Adaptive Activation Thresholding: Dynamic Routing Type Behavior for Interpretability in Convolutional Neural Networks

Yiyou Sun[†] Sathya N. Ravi[‡*] Vikas Singh[†]
[†]University of Wisconsin-Madison [‡]University of Illinois at Chicago
sunyiyou@cs.wisc.edu sathya@uic.edu vsingh@biostat.wisc.edu

Abstract

There is growing interest in strategies that can help us understand or interpret neural networks – that is, not merely provide a prediction, but also offer additional context explaining why and how. While many current methods offer tools to perform this analysis for a given (trained) network post-hoc, recent results (especially on capsule networks) suggest that when classes map to a few high level "concepts" in the preceding layers of the network, the behavior of the network is easier to interpret or explain. Such training may be accomplished via dynamic/EM routing where the network "routes" for individual classes (or subsets of images) are dynamic and involve few nodes even if the full network may not be sparse. In this paper, we show how a simple modification of the SGD scheme can help provide dynamic/EM routing type behavior in convolutional neural networks. Through extensive experiments, we evaluate the effect of this idea for interpretability where we obtain promising results, while also showing that no compromise in attainable accuracy is involved. Further, we show that the minor modification is seemingly ad-hoc, the new algorithm can be analyzed by an approximate method which provably matches known rates for SGD. Code is available at: https://github.com/sunyiyou/dynamic-k-activation.

1. Introduction

Machine learning and computer vision methods are now becoming closely interwined with our lives. There is consensus that this trend will clearly continue, but the need for human comprehension (i.e., **interpretability**) of what/why a model predicts will become a pressing issue – and potentially a key constraint that may limit broader adoption across in a number of different disciplines. Clearly, if a human can comprehend the model, then its utility increases [12, 10, 29, 16, 34]. But perhaps in other settings, this need is also driven by regulatory requirements, robustness or societal issues. Why did the system suggest that a person be selected for secondary screening at the airport? Why is a biopsy being requested based on a radiographic image? Why is the autonomous vehicle inadvertently deciding to change lanes when there is no apparent obstruction? Models that are easy to comprehend broadly fall under the umbrella of *interpretable* models [8], and recent work shows that apart from use in vision tasks such as visual question answering [2], interpretability also facilitates the development of fair/causal models [8].

At a high level, interpreting and understanding deep models in vision can be roughly partitioned into one or more of three broad categories. The first line of work seeks to generate **iconic examples** of what the network has learned. This idea relates to finding the pre-image of a trained model. Deep image priors [31], an inverse network [17] and Plug and play networks [21] are some examples of this line of work. But instead of interpreting the network as a whole, one may also seek **attribution**: in other words, what parts of an image are salient for a network and/or for individual examples. This can be approached by generating informative heatmaps such as CAM [36] and grad-CAM [27], or through back-propagation conditioned on the final prediction [29] and layer-wise relevance propagation [3]. *Network Dissection* [4] and variants can be used to quantify the interpretability of hidden units through segmentation, helping us identify what may be thought of as causal features. This work is also loosely related to the idea behind the third category of methods based on **semantic identification** which assumes that concepts (or neurons) activate only on a subset of images. This behavior induces equivalence classes in the image set. For example, the Net2Vec paper [9] seeks to associate concepts between concepts and filters in a given architecture.

Interpretability, explainability, post-hoc analysis: Most techniques that focus on interpretability approach the problem in a post hoc manner. This means that interpretability analysis is carried out on an already trained network. Recently, [24] investigated some of the pros/cons of

*Worked was performed while SNR was a dissertator at UW-Madison.

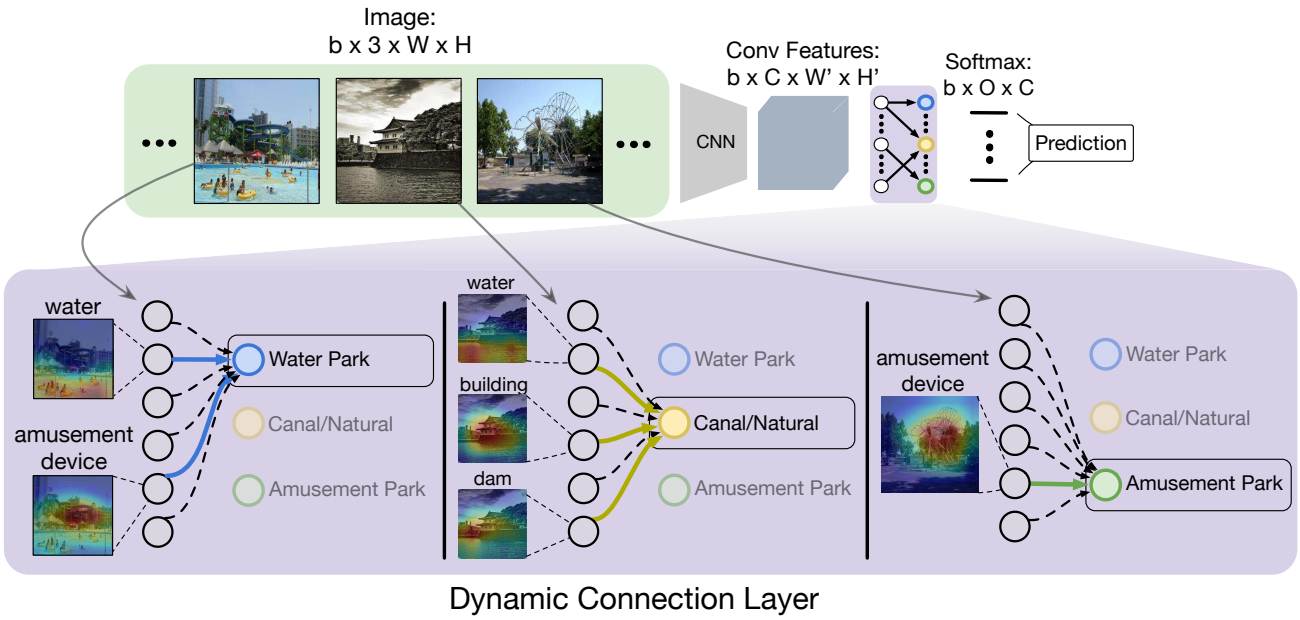

Figure 1: A simple illustration of the mechanism of dynamic connection layer. The route between nodes are dynamically chosen based on the input. Dynamic Connection Layer can be plugged into a fully connected layer or a convolution layer. Here we present a ResNet-liked structure whose final fc layer is replaced by a the Dynamic Connection Layer.

such an approach and concluded that while such approaches are very useful to better *interpret* or debug the model's behavior, the *explanation* emerging from various techniques may be incomplete and inaccurate. Instead, what is suggested is to build a model that is, *by design*, interpretable. Consistent with this sentiment, [40, 18] argues for using additive models whereas [1] shows that it is possible to compute a task embedding that is interpretable (also see [30, 15]. The authors in [35] seek to increase the interpretability of internal filters by incorporating the filter loss. Prototype network [6] proposes training a layer with prototypes for classification. Outside of vision, [7] suggests that interpretable models can be used for credit risk assessment.

EM routing: The general idea of semantic identification described above is also loosely related to the use of EM routing presented in a set of papers introducing capsule networks [25, 13]. Capsules can be thought of as richer (or more powerful) nodes within the network, and EM routing seeks to statistically associate *classes* with a *small number of capsules*. It is able to do so by utilizing only a small support of capsules using *dynamic routing*. In other words, the route can be considered "dynamic" – at the level of classes or even individual examples. While different, it is intuitively related to how, in an ideal setting, concepts create (non-exclusive) equivalence classes in the image corpus. However, rather than post-hoc analysis of a trained network, EM routing seeks to perform this at *training time*. It makes sense because in this case, hypothetically, the networks learns in a way that is more consistent with how one may understand images. A potential limitation of EM routing is that it is computationally demanding.

Intuition behind this work: EM routing with capsule networks performs well – capsules are rich representations and EM routing is an effective training scheme. In some cases, such a setup may even be more interpretable. This raises the following question: are there **some simple modifications** that can encourage EM routing type behavior in convolutional neural networks? If we could achieve this goal for standard CNN architectures, it is reasonable to expect that they may extend to capsule networks as well (although we do not study this extension here). Some recent work provides evidence [39] that evaluating units based on an ablation-based measure is related to the network's classification at the individual class level – in fact, there are 5-10 units which dominate the classification score. If a small set of individual units are relevant for classes, perhaps "dynamic" or EM style routing imposed at training time, will help. What is not known is whether, such a scheme if available, **(1)** will hurt accuracy **(2)** will be practical **(3)** and have any positive impact on interpretability. We provide promising answers to all three questions in this paper.

Contributions: This paper provides **a simple modification** to the SGD algorithm, motivated by numerous empirical observations reported in the literature already, which when adopted yields an EM routing type behavior in CNNs. Interestingly, the modification leads to **no significant reduction in accuracy** and **more interpretable results**. Further, we find that the modification (5-6 lines of code) gives slightly faster runtime (convergence). Interestingly, we can also theoretically analyze the convergence and other math-

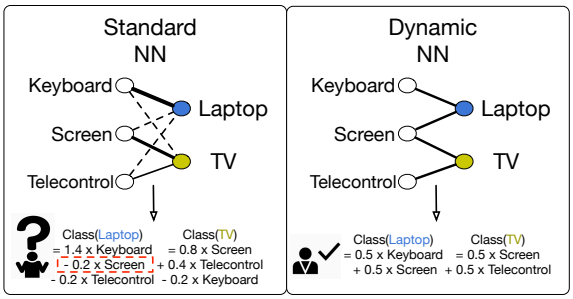

Figure 2: Comparison of weights estimated by standard connections versus dynamic connections. A toy network shows the different connections. A wider connection line represents a larger weight value. Dotted line represents a negative weight. We also visualize the weight for each class in function representation and find that the weight vector trained by standard NN can be hard to understand while the weight trained by dynamic connection is closer to the actual semantic representation.

ematical properties of this procedure.

Nomenclature: The reader will shortly see that our modification will involve adaptively thresholding the activations or the gradients based on certain criteria. But to make it easier to refer to and because of the dynamic routing type behavior it encourages, we will simply call our **adaptive activation thresholding** scheme as **Dynamic-K activation**, where K represents a vital thresholding parameter in the algorithm. The use of this algorithm will consequently lead to what we call a *Dynamic Connection* or *Dynamic Layer*. Analogously, a "Dynamic NN" is the network obtained using this adaptive activation thresholding or Dynamic-K activation scheme.

2. Preliminaries of EM/Dynamic Routing

First, we briefly review the high-level idea behind EM Routing [13] and discuss the relationship with our proposal. While the overall procedure is involved, we will present a simplified version of EM routing here, see Alg. 1.

Basic idea: Assume that the feature inputs are $X = \{X_i\}_{i=1}^{M}$, the parameters are $W \subseteq \mathbb{R}^{O\times M}$, and the feature outputs are $Y = \{Y_c\}_{c=1}^{O}$. Either X_i or Y_c can be thought of as a single scalar value (e.g., unit in a Fc layer) or a vector (e.g., in capsule networks) or a matrix (e.g., feature map in Conv layer). For simplicity, X_i or Y_c will be a scalar here. Basically, we can assume that the activation from an upper-layer's unit is calculated by estimating by the "mean" or average of the units' clusters in a lower-layer (Fig. 3 (a)). In EM routing, the upper-level units' activation only depends on (or driven by) a few lower-level units which are *inside* the range or support of the estimated covariance matrix. It may be argued that this has implications for interpretability due to sparsity. Let us briefly see why this is the case.

Consider a toy example shown in Fig. 2. The weight representation of a sparse connection ('dynamic' or variable w.r.t. samples or classes) on the right is easy to understand. On the contrary, by the nature of cross-entropy loss in the standard setting, the weight for the laptop and TV are encouraged to repel each other, which may lead to some confusion as to why the screen is negatively associated with a laptop. While a dynamic or EM routing scheme may not guarantee immunity to this behavior, one will empirically see a behavior more consistent with Fig. 2 (right).

Algorithm 1: A simplified EM/Dynamic routing scheme with routing variables $R \subseteq \mathbb{R}^{O\times M}$.

for $t = 1, 2, 3$ **do**
  for $c = 1, 2, ..., O$ **do**
    $Y_c = \frac{\sum_{i=1}^{M} R_i^c W_i^c X_i}{\sum_i R_i^c}$.
    Update R_i^c by a complicated distance function with modified kernel.
  end
end

Limitations of EM/Dynamic routing: A limitation of the original EM/dynamic routing algorithm is its sizable computational footprint, which becomes challenging on large-scale datasets. We will shortly see a minor modification that can provide significant benefits.

3. Separating Sparsity from Interpretability

Sparsity vis-à-vis Interpretability. Consider a classification task in vision. Here, there is agreement that models which use fewer (high level) features for predictions are naturally, more interpretable. So, if we **assume** that we have a good representation learner or feature extractor – let us say, provided by an oracle, then, models that are sparse may be more interpretable. It is important to note that in general, feature extractors cannot be assumed to be given. On the other hand, if we wanted to train the entire network **together with** the sparsity constraint, there are two known drawbacks. **First**, optimizing such models turns out to be much more demanding than a version without such constraints, especially in the large scale settings due to the optimization landscape [23]. lAnd **second**, when one performs end-to-end training (feature extraction with sparsity constraint), we often see a quantifiable loss in predictive accuracy[28], which is unacceptable in various scenarios.

Motivating a simpler alternative: Encouraging sparsity in a direct manner as described above is difficult. To see this conceptually, let $W : F \to S$ be an operator where $F \subseteq \mathbb{R}^p$ and $S \subseteq \mathbb{R}^q$ are the features and label (or output) spaces respectively. Then we may define,

$$W \text{ is } \alpha\text{-sparse} := \exists \alpha, \forall x \in F, \|w_i \odot x\|_0 \leq \alpha. \quad (1)$$

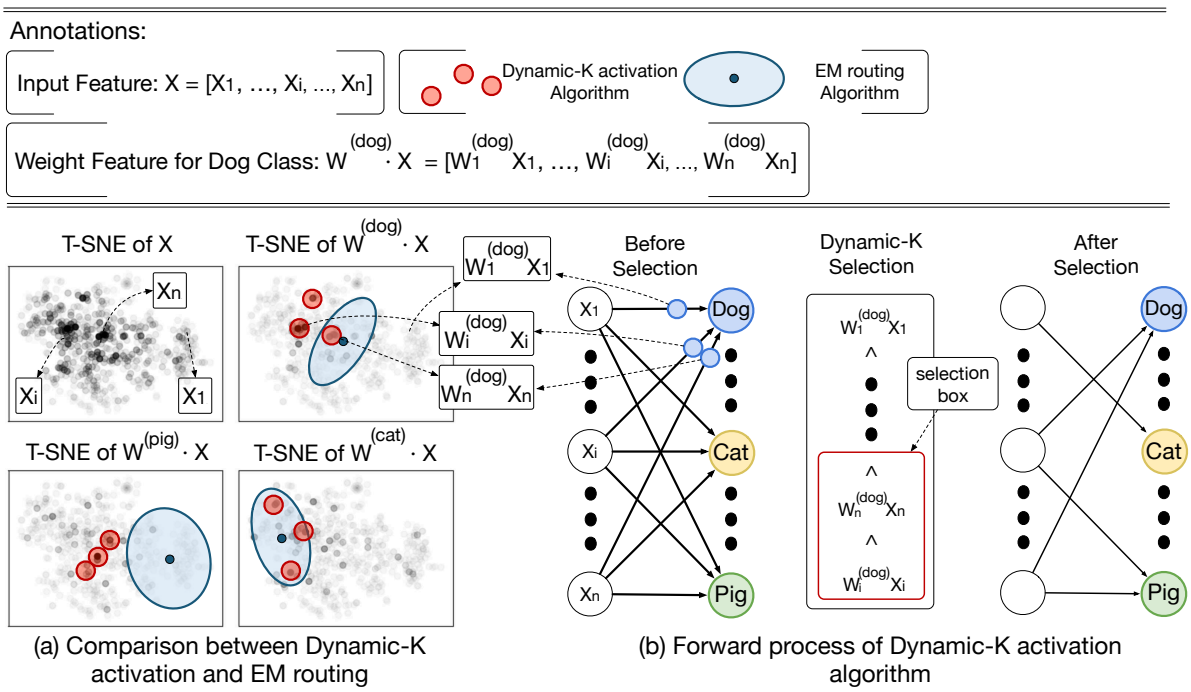

Figure 3: The difference between EM-Routing and Dynamic-K activation Algorithm. In (a), the blue shaded region corresponds to points that are within unit radius distance from a Guassian Distribution centered at μ as suggested by the EM-routing method. The red shaded region is the selection ball for the units selected by the Dynamic-K activation Method. The top left image shows the distribution of input units from the lower-layer(X could be a scalar value, in the plot, we show a single feature point X as a 7x7 feature map). The depth of the color represents the activation intensity of unit (X_i). Given an upper layer class unit $c \in \{\text{cat}, \text{dog}, \text{pig}\}$, the deeper color in other three images represents a larger value of $X_i \cdot W_i^c$, where W^c is the weight vector of class n. In (b), we visualize how the units from lower layer are selected by Dynamic-K activation algorithm and sent to the next layer.

We can interpret F as the space of equivalence class of images that is often used in the context of semantic identification. Choosing $F = \mathbb{R}^p$ in (1) is equivalent to imposing hard sparsity constraints, which has been extensively used in practice [14]. In general, to preserve the accuracy of a model, F needs to be the *support of the joint distribution* of **high level features** and **output classes**. Fortunately, results in [24] show that it is possible to train interpretable models that are also very accurate. Hence, it is safe to hypothesize that although it may be intractable to characterize the optimal F, the size of the optimal F may be small. For example, for a particular class, say dog, only few high level concepts/features in F (tail, nose etc.) are relevant [25].

Main idea: Our key idea is that since deep networks are trained using a first order method in most cases, interpretability may be ensured directly by a **very simple modification** to the training algorithm, that satisfies the following condition,

Requirement. *An update is* interpretable *if* **only** *the subset of parameters that are responsible for the prediction of the class for a given example are updated.*

The above requirement can be imposed by changing only a few lines of code in the optimization routine, and summarized in Alg. 2. It encourages interpretability simply by using activations to choose the parameters to update. Mechanistically, what the algorithm does and how it relates to EM routing is shown in Figure 3.

Algorithm 2: Dynamic-K activation with dynamic parameter K and routing variables, $R \subseteq \{0,1\}^{O \times M}$.

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. $||R^c||_1 \leq K$.
end

Fig. 3 (b) shows how easy it is to assess what the procedure does. It offers dynamic routing type behavior but does not need the two-phase EM scheme, which is known to be the primary runtime bottleneck. Interestingly, the benefits can be leveraged even with standard CNNs, without the full-blown capsule formulation.

Remark 1: Notice that the major difference with EM Routing is that we enforce that the column sum $\sum_{i=1}^{M} R_i^c$ adds up to no more than K instead of 1. We assume that each neuron can be more interpretable if responsible for multiple related predictions. As an illustration, for the ex-

ample in Fig. 2, 'Laptop' = 'Keyboard' + 'screen' and 'TV' = 'Telecontrol' + 'screen', where a 'screen' neuron may be responsible for both 'TV' and 'Laptop'. There are more examples available in [38] that suggest that allowing a neuron to map to more than one prediction is not a bad idea. Even in capsule networks, single routing (route to only one higher unit) is not always preferred in large-scale classification.

Remark 2: We should also observe that even though we combine units into groups as described above, it is different from the Group-ℓ_1 regularization method [26]. While [26] binds units to some specific groups ("concepts"), in contrast, we encourage one-to-one correspondences between units and concepts so that any number of units can be combined to form a new concept, as needed.

Intuition: The procedure described above, despite its simplicity, is more than just a implementation trick. From the theoretical standpoint, it turns out that our adaptive activation thresholding or Dynamic-K activation algorithm is (approximately) equivalent to a variant of SGD with interesting optimization and statistical guarantees. While it is not essential for implementation purposes, the update scheme can be interpreted in a rigorous manner, summarized in Algorithm 3. Essentially, as seen from Step 5 of Algorithm 3, the coordinates are first ranked or chosen based on the (local) curvature of the loss function measured using the Lipschitz constant along that coordinate. The parameter $s \in (0, 1]$ controls the sparsity of the updates: a large value of s encourages interpretable updates. In fact, Dynamic-K activation algorithm is exactly equivalent to Algorithm 3 for sufficiently large s. To avoid confusion, we remind the reader that this algorithm is explicitly written down only to facilitate mathematical analysis. In other words, Alg. 2 is "reinterpreted" as Alg. 3 so that it can be analyzed in terms of its local optimality and convergence rate.

The following theorem shows that Algorithm 3 and SGD provably converge at the **same rate**.

Theorem 1. *Let the loss function f be a smooth function with respect to W. Then, the iterates W_t generated by Algorithm 3 converges to a local minimizer at $O(1/t)$.*

Proof. (Sketch.) The proof follows the standard techniques involved in analysis of first order methods [22, 32, 5]. That is, we bound the expectation of the norm of the gradient W_t using the smoothness of the function and per iteration decrease of the loss. See appendix for full proof. □

Implementation: The only expensive step in Alg. 3 is to estimate L_{ij}'s, which can be intractable in the worst case. Luckily, the following Lemma shows that when we use a feed forward network with the cross entropy loss function, L_{ij}'s turn out to be bounded by activations.

Lemma 2. *For a feed forward network with cross entropy loss function and any parameter j on the final classification layer, an estimate of L_{ij} can be* **locally** *obtained from the activations, during back propagation.*

Proof. (Sketch.) Our proof proceeds in two steps. In the first step, we show a key technical lemma in which we bound the change in prediction for a small perturbation of parameters in the final layer. This step crucially relies on the structure of gradients obtained in the final layer which turns out to be satisfied by standard classification loss functions. The second step is essentially an inductive step over the hidden layers, and a standard calculation via arguments based on Section 2 in [20]. □

Practicality: Although Lemma 2 indicates that it is easy to estimate the Lipschitz constants, it is not clear if such an upper bound is always useful in practice. But it is satisfying that our minor adjustment can be nicely analyzed, and we can derive insights about its behavior. We will now provide extensive set of experimental results that show that we can obtain interpretable models using our algorithm, with minimal/no compromise in accuracy.

4. Experiments

SGD remains the defacto optimization method to train machine learning models in computer vision. Our theoretical analysis, in essence, suggests that only a simple modification to SGD is needed. At a high level, the goal of our experiments is to evaluate whether dynamic connection based routing such as ours can, in fact, make the model more interpretable while preserving the overall accuracy. We performed **two** sets of experiments to test the performance of

Algorithm 3: Interpretable SGD: Block Coordinate SGD with Lipschitz Sampling

1 **INPUT:** Dataset $\mathcal{Z} = \{(x_i, y_i)\}_{i=1}^n$, loss function f, trainable parameters $W_j, j = 1, ..., O \times M$; minibatch size B, sparsity level s, learning rate η, and block size β.
2 **for** $t = 1, 2, ..., T$ **do**
3 Estimate the Lipschitz constant L_{ij} of the gradient along the j-th coordinate of W using (x_i, y_i).
4 Set $L_i = \sum_{j=1}^{O \times M} L_{ij}$.
5 Interpretable Gradient from each sample $\tilde{g}_i \in \mathbb{R}^{O \times M}$: $\tilde{g}_{ij} = p_i q_j g_{ij}$ where $g_{ij} = \nabla f(x_i, y_i)$, p_i, and q_j are Bernoulli random variables with biases B/n and $\beta \left(L_{ij}/L_i\right)^{1/(1-s)}$, respectively.
6 Interpretable Update: $W \leftarrow W - \eta \frac{1}{B} \sum_{i=1}^n \tilde{g}_i$
7 **end**
8 **OUTPUT:** Parameters W.

our Dynamic-K activation algorithm. The **first** set of experiments intend to analyze the accuracy loss while using our algorithm for training. The **second** set of experiments are intended to analyze the interpretability of the models obtained using our algorithm. To measure the interpretability quantitively, we utilize proxies for the interpretablity of a network given in [19]: (A) **Sparsity**. In Fig. 4 and 5, we evaluate this property directly; (B) **Simulatability**. A human is able to simulate or work through its decision-making process. We show these results in Fig. 6-7 where dynamic routing plays a role. (C) **Modularity**. The meaningful portion (units) of its prediction making process can be interpreted independently. The 1-1 correspondence between units and independent concepts can be measured using [4], and we will show quantitative results in Fig. 5(c,d).

Experimental Setup. We used a variant of ResNet architecture in all our experiments. Since deeper convolution layers represent more explainable information [4], we only replaced the final fully-Connected layer with a dynamic connection layer to keep the presentation of our experimental results succinct. But our algorithm is applicable to the internal layers as well. These extended experimental results are in the appendix, and consistent with our main message.

We adopt the following naming convention throughout this section: (i) **"Res18-d$[K]$a"** denotes the ResNet18 [11] structure where the last layer uses Dynamic-K activation routing method; (ii) **"Res18"** denotes the original ResNet18 [11]; and (iii) **"Res18-L1"** denotes the original ResNet18 trained with ℓ_1 regularization on the last layer.

4.1. Performance on CIFAR10

Here, we evaluate if dynamic connections can lead to sparsity while ensuring high accuracy, simultaneously. We set the weight decay parameter to 0.0005 and the momentum parameter to 0.9. We trained our models on a *single* GPU with a mini-batch size of 256 to compute gradients and trained for 200 epochs. We employ the standard learning rate practice using 0.1 at the beginning, and reducing it by a factor every 70 epochs. We use data augmentation transformations in [11].

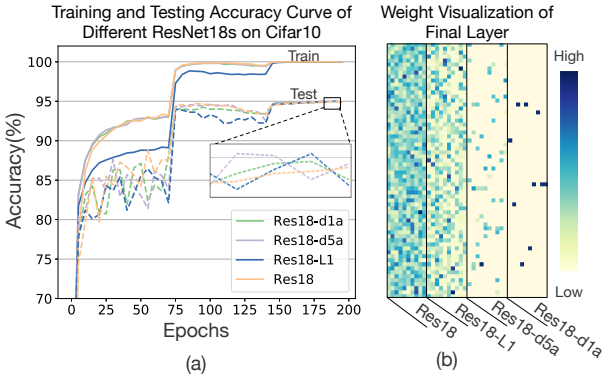

Figure 4: In (a), we compare the train and test accuracy curve over 200 epochs of dynamic CNN (Res18-d1a, Res18-d5a) and standard CNN (Res18, Res18-L1) trained on Cifar10. In (b), for all models, we visualize the final fully connected layer's weight (512×10) to compare the sparsity.

Model	Places365 Top1	Places365 Top5	Cifar
Res18(baseline)	53.69	83.78	95.10
Res18-d1a	27.83	57.56	95.02
Res18-d5a	53.60	84.15	95.12
Res18-d10a	53.64	84.06	**95.22**
Res18-d25a	**53.87**	84.25	95.04
Res18-d50a	53.58	**84.91**	95.06
Res18-d100a	53.81	83.92	95.12
Res50(baseline)	54.74	85.08	95.62
Res50-d25a	**55.00**	85.07	95.58

Table 1: Accuracy for different ResNets on Places365 and Cifar10.

Baseline. We use Res18-L1 as our baseline: this is the standard residual network with sparsity regularization. Fig. 4(a) shows our experimental results. We can clearly see that models trained by dynamic connections (Res18-d1a, Res18-d5a) converge at a similar speed as the standard network (Res18). Note that the accuracy of the models obtained using our algorithm and the baseline are nearly the same. Interestingly, the final layer obtained using our Dynamic-K activation algorithm is much sparser than the baseline, as in Fig. 4(b): (Res18-d1a $>$ Res18-d5a $\gg$ Res18-L1 $>$ Res18). Sparsity is, of course, a coarse way to estimate interpretability, in a small dataset like Cifar10. Still, we can conclude that Dynamic-K activation algorithm can be used to obtain models that are more interpretable than when using explicit sparsity terms in these settings.

Takeaway. Dynamic-K activation algorithm can boost interpretability without sacrificing the accuracy of the model obtained and training time.

4.2. Performance on Places365

Now we test our algorithm on the much larger dataset–Places365 [37]. The authors in [37] note that it is much easier to interpret units in models trained on Places365 dataset, as well as to evaluate interpretability schemes in general. On this dataset, we may simply use the NetDissection method [4]. The dataset contains 365 classes and each class includes ~5k training images and 1k testing images. For this experiment, we used a weight decay of 0.0001 keeping all the other hyperparameters the same as before.

From the accuracy perspective, we compare the following models: Res18-d$[K]$a with K in $\{1, 5, 10, 25, 50, 100\}$ and standard Res18, Res50-d25a and Res50. Table 1 shows the results of our experiment. We find that the validation accuracy on Places365 is even better than the original

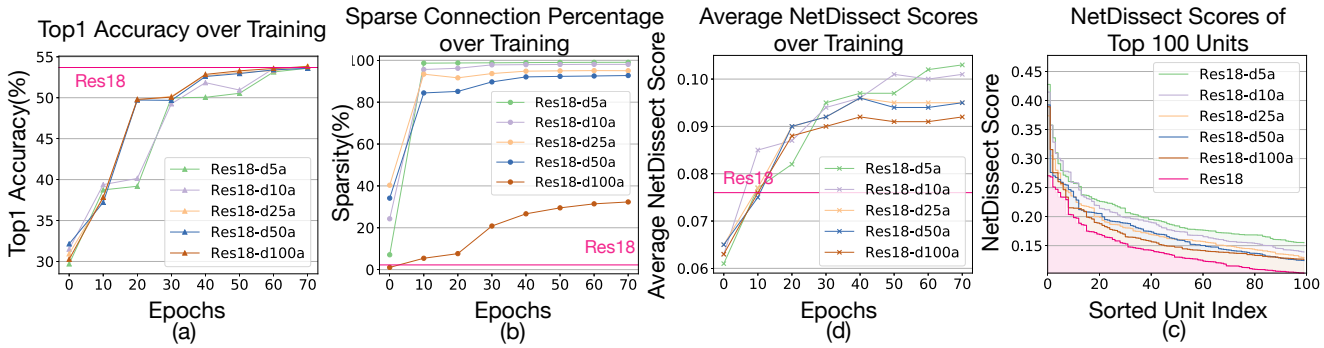

Figure 5: In (a), we compare the top1 accuracy curves of dynamic models(Res18-d[K]a) with K in $\{5, 10, 25, 50, 100\}$ trained on Places365 over 70 epochs. The baseline top1 accuracy of the standard ResNet18 is shown by red line in graph. In (b), we show how the percentage of sparse connection in five dynamic models with different K parameters changes over training. (the absolute value of weight smaller than 10^{-3} is considered as sparse connection.) The sparsity of connection in standard ResNet18 is also shown by red line in graph. In (c), we present the change of average NetDissect Scores over 512 units in layer4 of all test models during training. The standard Res18's score is also reported (red line). To see the distribution of NetDissect Scores of layer4's units in all dynamic models include Res18, we sort the unit by scores and choose top 100 units as shown in (d).

ResNet18 when K is 25 and 50, similar to the accuracy curve shown in Fig. 5(a). Clearly, the accuracy loss is negligible compared to the original model.

Now, let us focus on the sparsity levels of the models as training proceeds with different settings for the hyper-parameter K. We see in Figure 5(b), that if $K \geq 100$, the connection in the dynamic layer starts becoming denser, although still **much sparser** than Res18. Specifically, Res18-d5a can achieve within 0.1% best Top1 accuracy with 99% sparsity accuracy (within 0.4% of the best Top5 accuracy).

Interpreting results. We analyzed single units' interpretability using the NetDissection method as mentioned earlier [4]. In Fig. 5(c) and (d), we see that smaller values for K give larger IoU scores, which means that units are more interpretable. In essence, our algorithm encourages the preceding layers to be more "concept-specific".

Takeaways. There are two important takeaways from our large scale experiment: (i) Dynamic-K activation algorithm increases the interpretability of units; and (ii) the accuracy can be preserved at no additional cost.

4.3. Interpretabiliity

In the second set of experiments, we used two ways to analyze interpretability of models obtained using the Dynamic-K activation algorithm. First, we use *concept composability* where concept patterns that are important for predictions are analyzed. Secondly, we use the technique of *instance explanation* which allows us to visualize parts of the network that are important for a prediction.

Concept Composability. Here, our goal is to gain insight about what is learned by a model trained for each class. This is done by a direct examination of the hidden units and their corresponding concepts. We show illustrative examples in Figure 6. In these examples, every unit in the final convolutional layer is analyzed by NetDissect algorithm and labeled by the closest concept. Then, by simply inspecting the weights, we can characterize the decision boundaries of each class. For example, we can see that class "canal/nature" can be interpreted as a composition of **only** 4 concepts viz., "water", "dam", "castle" and "sea". Hence, the magnitude of the weights in the final layer indicate the importance of the concepts for predicting the class of canal/nature. Quantitatively, our experiments show that models obtained using our Dynamic-5 activation algorithm are easily interpretable: more than 95% classes can be represented with at most 5 concepts.

Takeaway. Our Dynamic-K algorithm can be used to obtain transparency of information flow in deep network that is *consistent* with human level understanding.

Instance Explanation. Here, we analyze the explanations provided for predictions at an instance level. To that end, we compare the explanations for predictions provided by models trained using Dynamic-K activation algorithm and a baseline SGD.

We show in Fig. 7 that the explanations obtained from the baseline model are spread across a wide range of unclear concepts. The difficulty of interpreting such a model may be mitigated by an external framework like [38], where we find that the residual drops to **45%**. We can obtain even better models directly using dynamic connections where very few concepts of units (fewer than 5) can explain about 98% of the image with **2%** inexplicable residual remaining.

Observe that the explanation by units in our Dynamic-K activation algorithm is significantly more consistent and accurate compared to the baseline approaches. For example, a model obtained from Dynamic-K algorithm explains a "stadium" using "stadium baseball" and "football field" concepts whereas the baseline model uses concepts such as

Figure 6: Visualize the connection from layer4 nodes to the final class in Res18-d5a trained on Places365. The width of line represents the strength of connection (weight value). For example, the top-3 units contributes to the Airplain Cabin class are unit237 (main concept "airplain cabin" with IoU score 0.22), unit483 (main concept "cockpit" with IoU score 0.19) and unit388 (main concept "laundromat" with IoU score 0.19). The connection of other units to this class is too weak to notice, thus not included. The visualization of each unit shown below the text is the top activated images and the corresponding regions[33].

"toilet", "bakery shop" that are obviously irrelevant for predicting the class.

Takeaway. Models that provide high quality instance level explanations can be obtained using Dynamic-K algorithm with no additional cost.

5. Conclusion

This paper proposes a simple adaptive activation thresholding or dynamic K routing rule based on choosing several top activated lower-level units for a higher-level unit. The idea is simple and inspired by the dynamic/EM routing algorithm introduced in [25, 13]. But our simplified version can be plugged into any current CNN network structures and involves minimal changes to the optimization code. The idea is easy to understand, works well, and we can rigorously analyze its convergence properties. Our experiments

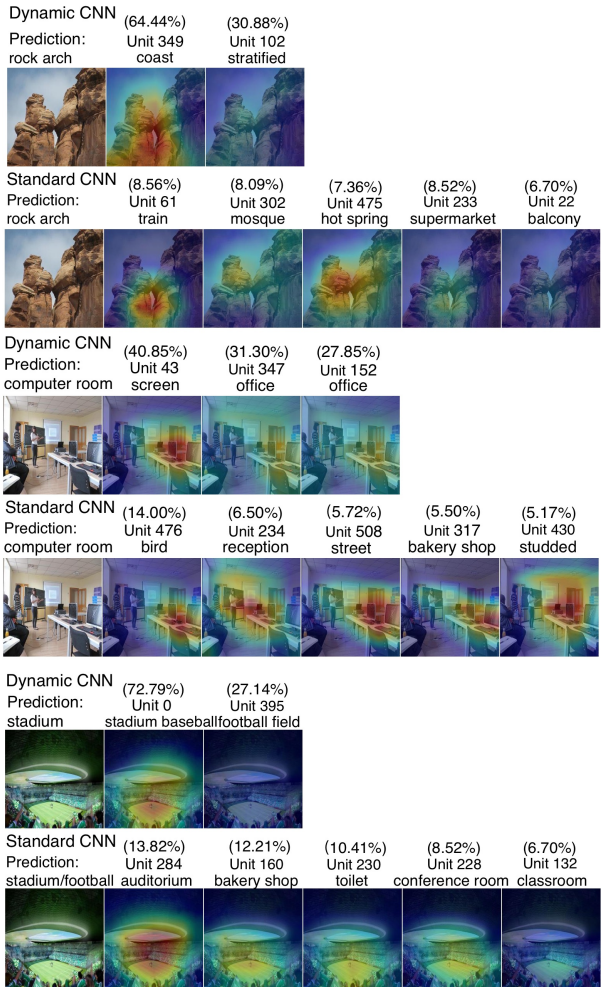

Figure 7: Visualize the decision explanation in Res18-d5a trained on Places365. The heatmap indicates the region corresponding to the units. Above each image, we show the contribution score, unit ID, concept name on the $1^{st}, 2^{nd}, 3^{rd}$ lines respectively. We see fewer than 5 concepts that are important for predictions while using Dynamic-K algorithm. To compute the contribution score, first we compute the output score for class c as $Y_c = \sum_{i=1}^{n} W_i^c X_i$, (where n is the #-units in layer4, X_i is the activation of unit i, W_i^c is the weight value on the edge between unit i and class c). Then, the contribution score of unit i to class c is computed as $W_i^c X_i / Y_c$.

show that this scheme can help speed up convergence, encourage sparsity of the network and increase interpretability, without sacrificing the overall accuracy.

Acknowledgments

This research was supported in part by UW CPCP AI117924 (NIH BD2K center grant), American Family Insurance, NSF CCF #1918211 and NSF CAREER award RI #1252725.

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichand ran, Subhransu Maji, Charless Fowlkes, Stefano Soatto, and Pietro Perona. Task2Vec: Task Embedding for Meta-Learning. Feb 2019.

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proc. CVPR*, 2015.

[3] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus Muller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS One*, 10(7), 7 2015.

[4] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proc. CVPR*, 2017.

[5] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization*, 23(4):2037–2060, 2013.

[6] Chaofan Chen, Oscar Li, Alina Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. *CoRR*, abs/1806.10574, 2018.

[7] Chaofan Chen, Kangcheng Lin, Cynthia Rudin, Yaron Shaposhnik, Sijia Wang, and Tong Wang. An interpretable model with globally consistent explanations for credit risk. *CoRR*, abs/1811.12615, 2018.

[8] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv*, 2017.

[9] Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[10] Abel Gonzalez-Garcia, Davide Modolo, and Vittorio Ferrari. Do semantic parts emerge in convolutional neural networks? *International Journal of Computer Vision*, pages 1–19, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[12] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European Conference on (ECCV)*, 2016.

[13] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with EM routing. In *International Conference on Learning Representations*, 2018.

[14] Chun-Guang Li, Chong You, and René Vidal. Structured sparse subspace clustering: A joint affinity learning and subspace clustering framework. *IEEE Transactions on Image Processing*, 2017.

[15] Lemao Liu, Taro Watanabe, Eiichiro Sumita, and Tiejun Zhao. Additive neural networks for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 791–801, 2013.

[16] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. *Proc. CVPR*, 2015.

[17] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[18] W. James Murdoch, Peter J. Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. *CoRR*, abs/1801.05453, 2018.

[19] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Interpretable machine learning: definitions, methods, and applications. *CoRR*, abs/1901.04592, 2019.

[20] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *ICLR*, 2018.

[21] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017.

[22] Julie Nutini. *Greed is good: greedy optimization methods for large-scale structured problems*. PhD thesis, University of British Columbia, 2018.

[23] Sathya N Ravi, Tuan Dinh, Vishnu Sai Rao Lokhande, and Vikas Singh. Constrained deep learning using conditional gradient and applications in computer vision. *arXiv preprint arXiv:1803.06453*, 2018.

[24] Cynthia Rudin. Please stop explaining black box models for high stakes decisions. *CoRR*, abs/1811.10154, 2018.

[25] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3856–3866. Curran Associates, Inc., 2017.

[26] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomput.*, 241(C):81–89, June 2017.

[27] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[28] S. Shalev-Shwartz, N. Srebro, and T. Zhang. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization*, 20(6):2807–2832, 2010.

[29] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *International Conference on Learning Representations Workshop*, 2014.

[30] Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo. Learning global additive explanations for neural nets using model distillation. *arXiv preprint arXiv:1801.08640*, 2018.

[31] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. *Proc. CVPR*, 2017.

[32] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[33] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[34] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Proc. ECCV*, 2014.

[35] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable cnns. *CoRR*, abs/1901.02413, 2019.

[36] Bolei Zhou, Aditya Khosla, Lapedriza. Agata, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

[37] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. *In Advances in Neural Information Processing Systems*, 2014.

[38] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[39] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Revisiting the importance of individual units in cnns via ablation. *CoRR*, abs/1806.02891, 2018.

[40] Hao Henry Zhou, Yunyang Xiong, and Vikas Singh. Building bayesian neural networks with blocks: On structure, interpretability and uncertainty. *arXiv*, 2018.