APPENDIX

A FULLY CONVOLUTIONAL NETWORK ARCHITECTURE

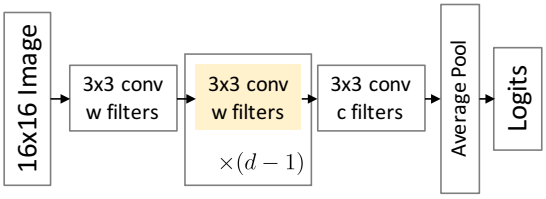


Figure 9: A diagram of the Fully-Conv Net used with width w, depth d, and c classes. Each convolutional layer (except for the last one) is followed by batch norm and LeakyReLU.

B RESNET ARCHITECTURE

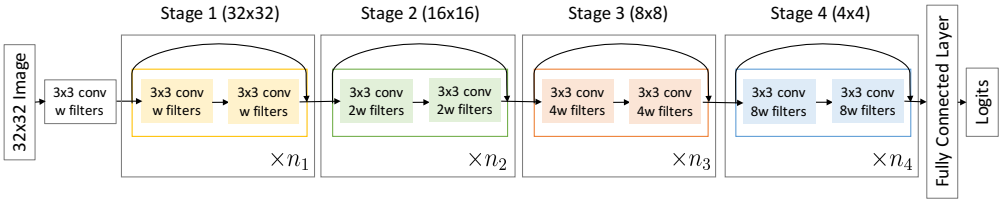


Figure 10: A diagram of the custom ResNet models used. The ith block is repeated n_i times in stage i. The first convolutional layer in stages 2, 3, and 4 downsamples the input using a stride of 2, while the remaining layers have a stride of 1.

C EXPERIMENTAL DETAILS

All models were trained on an NVIDIA TITAN RTX GPU using the PyTorch library.

An anonymized repository with code used for this paper can be found here: `https://anonymous.4open.science/r/ebf33ffa-565e-408d-ae5f-12d91f942000/`

Dataset	Network	Width	Activation	Optimizer	Init.	Train Size	# Epochs	Loss	Trials	Figure
CIFAR10	Fully-Conv	16	LeakyReLU	Adam lr=1e-4	Kaiming Normal	5000 /class	2000, or 100% accuracy	Cross Entropy	5	2
						5000 /class			5	
CIFAR10	FCN	4096								
ImageNet32	Fully-Conv	16				1300 /class			5	3
ImageNet32	FCN	4096				1300 /class			5	
CIFAR10	Fully-Conv	32							3	4
						5000 /class				
						1300 /class				
ImageNet32	Fully-Conv	32							3	
CIFAR10	ResNet	64	ReLU	SGD w/ momentum LR schedule from Yang et al (2020)	Default Pytorch	2500/class	500	MSE	3	1a, 12, 13ae
	ResNet	32							3	5a, 12, 13bf
	ResNet	16							3	5b, 12, 13cg
	ResNet	8							3	13dh, 14
	ResNet Increase Stage 1	32							1	15
	ResNet Increase Stage 3	32							1	15
	ResNet10 ResNet18 Varied kernel size	32							1	17
	ResNet	32				500/class	2000		1	16
CIFAR10	CNTK	Infinite	ReLU	N/A	N/A	100/class, 250/class	N/A	MSE	1	6, 18
Toy classification	Linear Conv	32	None	Adam lr=1e-4	Xavier Uniform	2, 8	Until Convergence	MSE	5	7, 19
Cifar10	Linear Conv	16	None	Adam lr=1e-4	Xavier Normal	1, 5	Until Convergence	MSE	5	8

Figure 11: Experimental details for all experiments conducted.

Model Depth	Blocks per Stage
10	1, 1, 1, 1
12	1, 1, 2, 1
14	1, 2, 2, 1
16	2, 2, 2, 1
18	2, 2, 2, 2
20	2, 2, 3, 2
22	2, 3, 3, 2
26	3, 3, 3, 3
30	3, 4, 4, 3
34	3, 4, 6, 3
38	3, 4, 8, 3
42	3, 4, 10, 3
46	3, 4, 12, 3
50	3, 4, 14, 3

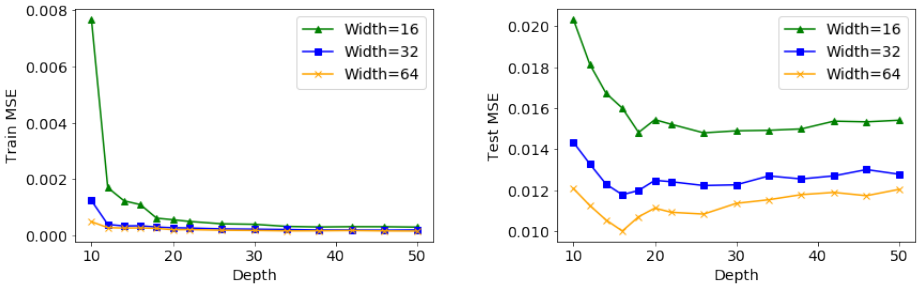
Table 1: Stage breakdown for all ResNet models used.

# Classes	Classes used
2	dog, cat
5	bird, cat, deer, dog, horse
10	all classes

Table 2: CIFAR10 classes considered in this work.

# Classes	Classes used
2	kit_fox, English_setter
5	2 classes + Siberian_husky, Australian_terrier, English_springer
10	5 classes + Egyptian_cat, Persian_cat, malamute, Great_Dane, Walker_hound

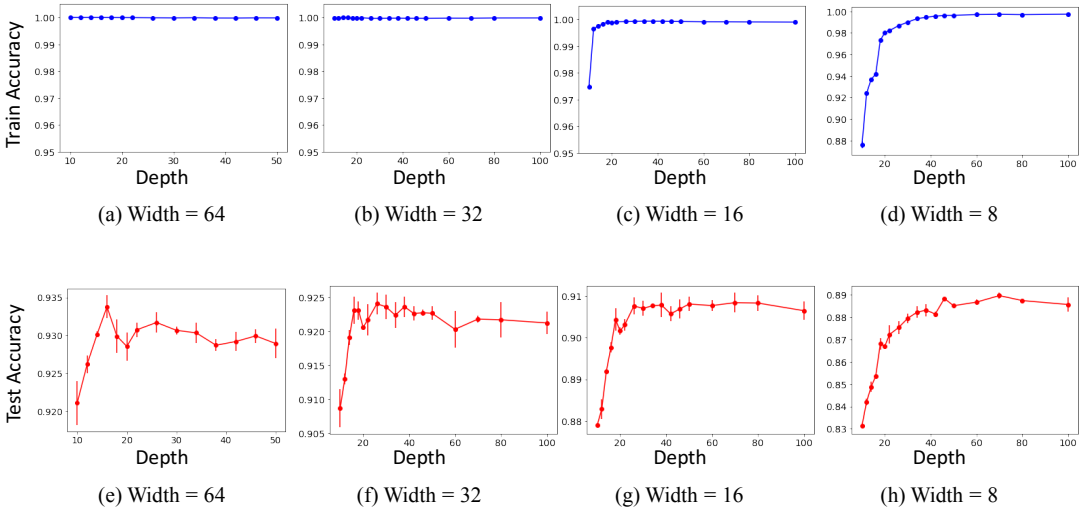
Table 3: ImageNet32 classes considered in this work.



(a) Train Losses of ResNet models with widths 16, 32, and 64.

(b) Test Losses of ResNet models with widths 16, 32, and 64.

Figure 12: Train and Test losses of the ResNet models for all widths.



(a) Width = 64 (b) Width = 32 (c) Width = 16 (d) Width = 8

(e) Width = 64 (f) Width = 32 (g) Width = 16 (h) Width = 8

Figure 13: Train and Test accuracies for the ResNet models of width 8, 16, 32, and 64, for increasing depths.

D Additional Experiments

D.1 Additional ResNet Plots

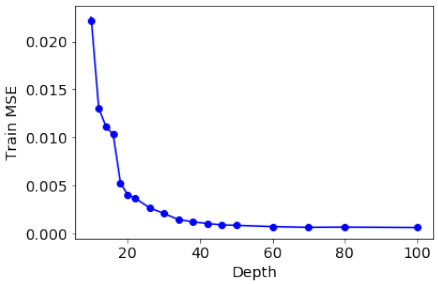
In Figure 12, we plot the train and test losses of all ResNet models used (for widths 16, 32, 64). Additionally, in Figure 13 we plot the accuracies of all ResNet models.

D.2 Effect of Depth in models with small widths

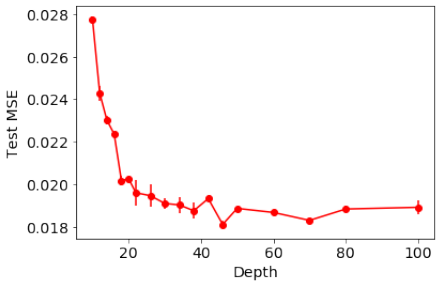
The only model in which test loss continues to increase is the width 8 model. We argue this is because the width 8 model is not sufficiently over-parameterized; in fact, in Figure 14, we see that the width 8 model is unable to reach zero training loss, while all the other models are after sufficient depth.

D.3 Effect of Downsampling

In Figure 15 we compare a ResNet model where we increase the number of blocks in the first stage versus a model where we increase the number of blocks in the third stage. We observe that the model where the third stage blocks are increased performs worse. This is likely because adding a block in a later stage, after downsampling, increases the effective depth of the model more than adding a block in an earlier stage.



(a) Train losses for the width 8 ResNet model, for increasing depths.



(b) Test losses for the width 8 ResNet model, for increasing depths.

Figure 14: Train and test losses for the width 8 ResNet model. We see that test loss decreases as model depth increases, but train loss has still not reached 0, even for large depths.

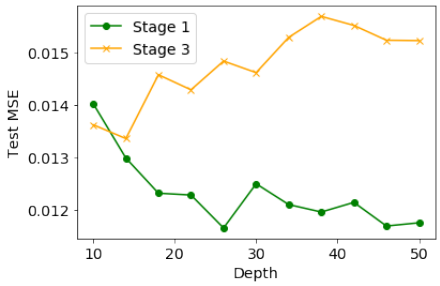


Figure 15: Test losses for the width 32 ResNet model where the 1st stage blocks are increased versus the model where the 3rd stage blocks are increased.

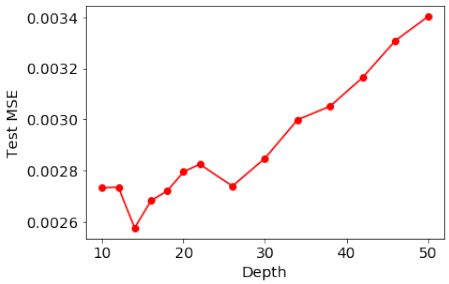


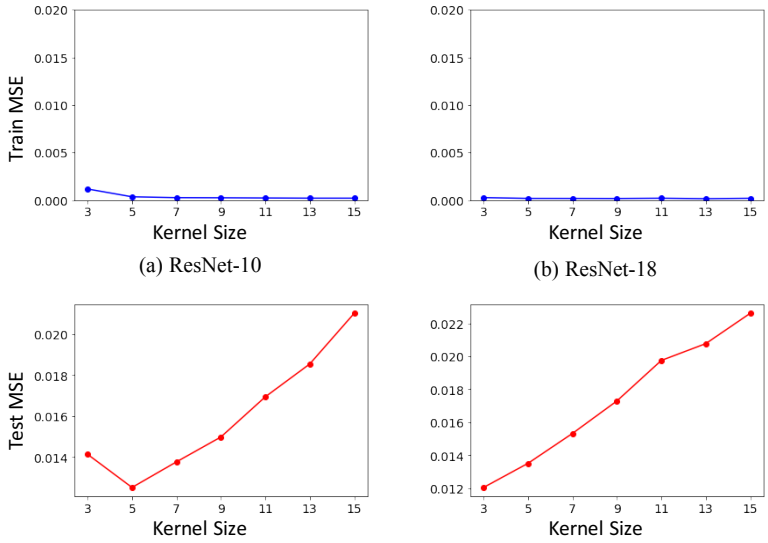
Figure 16: Test losses for the width 32 ResNet samples, trained on 500 samples per class.

D.4 Effect of Number of Samples

In Figure 16 is a plot of test losses when training the width 32 ResNet model on 500 samples per class ($\frac{1}{10}$ of CIFAR10). Number of training epochs is increased accordingly. We observe that test loss increases as depth increases, showing that this phenomenon is robust to change in sample size.

D.5 Effect of Kernel Size

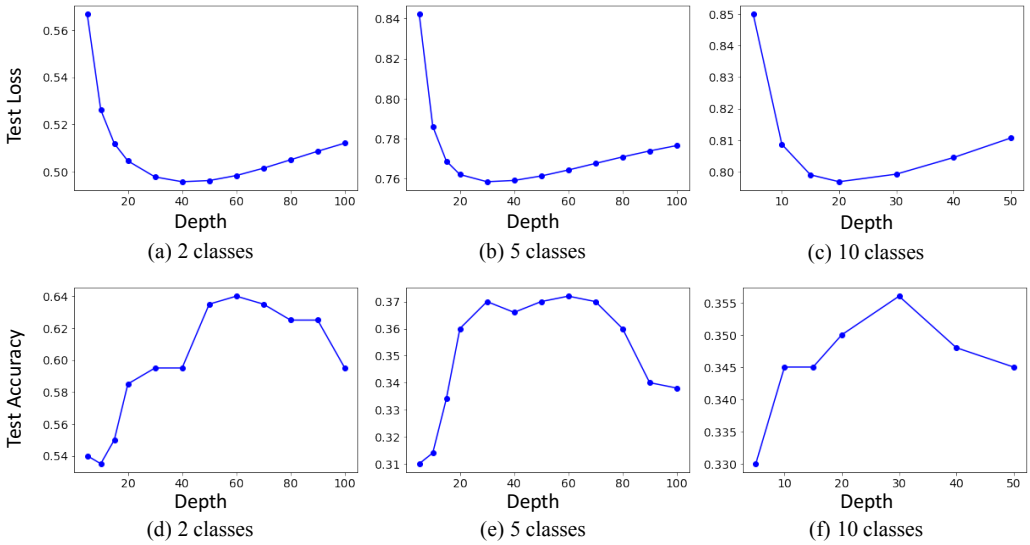
Another form of overparameterization is increasing the kernel size for convolutional filters. In Figure 17, we train ResNet-10 and ResNet-18 of width 32 and varying kernel sizes, and observe that as kernel size increases, test loss increases. This is consistent with our proposed explanation based on expressivity, since increasing kernel size increases representational power independent of depth.



(a) ResNet-10 (b) ResNet-18

(c) ResNet-10 (d) ResNet-18

Figure 17: Train and test losses for width 32 models for increasing kernel size. We observe that test loss increases as kernel size increases.



(a) 2 classes (b) 5 classes (c) 10 classes

(d) 2 classes (e) 5 classes (f) 10 classes

Figure 18: Test loss and test accuracy for the CNTK trained on subsets of Cifar10 with 2, 5 and 10 classes. We observe that generalization improves up to a critical depth, after which it worsens.

D.6 Additional CNTK Experiments

We also train the CNTK on subsets of CIFAR10 of varying number of classes. We use 100 train and 100 test examples per class, and train on both 2 classes (birds and deer), 5 classes (cats, dogs, horses, bids, and deer), and 10 classes (all of CIFAR10). The test losses and accuracies are shown in Figure 18. Again, we see that generalization is unimodel, with test loss decreasing until a critical depth and increasing afterwards, which is in agreement with our main CNTK experiment in Figure 6.

We note that training the CNTK for large depths is computationally prohibitive. The runtime scales quadratically in the number of training samples; furthermore, training the depth 500 CNTK on 1 GPU for 500 train and 500 test samples took approximately 2 days.

D.7 Additional Linear Convolutional Network Experiments

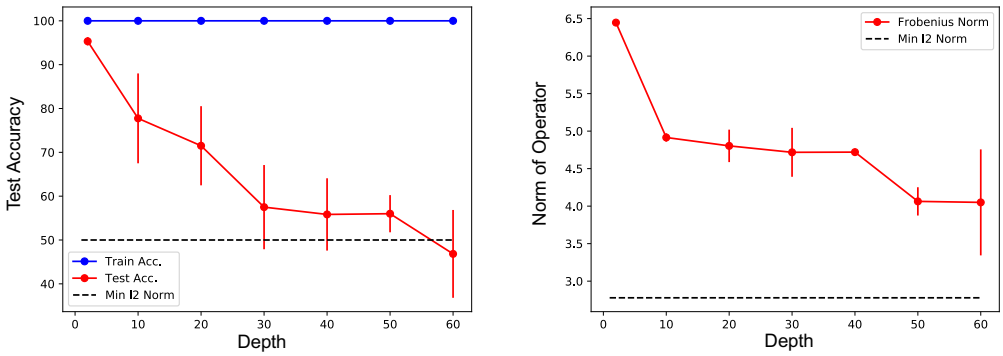


Figure 19: An additional toy example demonstrating that increasing depth in linear convolutional networks leads to operators of decreasing ℓ_2 norm, which manifests as a decrease in test accuracy. Instead of having only 1 training sample of each class, we now sample 4 from each class randomly from the upper left quadrant of a 6×6 square. Our network uses 64 filters per layer, with a kernel size of 3, and zero padding. (a) The training and test performance of linear convolutional networks of varying depth across 3 random seeds. The test accuracy of the minimum ℓ_2 norm solution for this problem is shown as a dashed black line. (c) The ℓ_2 norm of the operator with varying depth. The norm of the minimum ℓ_2 norm solution for this problem is shown as a dashed black line.