A REGULARIZATION THEORY

Theorem A.1. *Given a deep neural network $\mathcal{A}$ which consists of only convolution and linear layers. Let the network use one of $f(x) = \min\{x, 0\}$ (relu) or $f(x) = x$ (linear) as the activation function. Let the network be trained using the adjoined loss function as defined in Eqn. 3. Let $\mathbf{X}$ be the set of parameters of the network $\mathcal{A}$ which is shared across both the smaller and bigger networks. Let $\mathbf{Y}$ be the set of parameters of the bigger network not shared with the smaller network. Let $\mathbf{p}$ be the output of the larger network and let $\mathbf{q}$ be the output of the smaller network where $\mathbf{p}_i$, $\mathbf{q}_i$ represents their $\mathbf{i}^{th}$ component. Then, the adjoined loss function induces a data-dependent regularizer with the following properties.*

- *For all $x \in X$, the induced L_2 penalty is given by $\sum_i \mathbf{p}_i \big(\log' \mathbf{p}_i - \log' \mathbf{q}_i \big)^2$*
- *For all $y \in Y$, the induced L_2 penalty is given by $\sum_i \mathbf{p}_i \big(\log' \mathbf{p}_i \big)^2$*

Proof. We are interested in analyzing the regularizing behavior of the following loss function. $-y \log p + KL(p, q)$ y is the ground truth label, p is the output probability vector of the bigger network and q is the output probability vector of the smaller network. Recall that the parameters of smaller network are shared across both. We will look at the second order taylor expansion for the kl-divergence term. This will give us insights into regularization behavior of the loss function.

Let x be a parameter which is common across both the networks and y be a parameter in the bigger network but not the smaller one.

$D(x) = \sum_i p_i(x) \big(\log p_i(x) - \log q_i(x) \big)$ and $D(y) = \sum_i p_i(y) \big(\log p_i(y) - \log q_i \big)$

For the parameter y, q_i is a constant. Now, computing the first order derivative, we get that

$$D'(x) = \sum_i p_i'(x) \big(\log p_i(x) - \log q_i(x) \big) + p_i'(x) - \frac{q_i'(x) p_i(x)}{q_i(x)}$$

$$D'(y) = \sum_i p_i'(y) \big(\log p_i(y) - \log q_i \big) + p_i'(y)$$

Now, computing the second derivative for both the types of parameters, we get that

$$D''(x) = \sum_i p_i''(x) \big(\log p_i(x) - \log q_i(x) \big) + p_i'(x) \left(\frac{p_i'(x)}{p_i(x)} - \frac{q_i'(x)}{q_i(x)} \right) + p_i''(x)$$

$$- \frac{q_i(x) q_i'(x) p_i'(x) + q_i(x) q_i''(x) p_i(x) - q_i'(x) q_i'(x) p_i(x)}{q_i^2(x)}$$

$$D''(y) = \sum_i p_i''(y) \big(\log p_i(y) - \log q_i \big) + \frac{p_i'(y) p_i'(y)}{p_i(y)} + p_i''(y)$$

$$D''(x) = \sum_i \frac{p_i'(x) p_i'(x)}{p_i(x)} - \frac{2 p_i'(x) q_i'(x)}{q_i(x)} + \frac{q_i'(x) q_i'(x) p_i(x)}{q_i^2(x)}$$

$$= \sum_i p_i(x) \left(\frac{p_i'(x)}{p_i(x)} - \frac{q_i'(x)}{q_i(x)} \right)^2 = \sum_i p_i (\log' p_i - \log' q_i)^2 \tag{7}$$

Similarly, for the parameters only in the bigger network, we get that

$$D''(y) = \sum_i \frac{p_i'(y) p_i'(y)}{p_i(y)} = \sum_i p_i (\log' p_i)^2 \tag{8}$$

Note that y represents the over-parameterized weights of the model. The equations above show that the regularization imposed by the KL-divergence term on these parameters is such that if these parameters change a lot (on the log scale) then the penalty imposed on such parameters is more. Thus, the kl-divergence term encourages such parameters not to change by a lot. □

B DATA AUGMENTATION AND HYPERPARAMETERS

We use different data-augmentation techniques for different datasets. Below are the details.

- *CIFAR-100*
  We apply the following set of transforms for these datasets. (1) We horizontal flip the image with probability 0.5. (2) We normalize the image by using the following mean $[0.5071, 0.4867, 0.4408]$ and standard $[0.2675, 0.2565, 0.2761]$.
- *ImageNet*
  On these two datasets, we apply the following set fo transforms. (1) Random resize cropping - Crop a rectangular region with aspect ratio in $[3/4, 4/3]$ (selected uniformly at random) with area in $[0.08, 1.0]$ of the original area. (2) Flip the image horizontally with probability 0.5. (3) Normalize the image by dividing all pixel values of 255.0.

For CIFAR-100, input size is 32×32 for all the other datasets, the input size is 224×224. The above transforms are applicable for the training dataset. For validation, we use center crop - select the center of the image with 85% area, followed by a normalization step. Note that our data augmentation are not heavily optimized for accuracy. Rather our goal is to compare adjoined training with standard training. Hence, we use the same data augmentation steps for both the trainings. For standard training, our accuracies are still comparable to the accuracies reported in the literature on these datasets using the ResNet-18 and ResNet-50 architectures. However, the adjoined training methodology proposed in this paper outperforms the network trained in the standard way.

ImageNet was trained for 100 epochs using learning rate initially set to $4e-3$. We used adam optimizer with a cosine learning rate schedule with gradual warm-up for training on ImageNet. For CIFAR-100 we used SGD optimizer for 240 epochs with $lr = 0.05$. MultiStepLR was used to decay the learning rate by 0.1 at $150, 180, 210^{th}$ epoch.

More details of the data-augmentation, hyper-parameter settings can be found in the code provided with the supplementary materials.

C RESNET ARCHITECTURE DIAGRAM

$$\rightarrow \underbrace{\begin{bmatrix} \text{conv(3, 16)} \\ \text{conv(16, 64)} \\ \text{conv(64, 64)} \end{bmatrix}}_{stem} \rightarrow \text{MaxPool} \rightarrow \underset{\text{x3}}{\text{ResBlock}(16, 64)} \rightarrow \underset{\text{x4}}{\text{ResBlock}(64, 128)}$$

$$\rightarrow \underset{x6}{\text{ResBlock}(128, 256)} \rightarrow \underset{x3}{\text{ResBlock}(256, 512)} \rightarrow \text{AverageAdaptivePool} \rightarrow \text{Linear}$$

Figure 5: Architecture diagram for ResNet-50 network used in this paper. $conv(ni, no)$ is a combination of convolution layer with ni input and no output channels followed by a batch norm and relu layer. The ResBlocks are as defined in Fig. 6.

$$\longrightarrow \begin{bmatrix} \text{conv(4ni, no)} \\ \text{conv(no, no)} \\ \text{conv(no, 4no)} \end{bmatrix} \longrightarrow \underbrace{\begin{bmatrix} \text{conv(4no, no)} \\ \text{conv(no, no)} \\ \text{conv(no, 4no)} \end{bmatrix}}_{x\ (l-1)}$$

Figure 6: A ResBlock with l layers and input ni and output no.

The architecture for ResNet-50 is depicted in Figs. 5 and 6. Each $conv$ layer is actually a combination of three layers. A standard convolution layer followed by a batch normalization layer followed by a relu activation. The $ResBlock$ refers to the residual blocks in ResNet architecture. Note that

the skip connections are not shown in these diagrams. ResNet-100 is similar to ResNet-50 but istead of repeating blocks $3, 4, 6$ and 3 times, blocks are repeated for $6, 8, 12$ and 6 times. The same architecture has been used for searching in DAN-100. In ResNet-18 architecture, each resblock is repeated twice. Also, the resblock does not have a factor four in the convolution input and output.

For adjoined networks, the convolution parameters of all the last three resblocks are shared across both the original and the smaller architecture. Note that both the networks have different parameters for the batch-norm layers.

D DETAILED EXPERIMENTAL RESULTS

D.1 EXPERIMENTS ON CIFAR-10

Training with ResNet-56 on CIFAR-10					
Training paradigm	Masking matrix (M)	Top-1% Full	Top-1% Small	# Params (M)	FLOPs
Standard		76.8		23.7	338.59
AN (a)	a_2	94.71	93.68	0.24	63.94
AN (b)	a_4	93.32	90.15	0.09	48.33
AN (a)	a_2	95.01	94.49	0.37	95
AN (b)	a_4	94.83	93.67	0.24	87.1

Table 5: Accuracy for the various training paradigms for ResNet-56 trained on CIFAR-10. For AN training paradigm # Params and FLOPs represents number of parameters, FLOPs of the smaller model. For AN models, we replace the standard convolution operation by the adjoined convolution in the last 18, 36 layers for AN-56-Small (a) and AN-56-Small (b) respectively.

D.2 EXPERIMENTS ON CIFAR-100

Training with ResNet-50 on CIFAR-100					
Training paradigm	Masking matrix (M)	Top-1% Full	Top-1% Small	# Params (M)	FLOPs
Standard		76.8		23.7	338.59
AN	a_2	77.36	76.9	6.15	140.57
AN	a_4	76.8	75.38	1.8	88.94
AN	a_4	77.1	74.63	0.67	74.97
AN	a_{16}	76.8	72.1	0.38	70.95

Table 6: Accuracy for the various training paradigms for ResNet-50 trained on CIFAR-100. For AN training paradigm # Params and FLOPs represents number of parameters, FLOPs of the smaller model.

Training with ResNet-18 on CIFAR-100					
Training paradigm	Masking matrix (M)	Top-1% Full	Top-1% Small	# Params (M)	FLOPs
Standard		74.37		11.26	153.89
AN	a_2	74.8	73.62	3.00	78.62
AN	a_4	74	71.61	0.91	59.47
AN	a_8	74	68	0.39	54.29

Table 7: Accuracy for the various training paradigms for ResNet-18 trained on CIFAR-100. For AN training paradigm # Params and FLOPs represents number of parameters (in millions), FLOPs of the smaller model.

Training with DenseNet-121 on CIFAR-100					
Training paradigm	Masking matrix (M)	Top-1% Full	Top-1% Small	# Params (M)	FLOPs
Standard		79		6.96	888
AN	a_2	80.76	78.3	1.77	502
AN	a_4	80.8	76.38	0.7	411
AN	a_8	79	72.13	0.46	387

Table 8: Accuracy for the various training paradigms for ResNet-18 trained on CIFAR-100. For AN training paradigm # Params and FLOPs represents number of parameters (in millions), FLOPs of the smaller model.

D.3 Experiments on ImageNet

Training with ResNet-50 on ImageNet					
Training paradigm	Masking matrix (M)	Top-1% Full	Top-1% Small	# Params (M)	GFLOPs
Standard		76.1		25.5	4.7
AN	a_2	76.87	75.1	7.1	2.2
AN	a_4	75.84	71.84	2.2	1.6
AN	a_8	73.46	64.7	0.67	1.4

Table 9: Accuracy for the various training paradigms for ResNet-50 trained on ImageNet. For AN training paradigm # Params and FLOPs represents number of parameters (in millions), FLOPs of the smaller model.

D.4 EXPERIMENTS WITH DAN

Performance on architectures found by DAN					
Dataset	Network	Search Space	Params	FLOPs	Accuracy
CIFAR-100	ResNet-50		23.7	0.338	76.8
	DAN-50	$1, 2, 4, 8, 16$	1.64	0.084	75.1
	DAN-100	$1, 2, 4$	6.7	0.173	**77.62**
ImageNet	ResNet-50		25.5	4.7	76.1
	ResNet-100		46.99	8.47	77.3
	DAN-50	$2, 4$	3.49	1.74	73.33
	DAN-100	$2, 4, 8$	5.22	1.85	74.78
	DAN-100	$4, 8$	6.58	2.15	**75.43**
	DAN-100	$1, 2, 4$	12.5	2.95	75.71

Table 10: Performance on architectures found by DAN

E CHOOSING THE REGULARIZATION FUNCTION

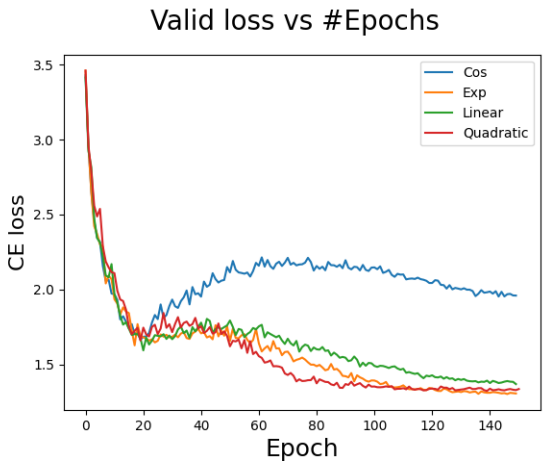

Figure 7: Validation cross entropy loss for various regularization functions. The networks were trained using Adj-4 mask matrix on CIFAR-100 using ResNet-18.

adjoined trained with different regularization functions		
Regularization function $\lambda(t)$	Top-1 Full	Top-1 Small
$1 - \cos(t)$	-2.86	-3.14
t	-0.1	-0.15
$\min\{4t^2, 1\}$	0.00	0.00
$\exp(t) - 1$	-0.37	$+0.38$

Table 11: The effect of training with different regularization functions on the top-1 accuracies of the bigger and the smaller networks. The quadratic function $\min\{4t^2, 1\}$ is used as the base for comparison.

Finally, we compare different choices of regularization function for the adjoined loss (Eqn. 3). For all the previous experiments, we use the 'quadratic' function $\lambda(t) = \min\{4t^2, 1\}$. In this section, we fix the architecture, dataset and mask matrix as ResNet-18, CIFAR-100 and a_4 respectively and vary the regularization function. We look at different functions which includes exponential and trigonometric functions. Table 11 and Fig. 7 both show the same trend. The cos function performs the worst while the rest have similar performance. We conjecture that any function that is close to zero for $t \leftarrow 0$ and grows to one eventually should be a reasonable choice for the regularization function. Note that throughout our discussion, we have used $\lambda(t) = c\min\{4t^2, 1\}$ with $c = 1$. Depending on the dataset, other values of c maybe more appropriate.

F Comparison against other works

Model Compression Results for ResNet-56 Architecture				
Method	# Params	GFLOPs	Accuracy	Reference
Gal-0.8	0.29	49.99	90.36	Lin et al. (2019)
Hrank	0.27	32.52	90.72	Lin et al. (2020a)
He at al	–	62	90.8	Lin et al. (2019)
GAL-0.6	0.75	78.3	92.98	Lin et al. (2019)
ENC	–	63.5	93.0	Kim et al. (2019)
NISP	0.49	81	93.01	Yu et al. (2018b)
L1	0.73	90.9	93.06	Li et al. (2017)
Hrank	0.49	62.72	93.17	Lin et al. (2020a)
ABC-Prunner	0.39	58.54	93.23	Lin et al. (2020b)
CaP	–	63.7	93.22	Minnehan & Savakis (2019)
KSE	0.38	60.96	93.23	Li et al. (2019)
FPGM	–	60.9	93.26	He et al. (2019)
GBN	0.4	50.4	93.43	You et al. (2019)
Hrank	0.71	88.72	93.52	Lin et al. (2020a)
Hinge	0.41	63.5	93.69	Li et al. (2020)
AN-56-Small α_2(a) (Our)	0.37	95	94.49	
AN-56-Small α_2(b) (Our)	0.27	63.94	93.68	

Table 12: The table shows the performance of various pruning methods when trained on the CIFAR-10 dataset for ResNet-56 architecture. a_α in AN-50-Small denotes the mask matrix as defined in Eqn. For AN models, we replace the standard convolution operation by the adjoined convolution in the last 18, 36 layers for AN-56-Small α_2(a) and AN-56-Small α_2(b) respectively. 2.

Model Compression Results for ResNet-110 Architecture				
Method	# Params	GFLOPs	Accuracy	Reference
ABC-Prunner	0.56	89.87	93.58	Lin et al. (2020b)
Hrank	1.04	148.7	94.23	Lin et al. (2020a)
Hrank	0.7	105.7	93.36	Lin et al. (2020a)
GAL-0.5	0.95	130.2	92.55	Lin et al. (2019)
L1	1.16	155	93.3	Li et al. (2017)
AN-110-Small α_2(a) (Our)	0.74	190.56	95	
AN-110-Small α_2(b) (Our)	0.49	127.64	94.73	

Table 13: The table shows the performance of various pruning methods when trained on the CIFAR-10 dataset for ResNet-110 architecture. a_α in AN-50-Small denotes the mask matrix as defined in Eqn. For AN models, we replace the standard convolution operation by the adjoined convolution in the last 36, 72 layers for AN-110-Small α_2(a) and AN-110-Small α_2(b) respectively. 2.

Sampling Filters According to Bernoulli's Distribution				
Probability (p)	AN-Large	AN-Small	AN-Large↓	AN-Small↓
0.25	92.8	90.98	0.71	1.47
0.5	93.04	92.25	0.47	0.2
0.75	92.89	92.58	0.62	-0.13

Table 15: We compare the fixed mask structure in AN against sampling mask M according to Bernoulli's Distribution. Probability (p) denotes probability of sampling a filter in each layer. AN-Large and AN-Small denote the accuracy of large and small models on CIFAR-10 dataset with ResNet-20. AN-Large↓ and AN-Small↓ denote the drop in accuracy of the respective models compared to the Adjoined model trained using fixed mask matrix ($\alpha = 2$).

G Normalization Constant (γ) for DAN

Comparison against various values of γ for DAN-50			
Normailization constant (γ)	# Param	FLOPs	Accuracy
0	4.49	120	75.5
1e-13	1.64	84	75.1
1e-6	0.38	71	72

Table 16: We compare the performance of DAN-50 models trained on CIFAR-100 dataset against various values of Normailization constant γ as Defn. in Eqn. 6. For all DAN models we use $\alpha = \{1, 2, 4, 8, 16\}$ as the search space.