Overview and Contents

In this supplementary material, we include data that either (1) we processed to produce the plots in the paper or (2) that we were not able to fit in the main body of the paper. The contents of these appendices are as follows:

Appendix A. The process by which we chose the "extreme" sparsity levels used in Section 4.

Appendix B. Details about the states of the unpruned networks and IMP subnetworks at the rewinding iterations, including full network accuracy, L_2 distance from initialization, L_2 distance to the trained weights, and the L_2 distance between trained weights under different data orders.

Appendix C. Linear interpolation instability data *throughout* training for ResNet-20 and VGG-16; that is, interpolating between the states at each epoch of networks trained on different dataorders.

Appendix D. Linear interpolation instability and test error across rewinding iterations for ResNet-20 and VGG-16 at all levels of sparsity (not just the extreme sparsity we analyzed in Section 4.3). This data was used to create Figure 9.

Appendix E. The error when linearly interpolating for all networks in all configurations (unpruned and sparse) at all rewinding iterations. This data was used to create the instability plots in Figures 3 and 6.

Appendix F. The training set instability for the sparse networks corresponding to the test set instability data that we present in Section 4 Figure 6.

Appendix G. Functions other than linear mode connectivity for comparing the networks that result from our instability analysis experiments: L_2 distance, cosine distance, number of identical classifications, and L_2 distance of losses.

A. Selecting Extreme Sparsity Levels for IMP

In this appendix, we describe how we select the extreme sparsity level that we examine in Section 4.3 for each IMP subnetwork. For each network and hyperparameter configuration, our goal is to study the most extreme sparsity level at which matching subnetworks are known to exist early in training. To do so, we use IMP to generate subnetworks at many different sparsities for many different rewinding iterations. We then select the most extreme sparsity level at which any IMP under any rewinding iteration produces a matching subnetwork.

In Figure 10, each plot contains the maximum accuracy found by any rewinding iteration in red. The black line is the accuracy of the unpruned network to one standard deviation. For each network, we select the most extreme sparsity for which the red and black lines intersect. As a basis for comparison, these plots also include the result of performing IMP with $k = 0$ (blue line), random pruning (orange line), and random reinitialization of the IMP subnetworks with $k = 0$ (green line).

Note that, for computational reasons, ResNet-50 and Inception-v3 are pruned using one-shot pruning, meaning the networks are pruned to the target sparsity all at once. All other networks are pruned using iterative pruning, meaning the networks are pruned by 20% after each iteration of IMP until they reach the target sparsity. Pruning 20% per iteration was the practice adopted by Frankle & Carbin (2019). This information is specified in the rightmost Table 1.

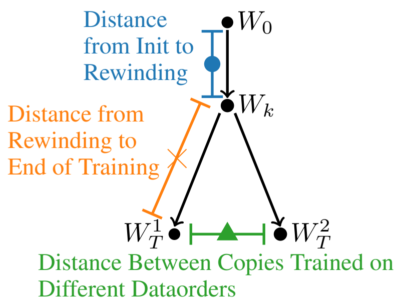
B. The State of the Network at Rewinding

B.1. Methodology

In the main body of the paper, we perform instability analysis by training to step k, making two copies of the network, optionally apply a pruning mask (as in Section 4), and training these two copies to completion under different samples of SGD noise. We find that, for a sufficiently large value of k, the trained networks will find the same, linearly connected minimum. In this appendix, we address the following question: what is the state of the network at the step k from which this linear connectivity results? Are the networks so far along in training that they are virtually fully optimized? Have they traveled the vast majority of the distance from initialization to the eventual minimum? In this sense, is the iteration at which the network becomes stable "trivial?" We address these questions in two ways.

Error at rewinding. In Figure 11, we present the error of the unpruned network at each rewinding iteration we consider in the main body of the paper. With this data, we investigate how close the network has come to its full accuracy when it becomes stable.

L_2 **distances.** In Figures 12 and 13, we measure various L_2 distances that capture how close the network is to initialization and to the end of training. In particular, we measure three distances as shown in the diagram below (which is an annotated version of Figure 1).


Distance from Init to Rewinding
W_0
W_k
Distance from Rewinding to End of Training
W_T^1
W_T^2
Distance Between Copies Trained on Different Dataorders

First, we measure the L_2 distance in parameter space from initialization to the state of the network at step k (blue circle in the diagram above and in Figures 12 and 13); for the sparse IMP subnetworks, we measure the L_2 distance after applying the pruning mask to both initialization and the state of the network at iteration k. This quantity captures the distance that the network has traversed from initialization by step k.

Second, we measure the distance from the state of the network at step k to its state at the end of training under one data order (orange x in the diagram above and in Figures 12 and 13). This quantity captures the distance that the network traverses after step k. If the network is very close to the minimum by the time it becomes stable, then we expect this quantity to be small compared to the L_2 distance between initialization and iteration k; that would indicate that the network has already traversed a large distance and has a relatively smaller distance to go.

Finally, we measure the distance between the final states of networks trained from step k under different data orders (green triangle in the diagram above and in Figures 12 and 13). This quantity captures the size of the linearly connected minimum found by the networks. We are interested in how this distance compares to the distance traveled by the networks and how this quantity changes as the rewinding iteration varies.

B.2. Results

Error at rewinding. These results appear in Figure 11. Recall that the unpruned networks become stable at a different (typically later) iteration than the IMP subnetworks, so we consider two rewinding points for each network.

Unpruned networks. ResNet-20 and VGG-16 become stable to SGD noise at iterations 2000 and 1000, at which point test error is about 25% (compared to final error 8.3%) for ResNet-20 and 20% (compared to final error 6.3%) for VGG-16. Train error is at a similar value to test error at these points; in both cases, train error eventually converges to 0%. We conclude that, at the iteration at which they become stable to SGD noise, these networks have not fully converged but are much closer to their final errors than to random guessing.

We see similar behavior for the unpruned ResNet-50 and Inception-v3 networks, which become stable to SGD noise at epochs 18 and 28. At these points, test error is 55% (compared to final error 24%) for ResNet-50 and 33% (compared to final error 22%) for Inception-v3. Both networks are most of the way to their final performance.

IMP pruned subnetworks. The IMP pruned subnetworks become stable to SGD noise earlier than the unpruned networks. ResNet-20 and VGG-16 become stable to SGD noise at iterations 500 and 1000, at which point error is 30% (compared to final error 8.3%) for ResNet-20 and 35% (compared to final error 6.3%) for VGG-16. These networks have not fully converged but are closer to their final errors than to random guessing. IMP subnetworks of ResNet-50 and Inception-v3 become stable to SGD noise much earlier than the unpruned networks—at epoch 5 and epoch 6, respectively. At these points, error is much higher—55% for ResNet-50 and 40% for Inception-v3—leaving these networks substantial room to further train. We did not evaluate the train accuracy at these checkpoints for the ImageNet networks due to storage and computational limitations.

L_2 distances. These results appear in Figures 12 and 13.

Unpruned networks. ResNet-20 and VGG-16 become stable to SGD noise at iterations 2000 and 1000, at which point they are closer to their initial weights than to their final weights. This indicates that they still have a substantial distance to travel on the optimization landscape and are still far from their final weights. This result is particularly remarkable considering our observation in Appendix C that stable networks follow the same, linearly connected trajectory throughout training (according to test error); the L_2 distance data suggests that they do so for a substantial distance.

The unpruned ResNet-50 and Inception-v3 networks are closer to their final weights than their initial weights when they become stable to SGD noise. In fact, it appears that distance from initialization begins to plateau and distance to the final weights only decreases slowly. This may indicate that the networks will make much slower progress for the remaining 80% of training iterations.

The green triangles in these plots show the distance between the weights of copies of the network trained from a rewinding iteration to completion on different data orders. In all cases, the distance between these copies is substantial, even after the networks become stable. As a point of comparison, we use the distance that the networks travel between initialization and the final weights, which is captured by the orange x for rewinding iteration 0. For ResNet-20, the distance between copies trained on different data orders from iteration 2000 (when it becomes stable) is more than half the distance that the network travels during the entirety of training. The same is true for VGG-16 from iteration 1000 (when it becomes stable). For ResNet-50 and Inception-v3, this distance is about a quarter and half (respectively) of the distance the networks travel over the course of training. These are remarkably large distances considering that any network on this line segment reaches full test accuracy.

IMP pruned subnetworks. We show the same data for the IMP subnetworks in Figure 13. Each L_2 distance in this figure is measured after applying the pruning mask to all weights. When ResNet-20 and VGG-16 become stable to SGD noise (iterations 2000 and 1000, respectively), they are about 2x (ResNet-20) and 3x (VGG-16) closer to their initial weights than their final weights. ResNet-50 and Inception-v3 are about equal distances from both points for the epochs at which they become stable.

Unique to the IMP subnetworks, we observe here and in Appendix G that the L_2 distance between copies trained on different data orders drops alongside instability, plateauing at a lower value when training from the rewinding iteration at which the subnetworks becomes stable. Even this lower distance is still a substantial fraction of the overall distance the network travels: 25%, 45%, 27%, and 28% for ResNet-20, VGG-16, ResNet-50, and Inception-v3.

C. Instability Throughout Training

In Section 3, we find stable networks that arrive at minima that are linearly connected. In this appendix, we study whether the trajectories they follow are also linearly connected. In other words, when training two copies of the same network with different noise, are the states of the network at each step t connected by a linear path over which test error does not increase? In the main body of the paper, we study this quantity only at the end of training (i.e., $t = T$). Here, we study it for all iterations t *throughout* training. To study this behavior, we linearly interpolate between the networks at each epoch of training and compute instability.

Figure 14 plots instability throughout training for ResNet-20 and VGG-16 from different rewinding iterations k for both train and test error for the unpruned networks and the IMP subnetworks. We begin with the unpruned networks. For $k = 0$ (blue line), instability increases rapidly. In fact, it follows the same pattern as error: as the train or test error of each network decreases, the maximum possible instability increases (since instability never exceeds random guessing). With larger values of k, instability increases more slowly throughout training. When k is sufficiently large that the networks are stable at the end of training, they are generally stable at every epoch of training ($k = 2000$, pink line). In other words, after iteration 2000, the networks follow identical optimization trajectories modulo linear interpolation.

The IMP subnetworks of ResNet-20 exhibit the same behavior as the unpruned network: when the network is stable at the end of training, it is stable throughout training, meaning two copies of the same network follow the same optimization trajectory up to linear interpolation. The IMP subnetworks of VGG-16 exhibit sightly different behavior at rewinding iterations 500 and 1000: instability initially spikes (meaning the networks rapidly become separated by a loss barrier) but decreases gradually thereafter. For rewinding iteration 1000, it decreases to 0, meaning the networks are stable by the end of training. For all other rewinding iterations, being stable at the end of training corresponds to being stable throughout training, so it is possible that rewinding iteration 1000 represents a transition point between the unstable rewinding iterations earlier and the stable rewinding iterations later.

D. Instability Data at All Sparsities

In Figure 6 in Section 4.3, we show the effect of rewinding iteration on instability and test error for sparse subnetworks. We specifically focus on the most extreme level of sparsity for which IMP at any rewinding iteration is matching (as selected in Appendix A). In this appendix, we present the relationship between rewinding iteration and instability/test error for all levels of sparsity for standard ResNet-20 (Figures 15 and 16) and VGG-16 (Figures 17 and 18) on CIFAR-10. Section 4.4 and Figure 9 summarize this data, so we defer analysis of this data to that section.

This data begins with 80% of weights remaining and includes sparsities attained by repeatedly pruning 20% of weights (e.g., 64% of weights remaining, 51% of weights remaining, etc.). We include these levels in particular because we use IMP to prune 20% of weights per iteration, meaning we have sparse IMP subnetworks for each of these levels. We include data for every sparsity level displayed in Appendix A, including those beyond the extreme sparsities we study in Section 4.3.

We only collected this data for standard ResNet-20 and VGG-16 on CIFAR-10. We determined that it was more valuable to spend our limited computational resources on these networks (whose instability and accuracy are sensitive to rewinding at the extreme sparsity level) than for the *low* and *warmup* variants (which are consistently stable and matching at the extreme sparsity level). We did not have the computational resources to compute this data on the ImageNet networks for all sparsities.

E. Full Linear Interpolation Data

In Figures 3 and 6, we plot the instability value derived from linearly interpolating between copies of the same network or subnetwork trained on different data orders. In this appendix, we plot the linear interpolation data from which we derived the instabilities in Figures 3 and 6. We plot this data for the unpruned networks (Figure 19), IMP subnetworks (Figure 20), randomly pruned subnetworks (Figure 21), and the randomly reinitialized IMP subnetworks (Figure 22).

F. Train Instability for Sparse Subnetworks

In Section 4, we only measure instability and error on the test set. We make this choice for simplicity after observing in Section 3 that train and test instability closely align. In this appendix, we present the data from Section 4 on the test set. Figures 23 and 24 examine the instability and error of the same IMP subnetworks as Figure 6, but it shows both the train and test sets. We did not compute the train set quantities for Inception-v3 due to computational limitations.

Train set and test set instability are nearly identical, just as we found in Section 3. Interestingly, the two coincide more closely for IMP subnetworks of ResNet-50 than they do for the unpruned networks in Section 3.

For networks that are unstable at rewinding iteration 0, train error and test error follow similar trends, starting higher when the subnetworks are unstable and dropping when the subnetworks become stable. In other words, the unstable IMP subnetworks are not able to fully optimize to 0% train error, while the stable IMP subnetworks are.

G. Alternate Distance Functions

Instability analysis involves training two copies of the same network on different data orders and comparing the networks that result. In the main body of the paper, our method of comparison is linear interpolation, which we find to offer valuable new insights into neural network optimization and the lottery ticket hypothesis. However, one could parameterize instability analysis with a wide range of other functions for comparing pairs of neural networks. In this appendix, we discuss four alternate methods for which we collected data using the MNIST and CIFAR-10 networks.

L_2 Distance. One simple way to compare neural networks is to measure the L_2 distance between the trained weights. The limitation of this function is that there is not necessarily any relationship between L_2 distance and the functional similarity of networks or the structure of the loss landscape. In other words, there is no clear interpretation of L_2 distance.

In Figure 25, we plot the L_2 distance function at all rewinding points for the unpruned networks. In Figure 26, we plot the L_2 distance function at all rewinding points for all three classes of sparse networks. We plot this data separately because L_2 distance is not necessarily comparable between sparse networks (which have fewer parameters) and dense networks (which have more parameters).

For the unpruned networks, distance decreases linearly as we logarithmically increase the rewinding iteration. We see no distinct changes in behavior when the networks become stable, and the L_2 distance remains far from 0 at this point.

For the IMP subnetworks, L_2 distance mirrors the behavior of instability. In cases where the IMP subnetworks are stable at all rewinding points (ResNet-20 low/warmup, VGG-16 low/warmup, and LeNet), the L_2 distance is at a lower level than the L_2 distance between the other baselines (random pruning and random reinitialization) and is consistent across rewinding points. In cases where the IMP subnetworks are unstable at initialization but become stable later (ResNet-20 and VGG-16), the L_2 distance begins high (at the same level as the L_2 distance for the randomly pruned and randomly reinitialized baselines) and drops when the subnetworks become stable, settling at a lower level.

Although stable IMP subnetworks are closer in L_2 distance than unstable IMP subnetworks and the baselines, the L_2 distance remains far from zero. In general, it is difficult to translate the results of this function into higher-level statements about the relationships between the networks.

Cosine distance. In Figures 27 (unpruned networks) and 28 (sparse networks), we plot the cosine distance in a manner similar to L_2 distance. The results are similar to those for L_2 distance, and the same interpretation applies.

Classification differences. This distance function computes the number of examples that are classified differently by two networks. Unlike linear interpolation and L_2/cosine distance, this function looks at the functional behavior of the networks rather than the parameterizations. This function is particularly valuable because it allows us to compare the dense and sparse networks directly.

In Figures 29 (test set) and 30 (train set), we plot this function for the unpruned and sparse networks across rewinding iterations. The unpruned networks generally classify the same number of examples differently no matter the rewinding iteration, although the number of different classifications decreases gradually for the latest rewinding iterations for ResNet-20 low and warmup. We see no relationship between this function and instability.

The behavior of the IMP sparse networks better matches instability. IMP subnetworks that are stable from initialization (ResNet-20 low and warmup, VGG-16 low and warmup, LeNet) consistently have the same distance no matter the rewinding iteration. This distance is lower than that for the randomly pruned and randomly reinitialized baselines.

IMP subnetworks that are unstable at iteration 0 (ResNet-20 and VGG-16) have the same number of different classifications as the baselines when rewinding to iteration 0. When the networks become stable, the number of different classifications drops substantially to a lower level.

One challenge with using this distance function is that it is inherently entangled with accuracy. As the accuracy of the networks improves, the number of different classifications might decrease simply because the networks will classify more examples correctly (and thereby, the same way). Consider the IMP subnetworks of ResNet-20 on the CIFAR-10 test set (the graph in the upper right of Figure 29, blue line). At rewinding iteration 0, the networks have about 11% error on the test set, meaning there are at most 2200 examples they could classify differently.[3] In Figure 29, we see that the networks are classifying about 1100 examples differently.

When ResNet-20 IMP subnetworks are stable, error decreases to 8.5%, meaning at most 1700 examples can be classified differently. However, in Figure 27, we see that only about 350 examples are being classified differently. Although this number is lower than the 1100 differences at rewinding iteration 0 in absolute terms, accuracy has improved as well, so we must consider these differences in context. At rewinding iteration 0, classification differences are 50% of their maximum possible value, while at rewinding iteration 1000, classification differences are at 21% of their maximum possible value. In summary, as the IMP subnetworks become stable, they behave in a more functionally similar fashion, even considering accuracy improvements.

Loss L_2 distance. This distance function computes the L_2 distance between the vector of cross-entropy losses aggregated by computing the loss on each example. This function again considers only the functional behavior of the networks, but it uses the per-example loss rather than the classification decisions, which may provide more information about the functional behavior of the networks. We plot this data in Figures 31 (test set) and 32 (train set). It largely mirrors the behavior from the classification difference function, and the same interpretations apply.

[3] In the worst case, all examples that one network misclassifies will be classified correctly by the other. Since each network misclassifies 1100 examples, 2200 examples will be classified differently in total.

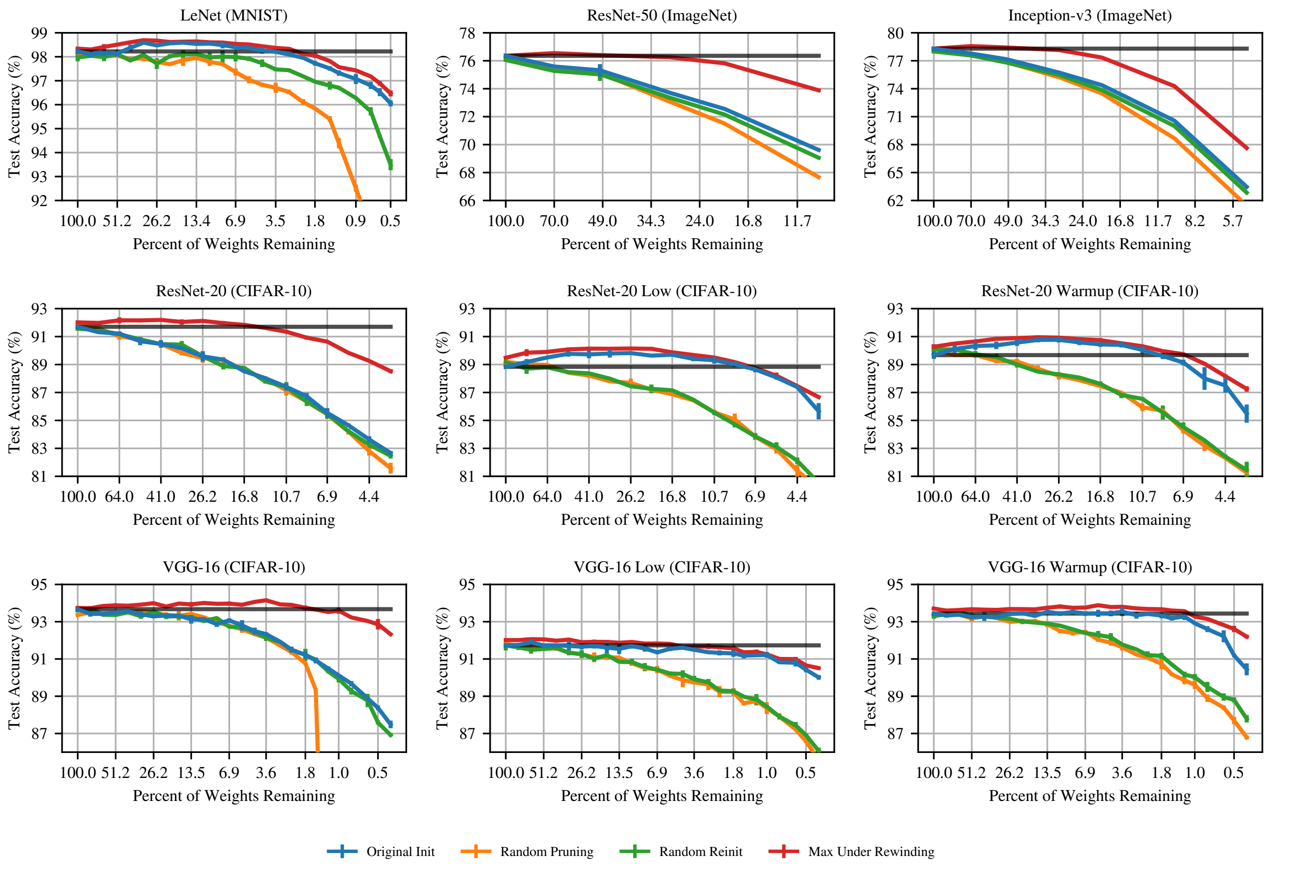


Figure 10. An illustration of the methodology by which we select the extreme sparsity levels that we study in Section 4. The red line is the maximum accuracy achieved by any IMP subnetwork under any rewinding iteration. The black line is the accuracy of the full network. We use the most extreme sparsity level for which the red and black lines overlap. Each line is the mean and standard deviation across three runs with different initializations.

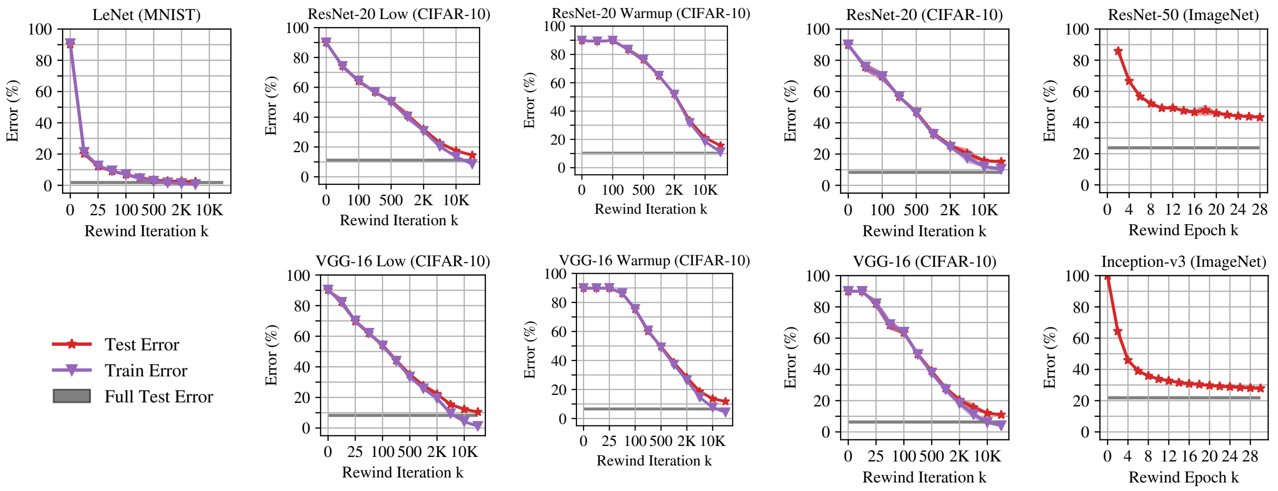


Figure 11. The error of the full networks at the rewinding iteration specified on the x-axis. For clarity, this is the error of the network at that specific iteration of training, before any copies are made or further training occurs. Each line is the mean and standard deviation across three initializations.

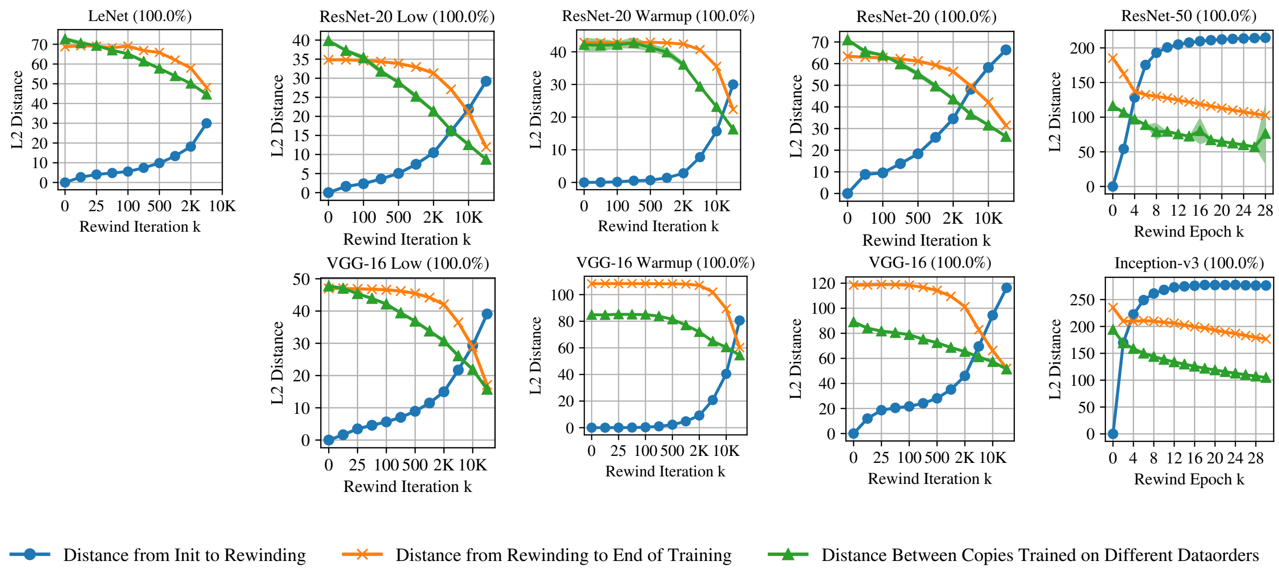


Figure 12. Various L_2 distances for the full networks at the rewinding iteration specified on the x-axis. Each line is the mean and standard deviation across three initializations.

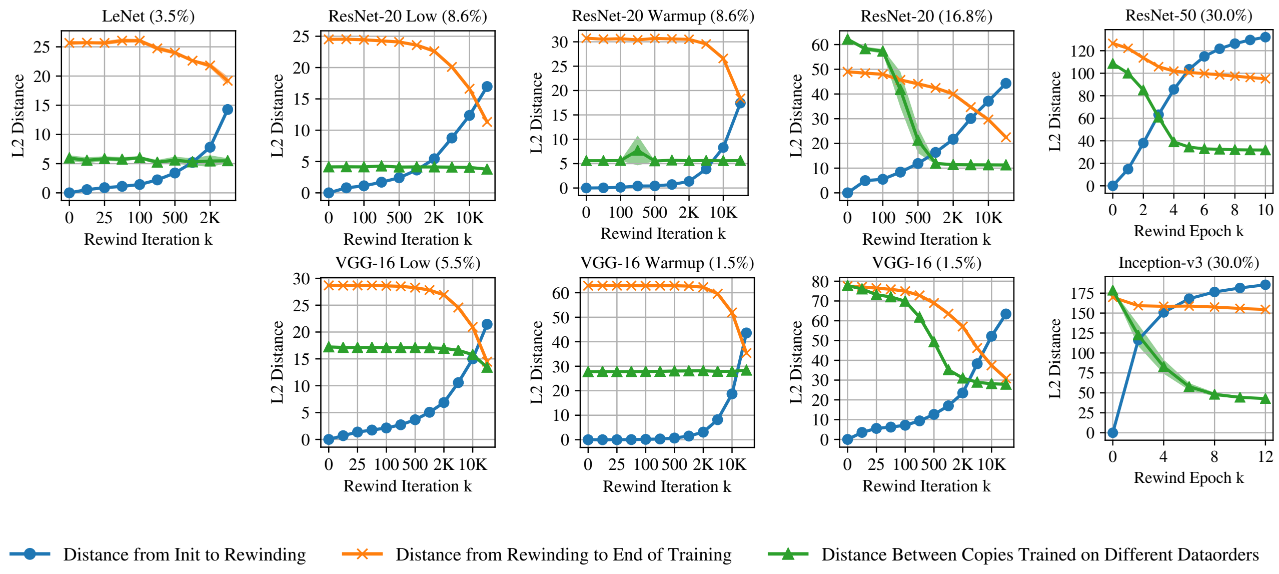


Figure 13. Various L_2 distances for the IMP subnetworks at the rewinding iteration specified on the x-axis. Each line is the mean and standard deviation across three initializations. Each L_2 distance is computed after applying the pruning mask to the states of the networks in question.

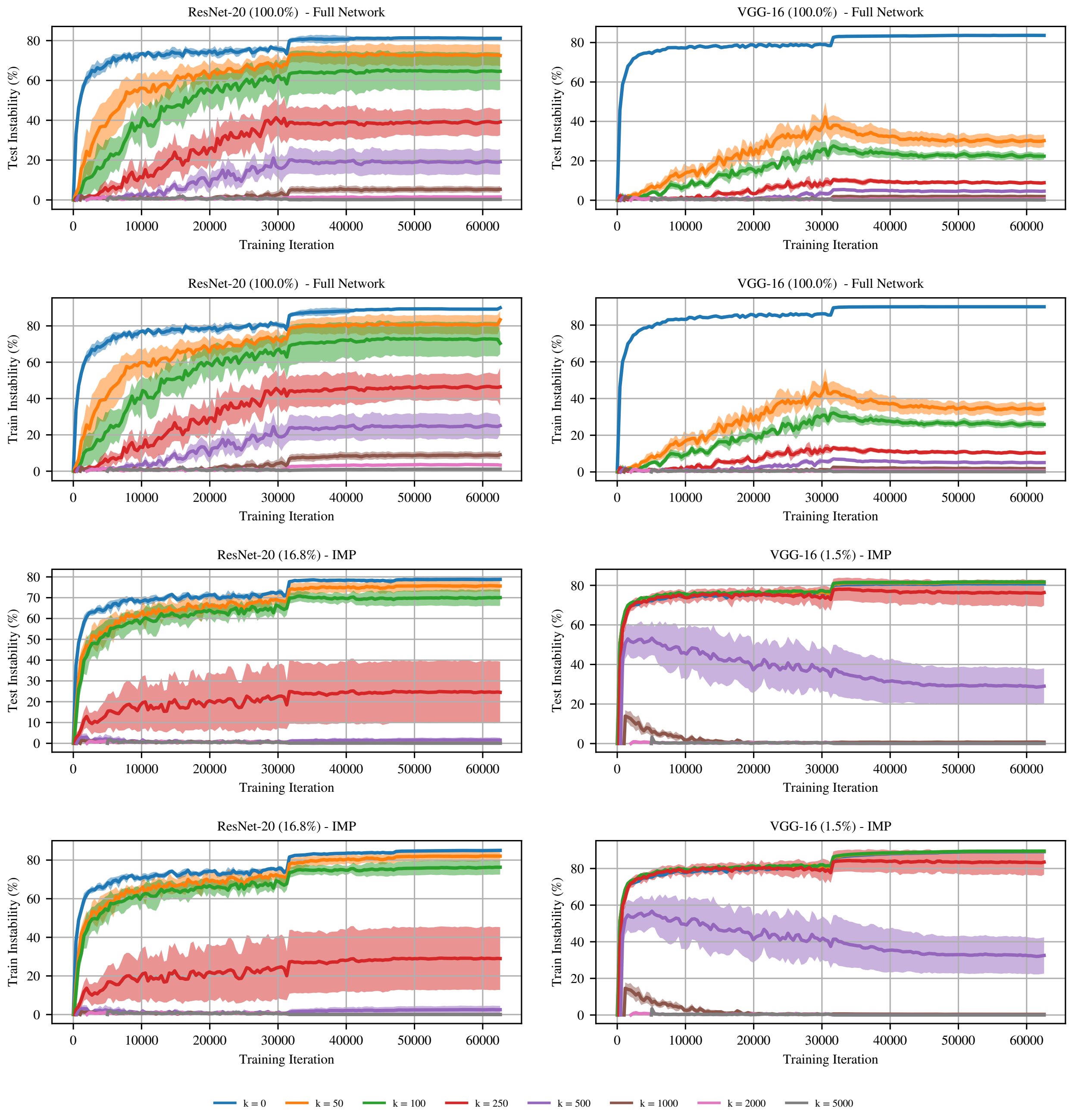


Figure 14. Instability throughout training for ResNet-20 and VGG-16 using both the unpruned networks and the IMP-pruned networks as computed on both the test set and train set. Each line involves training to iteration k and then training two copies on different data orders after. Each point is the instability when interpolating between the states of the networks at the training iteration on the x-axis.

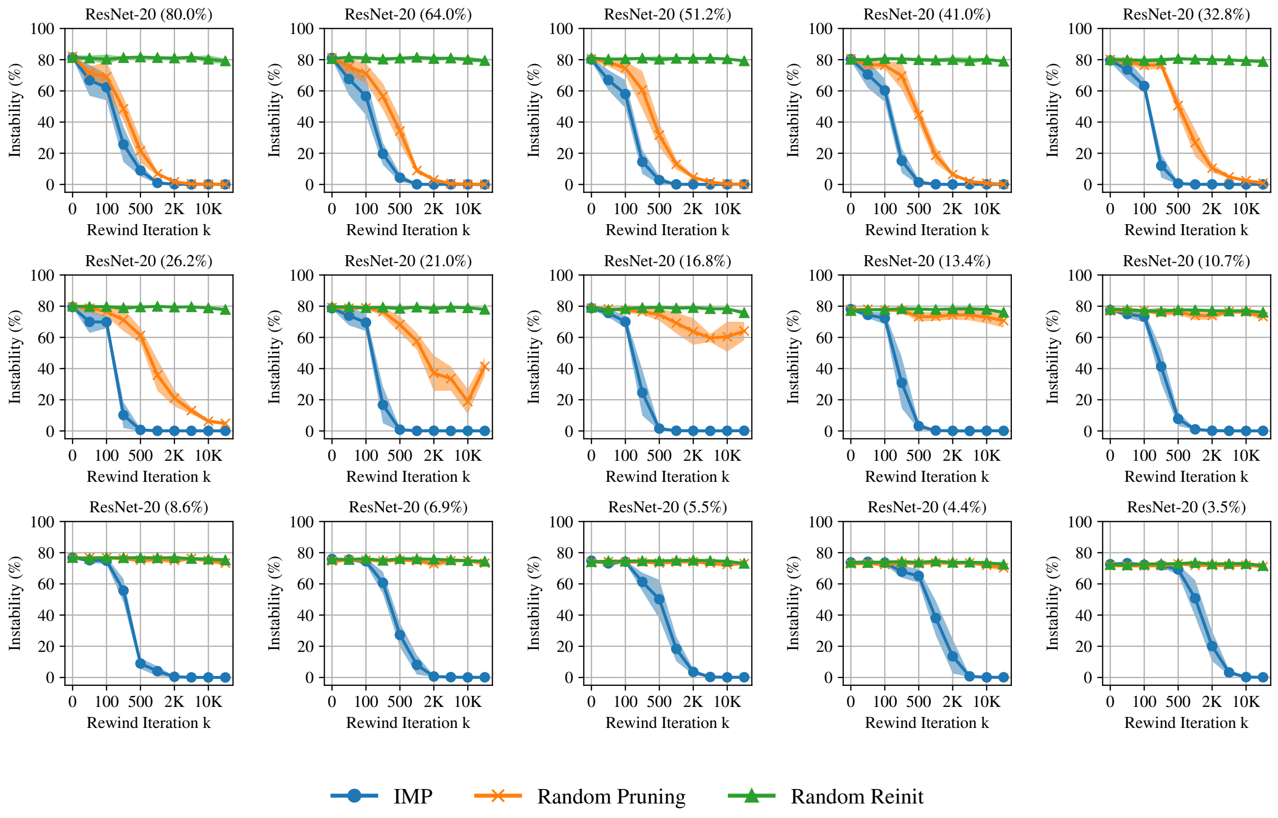


Figure 15. The instability of subnetworks of ResNet-20 created using the state of the full network at iteration k and trained on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples total). Percents are percents of weights remaining.

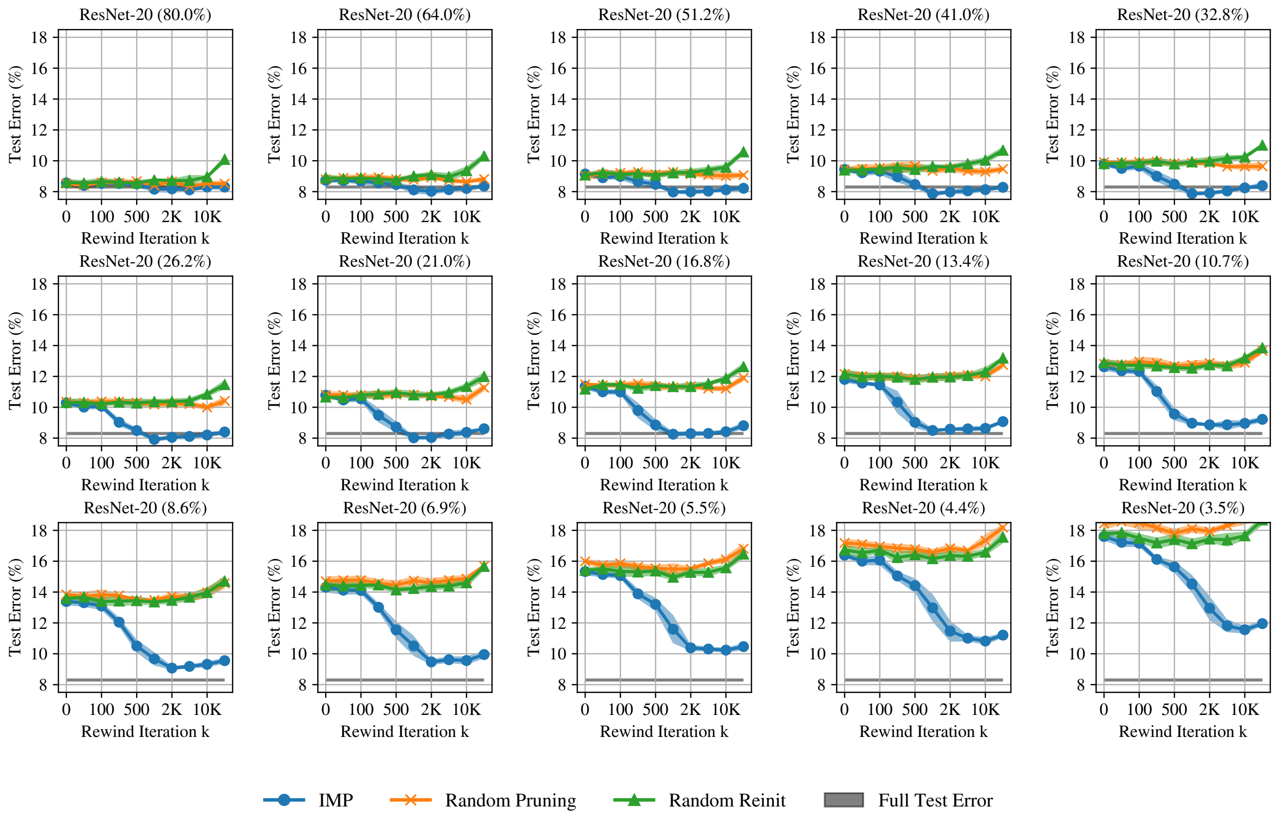


Figure 16. The test error of subnetworks of ResNet-20 created using the state of the full network at iteration k and trained on different data orders from there. Each line is the mean and standard deviation across three initializations. Gray lines are the accuracies of the full networks to one standard deviation. Percents are percents of weights remaining.

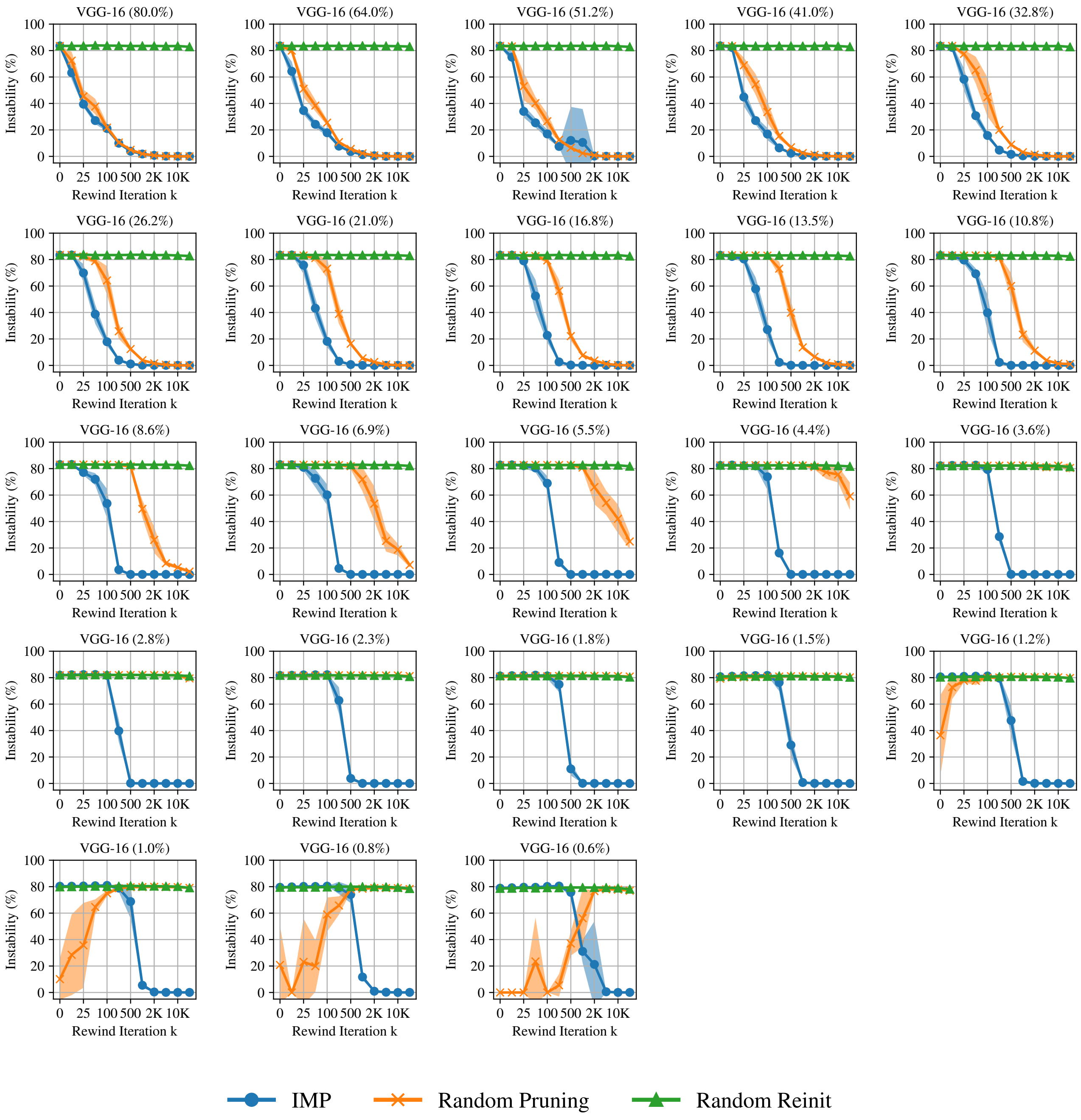


Figure 17. The instability of subnetworks of VGG-16 created using the state of the full network at iteration k and trained on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples total) Percents are percents of weights remaining.

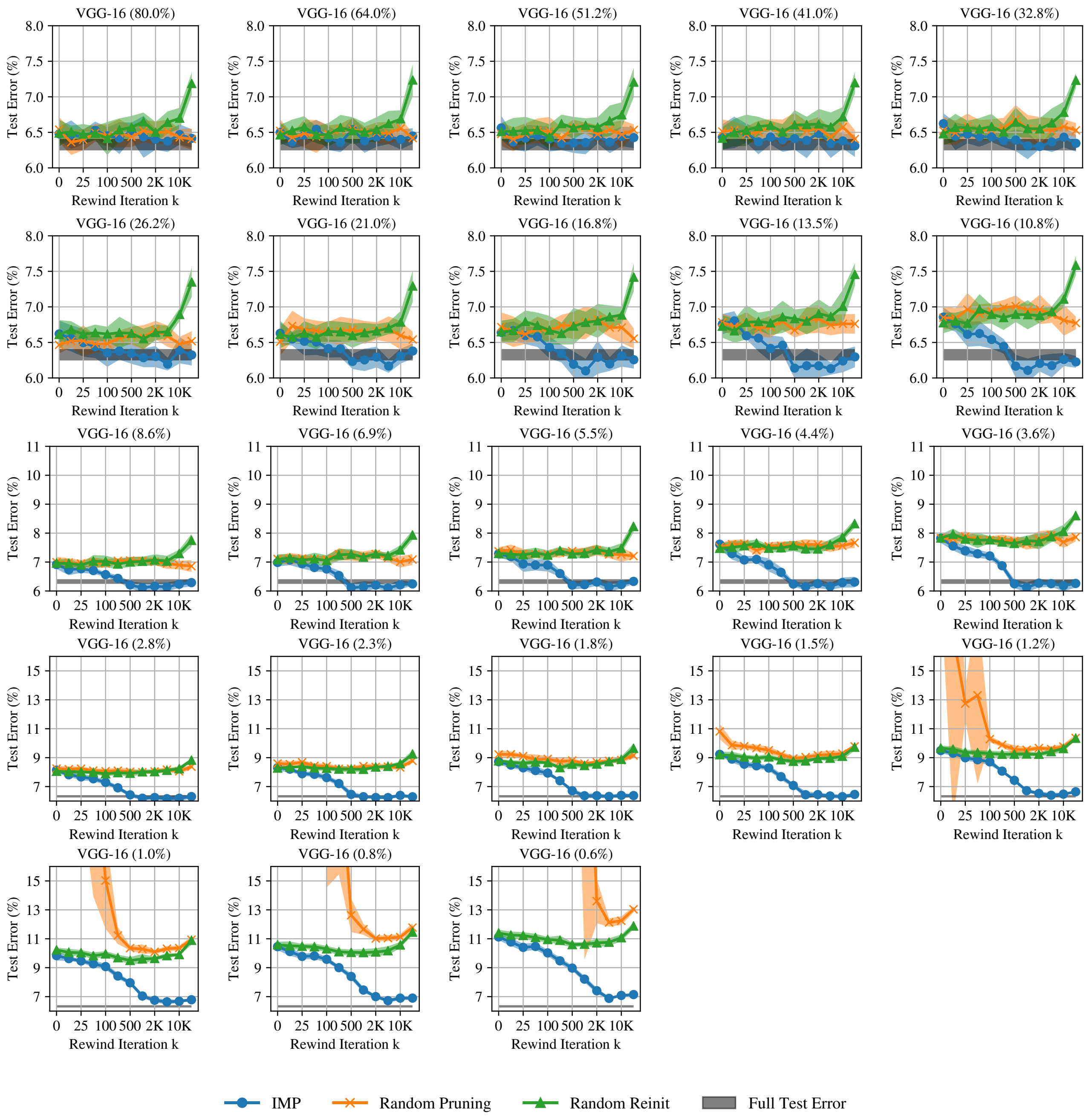


Figure 18. The test error of subnetworks of VGG-16 created using the state of the full network at iteration k and trained on different data orders from there. Each line is the mean and standard deviation across three initializations. Gray lines are the accuracies of the full networks to one standard deviation. Percents are percents of weights remaining.

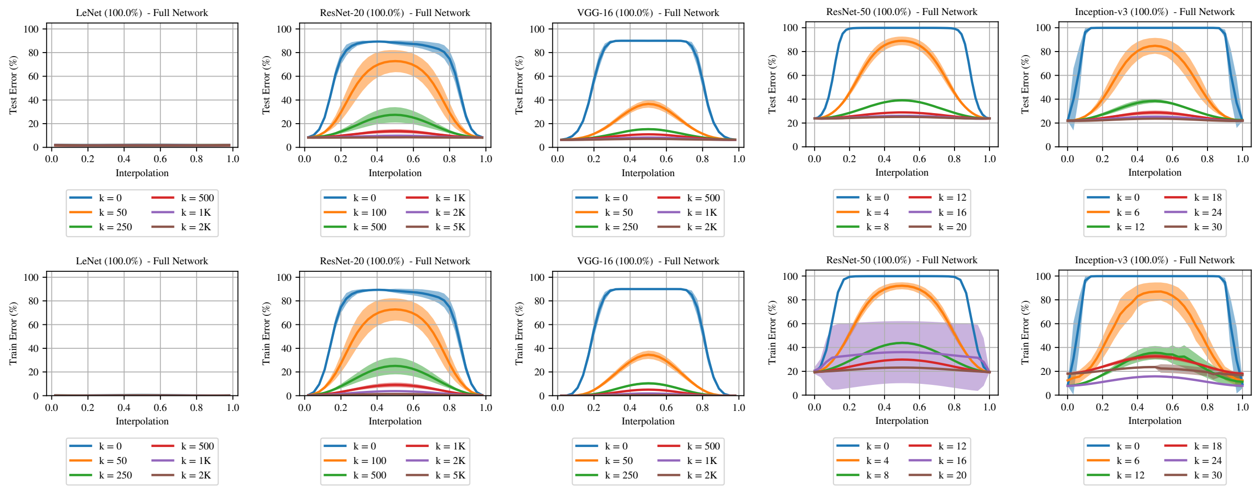


Figure 19. The error when linearly interpolating between the minima found by randomly initializing a network, training to iteration k, and training two copies from there to completion using different data orders. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). The errors of the trained networks are at interpolation = 0.0 and 1.0.

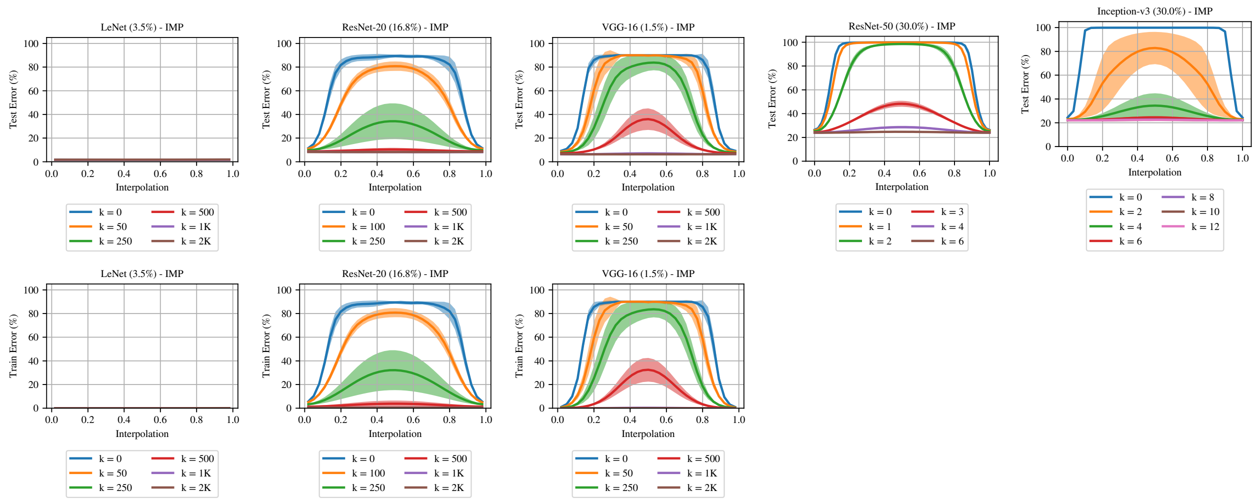


Figure 20. The error when linearly interpolating between the minima found by randomly initializing a network, training to iteration k, pruning according to IMP, and training two copies from there to completion using different data orders. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). The errors of the trained networks are at interpolation = 0.0 and 1.0. We did not interpolate using the training set for the ImageNet networks due to computational limitations.

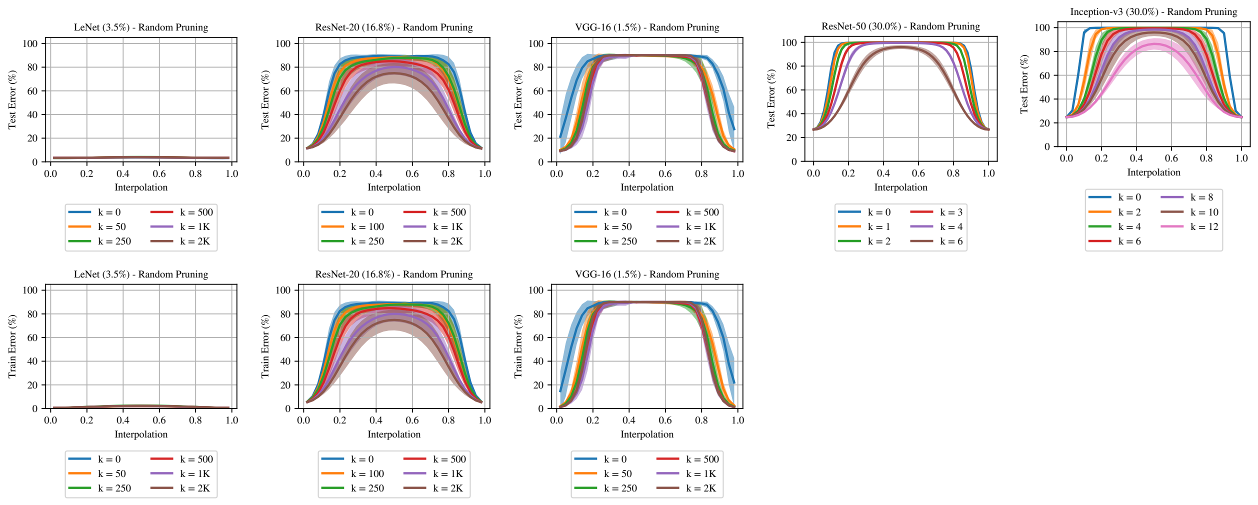


Figure 21. The error when linearly interpolating between the minima found by randomly initializing a network, training to iteration k, pruning randomly in the same layerwise proportions as IMP, and training two copies from there to completion using different data orders. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). The errors of the trained networks are at interpolation = 0.0 and 1.0. We did not interpolate using the training set for the ImageNet networks due to computational limitations.

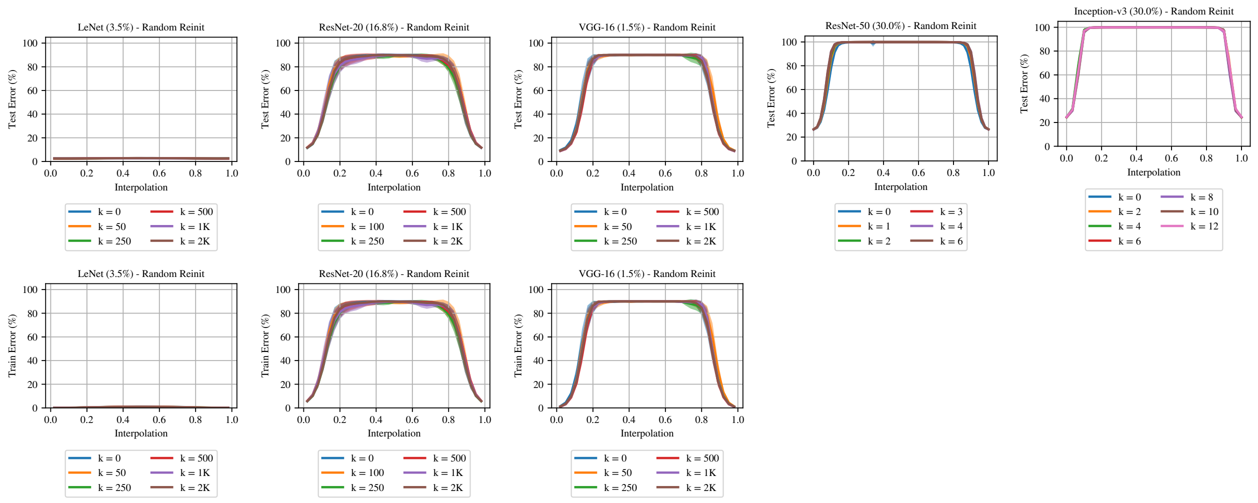


Figure 22. The error when linearly interpolating between the minima found by randomly initializing a network, training to iteration k, pruning according to IMP, randomly reinitializing, and training two copies from there to completion using different data orders. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). The errors of the trained networks are at interpolation = 0.0 and 1.0. We did not interpolate using the training set for the ImageNet networks due to computational limitations.

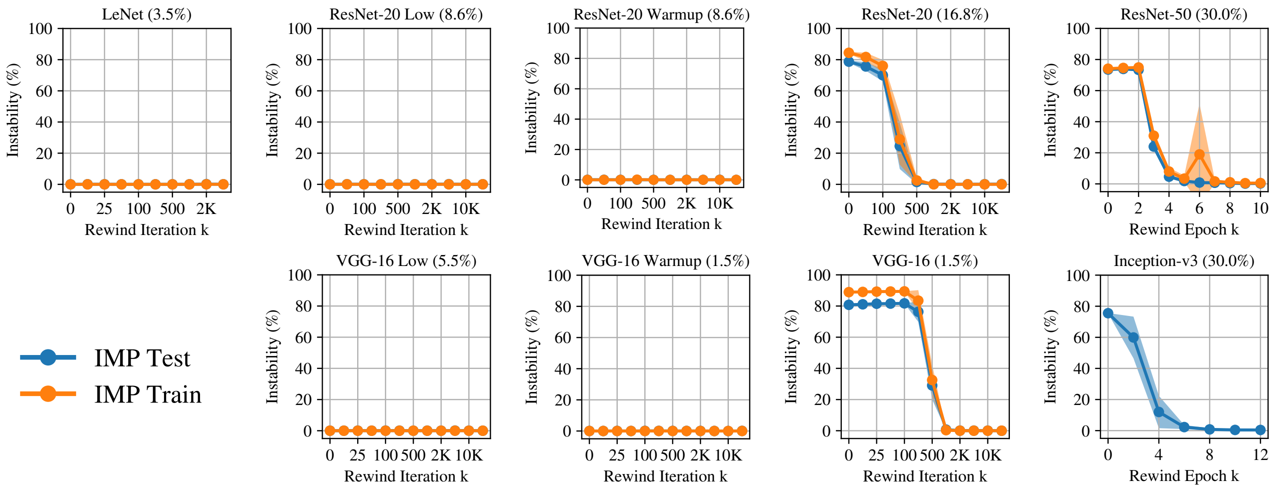


Figure 23. The train and test set instability of subnetworks that are created by using the state of the full network at iteration k, applying the pruning mask found by performing IMP with rewinding to iteration k, and training on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). Percents are percents of weights remaining. We did not compute the train set quantities for Inception-v3 due to computational limitations.

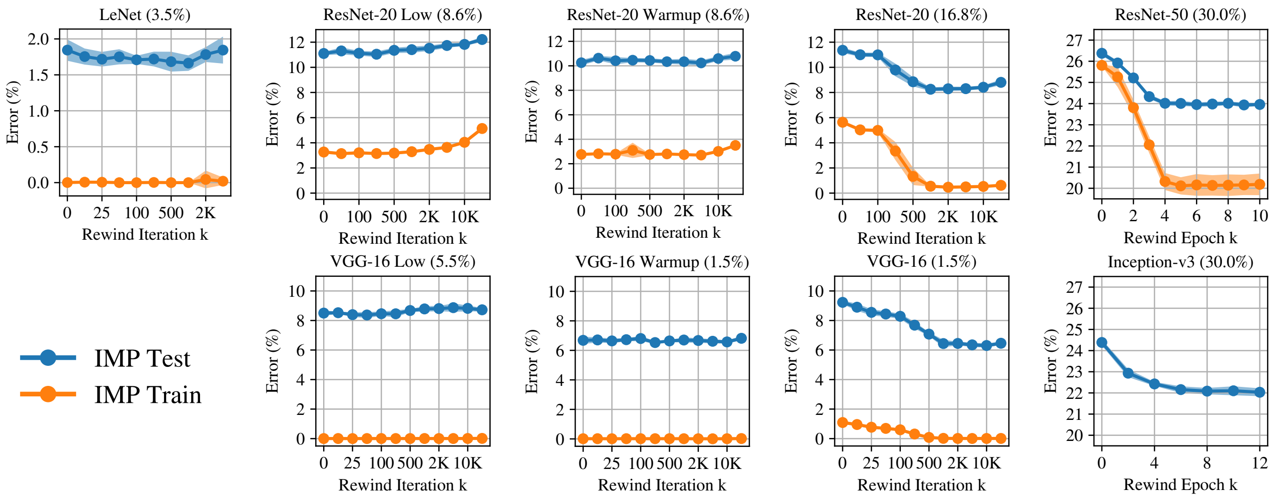


Figure 24. The train and test set error of subnetworks that are created by using the state of the full network at iteration k, applying the pruning mask found by performing IMP with rewinding to iteration k, and training on different data orders from there. Each line is the mean and standard deviation across three initializations. Percents are percents of weights remaining. We did not compute the train set quantities for Inception-v3 due to computational limitations.

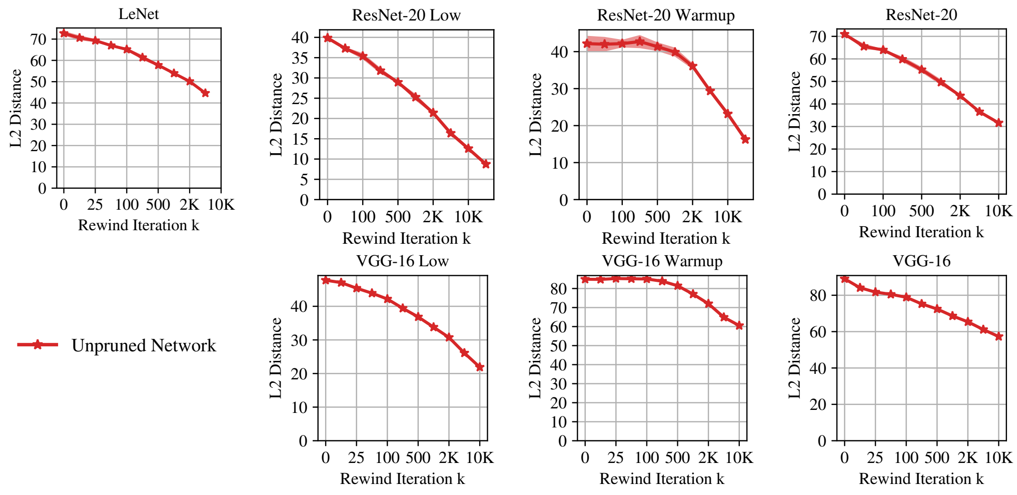


Figure 25. The L_2 distance between networks that are created by trained to iteration k, making two copies, and training on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). Percents are percents of weights remaining. We did not compute the train set quantities for the ImageNet networks due to computational limitations.

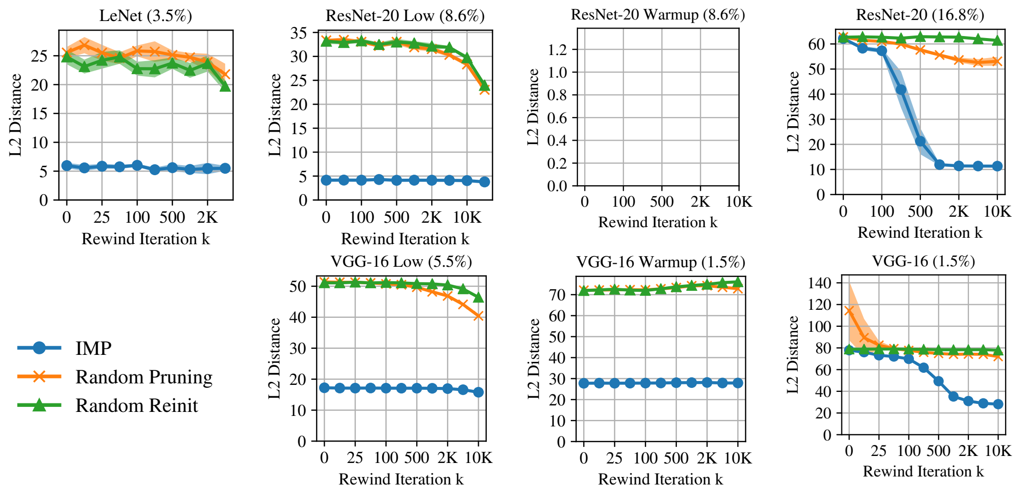


Figure 26. The L_2 distance between subnetworks that are created by using the state of the full network at iteration k, applying a pruning mask, and training on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). Percents are percents of weights remaining. We did not compute the train set quantities for the ImageNet networks due to computational limitations.

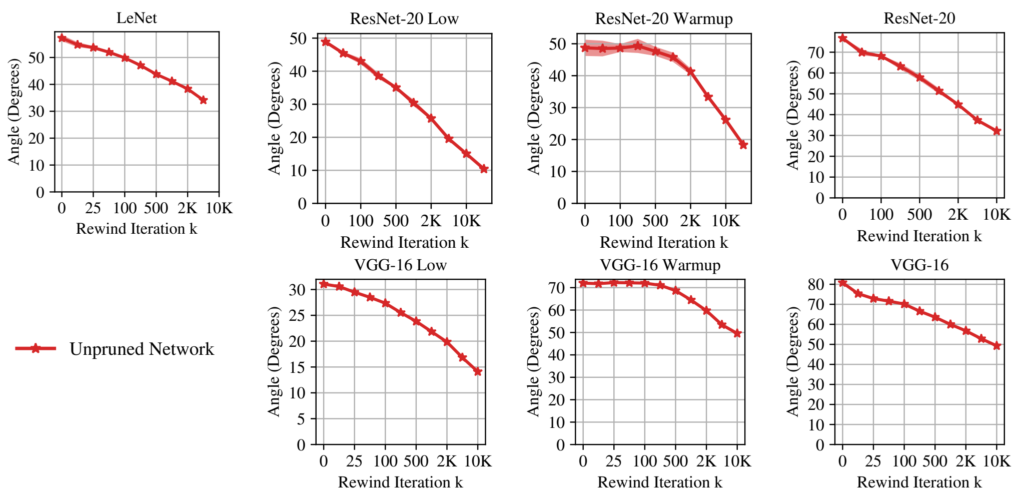


Figure 27. The cosine distance between networks that are created by trained to iteration k, making two copies, and training on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). Percents are percents of weights remaining. We did not compute the train set quantities for the ImageNet networks due to computational limitations.

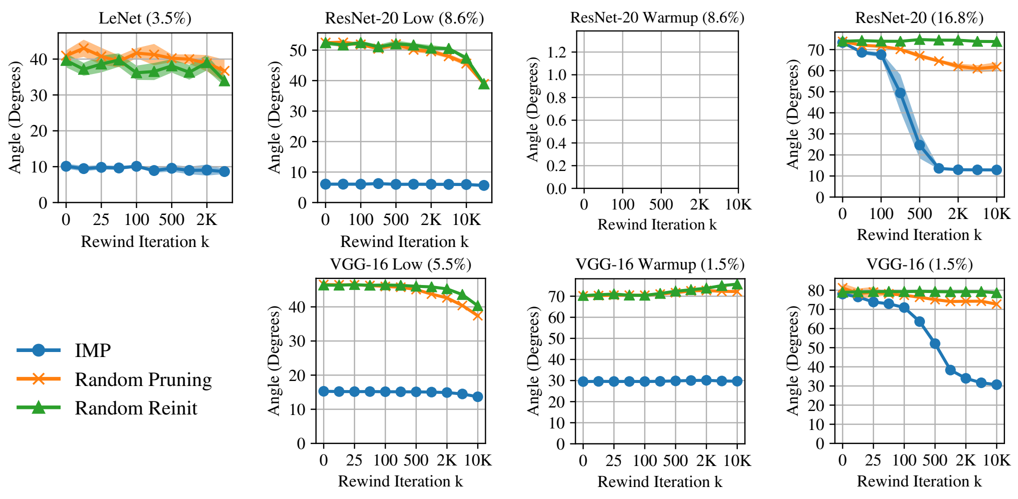


Figure 28. The cosine distance between subnetworks that are created by using the state of the full network at iteration k, applying a pruning mask, and training on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). Percents are percents of weights remaining. We did not compute the train set quantities for the ImageNet networks due to computational limitations.

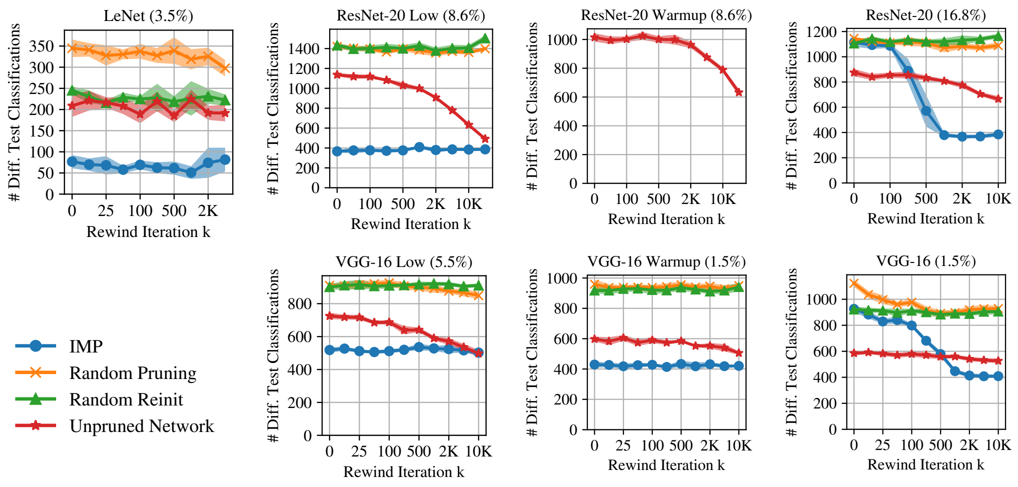


Figure 29. The number of different test set classifications between networks that are created by training the full network to iteration k, optionally applying a pruning mask, and training on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). Percents are percents of weights remaining.

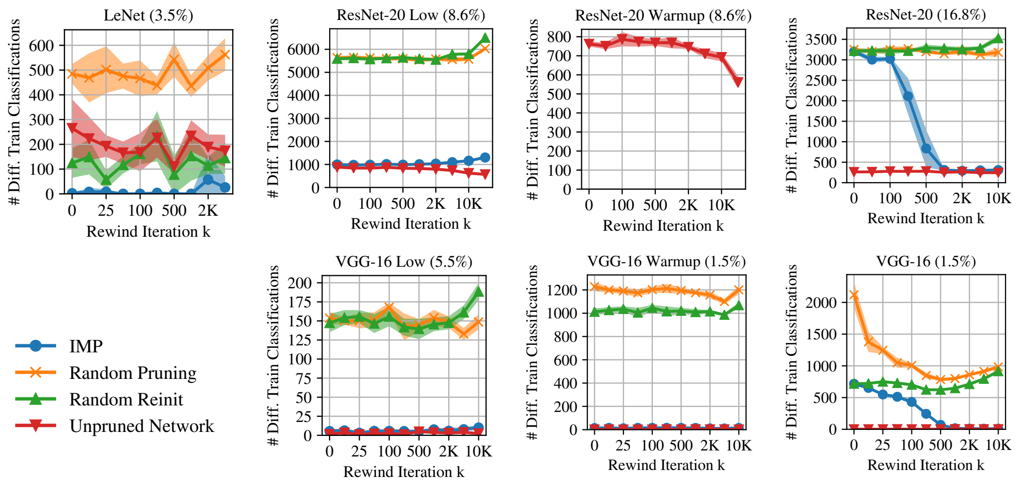


Figure 30. The number of different train set classifications between networks that are created by training the full network to iteration k, optionally applying a pruning mask, and training on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). Percents are percents of weights remaining.

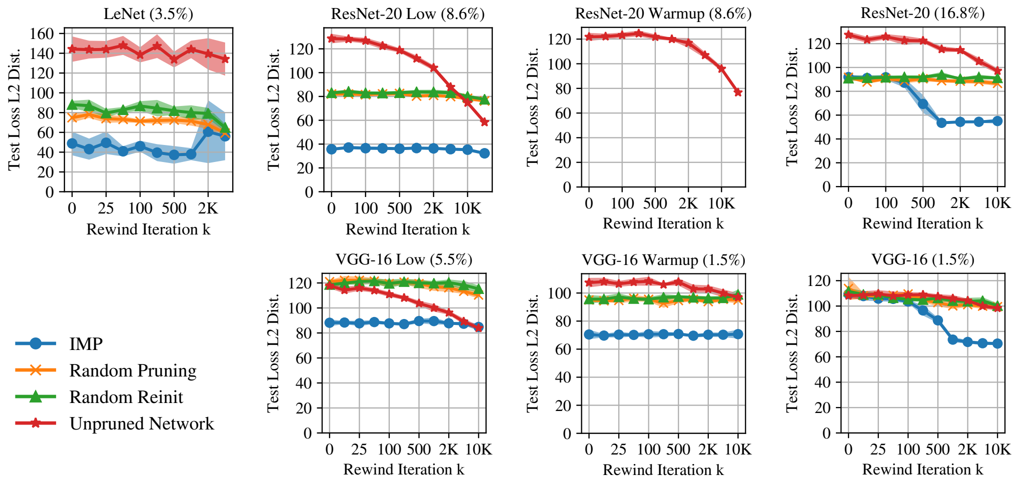


Figure 31. The L_2 distance between the per-example losses on the test set for networks that are created by training the full network to iteration k, optionally applying a pruning mask, and training on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). Percents are percents of weights remaining. We did not compute the train set quantities for the ImageNet networks due to computational limitations.

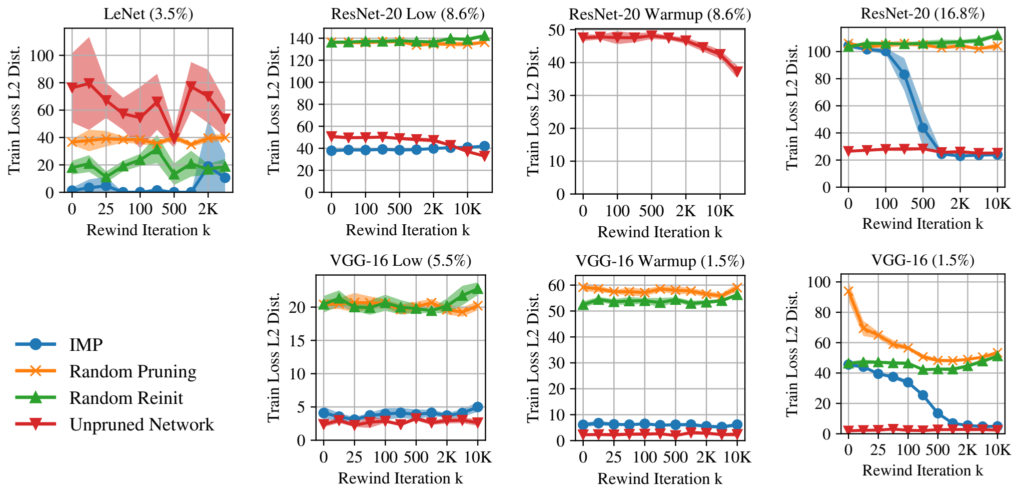


Figure 32. The L_2 distance between the per-example losses on the train set for networks that are created by training the full network to iteration k, optionally applying a pruning mask, and training on different data orders from there. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total). Percents are percents of weights remaining. We did not compute the train set quantities for the ImageNet networks due to computational limitations.