Low-loss connection of weight vectors: distribution-based approaches. Supplementary Materials

A. Experiments with different architectures

We made a few additional experiments to see how the considered methods perform on different architectures. In particular, we observe how the considered methods perform while we vary 1) the width of One Hidden layer network (Table 1) and 2) the depth of a dense network (Table 2). All experiments were done on CIFAR10.

In Table 1 we present experiments with underparameterized as well as overparameterized One Hidden layer networks. For any number of parameters, we observe approximately the same pattern of dependence of connection quality on the connection method. Performance of most connection methods degrades at smaller numbers of parameters, but this is to be expected from the general logic of the distributional approach.

in Table 2 we consider networks with 3, 5 or 7 layers. We use 6144 neurons in the first hidden layer, 2000 neurons in the second hidden layers, and 1000 neurons in each of the remaining layers. Perhaps the most interesting observation that one can make here is that increasing depth from 3 to 5 improves performance of almost all connection methods.

To connect minima with Garipov's curves we use the original implementation of their numerical algorithm (`https://github.com/timgaripov/dnn-mode-connectivity.git`).

B. Ensembling with Weight Adjustment

In Table 3 we compare WA ensemble methods against ensembles of independently trained networks. WA(n) in the table refers to Weight Adjusment procedure that is performed on the n'th layer counting from the last layer of neural network (e.g. WA(1) is an ensemble with the last layer adjusted). We can see from the table that the amount of diversity in the ensemble is crucial for the performance: the more diversity (i.e., the higher n), the more accurate the output is. However, it comes with a cost of additional computations on inference and required storage. Also, note that the method WA(1) slightly improves the results over one model, and it comes without the costs listed above.

These results were obtained for VGG16 (Simonyan & Zisserman, 2014) and PreResNet110 (He et al., 2016) trained with SGD for 400 epochs, with learning rate 0.01 and batch size 128. We use standard data augmentation as in (Huang et al., 2017). We train VGG16 without batch normalization.

We use the following implementations of VGG16 and PreResNet110.

- VGG16: `https://github.com/pytorch/vision/blob/master/torchvision/models/vgg.py`
- PreResNet110: `https://github.com/bearpaw/pytorch-classification/blob/master/models/cifar/preresnet.py`

C. Error rate dynamics along the path

In Fig. 1 we show how test error changes along the paths proposed by WA-based connection methods for VGG16 on CIFAR10 dataset. The observed oscillations are associated with the 15 intermediate layer-by-layer transitions.

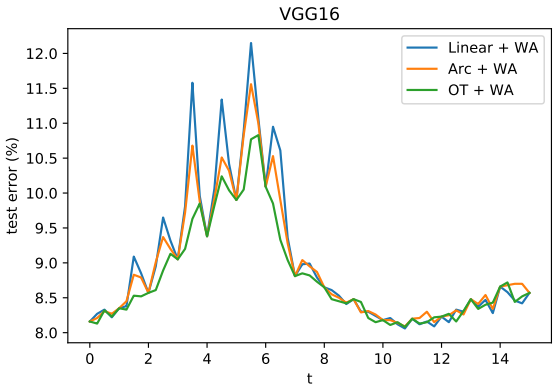

Figure 1: Test error of WA-based methods along the path on CIFAR10.

In Table 2 of the main text we see that some connection methods fail to connect two minima of VGG16 network. Namely, Linear, Arc, Linear + Butterfly and Arc + Butterfly has accuracy equals to random guess or even lower. OT + Butterfly method performs better, but has a high variance – we are currently investigating this issue. Note that neither of these methods uses Weight Adjustment procedure to improve the results. In Fig. 2 we show examples of paths with

Table 1: Test accuracy (%) of different methods for One Hidden layer networks with different width on CIFAR10.

	Width			
Methods	100	500	1000	2000
Linear	33.20 ± 2.03	35.39 ± 1.42	36.35 ± 1.68	39.34 ± 1.52
Arc	35.82 ± 1.64	36.73 ± 1.44	38.07 ± 1.41	41.34 ± 1.39
Linear + Weight Adjustment	45.89 ± 0.54	53.56 ± 0.33	55.55 ± 0.19	57.66 ± 0.26
Arc + Weight Adjustment	46.13 ± 0.46	53.84 ± 0.32	55.82 ± 0.19	57.88 ± 0.24
OT	53.73 ± 0.41	56.86 ± 0.40	56.18 ± 0.18	56.49 ± 0.46
OT + Weight Adjustment	55.10 ± 0.35	59.04 ± 0.17	58.95 ± 0.19	58.96 ± 0.21
Garipov (3)	53.94 ± 0.35	58.47 ± 0.21	58.99 ± 0.16	58.74 ± 0.23
Garipov (5)	53.81 ± 0.35	57.59 ± 0.27	57.89 ± 0.25	57.88 ± 0.32
End Points	56.47 ± 0.26	59.51 ± 0.32	59.14 ± 0.23	59.12 ± 0.26

Table 2: Test accuracy (%) of different methods for Dense networks with different depths on CIFAR10.

	Depth		
Methods	3	5	7
Linear	27.19 ± 1.12	30.70 ± 2.21	25.43 ± 2.04
Arc	40.17 ± 0.84	37.92 ± 1.84	35.94 ± 2.77
Linear + Butterfly	38.38 ± 0.84	50.66 ± 0.83	47.23 ± 1.10
Arc + Butterfly	49.63 ± 0.86	52.44 ± 4.42	47.47 ± 3.31
Linear + Weight Adjustment	51.87 ± 0.24	59.62 ± 0.13	58.12 ± 0.16
Arc + Weight Adjustment	58.86 ± 0.29	61.03 ± 0.17	60.15 ± 0.14
OT + Butterfly	59.11 ± 0.46	60.78 ± 0.39	59.89 ± 0.44
OT + Weight Adjustment	61.67 ± 0.49	61.29 ± 0.21	60.35 ± 0.24
Garipov(3)	61.38 ± 0.36	60.42 ± 0.19	58.95 ± 0.18
Garipov(5)	60.75 ± 0.32	59.51 ± 0.21	58.02 ± 0.27
End Points	63.25 ± 0.36	61.72 ± 0.21	61.02 ± 0.24

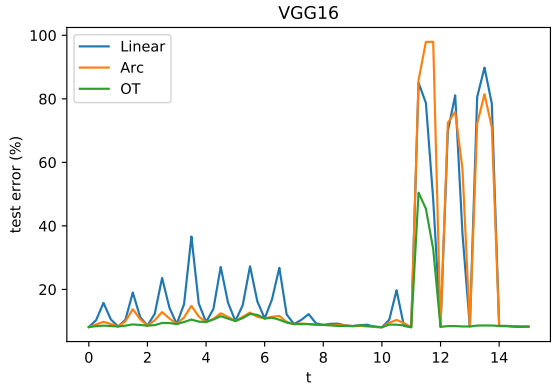

Figure 2: Test error of Butterfly methods along the connecting path (on CIFAR10).

failed Butterfly connections. As we can see, all Butterfly methods have low connection errors up to the 12th layer, after which the errors increase drastically.

Finally, we illustrate the variance of the OT + Butterfly method. In Fig. 3 we show accuracy on six different paths and observe that the main variance happens again on the 12'th layer.

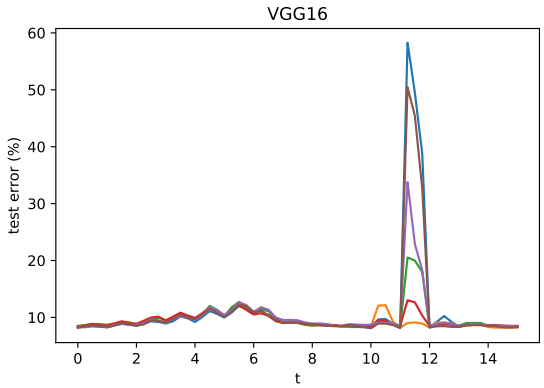

Figure 3: Test error of the method OT along six different paths (on CIFAR10).

Table 3: Test accuracy (%) of ensemble methods with respect to number of models in ensemble and architectures on CIFAR10.

		Number of models in ensemble			
Architecture	method	1	3	5	7
Dense 3	$WA(1)$	63.12	64.22	64.53	64.53
	$WA(2)$	63.12	65.35	66.27	66.69
	Ind	63.12	65.67	66.6	67.04
Dense 5	$WA(1)$	61.71	62.44	62.51	62.68
	$WA(2)$	61.71	62.48	62.77	62.82
	$WA(3)$	61.71	62.99	63.42	63.64
	$WA(4)$	61.71	63.1	63.71	64.19
	Ind	61.71	63.07	63.8	64.33
Dense 7	$WA(1)$	60.81	61.13	61.23	61.2
	$WA(2)$	60.81	61.54	61.74	61.89
	$WA(3)$	60.81	62.25	62.53	62.64
	$WA(4)$	60.81	63.	63.35	63.6
	$WA(5)$	60.81	62.97	63.46	63.63
	Ind	60.81	63.35	63.78	64.01
VGG16	$WA(1)$	91.52	91.54	91.58	91.59
	$WA(2)$	91.52	91.64	91.62	91.61
	Ind	91.52	92.88	93.12	93.4
PreResNet110	$WA(1)$	92.49	92.46	92.53	–
	Ind	92.49	93.81	94.08	–

REFERENCES

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

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

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