6. Supplementary Material

6.1. Update of τ

To meet the memory limitation (parameter constraint) of any platform, we only need to update τ in our approach. We show the proof of the update of τ as follows:

$$\tau_i = \tau_{i-1} - \Delta\tau_{i-1} \tag{12}$$

$$= \tau_{i-1} - \eta \frac{\partial \frac{1}{2}(h(f(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{\Phi}(\tau_{i-1}|\boldsymbol{\Theta}))) - \hat{\tau})^2}{\partial \tau_{i-1}} \tag{13}$$

$$= \tau_{i-1} - \eta(h(f(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{\Phi}(\tau_{i-1}|\boldsymbol{\Theta}))) - \hat{\tau}) \tag{14}$$

$$\cdot \frac{\partial h(f(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{\Phi}(\tau_{i-1}|\boldsymbol{\Theta})))}{\partial \boldsymbol{\Phi}(\tau_{i-1}|\boldsymbol{\Theta})} \frac{\partial \boldsymbol{\Phi}(\tau_{i-1}|\boldsymbol{\Theta})}{\partial \tau_{i-1}} \tag{15}$$

$$= \tau_{i-1} - \eta(h(f(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{\Phi}(\tau_{i-1}|\boldsymbol{\Theta}))) - \hat{\tau}) \sum_{l=1}^{L} \beta_l \alpha_l \tau_{i-1}^{\beta_l - 1}. \tag{16}$$

Note that (15) comes from the chain rule of derivative.

6.2. Accuracy Comparisons

In this section we show the tabulized comparison of VGG11-CIFAR100, MobileNetV2-CIFAR100 and ResNet18-TinyImageNet which was not shown in the main paper. As shown in Table 2, an accuracy gain of 5.85%, 2.40% and 3.04% is observed for VGG11, MobileNetV2 and ResNet18 on CIFAR100, TinyImageNet and CIFAR100 respectively.

6.3. Pre-Training Epochs P

The pruning of neural network is usually done on a pre-trained network. As we want our algorithm to be efficient in terms of search cost, we explore the possibility of reduction in time or epochs for network pre-training by tuning the pre-training epochs P. To our surprise, having a large P does not result in an architecture with the best performance. Here, we investigate how P affects the accuracy of the final configuration, proving that conventional wisdom on when to apply pruning might be flawed. Experiments will be shown on VGG11 and MobileNetV2 on CIFAR10 and CIFAR100 respectively. All results shown are based on the final (iteration=15) iteration of architecture descent.

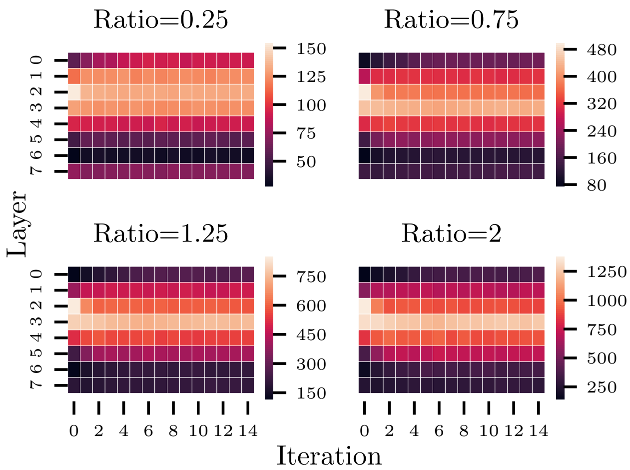
VGG11. CIFAR10 will be used for the experimentation on P for VGG11. We show architecture and results obtained by setting P to be 0, 2, 5, 10, 30 and 60. The searched architecture is shown in Figure 6. We next show the comparison plot using different pre-training epochs in Figure 7 accompanied by Table 3. For a simple network like VGG11, the number of pre-training epochs doesn't have too much of an impact in performance which can be clearly observed in the resulting filter configuration in Figure 6.

Table 2: Comparison of various network-dataset pairs.

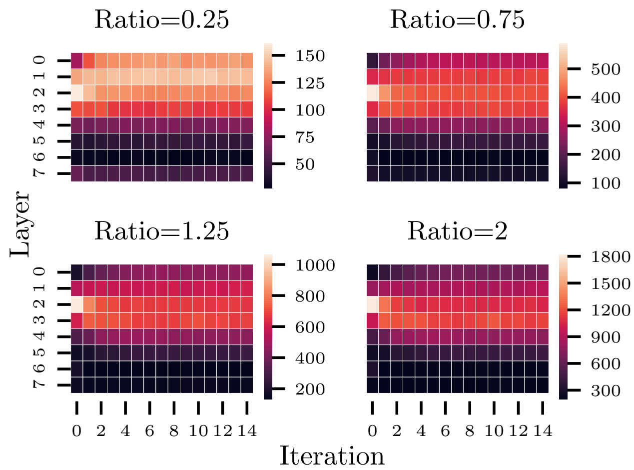
Method	Params	Latency	Accuracy (%)
VGG11 CIFAR100			
Uniform Scale (Baseline)	0.59M	1.30ms	60.22 ± 0.45
	5.23M	4.28ms	68.56 ± 0.21
	36.99M	18.83ms	71.94 ± 0.25
Li *et al.* [29]†	5.23M	4.77ms	68.41 ± 0.09
MorphNet [12] (Taylor-FO [35])	0.59M	1.78ms	64.85 ± 0.17
	5.21M	7.18ms	70.64 ± 0.38
	36.80M	41.52ms	72.72 ± 0.09
NeuralScale (Iteration = 1)	0.59M	1.95ms	65.71 ± 0.28
	5.23M	7.36ms	70.50 ± 0.16
	36.98M	33.24ms	**72.78 ± 0.19**
NeuralScale (Iteration = 15)	0.59M	2.52ms	**66.07 ± 0.21**
	5.23M	10.19ms	**70.70 ± 0.45**
	36.98M	43.95ms	**72.78 ± 0.13**
MobileNetV2 TinyImageNet			
Uniform Scale (Baseline)	0.23	8.53ms	44.22 ± 0.40
	1.52M	18.87ms	54.63 ± 0.46
Li *et al.* [29]†	1.52M	18.76ms	52.71 ± 0.28
MorphNet [12] (Taylor-FO [35])	0.23M	10.47ms	44.53 ± 0.50
	1.51M	28.88ms	53.08 ± 0.52
NeuralScale (Iteration = 1)	0.22M	14.96ms	**49.70 ± 0.73**
	1.49M	26.98ms	54.18 ± 0.57
NeuralScale (Iteration = 15)	0.22M	17.16ms	46.82 ± 0.89
	1.49M	41.20ms	**55.42 ± 0.44**
ResNet18 CIFAR100			
Uniform Scale (Baseline)	0.71M	2.53ms	68.10 ± 0.40
	6.32M	9.98ms	75.10 ± 0.34
	44.75M	47.04ms	78.39 ± 0.29
Li *et al.* [29]†	6.32M	10.18ms	73.91 ± 0.12
MorphNet [12] (Taylor-FO [35])	0.72M	3.73ms	69.34 ± 0.31
	6.29M	15.03ms	75.60 ± 0.40
	44.53M	98.54ms	**78.68 ± 0.17**
NeuralScale (Iteration = 1)	0.71M	4.51ms	70.63 ± 0.13
	6.38M	11.95ms	75.83 ± 0.15
	45.15M	50.24ms	78.39 ± 0.22
NeuralScale (Iteration = 15)	0.71M	5.71ms	**71.14 ± 0.45**
	6.36M	19.54ms	**76.35 ± 0.20**
	45.05M	90.18ms	78.62 ± 0.13

† Fine-tuned using pre-trained network (not trained from scratch).

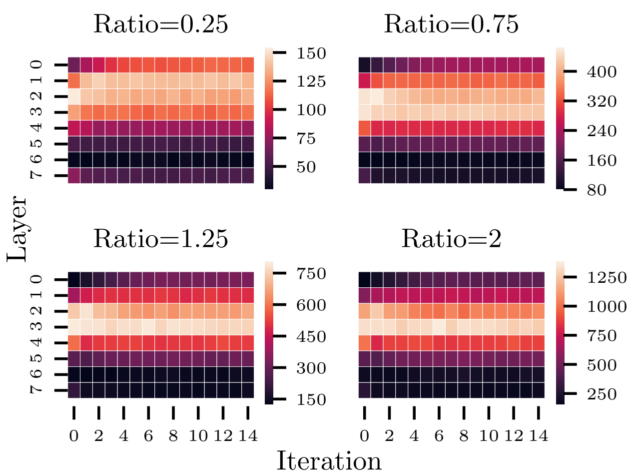
MobileNetV2. CIFAR100 will be used for the experimentation on P for MobileNetV2. We show architecture and results obtained by setting P to be 0, 2, 5, 10, 30 and



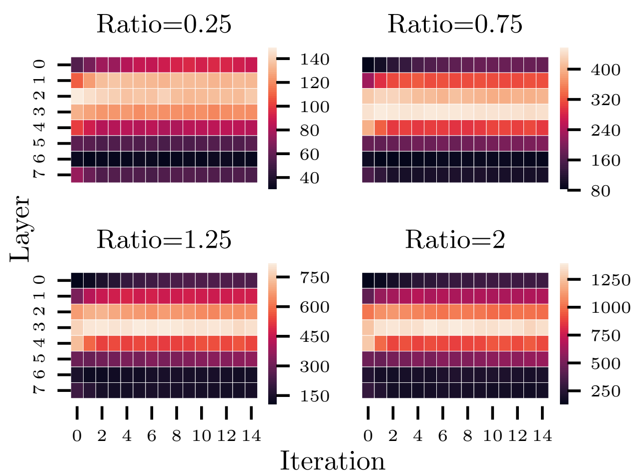
(a) $P = 0$



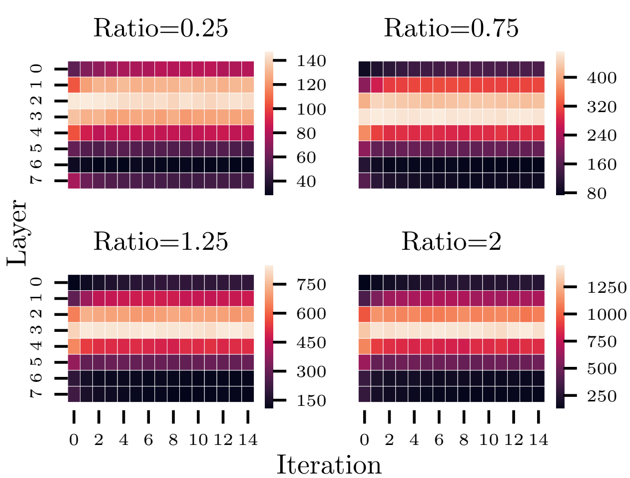
(b) $P = 2$



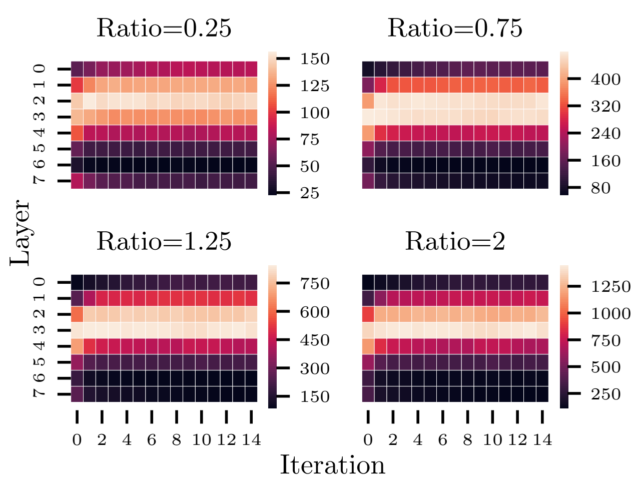
(c) $P = 5$



(d) $P = 10$

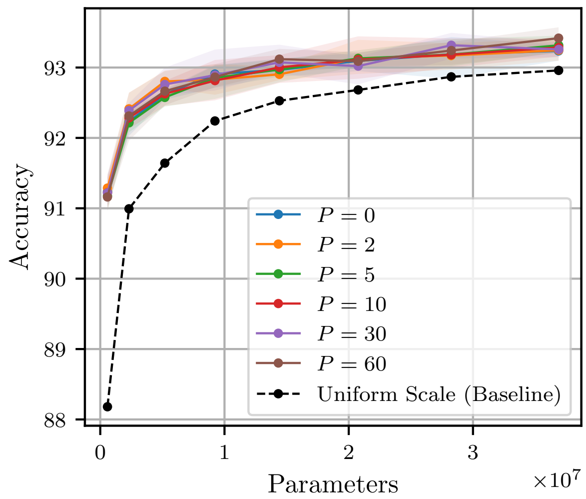


(e) $P = 30$

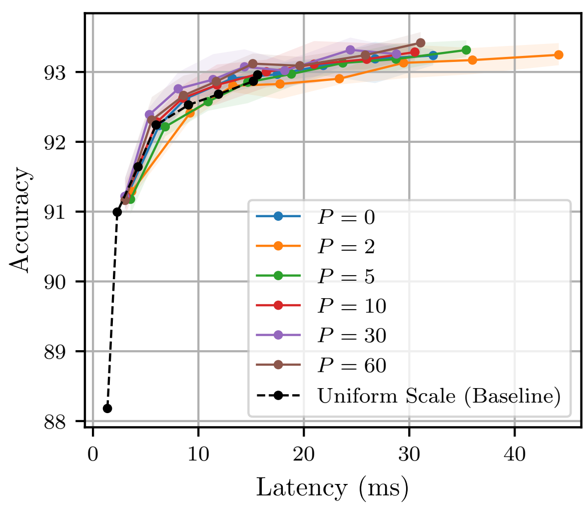


(f) $P = 60$

Figure 6: Showing the difference in searched architecture by running architecture descent on VGG11 for CIFAR10 using various value of pre-training epochs P.



(a) Accuracy vs Parameter.



(b) Accuracy vs Latency.

Figure 7: Accuracy comparison plot for VGG11 on CIFAR10 that uses different pre-training epochs P before pruning. (a) shows the accuracy comparison under different parameters using different value of P. (b) shows the comparison of accuracy under different latencies using different value of P.

60. The searched architecture is shown in Figure 8. We next show the comparison plot using different pre-training epochs in Figure 9 accompanied by Table 4. It is interesting to see that for a deeper and more complicated network like MobileNetV2, there's a notable variation in the distribution of filters with respect to the number of pre-training epochs. The accuracy comparison in Figure 9 shows that

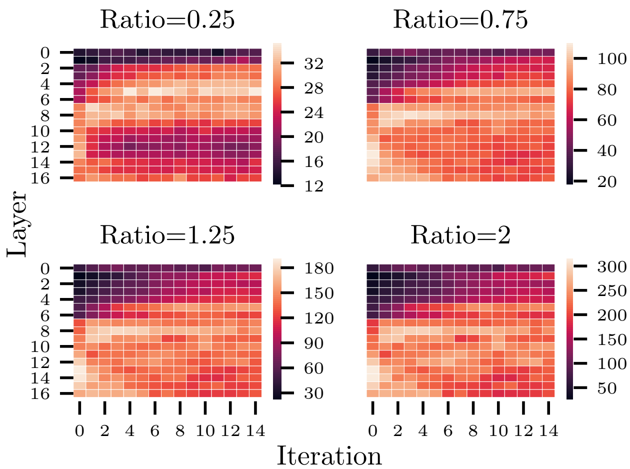
Table 3: Accuracy comparison on VGG11 for CIFAR10 using different pre-training epochs P.

Method	Params	Latency	Accuracy (%)
Uniform Scale (Baseline)	0.58M	1.30ms	88.18 ± 0.16
	5.20M	4.31ms	91.64 ± 0.10
	36.89M	19.50ms	92.96 ± 0.09
NeuralScale ($P = 0$)	0.58M	3.01ms	91.23 ± 0.05
	5.20M	12.35ms	92.62 ± 0.06
	36.89M	53.26ms	93.24 ± 0.09
NeuralScale ($P = 2$)	0.58M	3.49ms	$\mathbf{91.29 \pm 0.09}$
	5.20M	20.24ms	$\mathbf{92.80 \pm 0.09}$
	36.90M	81.27ms	93.25 ± 0.10
NeuralScale ($P = 5$)	0.58M	3.34ms	91.18 ± 0.13
	5.19M	17.30ms	92.58 ± 0.08
	36.90M	63.85ms	93.31 ± 0.08
NeuralScale ($P = 10$)	0.58M	2.93ms	91.22 ± 0.15
	5.20M	12.53ms	92.63 ± 0.12
	36.90M	55.44ms	93.29 ± 0.09
NeuralScale ($P = 30$)	0.58M	2.82ms	91.22 ± 0.15
	5.20M	11.85ms	92.76 ± 0.13
	36.90M	51.02ms	93.26 ± 0.08
NeuralScale ($P = 60$)	0.58M	2.85ms	91.16 ± 0.17
	5.20M	12.58ms	92.66 ± 0.18
	36.89M	61.14ms	$\mathbf{93.42 \pm 0.13}$

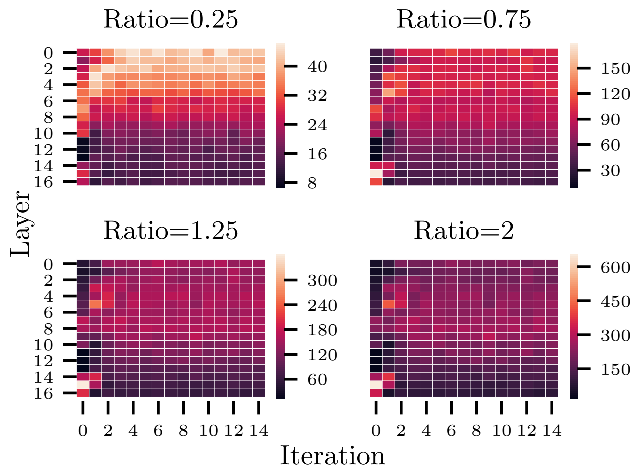
having large number of pre-training epochs doesn't help the efficiency in parameters and instead impedes it. It is shown that $P = 2$ or $P = 5$ gives us a configuration of filters that is the most efficient in terms of parameters for MobileNetV2 on CIFAR100. This is an interesting observation which sheds light on the number of pre-training iterations required prior to network pruning for optimal performance.

6.4. Using Convolutional Layers as Shortcut Connection

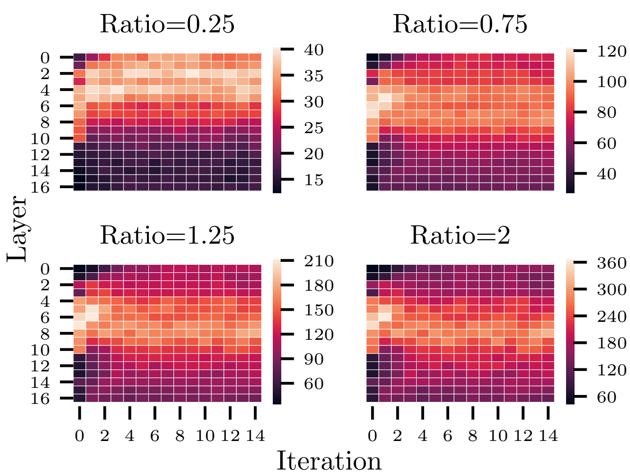
By default, MobileNetV2 has shortcut connections composed of identity mappings. By modifying the filter sizes of MobileNetV2, the shortcut connection has to be changed to a convolutional one instead to compensate the difference in filter sizes on both ends of the shortcut connection. A surprising finding is that the change from identity mapping to convolutional mapping affects the original performance significantly, despite the increase in parameter. We show experiments comparing two kinds of shortcut connection (identity and convolutional) on the original configuration which is uniformly scaled to different ratios. We name the method that uses convolutional shortcuts as *ConvCut*. A comparison plot comparing ConvCut with other scaling



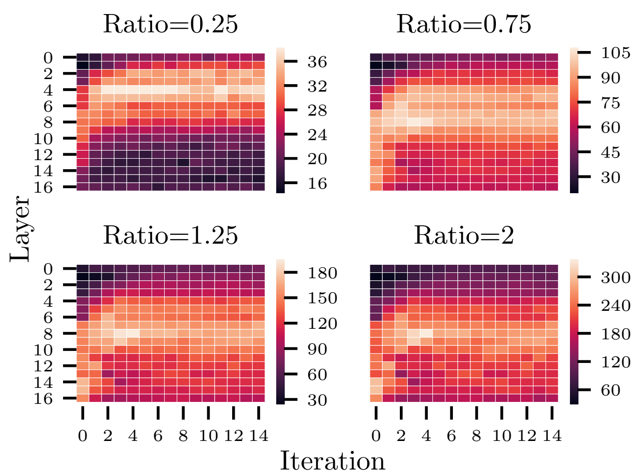
(a) $P = 0$



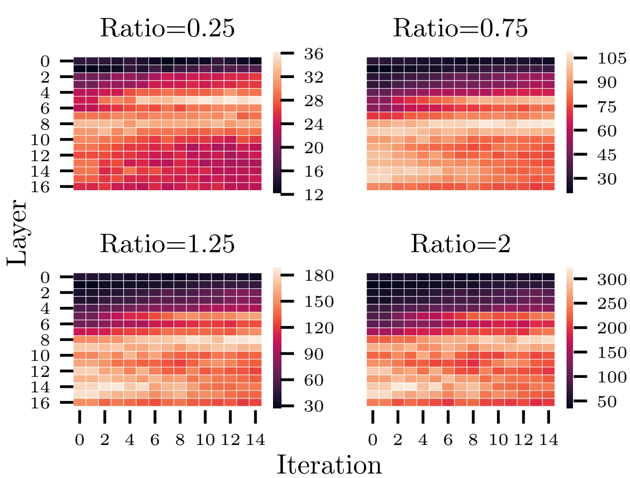
(b) $P = 2$



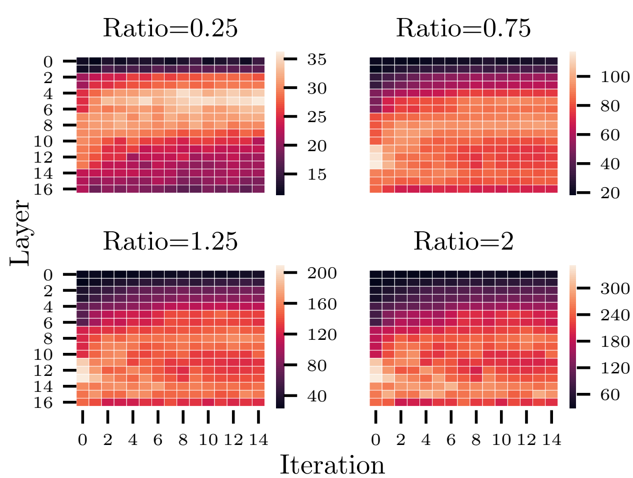
(c) $P = 5$



(d) $P = 10$

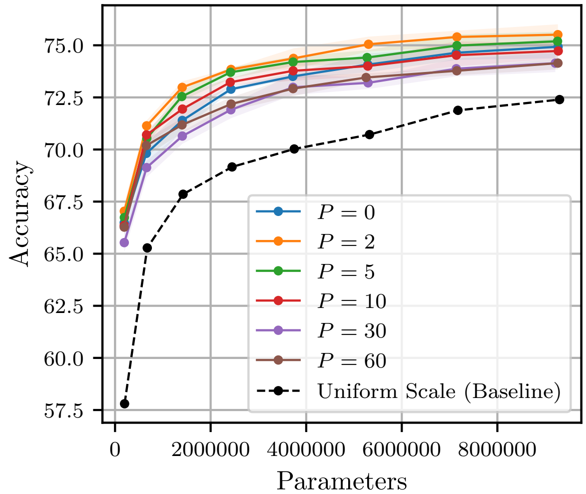


(e) $P = 30$

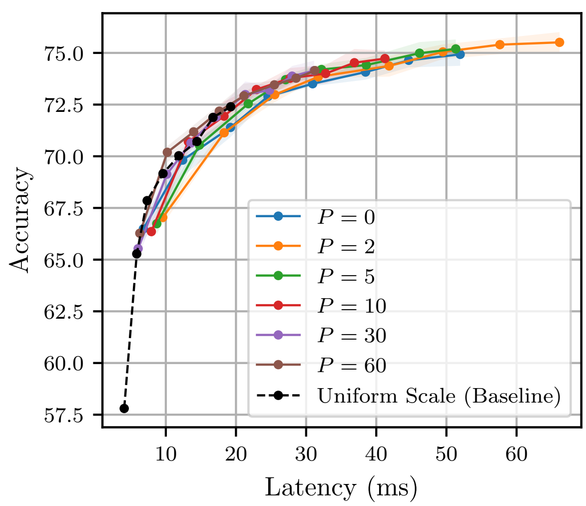


(f) $P = 60$

Figure 8: Showing the difference in searched architecture by running architecture descent on MobileNetV2 for CIFAR100 using various value of pre-training epochs P.



(a) Accuracy vs Parameters.



(b) Accuracy vs Latency.

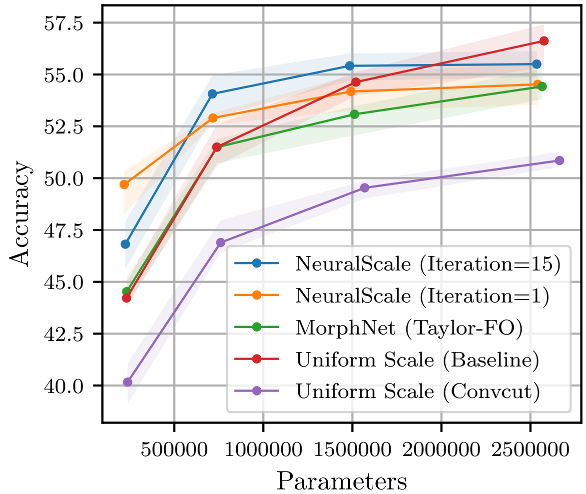
Figure 9: Accuracy comparison plot for MobileNetV2 on CIFAR100 that uses different pre-training epochs P before pruning. (a) shows the accuracy comparison under different parameters using different value of P. (b) shows the comparison of accuracy under different latencies using different value of P.

methods using ResNet18 and MobileNetV2 on TinyImageNet is shown in Figure 10 and 11 respectively. Results are summarized in Table 6 and Table 5 for ResNet18 and MobileNetV2 respectively. It can be observed that the switch from identity to convolutional mapping doesn't have drastic impact on the accuracy of ResNet18 but a significant drop in accuracy can be observed for MobileNetV2. Our conjecture

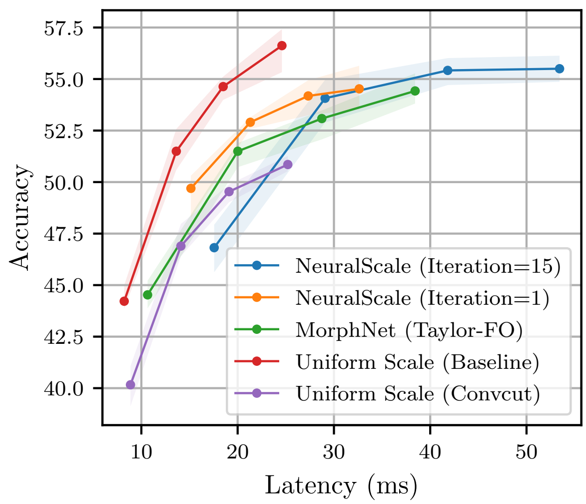
Table 4: Accuracy comparison on MobileNetV2 for CIFAR100 using different pre-training epochs P.

Method	Params	Latency	Accuracy (%)
Uniform Scale (Baseline)	0.20M	5.53ms	57.80 ± 0.31
	1.42M	7.56ms	67.85 ± 0.38
	9.30M	20.43ms	72.40 ± 0.22
NeuralScale ($P = 0$)	0.19M	6.64ms	66.49 ± 0.43
	1.40M	18.77ms	71.39 ± 0.45
	9.27M	52.49ms	74.93 ± 0.34
NeuralScale ($P = 2$)	0.19M	9.42ms	$\mathbf{67.04 \pm 0.28}$
	1.40M	24.95ms	$\mathbf{72.98 \pm 0.26}$
	9.27M	67.55ms	$\mathbf{75.51 \pm 0.41}$
NeuralScale ($P = 5$)	0.19M	8.61ms	66.74 ± 0.39
	1.40M	21.37ms	72.54 ± 0.18
	9.26M	51.63ms	75.19 ± 0.26
NeuralScale ($P = 10$)	0.19M	7.82ms	66.36 ± 0.28
	1.41M	18.02ms	71.94 ± 0.45
	9.27M	43.00ms	74.73 ± 0.26
NeuralScale ($P = 30$)	0.19M	6.20ms	65.53 ± 0.31
	1.41M	13.35ms	70.64 ± 0.23
	9.21M	31.77ms	74.14 ± 0.35
NeuralScale ($P = 60$)	0.19M	6.15ms	66.28 ± 0.13
	1.40M	13.74ms	71.18 ± 0.24
	9.27M	32.15ms	74.15 ± 0.18

is that the design of linear bottleneck layers in MobileNetV2 is to embed a low-dimensional manifold where switching from identity to convolutional mapping for shortcut layer that connects linear bottleneck layers introduces noise to this manifold which is harmful for information propagation and network training. Despite from the setback of accuracy drop through the introduction of convolutional shortcut layers, our approach is still able to induce accuracy gain in a low parameter count setting when compared to the baseline configuration setting, showing the importance of searching for the optimal configuration of filters. An unbiased comparison is to compare our approach with the convolutional shortcut (ConvCut) version of MobileNetV2 using the default set of filter configuration as shown in Figure 10 where both (ours and ConvCut) use convolutional layer as shortcut connection. On an apple-to-apple comparison, our approach shows superiority in parameter efficiency. This empirical study also explains the superiority in accuracy of iteration 1 when compared to iteration 15 of our approach as can be observed in Figure 10a. From our observation, iteration 1 of our approach generates a configuration composed repeated filters on some blocks, resulting in an architecture consisting of both identity and convolutional shortcut con-



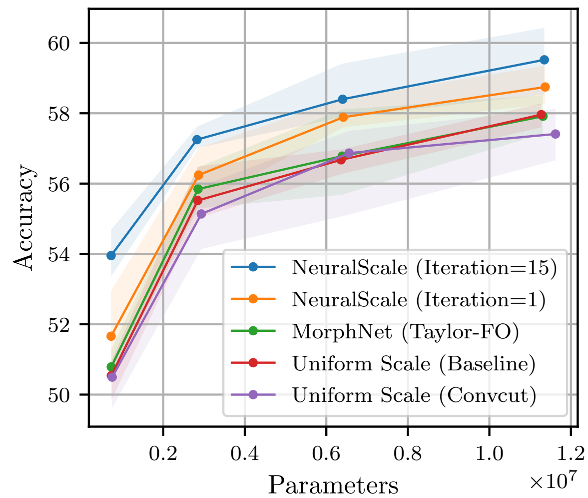
(a) Accuracy vs Parameter.



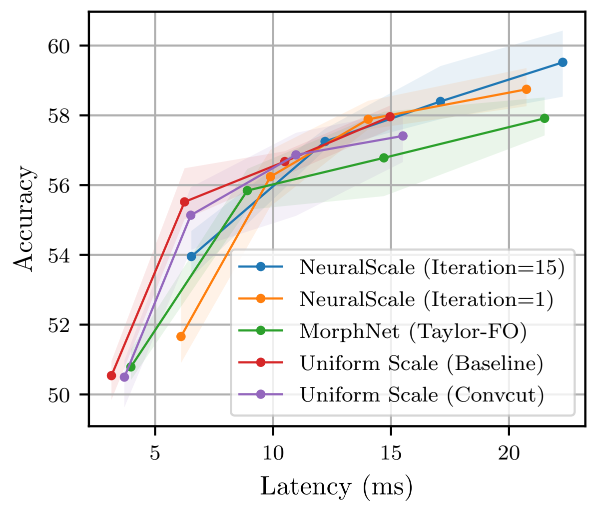
(b) Accuracy vs Latency.

Figure 10: Accuracy comparison plot for MobileNetV2 on TinyImageNet with inclusion of ConvCut.

nection. Hence, it is not surprising that iteration 1 outperforms iteration 15 of our approach as it has both traits: identity shortcut and optimized filter configuration.



(a) Accuracy vs Parameter.



(b) Accuracy vs Latency.

Figure 11: Accuracy comparison plot for ResNet18 on TinyImageNet with inclusion of ConvCut.

Table 5: Accuracy comparison on MobileNetV2 on Tiny-ImageNet (includes ConvCut).

Method	Params	Latency	Accuracy (%)
Uniform Scale (Baseline)	0.23M	8.53ms	44.22 ± 0.40
	1.52M	18.87ms	54.63 ± 0.46
	2.58M	24.70ms	**56.62 ± 0.70**
MorphNet [12] (Taylor-FO [35])	0.23M	10.47ms	44.53 ± 0.50
	1.51M	28.88ms	53.08 ± 0.52
	2.57M	38.45ms	54.42 ± 0.53
Uniform Scale (ConvCut)	0.24M	9.23ms	40.16 ± 0.63
	1.57M	19.23ms	49.54 ± 0.30
	2.66M	25.39ms	50.85 ± 0.27
NeuralScale (Iteration = 1)	0.22M	14.96ms	**49.70 ± 0.73**
	1.49M	26.98ms	54.18 ± 0.57
	2.54M	32.09ms	54.52 ± 0.72
NeuralScale (Iteration = 15)	0.22M	17.16ms	46.82 ± 0.89
	1.49M	41.20ms	**55.42 ± 0.44**
	2.54M	52.76ms	55.50 ± 0.51

Table 6: Accuracy comparison on ResNet18 on TinyImageNet (includes ConvCut).

Method	Params	Latency	Accuracy (%)
Uniform Scale (Baseline)	0.73M	3.02ms	50.54 ± 0.37
	6.36M	11.56ms	56.68 ± 0.28
	11.27M	15.46ms	57.96 ± 0.23
MorphNet [12] (Taylor-FO [35])	0.72M	3.80ms	50.79 ± 0.38
	6.39M	14.83ms	56.78 ± 0.85
	11.31M	22.07ms	57.91 ± 0.38
Uniform Scale (ConvCut)	0.75M	3.64ms	50.50 ± 0.46
	6.56M	11.99ms	56.87 ± 0.88
	11.62M	15.98ms	57.41 ± 0.58
NeuralScale (Iteration = 1)	0.72M	5.96ms	51.66 ± 0.80
	6.42M	14.58ms	57.89 ± 0.28
	11.37M	22.11ms	58.75 ± 0.37
NeuralScale (Iteration = 15)	0.72M	6.42ms	**53.95 ± 0.53**
	6.40M	17.52ms	**58.40 ± 0.54**
	11.35M	25.94ms	**59.52 ± 0.63**