On Implicit Filter Level Sparsity in Convolutional Neural Networks

Dushyant Mehta[1,3] Kwang In Kim[2] Christian Theobalt[1,3]
[1]MPI For Informatics [2]UNIST [3]Saarland Informatics Campus

Abstract

We investigate filter level *sparsity that emerges in convolutional neural networks (CNNs) which employ Batch Normalization and ReLU activation, and are trained with adaptive gradient descent techniques and L2 regularization or weight decay. We conduct an extensive experimental study casting our initial findings into hypotheses and conclusions about the mechanisms underlying the emergent filter level sparsity. This study allows new insight into the performance gap obeserved between adapative and non-adaptive gradient descent methods in practice. Further, analysis of the effect of training strategies and hyperparameters on the sparsity leads to practical suggestions in designing CNN training strategies enabling us to explore the tradeoffs between feature selectivity, network capacity, and generalization performance. Lastly, we show that the implicit sparsity can be harnessed for neural network speedup at par or better than explicit sparsification / pruning approaches, with no modifications to the typical training pipeline required.*

1. Introduction

In this work we show that filter[1] level sparsity emerges in certain types of feedforward convolutional neural networks.

In networks which employ Batch Normalization and ReLU activation, after training, certain filters are observed to not activate for any input. Importantly, the sparsity emerges in the presence of non sparsity inducing regularizers such as L2 and weight decay (WD), and vanishes when regularization is removed. We investigate how this sparsity manifests under different hyperparameter settings, and propose an experimentally backed hypothesis for the cause of this emergent sparsity, and the implications of our findings.

We find that adaptive flavours of SGD produce a higher degree of sparsity than (m)SGD, both with L2 regularization and weight decay (WD). Further, L2 regularization results in a higher degree of sparsity with adaptive methods than weight decay. Additionally, we show that a multitude of seemingly unrelated factors such as mini-batch size, network size, and task difficulty impact the extent of sparsity.

These findings are important in light of contemporary attempts to explain the performance gap between (m)SGD and adaptive variants. Any theoretical and practical explorations towards explaining the performance gap between SGD and adaptive variants should account for this inadvertent reduction in network capacity when using adaptive methods, which interplays with both the test accuracy and the generalization gap. Contemporaneous work [27] has also observed that Adam induces filter sparsity in ReLU networks, but lacks a thorough investigation of the causes.

Through a systematic experimental study, we hypothesize that the emergence of sparsity is the direct result of a disproportionate relative influence of the regularizer (L2 or WD) viz a viz the gradients from the primary training objective of ReLU networks. Multiple factors subtly impact the relative influence of the regularizer in previously known and unknown ways, and various hyperparameters and design choices for training neural networks interplay via these factors to impact the extent of emergent sparsity.

We show that understanding the impact of these design choices yields useful and readily controllable sparsity which can be leveraged for considerable neural network speed up, without trading the generalization performance and without requiring any explicit pruning [18, 13] or sparsification [14] steps. The implicit sparsification process can remove 70-80% of the convolutional filters from VGG-16 on CIFAR10/100, far exceeding that for [13], and performs comparable to [14] for VGG-11 on ImageNet.

2. Observing Filter Sparsity in CNNs

We begin with the setup for our initial experiments, and present our primary findings. In subsequent sections we further probe the manifestation of filter sparsity, and present an experimentally backed hypothesis regarding the cause.

2.1. Setup and Preliminaries

Our basic setup is comprised of a 7-layer convolutional network with 2 fully connected layers as shown in Figure 1.

[1]Filter refers to the weights and the nonlinearity associated with a particular feature, acting together as a unit. We use filter and feature interchangeably throughout the document.

This work was funded by the ERC Consolidator Grant 4DRepLy (770784).

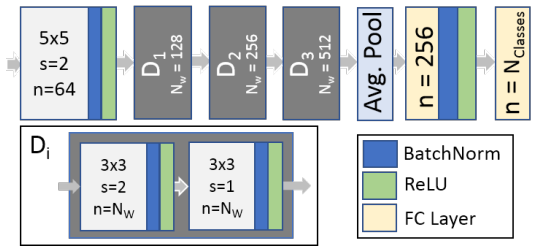

Figure 1. **BasicNet**: Structure of the basic convolution network studied in this paper. We refer to the individual convolution layers as C1-7. The fully connected head shown here is for CIFAR10/100 and ObjectNet3D [26] experiments, and a different fully-connected structure is used for TinyImageNet and ImageNet.

The network structure is inspired by VGG [23], but is more compact. We refer to this network as *BasicNet* in the rest of the document. We use a variety of gradient descent approaches, a mini-batch size of 40, with a method specific base learning rate for 250 epochs, and scale down the learning rate by 10 for an additional 75 epochs. We train on CIFAR10 and CIFAR 100 [12], with normalized images, and random horizontal flips. Xavier initialization [6] is used for the network weights, with the appropriate gain for ReLU. The base learning rates and other hyperparameters are as follows: Adam (1e-3, β_1=0.9, β_2=0.99, ϵ=1e-8), Adadelta (1.0, ρ=0.9, ϵ=1e-6), SGD (0.1, momemtum=0.9), Adagrad (1e-2). Pytorch [21] is used for training, and we study the effect of varying the amount and type of regularization on the extent of sparsity and test error in Table 1.

L2 regularization vs. Weight Decay: We make a distinction between L2 regularization and weight decay. For a parameter θ and regularization hyperparameter $1 > \lambda \geq 0$, weight decay multiplies θ by $(1 - \lambda)$ after the update step based on the gradient from the main objective. While for L2 regularization, $\lambda\theta$ is added to the gradient $\nabla L(\theta)$ from the main objective, and the update step is computed using this sum. See [16] for a detailed discussion.

Quantifying Feature Sparsity: We measure the learned feature sparsity in two ways, by per-feature activation and by per-feature scale. For sparsity by activation, for each feature we apply max pooling to the absolute activations over the entire feature plane, and consider the feature inactive if this value does not exceed 10^{-12} over the entire *training* corpus. For sparsity by scale, we consider the scale γ of the learned affine transform in the Batch Norm [8] layer. Batch normalization uses additional learned scale γ and bias β that casts each normalized convolution output $\hat{x}_i$ to $y_i = \gamma\hat{x}_i + \beta$. We consider a feature inactive if $|\gamma|$ for the feature is less than 10^{-3}. Explicitly zeroing the features thus marked inactive does not affect the test error, which ensures the validity of our chosen thresholds. The thresholds chosen are purposefully conservative, and comparable levels of sparsity are observed for a higher feature activation threshold of 10^{-4}, and a higher $|\gamma|$ threshold of 10^{-2}.

2.2. Primary Findings

Table 1 shows the overall feature sparsity by activation (Act.) and by scale (γ) for BasicNet. Only convolution features are considered. The following are the key observations from the experiments and the questions they raise. These are further discussed in Section 3.

1): The emergent sparsity relies on the strength of L2 regularization or weight decay. **No sparsity is observed in the absence of regularization, with sparsity increasing with increasing L2 or weight decay.** *What does this tell us about the cause of sparsification, and how does the sparsity manifest across layers?*

2): Regardless of the type of regularizer (L2 or weight decay), **adaptive methods (Adam, Adagrad, Adadelta) learn sparser representations than SGD for comparable levels of test error**, with Adam showing the most sparsity and most sensitivity to the L2 regularization parameter amongst the ones studied. Adam with L2 sees about 70% features pruned for CIFAR10, while SGD shows no sparsity for a comparable performance, with a similar trend for CIFAR100, as well as when weight decay is used. *What are the causes for this disparity in sparsity between SGD and adaptive methods?* We will focus on understanding the disparity between SGD and Adam.

3): SGD has comparable levels of sparsity with L2 regularization and with weight decay (for higher regularization values), while **for Adam, L2 shows higher sparsity for comparable performance than weight decay** (70% vs 40% on CIFAR10, 47% vs 3% on CIFAR100). *Why is there a significant difference between the sparsity for Adam with L2 regularization vs weight decay?*

4): The extent of sparsity decreases on moving from the simple 10 class classification problem of CIFAR10 to the comparatively harder 100 class classification problem of CIFAR100. *What does the task dependence of the extent of sparsity tell us about the origin of the sparsity?*

3. A Detailed Look at the Emergent Sparsity

Possible Cause of Sparsity: The analysis of Table 1 in the preceding section shows that the regularizer (L2 or weight decay) is very likely the cause of the sparsity, with differences in the level of sparsity attributable to the particular interaction of L2 regularizer (and lack of interaction of weight decay) with the update mechanism. The differences between adaptive gradient methods (Adam) and SGD can additionally likely be attributed to differences in the nature of the learned representations between the two. That would explain the higher sparsity seen for Adam in the case of weight decay.

Layer-wise Sparsity: To explore the role of the regularizer in the sparsification process, we start with a layer-wise breakdown of sparsity. For each of Adam and SGD, we con-

Table 1. Convolutional filter sparsity in *BasicNet* trained on CIFAR10/100 for different combinations of regularization and gradient descent methods. Shown are the % of non-useful / inactive convolution filters, as measured by activation over training corpus (max act. $< 10^{-12}$) and by the learned BatchNorm scale ($|\gamma| < 10^{-03}$), averaged over 3 runs. The lowest test error per optimizer is highlighted, and sparsity (green) or lack of sparsity (red) for the best and near best configurations indicated via text color. L2: L2 regularization, WD: Weight decay (adjusted with the same scaling schedule as the learning rate schedule). Note that for SGD with momentum, L2 and WD are not equivalent [16].

		CIFAR10			CIFAR100		
		% Sparsity		Test	% Sparsity		Test
	L2	by Act	by γ	Error	by Act	by γ	Error
SGD	2e-03	54	54	30.9	69	69	64.8
	1e-03	27	27	21.8	23	23	47.1
	5e-04	9	9	16.3	4	4	42.1
	2e-04	0	0	13.1	0	0	38.8
	1e-04	0	0	11.8	0	0	37.4
	1e-05	0	0	10.5	0	0	39.0
	0	0	0	11.3	0	0	40.1
Adam [11]	1e-02	82	85	21.3	87	85	69.7
	2e-03	88	86	14.7	82	81	42.7
	1e-03	85	83	13.1	77	76	39.0
	1e-04	71	70	10.5	47	47	36.6
	1e-05	48	48	10.7	5	5	40.6
	1e-06	24	24	10.9	0	0	40.5
	0	3	0	11.0	0	0	40.3
Adadelta [29]	1e-02	97	97	36.8	98	98	84.1
	2e-03	92	92	20.6	89	89	53.2
	1e-03	89	89	16.7	82	82	46.3
	5e-04	82	82	13.6	61	61	39.1
	2e-04	40	40	11.3	3	3	35.4
	1e-04	1	1	10.2	1	1	35.9
Adagrad [5]	2e-02	75	75	11.3	88	88	63.3
	1e-02	65	65	11.2	59	59	37.2
	5e-03	56	56	11.3	24	25	35.9
	1e-03	27	28	11.9	1	1	37.3
	1e-04	0	0	13.6	0	0	42.1
		CIFAR10			CIFAR100		
		% Sparsity		Test	% Sparsity		Test
	WD	by Act	by γ	Error	by Act	by γ	Error
SGD	1e-02	100	100	90.0	100	100	99.0
	1e-03	27	27	21.6	23	23	47.6
	5e-04	8	8	15.8	4	4	41.9
	2e-04	0	0	13.3	0	0	39.4
	1e-04	0	0	12.4	0	0	37.7
Adam [11]	1e-02	100	100	82.3	100	100	98.0
	1e-03	90	90	27.8	81	81	55.3
	5e-04	81	81	18.1	59	59	43.3
	2e-04	60	60	13.4	16	16	37.3
	1e-04	40	40	11.2	3	3	36.2

sider both L2 regularization and weight decay in Table 2 for CIFAR100. The table shows sparsity by scale ($|\gamma| < 10^{-3}$) for each convolution layer. For both optimizer-regularizer pairings we pick the configurations from Table 1 with the lowest test errors that also produce sparse features. For SGD, the extent of sparsity is higher for earlier layers, and decreases for later layers. The trend holds for both L2 and weight decay, from C1-C6. Note that the higher sparsity seen for C7 might be due to its interaction with the fully connected layers that follow. Sparsity for Adam shows a similar decreasing trend from early to middle layers, and increasing sparsity from middle to later layers.

Surprising Similarities to Explicit Feature Sparsification: In the case of Adam, the trend of layerwise sparsity exhibited is similar to that seen in explicit feature sparsification approaches (See Table 8 in [15] for Network Slimming [14]). If we explicitly prune out features meeting the $|\gamma| < 10^{-3}$ sparsity criteria, we still see a relatively high performance on the test set even with 90% of the convolutional parameters pruned. Network Slimming [14] uses explicit sparsity constraints on BatchNorm scales (γ). The similarity in the trend of Adam's emergent layer-wise sparsity to that of explicit scale sparsification motivates us to examine the distribution of the learned scales (γ) and biases (β) of the BatchNorm layer in our network. We consider layer C6, and in Figure 2 show the evolution of the distribution of the learned bias and scales as training progresses on CIFAR100. We consider a low L2 regularization value of 1e-5 and a higher L2 regularization value of 1e-4 for Adam, and also show the same for SGD with L2 regularization of 5e-4. The lower regularization values, which do not induce sparsity, would help shed light at the underlying processes without interference from the sparsification process.

Feature Selectivity Hypothesis: From Figure 2 the differences between the nature of features learned by Adam and SGD become clearer. For zero mean, unit variance BatchNorm outputs $\{\hat{x}_i\}_{i=1}^{N}$ of a particular convolutional kernel, where N is the size of the training corpus, due to the use of ReLU, a gradient is only seen for those datapoints for which $\hat{x}_i > -\beta/\gamma$. Both SGD and Adam (L2: 1e-5) learn positive γs for layer C6, however βs are negative for Adam, while for SGD some of the biases are positive. This implies that all features learned for Adam (L2: 1e-5) in this layer activate for $\leq$ half the activations from the training corpus, while SGD has a significant number of features activate for more than half of the training corpus, i.e., Adam learns more selective features in this layer. Features which activate only for a small subset of the training corpus, and consequently see gradient updates from the main objective less frequently, continue to be acted upon by the regularizer. If the regularization is strong enough (Adam with L2: 1e-4 in Fig. 2), or the gradient updates infrequent enough (feature too selective), the feature may be pruned away entirely. The propensity of later layers to learn more selective fea-

Table 2. Layerwise % filters pruned from BasicNet trained on CIFAR100, based on the $|\gamma| < 10^{-3}$ criteria. Also shown are pre-pruning and post-pruning test error, and the % of *convolutional* parameters pruned. C1-C7 indicate Convolution layer 1-7, and the numbers in parantheses indicate the total number of features per layer. Average of 3 runs. Color and highlighting indicates high and low sparsity for best and near best test errors, as in Table 1. Refer to the supplementary document for the corresponding table for CIFAR10.

					% Sparsity by γ or % Filters Pruned								% Param	
		Train Loss	Test Loss	Test Err	C1 (64)	C2 (128)	C3 (128)	C4 (256)	C5 (256)	C6 (512)	C7 (512)	Total (1856)	Pruned (4649664)	Pruned Test Err.
Adam	L2: 1e-3	1.06	1.41	39.0	56	47	43	68	72	91	85	76	95	39.3
	L2: 1e-4	0.10	1.98	36.6	41	20	9	33	34	67	55	47	74	36.6
	WD: 2e-4	0.34	1.56	37.3	55	20	3	4	2	16	26	16	27	37.3
	WD: 1e-4	0.08	1.76	36.2	38	4	0	0	0	0	5	3	4	36.2
SGD	L2: 1e-3	1.49	1.78	47.1	82	41	33	29	33	6	18	23	34	47.1
	L2: 5e-4	0.89	1.69	42.1	64	3	3	3	2	0	2	4	4	42.1
	WD: 1e-3	1.49	1.79	47.6	82	43	31	28	33	6	17	23	34	47.6
	WD: 5e-4	0.89	1.69	41.9	66	2	1	4	2	0	1	4	4	41.9

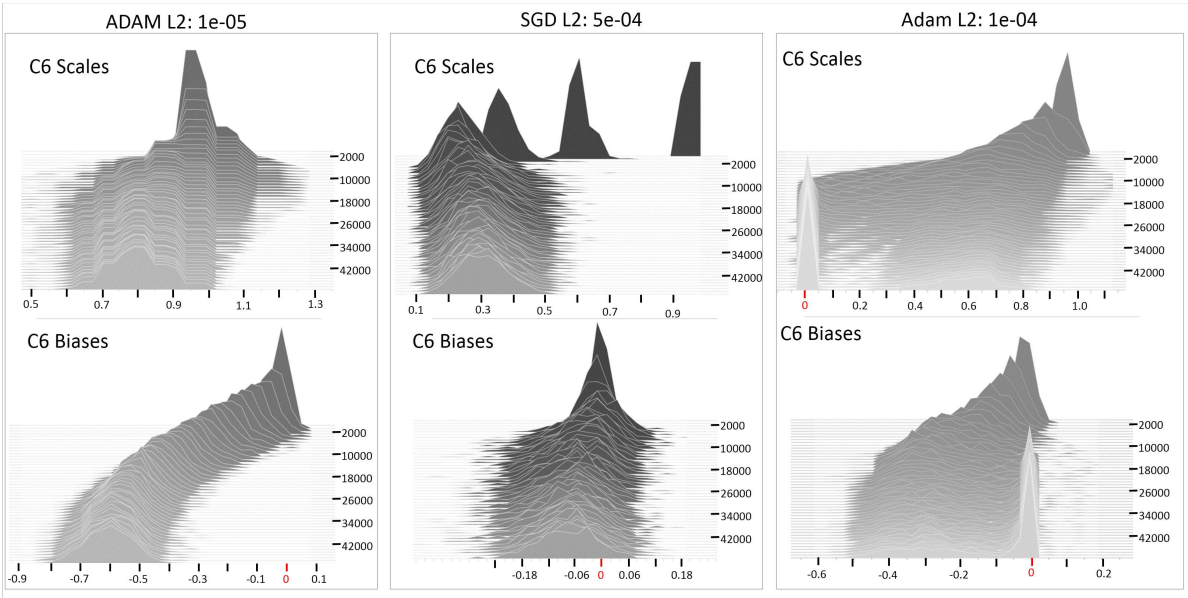

Figure 2. **Emergence of Feature Selectivity with Adam** The evolution of the learned scales (γ, top row) and biases (β, bottom row) for layer C6 of *BasicNet* for Adam and SGD as training progresses. Adam has distinctly negative biases, while SGD sees both positive and negative biases. For positive scale values, as seen for both Adam and SGD, this translates to greater feature selectivity in the case of Adam, which translates to a higher degree of sparsification when stronger regularization is used. Note the similarity of the final scale distribution for Adam L2:1e-4 to the scale distributions shown in Figure 4 in [14]

tures with Adam would explain the higher degree of sparsity seen for later layers as compared to SGD. Understanding the reasons for emergence of higher feature selectivity in Adam than SGD, and verifying if other adaptive gradient descent flavours also exhibit higher feature selectivity remains open for future investigation.

Quantifying Feature Selectivity: Similar to feature sparsity by activation, we apply max pooling to a feature's absolute activations over the entire feature plane. For a particular feature, we consider these pooled activations over the entire training corpus and normalize them by the max of the pooled activations over the entire training corpus. We then consider the percentage of the training corpus for which this normalized pooled value exceeds a threshold of 10^{-3}. We refer to this percentage as the feature's *universality*. A feature's selectivity is then defined as 100-*universality*. Unlike the selectivity metrics employed in literature [19], ours is class agnostic. In Figure 3, we compare the 'universality' of features learned with Adam and SGD per layer on CIFAR100, for both low and higher regularization values. For the low regularization case, we see that in C6 and C7 both Adam and SGD learn selective features, with Adam showing visibly 'more selectivity for C6 (blue bars shifted left). The disproportionately stronger regularization effect of L2 coupled with Adam becomes clearer when moving to a higher regularization value. The selectivity for SGD in C6 remains mostly unaffected, while Adam sees a large fraction (64%) of the features inactivated (0% universality).

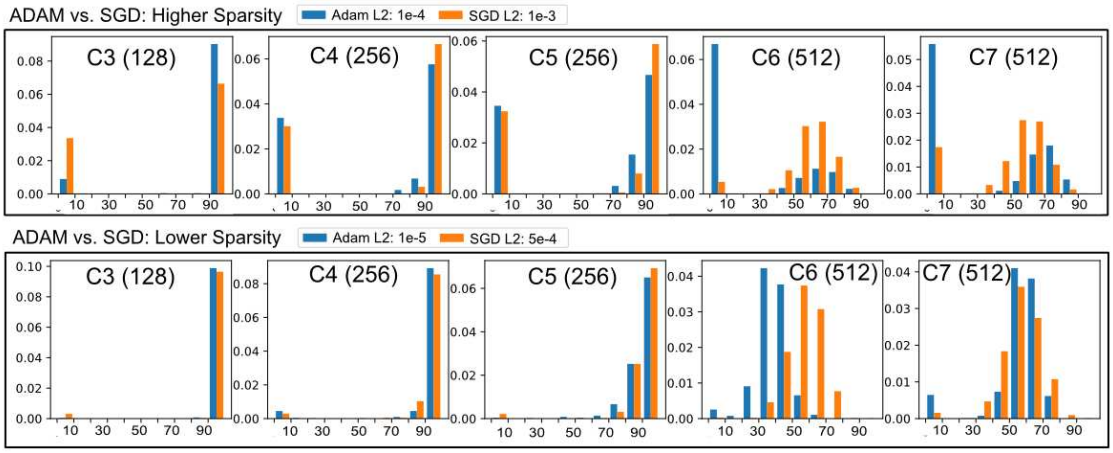

Figure 3. **Layer-wise Feature Selectivity** Feature universality for CIFAR 100, with Adam and SGD. X-axis shows the universality and Y-axis ($\times 10$) shows the fraction of features with that level of universality. For later layers, Adam tends to learn less universal features than SGD, which get pruned by the regularizer. Please be mindful of the differences in Y-axis scales between plots. Refer to the supplementary document for a similar analysis for CIFAR10

Similarly for C7, the selectivity pattern remains the same on moving from lower regularization to higher regularization, but Adam sees more severe feature inactivation.

Interaction of L2 Regularizer with Adam: Next, we consider the role of the L2 regularizer vs. weight decay. We study the behaviour of L2 regularization in the low gradient regime for different optimizers. Figure 4 shows that coupling of L2 regularization with ADAM update equation yields a faster decay than weight decay, or L2 regularization with SGD, even for smaller regularizer values. This is an additional source of regularization disparity between parameters which see frequent updates and those which don't see frequent updates or see lower magnitude gradients. It manifests for certain adaptive gradient descent approaches.

Task 'Difficulty' Dependence: As per the hypothesis developed thus far, as the task becomes more difficult, for a given network capacity, we expect the fraction of features pruned to decrease corresponding to a decrease in selectivity of the learned features [30]. This is indeed observed in Table 1 for *BasicNet* for all gradient descent methods on moving from CIFAR10 to CIFAR100. For Adam with L2 regularization, 70% sparsity on CIFAR10 decreases to 47% on CIFAR 100, and completely vanishes on ImageNet (See Table 5). A similar trend is evident for VGG-16 in Tables 7 and 8. In Figure 5 note the distinct shift towards less selective features in *BasicNet* with increasing task difficulty.

Since the task difficulty cannot be cleanly decoupled from the number of classes, we devise a synthetic experiment based on grayscale renderings of 30 object classes from ObjectNet3D [26]. We construct 2 identical sets of $\approx 50k$ 64×64 pixel renderings, one with a clean background (BG) and the other with a cluttered BG. We train *BasicNet* with a mini-batch size of 40, and see that as expected there is a much higher sparsity (70%) with the clean BG set than with the more difficult cluttered set (57%). See the supplemental document for representative images and a list of the object classes selected.

4. Related Work

Effect of L2 regularization vs. Weight Decay for Adam: Prior work [16] has indicated that Adam with L2 regularization leads to parameters with frequent and/or large magnitude gradients from the main objective being regularized less than the ones which see infrequent and/or small magnitude gradients. Though weight decay is proposed as a supposed fix, we show that there are rather two different aspects to consider. The first is the disparity in effective regularization due to the frequency of updates. Parameters which update less frequently would see more regularization steps per actual update than those which are updated more frequently. This disparity would persist even with weight decay due to Adam's propensity for learning more selective features, as detailed in the preceding section. The second aspect is the additional disparity in regularization for features which see low/infrequent gradient, due to the coupling of L2 regularization with Adam.

Attributes of Generalizable Neural Network Features: Dinh et al. [4] show that the geometry of minima is not invariant to reparameterization, and thus the flatness of the minima may not be indicative of generalization performance [9], or may require other metrics which are invariant to reparameterization. Morcos et al. [19] suggest based on extensive experimental evaluation that good generalization ability is linked to reduced selectivity of learned features. They further suggest that individual selective units do not play a strong role in the overall performance on the task as compared to the less selective ones. They connect the ablation of selective features to the heuristics employed in neural network feature pruning literature which prune fea-

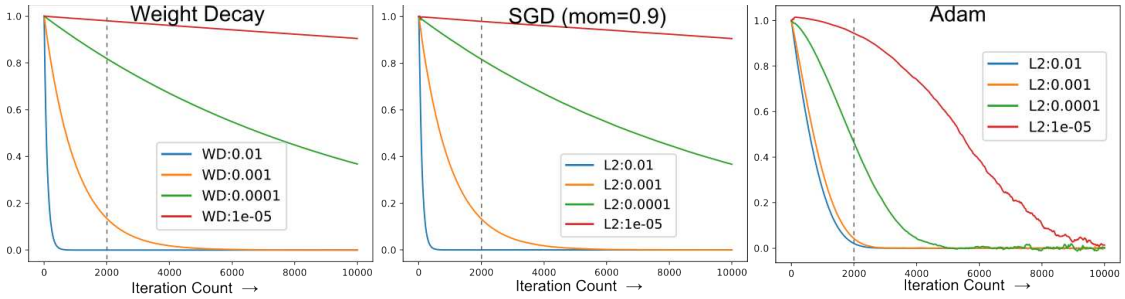

Figure 4. The action of regularization on a scalar value, for a range of regularization values in the presence of simulated low gradients drawn from a mean=0, std=10^{-5} normal distribution. The gradients for the first 100 iterations are drawn from a mean=0, std=10^{-3} normal distribution to emulate a transition into low gradient regime rather than directly starting in a low gradient regime. The learning rate for SGD(momentum=0.9) is 0.1, and the learning rate for ADAM is 1e-3. We show similar plots for other adaptive gradient descent approaches in the supplementary document.

tures whose removal does not impact the overall accuracy significantly [18, 13]. The findings of Zhou et al. [30] concur regarding the link between emergence of feature selectivity and poor generalization performance. They further show that ablation of class specific features does not influence the overall accuracy significantly, however the specific class may suffer significantly. We show that the emergence of selective features in Adam, and the increased propensity for pruning the said selective features when using L2 regularization presents a direct tradeoff between generalization performance and network capacity which practitioners using Adam must be aware of.

Observations on Adaptive Gradient Descent: Several works have noted the poorer generalization performance of adaptive gradient descent approaches over SGD. Keskar et al. [10] propose to leverage the faster initial convergence of ADAM and the better generalization performance of SGD, by switching from ADAM to SGD while training. Reddi et al. [22] point out that exponential moving average of past squared gradients, which is used for all adaptive gradient approaches, is problematic for convergence, particularly with features which see infrequent updates. This short term memory is likely the cause of accelerated pruning of selective features seen for Adam in Figure 4(and other adaptive gradient approaches), and the extent of sparsity observed would be expected to go down with AMSGrad which tracks the long term history of squared gradients.

Feature Pruning/Sparsification: Among the various explicit filter level sparsification heuristics and approaches [13, 24, 7, 25, 18, 20, 14, 28], some [28, 14] make use of the learned scale parameter γ in Batch Norm for enforcing sparsity on the filters. Ye et al. [28] argue that BatchNorm makes feature importance less susceptible to scaling reparameterization, and the learned scale parameters (γ) can be used as indicators of feature importance. We find that Adam with L2 regularization, owing to its implicit pruning of features based on feature selectivity, makes it an attractive alternative to explicit sparsification/pruning approaches. The link between ablation of selective features and explicit feature pruning is also established in prior work [19, 30].

5. Further Experiments

We conduct additional experiments on various datasets and network architectures to show that the intuition developed in the preceding sections generalizes. Further, we provide additional support by analysing the effect of various hyperparameters on the extent of sparsity. We also compare the emergent sparsity for different networks on various datasets to that of explicit sparsification approaches.

Datasets: In addition to CIFAR10 and CIFAR100, we also consider TinyImageNet [2] which is a 200 class subset of ImageNet [3] with images resized to 64×64 pixels. The same training augmentation scheme is used for TinyImageNet as for CIFAR10/100. We also conduct extensive experiments on ImageNet. The images are resized to 256×256 pixels. and random crops of size 224×224 pixels used while training, combined with random horizontal flips. For testing, no augmentation is used, and 1-crop evaluation protocol is followed.

Network Architectures: The convolution structure for *BasicNet* stays the same across tasks, while the fully-connected (*fc*) structure changes across task. We will use '[n]' to indicate an *fc* layer with n nodes. Batch Norm and ReLU are used in between *fc* layers. For CIFAR10/100 we use Global Average Pooling (GAP) after the last convolution layer and the *fc* structure is [256][10]/[256][100], as shown in Figure 1. For TinyImagenet we again use GAP followed by [512][256][200]. On ImageNet we use average pooling with a kernel size of 5 and a stride of 4, followed by [4096][2048][1000]. For VGG-11/16, on CIFAR10/100 we use [512][10]/[512][100]. For TinyImageNet we use [512][256][200], and for ImageNet we use the structure in [23]. For VGG-19, on CIFAR10/100, we use an *fc* structure identical to [14]. Unless explicitly stated, we will be using Adam with L2 regularization of 1e-4, and a batch size of 40. When comparing different batch sizes, we ensure the same number of training iterations.

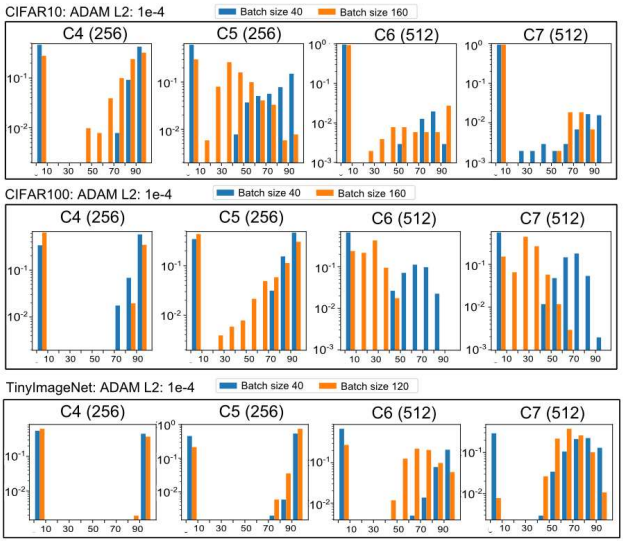

Figure 5. **Feature Selectivity For Different Mini-Batch Sizes for Different Datasets** Feature universality (1 - selectivity) plotted for layers C4-C7 of *BasicNet* for CIFAR10, CIFAR100 and TinyImagenet. Batch sizes of 40/160 considered for CIFAR, and 40/120 for TinyImagenet.

5.1. Analysis of Hyperparameters

Having established in Section 3 (Figures 3 and 2) that with Adam, the emergence of sparsity is correlated with feature selectivity, we investigate the impact of various hyperparameters on the emergent sparsity.

Effect of Mini-Batch Size: Figure 5 shows the extent of feature selectivity for C4-C7 of *BasicNet* on CIFAR and TinyImageNet for different mini-batch sizes. For each dataset, note the apparant increase in selective features with increasing batch size. However, a larger mini-batch size is not promoting feature selectivity, and rather preventing the selective features from being pruned away by providing more frequent updates. This makes the mini-batch size a key knob to control tradeoffs between network capacity (how many features get pruned, which affects the speed and performance) and generalization ability (how many selective features are kept, which can be used to control overfitting). We see across datasets and networks that increasing the mini-batch size leads to a decrease in sparsity (Tables 3, 4, 5, 7, 8, 9, 10).

Network Capacity: Task 'difficulty' is relative to the network's learning capacity. In the preceding section we directly manipulated the task difficulty, and here we consider variations of *BasicNet* in Table 6 to study the complementary effect of network capacity. We indicate the architecture presented in Figure 1 as '64-1x', and consider two variants: '64-0.5x' which has 64 features in the first convolution layer, and half the features of *BasicNet* in the remaining convolution layers, and '32-0.25x' with 32 features in the first channel and a quarter of the features in the remaining layers. The fc-head remains unchanged. We see a consistent decrease in the extent of sparsity with decreasing network width in Table 6. Additionally note the decrease in sparsity in moving from CIFAR10 to CIFAR100.

Table 3. BasicNet sparsity variation on CIFAR10/100 trained with Adam and L2 regularization.

		CIFAR 10				CIFAR 100			
	Batch Size	Train Loss	Test Loss	Test Err	%Spar. by γ	Train Loss	Test Loss	Test Err	%Spar. by γ
L2: 1e-3	20	0.43	0.45	15.2	82	1.62	1.63	45.3	79
	40	0.29	0.41	13.1	83	1.06	1.41	39.0	76
	80	0.18	0.40	12.2	80	0.53	1.48	37.1	67
L2: 1e-4	20	0.17	0.36	11.1	70	0.69	1.39	35.2	57
	40	0.06	0.43	10.5	70	0.10	1.98	36.6	46
	80	0.02	0.50	10.1	66	0.02	2.21	41.1	35
	160	0.01	0.55	10.6	61	0.01	2.32	44.3	29

Table 4. Convolutional filter sparsity for BasicNet trained on TinyImageNet, with different mini-batch sizes.

	Batch Size	Train Loss	Val Loss	Top 1 Val Err.	Top 5 Val Err.	% Spar. by γ
SGD	40	0.02	2.63	45.0	22.7	0
Adam	20	1.05	2.13	47.7	22.8	63
	40	0.16	2.96	48.4	24.7	48
	120	0.01	2.48	48.8	27.4	26

Table 5. Convolutional filter sparsity of BasicNet on ImageNet.

Batch Size	Train Loss	Val Loss	Top 1 Val Err.	Top 5 Val Err.	% Sparsity by γ
64	2.05	1.58	38.0	15.9	0.2
256	1.63	1.35	32.9	12.5	0.0

Table 6. Effect of varying the number of features in BasicNet.

	CIFAR 10				CIFAR 100			
Net Cfg.	Train Loss	Test Loss	Test Err	%Spar. by γ	Train Loss	Test Loss	Test Err	%Spar. by γ
64-1x	0.06	0.43	10.5	70	0.10	1.98	36.6	46
64-0.5x	0.10	0.41	11.0	51	0.11	2.19	39.8	10
32-0.25x	0.22	0.44	13.4	23	0.51	2.05	43.4	0

5.2. Comparison With Explicit Feature Sparsification / Pruning Approaches

For VGG-16, we compare the network trained on CIFAR-10 with Adam using different mini-batch sizes against the handcrafted approach of Li et al. [13]. Similar to tuning the explicit sparsification hyperparameter in [14], the mini-batch size can be varied to find the sparsest representation with an acceptable level of test performance. We see from Table 7 that when trained with a batch size of 160, 83% of the features can be pruned away and leads to a better performance that the 37% of the features pruned for [13]. For VGG-11 on ImageNet (Table 9), by simply varying the mini-batch size from 90 to 60, the number of convolutional

Table 7. Layerwise % Sparsity by γ for VGG-16 on CIFAR10 and 100. Also shown is the handcrafted sparse structure of [13]

		CIFAR 10				CIFAR 100		
Conv	#Conv	Adam, L2:1e-4			Li et	Adam, L2:1e-4		
Layer	Feat.	B: 40	B: 80	B: 160	al.[13]	B: 40	B: 80	B: 160
C1	64	64	0	0	50	49	1	58
C2	64	18	0	0	0	4	0	8
C3	128	50	47	51	0	29	40	54
C4	128	12	5	6	0	0	0	3
C5	256	46	40	36	0	10	5	27
C6	256	71	66	63	0	26	5	7
C7	256	82	80	79	0	44	12	0
C8	512	95	96	96	50	86	74	55
C9	512	97	97	97	50	95	90	94
C10	512	97	97	96	50	96	93	93
C11	512	98	98	98	50	98	97	96
C12	512	99	99	98	50	98	98	99
C13	512	99	99	99	50	98	98	96
%Feat. Pruned		86	84	83	37	76	69	69
Test Err		7.2	7.0	6.5	6.6	29.2	28.1	27.8

Table 8. Sparsity by γ on VGG-16, trained on TinyImageNet, and on ImageNet. Also shown are the pre- and post-pruning top-1/top-5 single crop validation errors. Pruning using $|\gamma| < 10^{-3}$ criteria.

	# Conv	Pre-pruning		Post-pruning	
TinyImageNet	Feat. Pruned	top1	top5	top1	top5
L2: 1e-4, B: 20	3016 (71%)	45.1	21.4	45.1	21.4
L2: 1e-4, B: 40	2571 (61%)	46.7	24.4	46.7	24.4
ImageNet					
L2: 1e-4, B: 40	292	29.93	10.41	29.91	10.41

Table 9. Effect of different mini-batch sizes on sparsity (by γ) in VGG-11, trained on ImageNet. Same network structure employed as [14]. * indicates finetuning after pruning

	# Conv	Pre-pruning		Post-pruning	
	Feat. Pruned	top1	top5	top1	top5
Adam, L2: 1e-4, B: 90	71	30.50	10.65	30.47	10.64
Adam, L2: 1e-4, B: 60	140	31.76	11.53	31.73	11.51
Liu et al. [14] from [15]	85	29.16		31.38*	-

Table 10. Sparsity by γ on VGG-19, trained on CIFAR10/100. Also shown are the post-pruning test error. Compared with explicit sparsification approach of Liu et al. [14]

	CIFAR 10			CIFAR 100		
	Adam, L2:1e-4		Liu et	Adam, L2:1e-4		Liu et
	B: 64	B: 512	al.[14]	B: 64	B: 512	al.[14]
%Feat. Pruned	85	81	70	75	62	50
Test Err	7.1	6.9	6.3	29.9	28.8	26.7

features pruned goes from 71 to 140. This is in the same range as the number of features pruned by the explicit sparsification approach of [13], and gives a comparable top-1 and top-5 validation error. For VGG-19 on CIFAR10 and CIFAR100 (Table 10), we see again that varying the mini-batch size controls the extent of sparsity. For the mini-batch sizes we considered, the extent of sparsity is much higher than that of [14], with consequently slightly worse performance. The mini-batch size or other hyper-parameters can be tweaked to further tradeoff sparsity for accuracy, and reach a comparable sparsity-accuracy point as [14].

6. Discussion and Future Work

Our findings relate to the anecdotally known and poorly understood 'dying ReLU' phenomenon [1], wherein some features in ReLU networks get cut off while training, leading to a reduced effective learning capacity of the network. Ameliorating it with Leaky ReLU [17] is ineffective because it does not address the root cause. *BasicNet* with Leaky ReLU (negative slope of 0.01) on CIFAR-100 only marginally reduces the extent of sparsity in the case of Adam with L2: 10^{-4} (41% feature sparsity vs. 47% with ReLU). Reducing the learning rate of BN parameter γ is much more effective (33% sparsity). See Tables 2, 3 in the supplemental document.

Our work opens several avenues of future investigation. Understanding why features learned with Adam (and perhaps other adaptive methods) are more selective than with (m)SGD can further shed light on the practical differences between adaptive methods and SGD. Also, our insights will lead practitioners to be more aware of the implicit tradeoffs between network capacity and generalization being made below the surface, while changing hyperparameters such as mini-batch size, which are seemingly unrelated to network capacity. Also, we show that Adam with L2 regularization works out of the box for speeding up neural networks and is a strong baseline for future efforts towards filter-sparsification-for-speedup approaches.

7. Conclusion

We show through extensive experiments that the root cause for the emergence of filter level sparsity in CNNs is likely the disproportionate regularization (L2 or weight decay) of the parameters in comparison to the gradient from the primary objective. We identify how various factors influence the extent of sparsity by interacting in subtle ways with the regularization process. We show that adaptive gradient updates play a crucial role in the emergent sparsity (in contrast to SGD), and Adam not only shows a higher degree of sparsity but the extent of sparsity also has a strong dependence on the mini-batch size. We show that this is caused by the propensity of Adam to learn more selective features, and the added acceleration of L2 regularization interacting with the adaptive updates in low gradient regime.

Due to its targeting of selective features, the emergent sparsity can be used to trade off between network capacity, performance and generalization ability as per the task setting, and common hyperparameters such as mini-batch size allow direct control over it. We leverage this finegrained control and show that Adam with L2 regularization can be an attractive alternative to explicit network slimming approaches for speeding up test time performance of CNNs, without any tooling changes to the traditional neural network training pipeline supported by popular frameworks.

References

[1] CS231n convolutional neural networks for visual recognition. http://cs231n.github.io/neural-networks-1/.

[2] Tiny imagenet visual recognition challenge. https://tiny-imagenet.herokuapp.com/.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[4] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *ICML*, 2017.

[5] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. volume 12, pages 2121–2159, 2011.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[7] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, volume 32, 2015.

[9] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

[10] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[13] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017.

[14] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2755–2763. IEEE, 2017.

[15] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *ICLR*, 2019.

[16] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017.

[17] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, page 3, 2013.

[18] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. 2017.

[19] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. 2018.

[20] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, pages 107–115, 1989.

[21] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[22] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *ICLR*, 2018.

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

[24] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. In *BMVC*, 2016.

[25] Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and fisher pruning. In *ICLR*, 2017.

[26] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. Objectnet3d: A large scale database for 3d object recognition. In *European Conference Computer Vision (ECCV)*. 2016.

[27] Atsushi Yaguchi, Taiji Suzuki, Wataru Asano, Shuhei Nitta, Yukinobu Sakata, and Akiyuki Tanizawa. Adam induces implicit weight sparsity in rectifier neural networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 318–325. IEEE, 2018.

[28] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *ICLR*, 2018.

[29] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[30] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*, 2018.