, the computational complexity scales with the number of tasks.

2. **Why do you call it continual learning, instead of, say, incremental or lifelong learning?**
The current literature is quite inconclusive about the formal distinction between continual, incremental and lifelong learning. We have chosen to call our problem "continual learning" and, by that, we simply mean that the learner gets access to tasks sequentially instead of having access to all tasks before training begins.

3. **Why are you not using the same neural architectures as those in the existing literature? Perhaps the methods in this paper work better because you use a larger/different neural architecture.**
We use a small deep network (WRN-16-4 with 3.6M weights) for all our experiments. In particular, this is smaller than the Resnet-12 or Resnet-18 architectures that are used in a number of continual learning experiments (see Kaushik et al. (2021)) and the Model Zoo has a comparable number of weights. The exceptional performance of Model Zoo indicates that these observations indicate that the significant gains in accuracy of Model Zoo are not simply a result of using a larger model. We also demonstrate results on continual learning with a much smaller model, a CNN with 0.12M weights (which entails that Model Zoo has about 2.42M weights). This is an extremely small model, and even this model, under all problem settings, improves the accuracy of continual learning over existing methods.

4. **Why not compare Model Zoo to ensemble versions of other methods?**
We compare the performance of Model Zoo with ensemble versions of Isolated in Fig. 4. We observe that Model Zoo performs better than an ensemble of Isolated models. We did not compare against ensemble variants of existing continual learning methods because as our results show in multiple places, Isolated significantly outperforms the state of the art as a continual learner. We therefore expect that Model Zoo will also outperform ensembles of existing methods.

5. **Boosting is not novel.**
We do not claim any novelty in developing boosting and moreover our method is only loosely inspired by it. The key property of Model Zoo that makes it effective is the ability to split the capacity of the learner across different sets of tasks, the ones that are chosen at each round. This entails that the implementation of Model Zoo is similar to that of boosting-based algorithms such as AdaBoost, but that is the extent of the similarity between the two. In particular, Model Zoo only uses the models that were trained on a particular task in order to make predictions for it. Unlike AdaBoost which combines all the weak-learners using specific weights, we simply average the predictions of all models trained on each task. To emphasize, boosting is not novel, but the ability of Model Zoo to split learning capacity across multiple models, one from each round, trained on a set of tasks, *is* novel.

6. **Identifying that tasks compete is not novel.**
See §6 and the references in §2.1. The fact that tasks compete with each other is broadly appreciated–if not rigorously studied–in the theoretical machine learning literature. It is also appreciated broadly under the name of catastrophic forgetting in continual learning. Theorem 2 elucidates this competition and shows, together with Fig. 2, that it can be quite non-trivial. Even if some tasks compete, i.e., a hypothesis that is optimal for one performs poorly on the other, they may benefit each other if we have access to lots of samples from each task. An effective way to resolve this competition has been missing. Model Zoo is a simple and effective framework to tackle task competition; such a mechanism, and certainly its use for continual learning, is novel to our knowledge.

7. **Why does the rate of convergence in Theorem 2 depend upon $\rho_{\max}$, this seems quite inefficient.**
The convergence rate in Theorem 2 which depends on $\rho_{\max}$ indeed seems pessimistic if one chooses a bad set of tasks to train together. But this may be a fundamental limitation of non-adaptive methods, e.g., that pool data from all tasks together to compute $\hat{h}^k$. If the

learner uses adaptive methods, e.g., if it has access to ρ_{ij} and iteratively restricts the search space at iteration k to only consider hypotheses that achieve a low empirical risk $\hat{e}_{S_{(i)}}$ on all tasks closer than $\rho_{(k)}$, then as (Hanneke and Kpotufe, 2020) shows, we can get better convergence rates if all tasks have the same optimal hypothesis. Let us note that we have chosen some drastic inequalities in Appendix C in order to elucidate the main point, and it may be possible to improve upon the rate.

8. **Can you give some intuition for the transfer exponent?**
The transfer exponent discussed in (5) is inspired by the work of Hanneke and Kpotufe (2020) and is defined by the smallest value such that

$$c\,\mathcal{E}_{P_i}^{1/\rho_{ij}}(h) \geq \mathcal{E}_{P_j}(h, h_i^*) = \mathcal{E}_{P_j}(h) + e_{P_j}(h_j^*) - e_{P_j}(h_i^*)$$

for all $h \in H$. This should be understood as a measure of similarity between tasks that incorporates properties of the hypothesis space. A small value of $\rho_{ij} \approx 1$ suggests that minimizing the excess risk on task P_i (the left-hand side) is a good strategy if we want to minimize the excess risk on task P_j (the right-hand side). But there may be instances when we can only reduce the left hand-side up to an additive term

$$e_{P_j}(h_j^*) - e_{P_j}(h_i^*)$$

that may be non-zero (or large) if the optimal hypotheses h_j^* and h_i^* perform very differently on samples from P_j. Mathematically, ρ_{ij} is seen as the rate of convergence of the concentration term in Theorem 2 if samples from P_i are used to select a hypothesis for P_j; larger the transfer exponent, more inefficient these samples, even if this additive term is zero.