2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA) | 978-1-6654-4337-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICMLA52953.2021.00011

TopKConv: Increased Adversarial Robustness Through Deeper Interpretability

Henry Eigen
Department of EECS
University of Tennessee, Knoxville
Knoxville, TN
heigen@vols.utk.edu

Amir Sadovnik
Department of EECS
University of Tennessee, Knoxville
Knoxville, TN
asadovnik@utk.edu

***Abstract*—Vulnerability to adversarial inputs remains an issue for deep neural networks. Attackers can slightly modify inputs in order to cause adverse behavior in otherwise highly accurate networks. In addition to making these networks less secure for real world applications, this also emphasizes a misalignment between the features the network uses to make decisions and the ones humans use. In this work we propose that more interpretable networks should yield more robust ones since they are able to rely on features that are more understandable to humans. More specifically, we take inspiration from interpretability based approaches to adversarial robustness, and propose a sparsity based defense to counter the impact of overparameterization on adversarial vulnerability. Building off of the work of the Dynamic-K algorithm, which introduces dynamic routing to fully connected layers in order to encourage sparse, interpretable predictions, we propose TopKConv, a novel method of reducing the number of activation channels used to construct each convolutional feature map. The incorporation of TopKConv alongside Dynamic-k results in a significant increase in adversarial accuracy at no cost to benign accuracy. Further, this is achieved with no fine tuning of or adversarial training.**

***Index Terms*—Adversarial defense, sparse training, dynamic routing**

I. Introduction

It has been well established that vulnerability to adversarial inputs–inputs slightly modified to maximize the loss of a network and cause misclassification–is a natural consequence of the algorithms we currently use to train neural networks [1].

In an attempt to combat this vulnerability, significant effort has been dedicated to exploring methods which can reduce the fragility of neural networks. Adversarial training [2], the most successful method to date, still falls short insofar as it fails to bring adversarial accuracy to the level of benign accuracy. Additionally, some argue that this method inevitably forces a trade-off between these two accuracies, with increased robustness effectively compromising benign accuracy [3] [4]. Various other approaches have been explored, either motivated by the significant computational cost of adversarial training, or by its shortcomings in producing robust models.

Another relatively successful line of work explores the use of pruning to create sparser models which have been shown to have superior performance against adversarial attacks. These methods remove certain connections from the network, hopefully leaving the highly predictive ones and thus creating a more robust network. However, as these networks are not trained in a fashion which enforces sparsity, they require fine tuning or adversarial training, and the results are still far from optimal.

In this work we focus mainly on the tie between interpretability and robustness. The fact that small perturbations, which are imperceptible to humans, cause large shifts in the classifiers output reveal a misalignment between the features the network uses to make decisions and the features a human would expect it to use. Since a common definition of a model's interpretability is the degree to which its decisions are understandable to human observers [5], this means that networks which are susceptible to adversarial attacks are not only less secure for real world applications, but are also less trustworthy and less interpretable since they make predictions which a human cannot understand. Following from this, it is clear that interpretability and adversarial fragility are at odds with one another. With this in mind, we draw inspiration from works concerning interpretable models in order to develop methods to train more robust models.

More specifically, in this work we present a novel modification of the traditional convolutional layer, which extends the ideas of the the Dynamic-K algorithm [6], an expectation maximization based dynamic routing schema for fully connected layers. As this method has been shown to increase interpretability in fully connected layers, we formulate a way to use in convolutional layers as well in order to train models which make predictions based on fewer, more meaningful features.

We demonstrate that, without any exposure to adversarial examples, which is necessary for adversarial training, models trained using these dynamic routing methods can outperform those achieved by pruning both in terms of increasing adversarial accuracy, and in minimizing decreases in benign accuracy.

II. Previous Work

A. Adversarial attacks

The first adversarial attack demonstrating the fragility of state of the art networks was the Fast Gradient Sign Method (FGSM) attack [1]. It is a single step of gradient ascent, taking a step in the direction of the gradient of the loss with respect to input, thereby increasing loss.

$$x = x + \epsilon sgn(\nabla_x L(\theta, x, y)) \tag{1}$$

Where x is the input image, y is the ground truth label, θ are the network parameters, ϵ is the size of the step, and $L()$ is the loss function. While FGSM is a relatively weak attack, and one that can be largely defended against through adversarial training, a slight modification to it will yield the Projected Gradient Descent (PGD) attack, a multi-step variant of FGSM which is much more difficult to defend against.

$$x^{t+1} = \Pi_{x+S}(x^t + \alpha sgn(\nabla_x L(\theta, x, y))) \tag{2}$$

Where Π is a clipping operator which subjects x^{t+1} to $\|x^{t+1} - x\| \leq S$. Because it is a multi-step attack, α acts as the step size.

Although a number of stronger attacks have since been created which minimize the perturbation required to fool the network, for the purposes of this paper, we mainly consider robustness against these two attacks.

B. Adversarial defenses

Adversarial training (AT) is arguably the most successful defense against adversarial attacks to date. AT consists of augmenting the dataset used to train a network, typically by include adversarial images [7] [1] [2].

Attacks like FGSM and PGD (referred to as "white box" attacks) rely on a gradient which can be used to ascend the loss curve; without a useful gradient, these attacks would be unable to cause adverse behavior in networks. As a result of this fact, methods of obscuring or obfuscating gradients have been proposed as a means of reducing the efficacy of such attacks. Either by employing non-differentiable operations, or by introducing randomness into the backpropagation, the ultimate result is gradients which are unstable, incorrect, or nonexistent [8]. While variations of these defenses outperformed other methods at the time, they proved ineffective against subsequently developed "Black Box" attacks, those which do not rely on knowledge of a network's weights or its gradient [9].

There have been several attempts to use dynamic routing as as a defense against adversarial robustness, most notably SAP (Stochastic Activation Pruning) [10]. SAP seeks to introduce randomness into the model's routing in order to reduce the effectiveness of adversarial noise by making predictions with a different set of weights than were used to generate the attack gradient. Activations are sampled with replacement from a distribution with a likelihood corresponding to their magnitude. The resulting set of activations is then scaled to closely resemble the expected value of the initial set. This is performed exclusively in the testing stage, and the goal is to maximize the number of activations removed while minimizing the change in expected value of the output.

Several works have suggested a relationship between sparsity and robustness, citing overparameterization as a primary cause of adversarial vulnerability [11] [12]. These and other similar works have shown that sparsity, either achieved by pruning or weight regularization, can lead to greater robustness against adversarial attacks.

III. Methods

A. Dynamic-K

The Dynamic-k algorithm [6] was designed with model interpretability in mind. The authors show that its incorporation into a model's final fully connected layer leads to a simplified learned representation by restricting the number of features at the model's disposal. Further, the authors found that the increase in a model's interpretability as a result of Dynamic-k does not come at the cost of accuracy.

The Dynamic-k algorithm modifies a fully connected layer by constraining the number of activations used to calculate each class probability. This is accomplished by setting each the output of each class to the max-k-sum of its activations, the max-k-sum being the sum of the k largest numbers in a set. Specifically, the max-k-sum is found by applying a binary mask to each channel's activations, setting those included in the top-k set to 1, and the rest to 0. Summing along the masked activations then yields the max-k-sum of each output channel (Algorithm 1). In terms of a channel's output, this is equivalent to setting the weights responsible for those activations to zero. Because the top-k set is determined by the layer's activations, however, that set, and by extension the set of weights used by the layer, varies with the input. This dynamic routing mechanism also impacts backpropagation, as loss is only attributed to the weights used to make a given prediction. Because Dynamic-k is applied during training, the network learns a set of weights which maximizes these output channels.

Algorithm 1: TopKDense

```
for $c$ = 1, 2, ... O do
    $Y_c = max_{R^c} \sum_{i=1}^{m} R_i^c W_i^c X_i \; s.t. |R^c| \leq k$
end
```

B. Dynamic-K Conv

The original Dynamic-k paper was primarily interested in demonstrating how applying this dynamic routing scheme to a network's final fully connected layers improved interpretability. In addition, the authors do suggest that by treating a convolutional layer with a 1×1 kernel as a type of fully connected layer, a pixelwise top-k sum could be added between existing convolutional layers earlier in the network. However, the effect of using this in those layers is not investigated, and it is limited to a 1×1 since it is not clear how this technique would be applied to larger kernels. In this work we propose a novel way to apply channel-wise Dynamic-k routing to convolutional kernels which we call TopKConv. We hypothesize that there is no inherent reason why the ideas of dynamic routing cannot be applied to earlier convolutional layers, and we should expect that this would yield even more robust networks since the convolutional layers will also be able to learn more interpretable features. Our novel method allows us to apply Dynamic-k routing to convolutional kernels of any size at any layer.

There are several features of the convolutional layer which prevent the Dynamic-k algorithm from being directly applied. Unlike the fully connected layers considered by Dynamic-k, weights and activations do not have a one-to-one relationship. As discussed above, Dynamic-k works because setting activations outside of the top-k to zero is equivalent to setting that activation's weights to zero in terms of both the feed forward and backpropagation stages. When a single weight yields multiple activations, as is the case in convolutional layers (see Figure 1), the weight is treated as non-zero during backpropagation if any of its activations are non-zero. This would violate the constraints on the number of weights updated at any given time, therefore undermining the effectiveness of Dynamic-k. In order to avoid this issue of shared weights, we use depthwise convolution as the basis for our method [13]. In depthwise convolution, each filter in a kernel is convolved along a single corresponding input channel. The resulting channels are then stacked and returned. This is typically followed by a $1x1$ convolution which reduces the stack to a single output channel. When used in concert, the combination is called a depthwise separable convolution. What we are interested in, however, is the depthwise convolution's one-to-one relationship between kernel filters and stack channels. We do without the $1x1$ convolution, instead treating the stacked output as our activations and perform a top-k-sum to reduce them to a single output channel. This allows us to avoid the issue of shared weights.

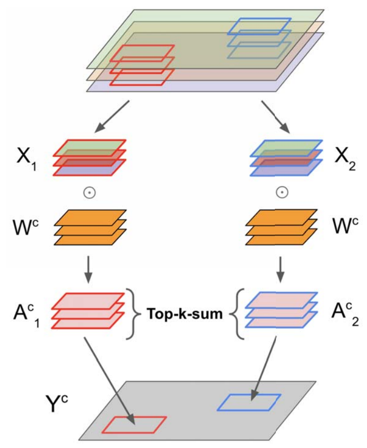

Fig. 1. In the standard convolutional layer shown above, a single weight will yield multiple activations, as the input at each position of the sliding convolution window is independently multiplied by a kernel and summed. In the image above, the top-k activation sets of both A_1^c and A_2^c will impact the gradient w.r.t. W^c ultimately allowing more than k filters to be updated. This is incompatible with the orinigal idea of Dynamic-k.

The depthwise convolution approach introduces a new problem. Since the activations of depthwise convolution have height and width dimensions, the max-k-sum of activations cannot be directly solved, meaning that the process for selecting the top-k channels must be modified to accommodate this. We propose using channels' max as the criteria for top-k selection (Figure 2). Our reasons for using the max, as opposed to other dimensionality reduction operations, is discussed in next section.

We implement our proposed TopKConv layer as the composition of two function. The first function, f, is the depthwise convolution function. Letting W be the layer's weights:

$$f : \underset{(h\times w\times n)}{X}, \underset{(h\times w\times n\times o)}{W} \to \underset{(h\times w\times n\times o)}{A}$$

Where h and w are height and width respectively, n is the number of input channels, and o is the number of output channels. Because we will be summing along the n axis of A to find the layer's output, we consider the output of f to be our activations. The second function, M^k, finds performs a top-k summation on the activations.

$$M^k : \underset{(h\times w\times n\times o)}{A} \to \underset{(h\times w\times o)}{Y}$$

In particular, for each output channel c, M^k finds and applies applies a binary mask R^c which maximizes the sum along A's n axis. This process is explicitly defined in Algorithm 2.

Algorithm 2: TopKConv: M^k

```
for c = 1, 2, ... O do
    $R_c \leftarrow \arg\max_{R_c} \sum_{i=1}^{n} R_i^c \max(A_i^c)\ s.t. |R_c| \leq k$
    $Y_c \leftarrow \sum_{i=1}^{n} R_i^c A_i^c$
end
```

Note that absent the mask R, M^k is simply a summation of A. In this case, Y is equivalent to the output of a standard convolutional layer given the same X and W.

$$Y \leftarrow M^k(f(X,W)) = \sum_{i=1}^{n} f(X,W)_{hwio}$$

It is as a result of this fact that we are able to represent our top-k convolutional layer as the top-k summation of depthwise convolution. The entire TopKConv pipeline is shown in Fig. 2.

C. top-k criteria

Because the activation channels are multidimensional, a reduction operation must be used to rank them in order to find a top-k-sum. Although choosing the top-k-sum by summing across each channel would be truest to Dynamic-K's EM design, we chose to select channels based on each

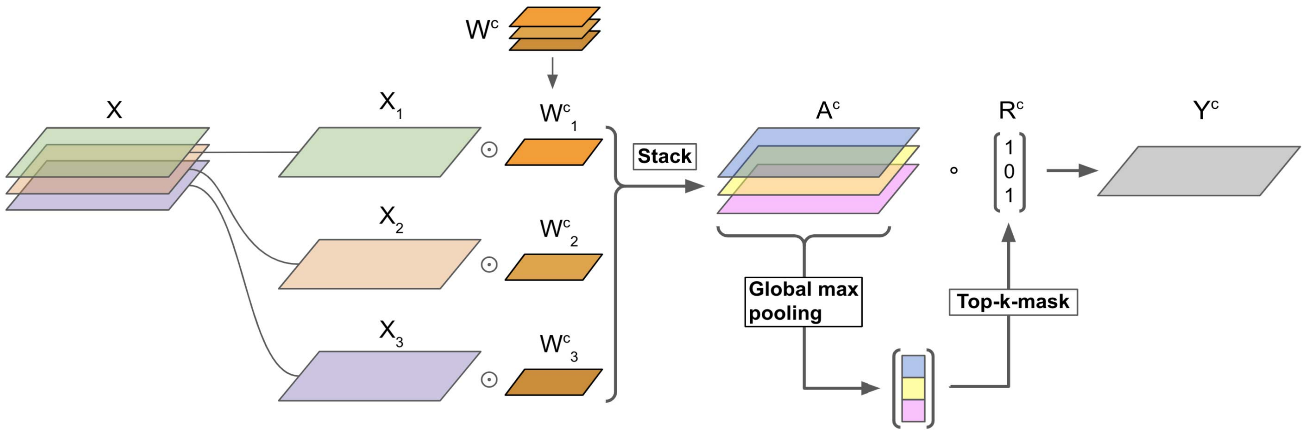

Fig. 2. Our proposed TopKConv method starts with depthwise convolution, as it allows us to examine each feature map independently to select the ones with the highest activation. The A^c stack in the figure above, which has the same number of channels as the input, represents the output of the depthwise convolution, or what we consider to be our feature activations. Because this convolution output results in two dimensional activations for each feature, a top-k mask (R as shown in Algorithm 1) cannot be directly derived. Instead, we create the top-k binary mask, R^c, by first performing global max pooling so that we can have a single value for each feature. Finally, this map is multiplied by each corresponding channel to yield the final convolution result.

channel's max. This decision was motivated by intuition, and is supported by empirical results (as shown in Table III).

If we are considering channel's to be representative of features, sum maximization doesn't make sense as a goal. Applying reduce-sum to channels means channels with large areas of activation are selected in the top-k-sum. While large areas of activation makes sense for certain types of features like texture, many high level features, like an ear, should only activate the specific regions in which they occur. Because a max-reduce is more receptive to channels with areas of high activation, rather than broad activation, it makes for a better reduction operation.

Additionally, since top-k routing is used during training, the model learns features which will be selected in the top-k sum. This means that max-reduce encourages the model not to learn features which highly activate in specific circumstances (Figure 3). Moreover, selecting based on the maximum value seems to encourage a more diverse selection of weights. While small variations in an activation with a large sum are unlikely to have a meaningful impact on its sum, the presence of a feature in a small region of an image will have great impact on a channel's relative max ranking.

Our tests show that a model trained with the sum-reduce top-k slightly under-performs a model trained using max-reduce in terms of accuracy on clean images. In the case of adversarial accuracy, it not only under-performs, but actually worsens baseline accuracy.

D. Synapses vs. Neurons

Any references or comparisons to pruning up to this point have been in reference to synapse pruning, that is, pruning the connections between weights. Although synapse pruning is by far the most common approach, there is an alternative: neuron pruning. These two classifications of pruning methods, synapse and neuron pruning, refer to which values are removed from calculation. To prune synapses, or the connections between weights, is to remove values from the activations prior to summation, affecting the value of an output channel. Pruning neurons refers to removing a neuron, or kernel in this context, entirely, resulting in an output channel value of zero.

Dynamic-k, as well as our proposed TopKConv method, carry out a form of dynamic synapse pruning. We also consider the possibility of adapting our TopKConv method to a neuron pruning framework. TKC-neuron, as we'll refer to this method, is considerably simpler than it's synapse pruning counterpart. Just as in TopKConv, routing decisions must be a function of activations, as weights are static across inputs. Instead of selecting individual activations to preserve, TKC-neuron selects the top-k sums of activations, each of which represent the output of a single kernel. This then has the effect of dynamically pruning entire kernels responsible for low channels. Here too, we use channels' max as the criteria for top-k selection. Note that this is not a top-k summation as has been the case in previous methods. Our desire to adapt a neuron pruning approach is primarily motivated by computational cost considerations, as the top-k selection takes place only a single time on the layer's output, rather than once per output channel.

IV. Experiments and Analysis

A. Setup

Since our work aims to highlight the benefits of updating only a small number of parameters in a given training step, we wish to use a large network, as overparameterization becomes more of a concern with more features. In addition, small images like ones from the cifar [14] database which are 32×32 result in feature maps reduced to only a few pixels wide in the later convolutional layers, which make the use of top-k selection less meaningful. Therefore, we use a dataset with large images which would justify a larger network.

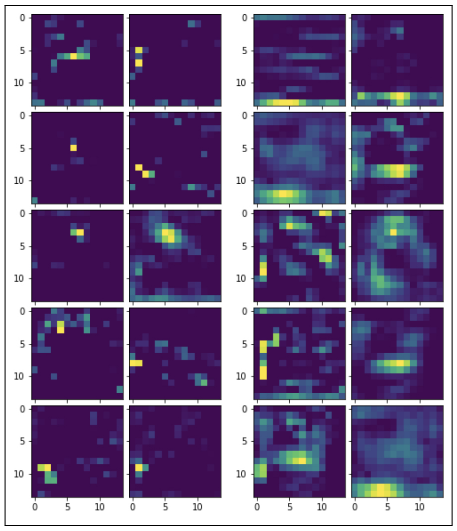

Fig. 3. Feature maps output by TopKConv layers using different top-k criteria. Feature maps on the left were output by a layer using channel-max based selections. Feature maps on the right were output by a layer using channel-sum based selections. As can be seen, when using max criteria, TopKConv has greater success in both learning particular features and locating them with accuracy.

More specifically, we use the Cinic-10 dataset [15]. Cinic-10 contains the same classes as the Cifar-10 dataset, however it draws the images themselves from the Imagenet dataset. This yields a 10 class dataset of images with size 224x224, which allows us to use large images while reducing the training and testing as compared to using the 1000 classes in the Imagenet dataset. For our network we used a slight variant of VGG16 [16]: given our 10 class dataset, we used only two fully connected layers (512 neurons and 10 neurons). Because of the size of our network and the number of experiments, we sped up training by initializing the weights of networks to those of a pretrained network.

We primarily consider robustness against the FGSM and PGD attacks when evaluating robustness. These attacks provide a good point of comparison as they are frequently used to evaluate methods in other papers. Additionally, the defenses we are comparing against are vulnerable enough to these relatively ”weak” attacks, that stronger attacks aren't necessary to demonstrate improved performance. For both the FGSM and PGD attacks, we use an L_∞ $\epsilon = 8$. This means that a given pixel can be changed by a value of up to 8. The PGD is a multistep attack, so it has the additional parameters, iterations = 16 and step-size = 0.75. Note that iterations * step-size $>$ ϵ. Should the change in a pixel exceed ϵ, PGD projects the attack noise to the nearest point on the L_∞ ball.

B. Baselines

As we focus in this work on learning more robust features without the need for adversarial training, we wish to compare how Dynamic-k's performed in relation to other defenses which were not exposed to adversarial data. In addition to comparing to regularly trained networks, we also examined the performance of other sparsity based approaches to robustness. More specifically, we selected LWM [17] as the pruning method against which to compare. This is motivated by LWM's frequent appearance as a baseline in works proposing new pruning methods. Specifically, we used a polynomial decay pruning schedule [18] which gradually increases the sparsity up to a set value over the course of several epochs of fine tuning.

We also compare against SAP, using SAP-80, which performs best according to the original paper. As mentioned in the Previous Works section, SAP samples activations randomly. While both Dynamic-k and SAP perform dynamic activation pruning, they fundamentally differ in several ways. For one, SAP is not used in training, only in testing. This means that it does not benefit from the altered backpropagation. Because Dynamic-k is incorporated into the training loop, it ultimately learns to produce a different set of activations from which to choose. A model, in theory, has the ability to learn more robust features as a result of Dynamic-k modified gradients, and as we show in our results there is evidence to suggest it does. SAP, on the other hand, only modifies how the model makes predictions in the testing phase, and as a result, is limited as a defense to moderate gradient masking. A similar point is made about pruning in the Discussion section.

C. Dynamic-k Performance

We first test the robustness of our algorithm when using the Dynamic-k routing algorithm in fully connected layers, as was discussed in the original paper. Although this is the same method as the one used in that original paper, it was not tested there for adversarial robustness, and therefore we present our results here. Our penultimate Dynamic-k layer has $k = 250$ with an input dimension of (25088 x 1), meaning 1% of activations were kept. Our final Dynamic-k layer has $k = 10$ with an input dimension of (512 x 1) meaning 2% of activations were kept. A hyperparameter search determined these to be the best k values.

Our results are shown in Table I and clearly show that using Dynamic-k outperforms the baselines. It is interesting to note that the increase in robustness as the result of the inclusion of Dynamic-k did not come at the cost of the model's benign accuracy. Dynamic-k's ability to impose sparsity constraints without hurting the model's performance is a point emphasized by its authors. It is even more notable in this context, as many adversarial defenses create a trade-off between benign and adversarial accuracy.

We found that by applying Dynamic-k to the penultimate layer in addition to the final layer, we were able to further improve the performance. As results from the pruning defense suggested that the application of Dynamic-k even further back could be beneficial (see Table II), and since our model had only two fully connected layers, Dynamic-k's restriction to fully connected layers motivated the development of our TopKConv method.

Attack	Baseline	SAP	Dropout	Pruning	Pruning + LogitAug	Dynamic-k
None	0.895	0.896	0.858	0.884	0.884	0.921
FGSM	0.192	0.207	0.233	0.340	0.369	0.388
PGD	0.084	–	0.108	0.223	0.245	0.324

TABLE I

DYNAMIC-K COMFORTABLY OUTPERFORMS OTHER METHODS IN TERMS OF ADVERSARIAL ROBUSTNESS. IT IS INTERESTING TO NOTE ACCURACY AGAINST CLEAN IMAGES IN THE "NONE" ROW. WHILE OTHER METHODS SAW SIMILAR OR SLIGHTLY DECREASED ACCURACY IN THIS CONTEXT, DYNAMIC-K ACTUALLY PERFORMED BETTER THAN THE BASELINE NETWORK WHICH EMPLOYED NO ADVERSARIAL DEFENSE. THIS IS CONSISTENT WITH RESULTS PUBLISHED BY DYNAMIC-K'S AUTHORS WHO NOTED THAT IT HAS A TENDENCY TO CONVERGE MORE QUICKLY, OFTEN OUTPERFORMING THE BASELINE. ALSO INTERESTING IS THE RELATIVELY SMALL DIFFERENCE IN DYNAMIC-K'S PERFORMANCE AGAINST FGSM AND PGD, COMPARED TO THE DROP SEEN BY OTHER METHODS. PGD PERFORMANCE IS PERHAPS THE MORE IMPORTANT METRIC HERE, AS ROBUSTNESS AGAINST PGD IS MORE INDICATIVE OF GENERAL ROBUSTNESS ACROSS ALL ATTACKS.

Method	Layers Affected	Adv Acc
Pruning	Final layer only	15.2
	Final two layers	16.1
	All layers	19.8
Dynamic-K	Final layer only	27.8
	Final two layers	31.1

TABLE II

D. Top-k-Conv Performance

In evaluating the performance of our TopKConv method, we are primarily interested to see the degree to which it can enhance the Dynamic-k defense, and so we use the Dynamic-k network (as described in Sec. IV-C as the baseline.

We show results for implementations of TopKConv using both max and sum as top-k selection criteria, denoted TKC-max and TKC-sum respectively. We also consider the neuron pruning adaptation, TKC-neuron. In all experiments, variations of the TopKConv are used in addition to Dynamic-k.

In all cases shown in Table 2, TopKConv was used in place of the standard convolutional layer in the two layers immediately preceding the fully connected layers, and had k values corresponding to 10% of activations. While much stricter sparsity constraints could be applied to the final FC layers, we found that values lower than 10% in the convolutional layers created difficulty in training. Additionally, we found that the use of TopKConv layers in earlier convolutional blocks could prevent the model from converging altogether.

It is interesting to note that the addition of TKC-sum layers actually lowered accuracy against PGD attacks. This is inline with our intuition concerning top-k criteria.

V. DISCUSSION

A. Comparison to pruning

A key difference in a top-k trained model and a pruned model is the top-k model's incorporation of the sparsity constraints into the training loop. Generally speaking, pruning occurs after the model has converged, and it takes the form of a two step iterative process of removing weights and fine tuning the resulting network. During fine tuning, the values of convolutional layers' kernels are frozen, and only the model's final fully connected layers are updated. Because pruning doesn't take place until a model has converged, the model's weights prior to pruning are ideal. The goal of pruning is then to find the subset of weights whose behavior most closely resembles that of the full set.

In a top-k network, the activation pruning takes place during training, meaning that the model learns features which work well given the constraints. When dynamic routing is applied to a model trained without it, the model's accuracy is compromised considerably. Similarly, a model trained using dynamic routing performs poorly when it is removed during the test phase. By restricting the model to the use of a small number of features at a given layer, the model will learn fewer, more predictive features. Pruning yields a sparse approximation of a model's non-sparse representation, while top-k training yields a sparse representation to begin with.

Additionally, dynamically pruning weights via their activations gives the routing more flexibility. When pruning, a weight's value to the model is a function of its performance on the entire distribution of training data. Dynamically pruning allows an equally small number of weights to be used in prediction, while still allowing a larger set of features to be available, by choosing the sparse set most useful to each input.

B. Why top-k routing works

Despite significant effort, adversarial vulnerability is still poorly understood, leaving a variety of theories, some in agreement and others in conflict. Although none have provided a conclusive explanation, it helps to have some conceptual framework for understanding adversarial vulnerability to guide one's approach to fixing it. Our intuition about what allows for adversarial attacks, and why the methods presented above can help thwart them, is largely shaped by the work of Ilyas et al [19].

This work presents the issue of adversariality as fundamentally an issue of interpretability. They theorize that the ability to change a model's prediction through the addition of adversarial noise considered meaningless to humans isn't evidence of a failure of the models to generalize to the training data, but rather a disconnect in which features are meaningful to humans as opposed to vision models.

They demonstrate that, not only do models learn a variety of non-robust features, features which are statistically significant but largely meaningless to a human, but that the non-robust

Attack	Baseline	Dynamic-k	TKC-neuron	TKC-max	TKC-sum
None	0.895	0.921	0.912	0.903	0.895
FGSM	0.192	0.388	0.398	0.401	0.398
PGD	0.084	0.324	0.385	0.421	0.296

TABLE III
BOTH TKC-NEURON AND TKC-MAX CONTRIBUTED TO FURTHER INCREASES IN ROBUSTNESS WHEN USED IN ADDITION TO DYNAMIC-K. ALTHOUGH TKC-NEURON'S PERFORMANCE FELL SHORT OF TKC-MAX'S, IT COULD BE FAVORABLE IN CERTAIN CIRCUMSTANCES GIVEN ITS CONSIDERABLY FASTER RUNTIME.

features present in a dataset are sufficient as training data for a model to achieve high accuracy on the full dataset.

Interpretability methods are useful insofar as they make a model's behavior interpretable to a human. As these non-robust features are, by definition, meaningless to a human, effective interpretability methods would ignore them, despite their pronounced impact on the model's ability to learn and predict the dataset. Paradoxically, this means that current interpretability methods don't necessarily truly capture how the model makes predictions, meaning that they are not as good of methods as we expect them to be.

Learning a representation which is not reliant on non-robust features is therefore a prerequisite for a truly interpretable model. Their paper makes a this point and states, "producing human-meaningful explanations that remain faithful to underlying models cannot be pursued independently from the training of the models themselves" (pg 2).

They suggest training such a model by constructing a dataset free from the presence of non-robust features as shown in the distribution below, where D is the original dataset, D_R is the robust dataset, F is the set of penultimate activations corresponding to robust features.

$$\mathbb{E}_{(x,y)\sim D_R}[f(x)\cdot y] = \begin{cases} \mathbb{E}_{(x,y)\sim D}[f(x)\cdot y] & iff \in F_c \\ 0 & otherwise \end{cases}$$

If it were the case that robust features were more likely to activate highly in relation to non robust features, these top-k methods could ultimately learn activations like those in D_R above. This would be accomplished without creating a new dataset, however. Since only the weights corresponding to the top-k activations are updated during backpropagation, only features which result in the highest activations are learned.

In future works, we hope to attain a more rigorous understanding of how non-robust features are handled by these top-k routing methods and draw a stronger comparison.

VI. CONCLUSION

In this paper, we propose TopKConv, a Dynamic-k inspired convolutional layer. Additionally, we demonstrate that models trained using top-k dynamic routing are able to learn more robust representations without suffering a loss of benign accuracy. Further, they are able to do so without any fine tuning. In future works, we hope to better understand why these sparsity constraints result in increased robustness.

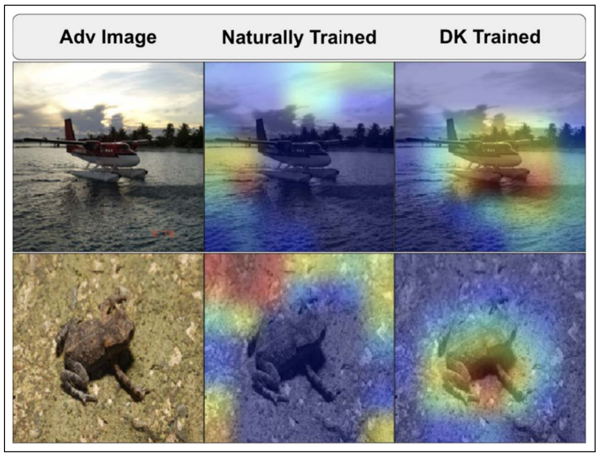

Fig. 4. GradCams [20] for models' predictions on adversarial images. Here we show specifically the class activation maps for the correct class (airplane, frog) where the network classified the adversarial image incorrectly. The center column shows the regions of interest to a naturally trained model. The right column shows the regions of interest to a model trained with Dynamic-k. Illyas et al. demonstrated that a model will make predictions based on non-robust, easily flipped features even in the presence of robust features. One possible interpretation of these attention maps is that the baseline model has learned non-robust features which can be simulated in the background by adversarial noise, while the DK trained model has not learned non-robust features, and will therefore make predictions based on the robust features present in subject of the image.

REFERENCES

[1] J. Shlens I. Goodfellow and C. Szegedy, "Explaining and harnessing adversarial examples," *International Conference on Learning Representations*, 2015.
[2] L. Schmidt D. Tsipras A. Madry, A. Makelov and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *International Conference on Learning Representations*, 2018.
[3] J. Jiao E. Xing L. E. Ghaoui M. Jordan H. Zhang, Y. Yu, "Theoretically principled trade-off between robustness and accuracy.," vol. 97, pp. 7472–7482, 2019.
[4] L. Engstrom A. Turner A. Madry D. Tsipras, S. Santurkar, "Robustness may be at odds with accuracy.," 2019.
[5] C. Cotton O. Biran, "Explanation and justification in machine learning: A survey," *Workshop on Explainable Artificial Intelligence*, p. 8–13, 2017.
[6] S. Vikas S.Yiyou, R.Sathya, "Adaptive activation thresholding: Dynamic routing type behavior for interpretability in convolutional neural networks," *ICCV.2019.00504*, pp. 4937–4946, 2019.
[7] N. Papernot D. Boneh P. McDaniel F. Tramer, A. Kurakin, "Ensemble adversarial training: Attacks and defenses," *In International Conference on Learning Representations*, 2018.
[8] Carlini N. Wagner D. Athalye, A., "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples.," *International Conference on Machine Learning*, pp. 274–283, 2018.
[9] I. Goodfellow S. Jha Z. Celik A. Swami N. Papernot, P. McDaniel, "Practical black-box attacks against machine learning.," *In Proceedings*

of the 2017 ACM on Asia conference on computer and communications security, pp. 506–519, 2017.

[10] J. D. Bernstein J. Kossaifi A. Khanna Z. C. Lipton G. S. Dhillon, K. Azizzadenesheli and A. Anandkumar, “Stochastic activation pruning for robust adversarial defense.,” *In International Conference on Learning Representations*, 2018.

[11] H. Yang C. Yu Z. Wang J. Liu S. Gui, H. Wang, “Model compression with adversarial robustness: A unified optimization framework,” 2019.

[12] C. Zhang Y. Chen Y. Guo, C. Zhang, “Sparse dnns with improved adversarial robustness.,” *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, p. 240–249, 2018.

[13] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017.

[14] G. Hinton A. Krizhevsky, V. Nair, “Cifar-10 (canadian institute for advanced research),” .

[15] A. Antoniou A. Storkey L. Darlow, E. Crowley, “Cinic-10 is not imagenet or cifar-10,” *arXiv preprint*, vol. arXiv:1810.03505, 2018.

[16] A. Zisserman K. Simonyan, “Very deep convolutional networks for large-scale image recognition,” *International Conference on Learning Representations*, 2015.

[17] J. Tran S. Han, J. Pool and W. Dally., “Learning both weights and connections for efficient neural networks.,” *In Advances in neural information processing systems*, p. 1135–1143, 2015.

[18] Michael Zhu and Suyog Gupta, “To prune, or not to prune: exploring the efficacy of pruning for model compression,” *preprint arXiv:1710.01878*, 2017.

[19] D. Tsipras L. Engstrom B. Tran A. Madry A. Ilyas, S. Santurkar, “Adversarial examples are not bugs, they are features,” *Advances in Neural Information Processing Systems*, pp. 125–136, 2019.

[20] A. Das R. Vedantam D. Parikh D. Batra R. Selvaraju, M. Cogswell, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” *International Journal of Computer Vision*, 2019.