SINF: Semantic Neural Network Inference with Semantic Subgraphs

A.Q.M. Sazzad Sayyed [1] **Francesco Restuccia** [1]

Abstract

This paper proposes *Semantic Inference* (SINF) that creates semantic subgraphs in a deep neural network (DNN) based on a new Discriminative Capability Score (DCS) to drastically reduce the DNN computational load with limited performance loss. We evaluate the performance of SINF on VGG16, VGG19, and ResNet50 DNNs trained on CIFAR100 and a subset of the ImageNet dataset. Moreover, we compare its performance against 6 state-of-the-art pruning approaches. Our results show that (i) on average, SINF reduces the inference time of VGG16, VGG19, and ResNet50 respectively by up to 29%, 35%, and 15% with only 3.75%, 0.17%, and 6.75% accuracy loss for CIFAR100 while for ImageNet benchmark, the reduction in inference time is 18%, 22%, and 9% for accuracy drop of 3%, 2.5%, and 6%; (ii) DCS achieves respectively up to 3.65%, 4.25%, and 2.36% better accuracy with VGG16, VGG19, and ResNet50 with respect to existing discriminative scores for CIFAR100 and the same for ImageNet is 8.9%, 5.8%, and 5.2% respectively. Through experimental evaluation on Raspberry Pi and NVIDIA Jetson Nano, we show SINF is about 51% and 38% more energy efficient and takes about 25% and 17% less inference time than the base model for CIFAR100 and ImageNet.

1. Introduction

State-of-the-art DNNs employ a larger number of parameters than what mobile devices can tolerate today. For example, YoLov10, the state of the art DNN for object detection, uses a DNN backbone with 29.5M parameters (Ao Wang, 2024). Approaches such as pruning (Han et al., 2015b; Chen et al., 2023), quantization (Han et al., 2015a; Qin et al., 2022), and coding (Gajjala et al., 2020; Han et al., 2015a) incur in excessive performance loss and most often require fine-tuning.

[1]Northeastern University, United States. Correspondence to: A.Q.M. Sazzad Sayyed <sayyed.a@northeastern.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

In this paper, we propose an approach based on *cluster-level semantic DNN subgraphs* to reduce the computing load without compromising the DNN accuracy and without retraining. As detailed in Section 3, semantically similar inputs share a significant number of filter activations compared to semantically dissimilar inputs. For example, as shown in Section 3, images of seals share significantly more filter activations with images of dolphins than with images of tables. As such, if a semantic subgraph were to be available, we could only execute that subgraph related to the current input and "turn off" the rest of the DNN.

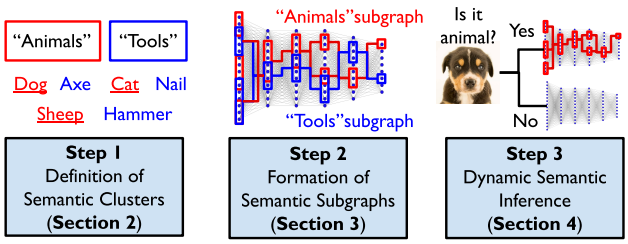


Figure 1: Overview of proposed SINF framework.

We build on top of this interpretability insight and propose *Semantic Inference* (SINF), whose main components are illustrated in Figure 1. In a nutshell, SINF transforms a *pre-trained and static* DNN into a *dynamic* DNN by (i) creating subgraphs corresponding to each semantic cluster, which are defined based on the current DNN task (steps 1 and 2); (ii) selecting the semantic-relevant subgraphs at inference time based on a preliminary cluster-based classification of the input image (step 3). Conversely from pruning approaches, SINF does *not* require changing any portion of the DNN to decrease computing load.

This paper makes the following novel contributions:

• We propose a new inference framework called SINF, which logically partitions the DNN into semantic subgraphs based on a new Discriminative Capability Scoring (DCS) that finds the filters most associated with a given semantic cluster (**Section 3**). Finally, SINF pre-classifies an image based on its relevant cluster so that only its semantic-specific subgraph is activated (**Section 4**);

• We benchmark the performance of SINF on VGG16, VGG19, and ResNet50 DNNs trained on the CIFAR100 and

ImageNet dataset. We compare DCS against several state-of-the-art discriminative algorithms (Mittal et al., 2019; Molchanov et al., 2019; Hu et al., 2016; Sui et al., 2021; Lin et al., 2020) (**Section 5**). Although the target of SINF is not to perform pruning but to extract subgraphs, we use DCS as a pruning approach and compare it against (Murti et al., 2023) which does not require fine-tuning. We have implemented SINF on real-world platforms Raspberry Pi and NVIDIA Jetson Nano and evaluated its latency and energy consumption;

Key Results: On average, executing the semantic subgraphs reduces the inference time of VGG16, VGG19, and ResNet50 respectively by up to 29%, 35%, and 15% with only 3.75%, 0.17%, and 6.75% accuracy loss for CIFAR100 while for ImageNet benchmark, the reduction in inference time is 18%, 22%, and 9% for accuracy drop of 3%, 2.5%, and 6%. Moreover, DCS achieves respectively up to 3.65%, 4.25%, and 2.36% better accuracy with VGG16, VGG19, and ResNet50 with respect to existing discriminative scores for CIFAR100 and the same for ImageNet is 8.9%, 5.8%, and 5.2% respectively. Finally, SINF is about 51% and 38% more energy efficient and takes about 25% and 17% less inference time than the base DNNs for CIFAR100 and ImageNet, respectively.

2. Related Work

Table 1 compares SINF with prior approaches, which are detailed in the rest of the section. Unlike early exit methods, which require multiple classifier heads, SINF requires a single auxiliary classifier. As for pruning and quantization, these are considered orthogonal to SINF and can be used in addition to SINF to further improve performance.

	Dynamic	**Fine-tuning**	**Semantic**
Pruning	✗	✓	✗
Quantization	✗	✓	✗
Early-exit	✓	N/A	✗
SINF	✓	✗	✓

Table 1: SINF vs other related approaches.

Pruning: Han et al. (Han et al., 2015b) proposed weight-norm-based unstructured pruning. Li et al. (Li et al., 2017) used the L_1 norm of the kernel weights to prune entire filters. However, weight-norm-based strategies do not directly take into account the importance of the filters or parameters to preserve the DNN accuracy. Another approach employs first-order gradient which estimates the importance of the filters based on the gradient of the loss function (Molchanov et al., 2019). Another class of techniques find and prune duplicate or redundant filters. To find such filters, Sui et al. (Sui et al., 2021) use the change in the nuclear norm of the matrix formed from the activation maps when individual filters are removed from a layer. Lin et al. (Lin et al., 2020) use the expected rank of the feature maps, while Chen et al. (Chen et al., 2023) explain the soft-threshold pruning as an implicit case of Iterative Shrinkage-Thresholding. *Although these methods determine the redundant filters, they fail to focus on the filters that are necessary to distinguish among the classes. Moreover, all of these methods require fine-tuning after pruning*. When fine-tuning is not possible, these methods do not provide satisfactory performance. Recently, Murti et al. (Murti et al., 2023) proposed a retrain-free *IterTVSPrune* approach based on Total Variational Distance (TVD) (Verdú, 2014). Here, we extract subgraphs most representative of classes belonging to a given semantic cluster.

Quantization and Coding: The seminal work by (Han et al., 2015a) compressed the DNN through quantization and Huffman coding to reduce the memory footprint. Among more recent work, post-training quantization (Cai et al., 2020) and quantization-aware training (Zhong et al., 2022) have been proposed. Qin et al. (Qin et al., 2022) pushes the boundary using single-bit quantization of the popular language model Bidirectional Encoder Representations from Transformers (BERT). Lin et al. (Li et al., 2020) designed a layer-wise symmetric quantizer with the learnable clip value only for high-level feature extraction module. Tu et al. (Tu et al., 2023) recently designed an algorithm for network quantization catered to the needs of image super-resolution. Both quantization and coding are orthogonal to SINF and can be used to achieve further improvement.

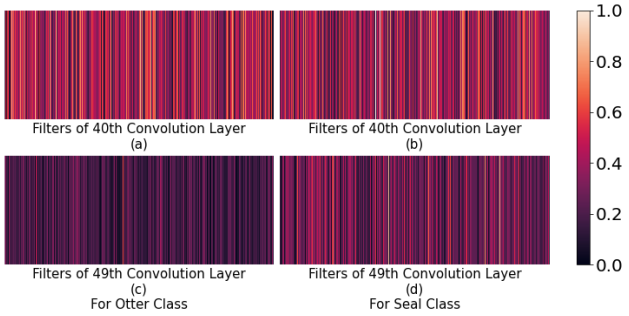
Early Exiting: Early exiting was proposed by (Teerapittayanon et al.) to make the DNN inference dynamic by using auxiliary (and relatively small) classifiers attached to the output of the DNN layers (Matsubara et al., 2021). Based on their confidence, the decision to traverse the remaining layers is made (Matsubara et al., 2021). The training of the auxiliary classifiers can be done jointly with the backbone network (Pomponi et al., 2022) or separately (Garg & Moschitti, 2021). The classifiers can be trained using either cross-entropy loss (Wang et al., 2019), or knowledge distillation (Phuong & Lampert, 2019). Recently, (Han et al., 2023) proposed using block-dependent loss from a subset of the exits close to a block to train the classifiers. Dong et al. (Dong et al., 2022) predicts which early exit to use using a lightweight "Exit Predictor". (Narayan et al., 2023) modeled the exit selection as an online learning problem and chooses the exit in an unsupervised way. Conversely, SINF uses an auxiliary classifier to select the subgraph according to the input.

3. Dividing a DNN into Semantic Subgraphs

We define the concept of *semantic cluster*. Let $\mathcal{D}$ be a labeled dataset with class labels $\mathcal{K}$. We define K semantic

clusters, each composed by a subset of classes $\{\gamma_1, \ldots, \gamma_K\}$ such that $\gamma_1 \cup \gamma_2 \cup \cdots \gamma_K = \mathcal{K}$. We primarily assume that these clusters are formed based on similarity of the semantics of their member classes. These semantics can be defined at the application level. The clusters can also be pre-defined at the dataset level (e.g., as in the CIFAR100 dataset).

We performed a series of experiments to validate the intuition behind our proposed SINF approach (we refer to supplementary section A for the details). It is well known that filters of DNN identify parts of objects, colors or concepts. Many of these filters are shared among classes (Bau et al., 2017). On the other hand, filter activations become sparser as the DNN becomes deeper, with filters reacting only to specific inputs belonging to specific classes. This phenomenon can be observed in the top portion of 2, which shows the average filter activation strength for the "otter" and "seal" classes in the the 40th and 49th convolutional layers of ResNet50 trained on CIFAR100.



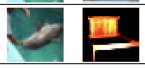
	Inputs	Layer 6	Layer 11	Layer 13
(a)		57.01%	54.97%	52.25%
(b)		15.5%	11.9%	9.22%

Figure 2: (top) Filter activations in ResNet50; (bottom) Percentage of filters shared between (a) semantically similar "dolphin" and "whale"; (b) semantically dissimilar "dolphin" and "table".

Key Observations. *This experiment indicates that filters in earlier layers are less specialized than filters in deeper layers*. Moreover, it remarks that filters from semantically similar classes get similarly activated, especially in earlier layers. *To put it in more quantitative terms, the L_1 distance of the activation maps of the mentioned classes in the 40^{th} layer is 0.028, while the same for the 49^{th} layer is 0.111*. To further investigate this critical aspect, we have performed additional experiments where we have computed the percentage of filters "shared" among different classes for each layer of VGG16. Specifically, we have tagged each filter with the top 20 classes for which it gets activated. For each pair of classes, their similarity is calculated as the number of filters tagged with both classes over the number of filters tagged with at least one of the classes. The results are shown in the bottom portion of Figure 2, where the first row shows the filters shared between the "dolphin" and "whale" classes – two semantically similar classes. The second row shows the filter sharing between two semantically dissimilar classes - "dolphin" and "table". *We notice that the semantically similar classes share more filters*.

A key issue is to define the subgraphs corresponding to each semantic cluster. We formalize this step as Semantic DNN Subgraph Problem (SDSP). We consider a DNN $\mathcal{F}$ trained on dataset $\mathcal{D}$ as a computation graph, while the filters of the DNN work as the nodes of the graph.

Semantic DNN Subgraph Problem (SDSP)

Find K subgraphs $\mathcal{F}_{\gamma_i} \cdots \mathcal{F}_{\gamma_K}$ such that

$$\mathcal{B}_{eval}(\mathcal{F}, \mathcal{D}_{\gamma_i}) \leq \mathcal{B}_{eval}(\mathcal{F}_{\gamma_i}, \mathcal{D}_{\gamma_i}) + \epsilon, \quad (1)$$

where ϵ is an error margin and $\mathcal{F}_{\gamma_i} \subset \mathcal{F}$ and $\mathcal{D}_{\gamma_i} \subset \mathcal{D}$ are respectively the proper subgraphs of $\mathcal{F}$ and subset of data corresponding to the semantic cluster γ_i. The function $\mathcal{B}_{eval}$ is the metric to measure the performance of the DNN on the subsets of dataset corresponding to semantic clusters. A higher value of $\mathcal{B}_{eval}$ corresponds to better performance.

In other words, the subgraph $\mathcal{F}_{\gamma_i}$ contains the nodes of $\mathcal{F}$ which best classifies the members of the semantic cluster γ_i within the error margin of ϵ.

4. Discriminative Capability Score

We design a novel algorithm named Discriminative Capability Score (DCS), which aims at satisfying Equation 1 by extracting the filters from each layer of a DNN which best discriminate among the members of a semantic cluster γ. We describe the DCS for a given layer l and given semantic cluster γ_m in algorithm 1 (in Appendix). We start by considering the activation map $\mathbf{A}_l^j \in \mathbb{R}^{C_{out}^l \times k \times k}$ of a generic layer l of a DNN for input X^j(with target label t^j) $\in \mathcal{D}_\gamma$. Here, C_{out}^l, and k is the number of channels and size of a single channel of the activation map. We apply an adaptive average pooling operation $\mathcal{P}$ to obtain $\tilde{A}_l^j \in \mathbb{R}^{C_{out}^l \times k' \times k'}$, where k' is the reduced size of a single channel of the activation map. We then flatten the activation map to obtain feature map $\mathbf{F_l^j} \in \mathbb{R}^{C_{out}^l k'^2}$ for the layer l and input X^j. Our goal is to first learn a linear transformation $\mathbf{W}_l \in \mathbb{R}^{|\gamma| \times C_{out}^l k'^2}$ ($|\gamma|$ = cardinality of set γ) that can distinguish the members of γ from the feature maps $\mathbf{F}_l^j$. To this end, we minimize the objective function $\mathcal{L}_{DOF}$:

$$\mathbf{W}_l^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{|\mathcal{D}_{\gamma_m}|} \sum_{j=1}^{j=|\mathcal{D}_{\gamma_m}|} \mathcal{L}_{DOF}(\mathbf{W}_l \cdot \mathbf{F}_l^j, t^j), \quad (2)$$

Once the transformation $\mathbf{W}_l$ is learned, the importance of the features and in turn the filters, are encoded in $\mathbf{W}_l$.

We provide an example of DCA in Figure 3. For simplicity, we show the feature vector and the weight matrix in transposed form. Notice that each column of the weight matrix $\mathbf{W}_l$ connects a single feature to the outputs. The weights of these connections can be used to directly measure the importance of the feature. The importance of the i-th feature in discriminating among the members of the cluster depends on the weight of its connections to the outputs and on the sensitivity of those weights, i.e., the gradient of the objective function with respect of those weights. We formalize this notion in Proposition **??**.

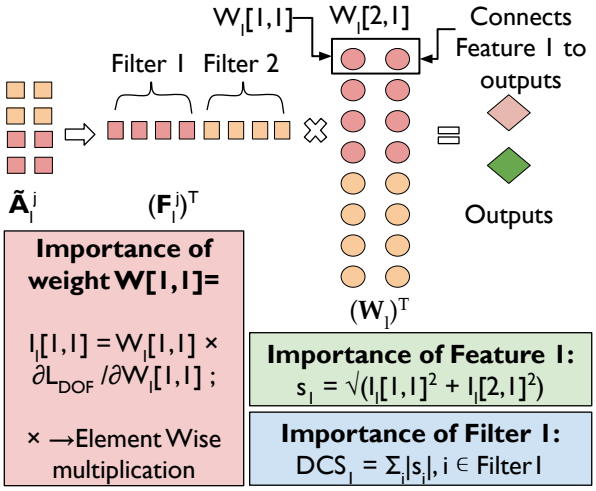


Figure 3: Example of computation of DCS score.

As a result, the importance of i-th feature can be calculated as $s_i = \sqrt{\sum_{j=1}^{C_{out}^l k'^2} \mathbf{I}[j,i]^2}$. As k'^2 consecutive features come from the same filter, we calculate the DCS of the i-th filter of the l-th layer as $\text{DCS}_i^l = \sqrt{\sum_j |s_j|}, j \in$ Filter i, where j denotes indices of the features that come from the i-th filter.

Figure 4 shows the DCS distributions obtained in layers 6, 9, 11, and 13 of VGG16 by considering the cluster "fish" of CIFAR100. Figure 4 confirms that deeper layers are more specialized for individual classes, and thus the average DCS for the filters in the deeper layers is smaller, i.e., 0.68 for layer 6 vs 0.39 for layer 13. As such, DCS captures the filter activation pattern of the DNN.

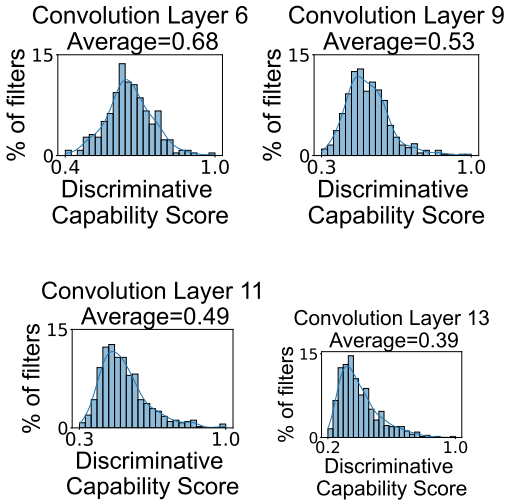


Figure 4: The DCS distribution for cluster "fish" of CIFAR100 in different layers of VGG16.

5. Dynamic Semantic Inference (SINF)

A step-by-step overview of the main operations of SINF is summarized in the top portion of Figure 5. A key challenge is assigning each incoming input to a semantic cluster. For this reason, we divide the DNN into two portions – a *Common Feature Extractor* (CFE) and the *Semantic Subnetworks* (SSN). The output of the CFE is used by a *Semantic Route Predictor* (SRP) that assigns the input to a semantic cluster (**step 1**). To this end, the features extracted by the CFE are passed to the SRP (**step 2**). The SRP, detailed later in this section, provides both the predicted semantic cluster and its confidence in its prediction to the *Feature Router* (FR) (**step 3**). Based on the SRP output, the features extracted by the CFE will be routed to the selected semantic subgraph by using the FR (**step 4**). Finally, the extracted subgraphs predict the output (**step 5**). Although Figure 5 represents each subgraph separately for better graphical clarity, no additional memory beyond the annotations is needed to characterize each subgraph.

Semantic Route Predictor. The SRP predicts the semantic clusters using an auxiliary classifier $\boldsymbol{\chi}$, which is attached after the $M-1$-th layer of $\mathcal{F}$. In our experiments, we chose as M the earliest layer providing classification accuracy of 75%. As such, the layers of $\mathcal{F}$ up to the $M-1$-th layer become the CFE. In our current design, the architecture of the auxiliary classifier consists of two convolutional layers, followed by an adaptive average pooling layer stacked on top of three fully connected layers. We use the convolutional layers to tailor the activation map from layer l of $\mathcal{F}$ for the classification of the semantic clusters. To train the auxiliary classifier $\boldsymbol{\chi}$, the first $M-1$ layers of $\mathcal{F}$ are frozen and the classifier is trained in supervised fashion

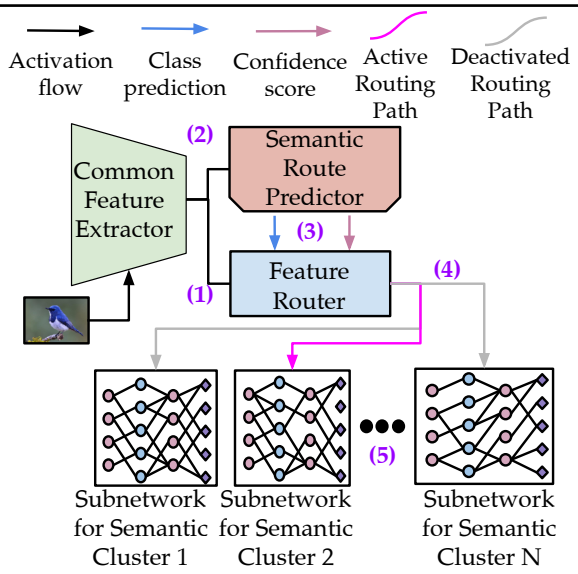


Figure 5: Overview of SINF.

using $\{\mathbf{A}^j_{M-1}, \gamma^j_m\}_{j=1}^{j=|\mathcal{D}|}$ as the dataset. Here, $\mathbf{A}^j_{M-1}$, and γ^j_m are respectively the activation of the $M-1$-th layer of the $\mathcal{F}$ and the ground truth semantic cluster for the j-th sample. As we are considering a pre-trained DNN, we train the auxiliary classifier separately from the DNN using the activations obtained from the $M-1$-th layer. The output of the SRP is the probability distribution over the K different semantic clusters, and the input is assigned to the semantic cluster with the highest probability.

Extraction of Subgraphs. The extraction of the subgraph follows the procedure described in Algorithm 2 in Appendix C. We define L and M as respectively the last layer of the base model $\mathcal{F}$ and the layer after the CFE. We define r_l as the percentage of retained filters in generic layer l. For semantic cluster γ_i, we iterate from layer L to layer M to extract the subgraph. For each layer $M \leq l \leq L$, we calculate $r_l (r_L \leq r_l \leq r_M)$, as well as the DCS score of the filters using DCS algorithm. For each cluster, we rank the filters based on the DCS score, and the indices of the top r_l percent filters are saved. If the average accuracy of the subgraphs is above threshold τ_{acc}, the indexes of the filters belonging to the subgraphs are stored. This procedure is performed for different values of r_L and r_M. Further details are provided in Section D.

Feature Router. The effectiveness of the DCS score can be improved by conditioning the outputs of the SRP to the confidence of the SRP $\boldsymbol{\chi}$. A higher confidence value represents a higher probability that the SRP is able to correctly place the input in the proper semantic cluster. The Feature Router (FR) calculates this confidence by taking the activation map from $\boldsymbol{\chi}$ along with the probability distribution from its prediction layer. To compute the confidence of the classifier on individual decisions, the FR employs the lightweight metric

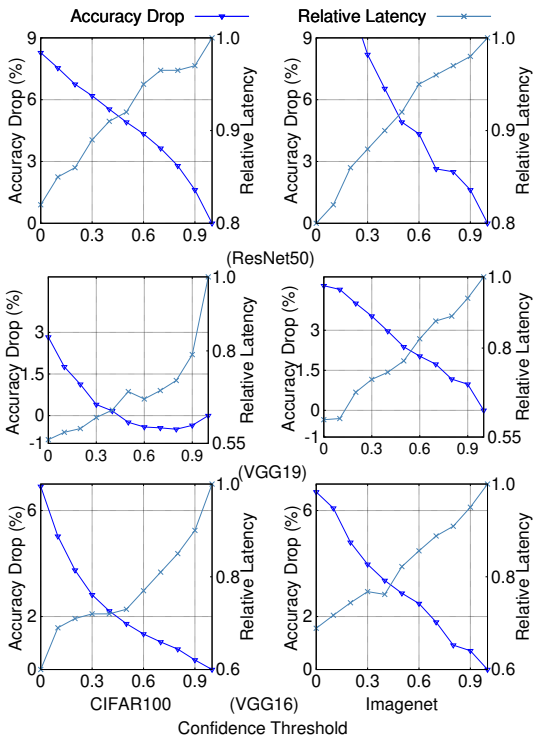


Figure 6: Accuracy and inference time vs confidence threshold for CIFAR100 (left) and ImageNet (right).

in (Park et al., 2015). The confidence score can be calculated as $C_\chi = P_h - P_{sh}$, using the highest (P_h) and the second highest probabilities (P_{sh}) for individual semantic clusters. If the confidence score exceeds a threshold, the activation map is routed to the subgraph corresponding to the predicted semantic cluster. Otherwise, $\mathcal{F}$ is fully executed to obtain the inference task output.

6. Experimental Results

We perform an extensive set of experiments to understand the utility of SINF. We defer the experimental setup to supplementary section D and discuss the experimental results here.

6.1. Impact of Confidence Threshold

We first evaluate the impact of the confidence threshold α. The top row of Figure 6 shows the decrease in accuracy and the relative latency with respect to the original DNN as a function of α. As expected, increasing α increases the accuracy while also decreasing the gain in latency. As such, the confidence threshold α acts as a hyperparameter to find the needed trade-off between accuracy and latency. We notice that with VGG19, the overall accuracy actually increases by up to 0.49% for $0.4 \leq \alpha \leq 0.9$. In the best case, SINF reduces the inference time by up to 35%, 29% and 15% with only 0.17%, 3.75%, and 6.75% accuracy loss

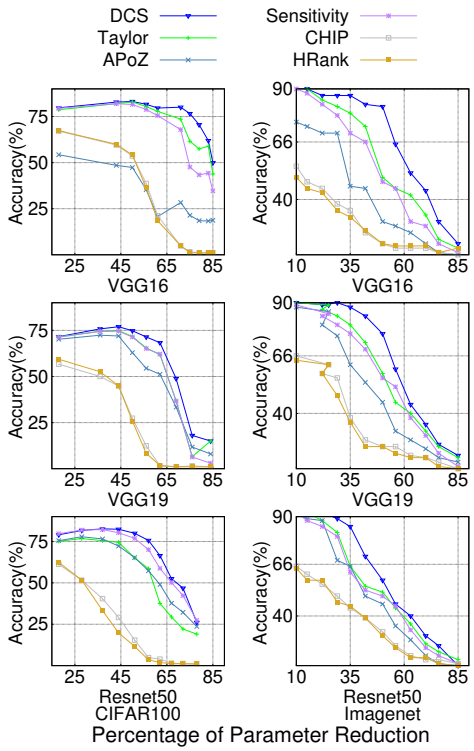


Figure 7: Comparison between DCS and other parameter reduction approaches on CIFAR100 (left) and ImageNet (right).

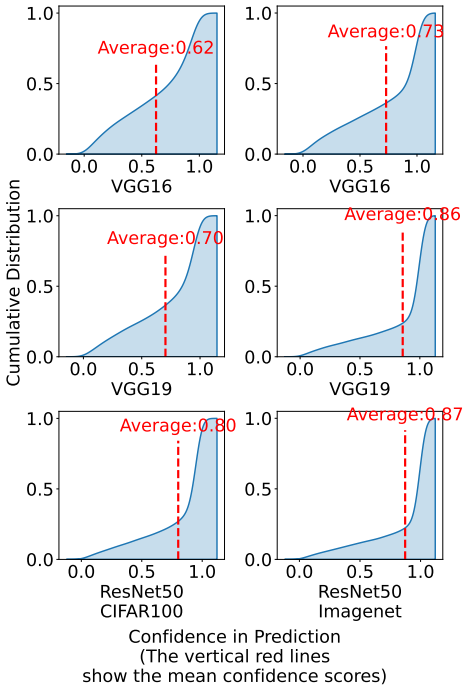


Figure 8: Cumulative distribution of the confidence values of SINF for the VGG16, VGG19, and ResNet50 DNNs for CIFAR100 (left) and ImageNet (right) benchmarks respectively.

for VGG19, VGG16, and ResNet50 respectively. For the ImageNet benchmark, we observe that SINF achieves 18%, 22%, and 9% less inference time with an accuracy degradation of 3%, 2.5%, and 6% respectively. This effect may be due to many filters being shared among the semantic clusters creating polysemantic neurons (Olah et al., 2017). As a result, the gain from such partitioning is smaller than the same for datasets like CIFAR100. This gain in performance depends on the frequency of activation of the semantic subgraphs. As such, we analyze the cumulative distribution of the confidence values in Figure 8 with CIFAR100 (top) and ImageNet (bottom) datasets. Figure 8, shows that for both CIFAR100 and ImageNet datasets, we obtain high confidence value. The distribution of confidence values is skewed to the right with a high mean, which translates to a high frequency of activation of the subgraphs.

6.2. DCS vs Existing Discriminative Metrics

To evaluate the effectiveness of DCS with respect to prior approaches, we use the metrics proposed in existing work while keeping the same inference structure of SINF. Figure 7 compares DCS against gradient-based approaches *Sensitivity* by (Mittal et al., 2019) and *Taylor* by (Molchanov et al., 2019), sparsity of activation based approach *APOZ* by (Hu et al., 2016), channel-independence based approach *CHIP* by (Sui et al., 2021), and an approach based on channel importance named *HRANK* by (Lin et al., 2020). All the approaches are compared without retraining the DNN. Figure 7 shows that in the best case, DCS has 15% higher accuracy than the second-best approach *Taylor* for VGG16 with 75% sparsity (i.e., percentage of parameters dropped). For VGG19, DCS achieves in the best case 6.54% higher accuracy than the second-best approach *Taylor* at 63% sparsity. Lastly, in the case of ResNet50, the best case is attained at 51% sparsity, where DCS presents 14.87% more accuracy than the second-best approach *Sensitivity*. On average, SINF achieves 3.65%, 4.25%, and 2.36% better accuracy than the second-best approaches for VGG16, VGG19, and ResNet50 respectively on CIFAR100 dataset. For ImageNet dataset, SINF outperforms the second best approaches by 8.9%,5.8%, and 5.2% on average for VGG16, VGG19, and ResNet50.

6.3. Visualization of Discriminative Features from Subgraphs

To understand if the extracted subgraphs have sufficient discriminative capability, we utilize the feature map generated from the convolution layer backbone of the DNN. Our objective is to evaluate whether the features are sufficiently separated from each other. We plot the t-distributed stochastic neighbor embedding (t-SNE) visualization of the features extracted by VGG16 for the "flowers" cluster of CIFAR100. For comparison, we plot the t-SNE visualization

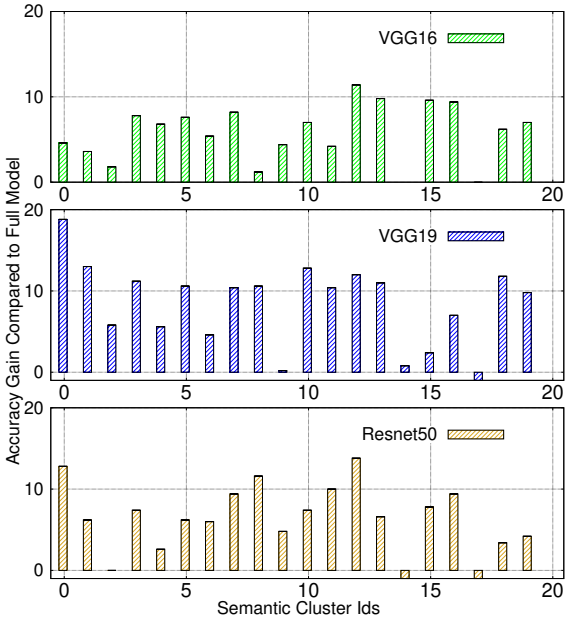


Figure 9: Performance gain compared to the original DNN.

for the original DNN in Figure 10 (a). From Figure 10 (b), we observe that the feature maps are sufficiently separated to obtain good performance.

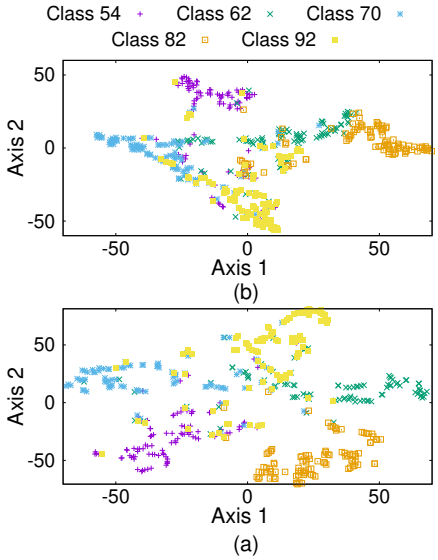


Figure 10: Comparison of the original model (a) and extracted subgraph (b) using TSNE visualization of the extracted features.

6.4. Per-Cluster Accuracy Gain

We ask the question: "*Can SINF perform better than the original DNN when considering the accuracy obtained in individual clusters?*". Figure 9 shows the accuracy gain obtained on the individual clusters by those subgraphs as compared to the original VGG16, VGG19, and ResNet50

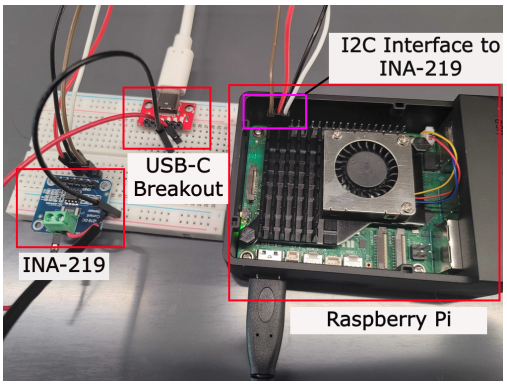


Figure 11: The setup for power measurement.

DNNs. In these experiments, we find the subgraphs with the lowest percentage of parameters retained while satisfying the constraint on the evaluation criterion posed in Equation 1. Intriguingly, SINF provides on the average 5.73%, 8.38% and 6.36% better per-cluster accuracy than the original VGG16, VGG19, and ResNet50 DNNs, respectively, notwithstanding that the number of parameters have been reduced by 30%, 50%, and 44%. We believe the reason behind this improvement is that the semantic partitioning performed by SINF improves the DNN explainability saving the DNN from being "less confused" among different semantic clusters, which justifies better results when considering per-cluster accuracy.

6.5. Latency and Energy Consumption on Mobile Devices

We considered Raspberry-Pi-5 and Nvidia Jetson-Nano as example devices for running mobile computer vision (CV) applications. Our experimental setup is shown in Figure 11. The Raspberry Pi runs a quad-core ARM A76 SoC running at upto 2.4 GHz with 8 GB LPDDR4 memory. In addition to being powered by a quad-core ARM Cortex A57 CPU, Jetson-Nano is equipped with 128 core Maxwell GPU.

Energy Consumption: We measure the power consumption of SINF and compare against the power consumption of the considered baselines. From 12, we show that SINF outperforms the other methods in terms of power efficiency. Specifically, for Raspberry Pi 5 and Jetson Nano devices SINF respectively achieves 50% and 52% better energy efficiency on CIFAR100 benchmark and about 21% and 22% better energy efficiency on the ImageNet benchmark as compared to the second best approach *Taylor*. When comparing with the base (static) model, SINF uses about 50% less power for CIFAR100 and 38% less power for ImageNet benchmark.

Inference Latency: We consider the time introduced by the common feature extractor (static part), semantic route predictor, and the semantic cluster specific subgraph (dy-

DNN	Dataset	Pruned	Criterion	Accuracy Loss	Difference
VGG16	CIFAR100	40.2% 43.8%	IterTVSPrune DCS	18.59% 8.84%	+9.75% Accuracy -3.6% Parameters
VGG16	CIFAR10	37.6% 42%	IterTVSPrune DCS	1.9% 1.29%	+0.61% Accuracy -4.4% Parameters
VGG19	CIFAR100	59% 59%	IterTVSPrune DCS	5.2% 5.15%	+0.05% Accuracy +0% Parameters
VGG19	CIFAR10	49% 49.65%	IterTVSPrune DCS	1.3% 0.9%	+0.4% Accuracy -0.65% Parameters
ResNet50	CIFAR10	34.1% 39.92%	IterTVSPrune DCS	9.94% 1.81%	+8.13% Accuracy -5.82% Parameters
ResNet50	ImageNet	9.98% 12%	IterTVSPrune DCS	10.21% 7.8%	+2.41% Accuracy -2.02% Parameters

Table 2: Using DCS as a Pruning Criterion vs *IterTVSPrune* (ICLR 2023).

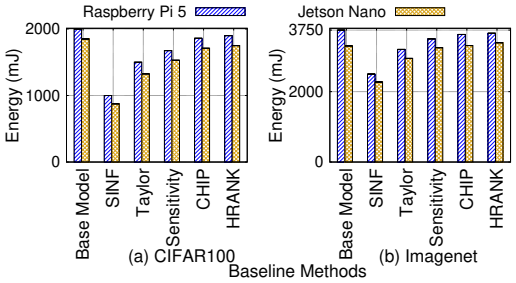


Figure 12: Power, SINF vs baselines.

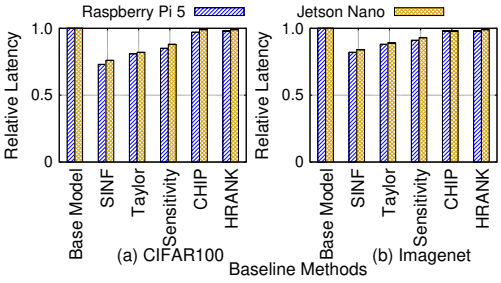


Figure 13: Latency, SINF vs baselines.

namic part). Our experiment shows that on an average, SINF achieves the lowest inference latency compared to the baselines which includes the base model itself. Specifically, SINF takes about 27% and 24% less time than the base model for the CIFAR100 benchmark while running on the Raspberry Pi 5 and Jetson Nano respectively. For the ImageNet Benchmark, SINF takes about 18% and 16% less inference time on Raspberry Pi 5 and Jetson Nano respectively than the base model itself. When considering the task adaptive deployment setting, we measure the latency of communication. We observe that sending the semantic subgraph reduces the average communication latency about 52% than sending the base model for CIFAR100 benchmark. For the ImageNet benchmark, the communication latency is reduced by about 30%.

6.6. DCS as Pruning Criterion

Viewing the dataset $\mathcal{D}$ as a single macro-cluster DCS can be applied to determine the most relevant filters effectively acting as a pruning criterion. For comparison, we consider the state-of-the-art *IterTVSPrune* by (Murti et al., 2023) published at ICLR 2023, which also does not require fine-tuning. For a fair comparison, we have taken the percentage of parameters pruned by *IterTVSPrune* at each layer and set the same pruning threshold for DCS. 2 summarizes the performance achieved by *DCS* and *IterTVSPrune*. We did not compare performance on CIFAR100 with ResNet50 as the authors of (Murti et al., 2023) did not provide the performance of their approach on ResNet50 trained with CIFAR100. We notice that for different DNN structures and datasets, DCS achieves substantially better performance in three out of five settings considered while achieving similar performance in the remaining two settings. For VGG19, both our technique and the IterTVSPrune have pruned a significant amount of weights – respectively about 50% and 60% for CIFAR10 and CIFAR100 – possibly causing the DNN to reach a lower bound on its predictive capability thereby causing similar performance of both techniques. The best results are obtained in the case of VGG16 trained on CIFAR100 and ResNet50 trained on CIFAR10, where we see respectively 9.75% and 8.13% accuracy gain with 3.6% and 5.82% less parameters. On average, DCS presents 3.78% better accuracy and 2.89% less parameters than *IterTVSPrune*.

7. Concluding Remarks

In this paper, we propose *Semantic Inference* (SINF), a framework for accelerating DNNs without compromising accuracy. Central to SINF is the *Discriminative Capability Score* (DCS), which we employ to identify subgraphs within large DNNs that discriminate specific semantic clusters. Unlike model compression (e.g., pruning, quantization), SINF requires no fine-tuning and operates at the cluster level. We have benchmarked SINF on VGG16, VGG19, and ResNet50 using CIFAR100 and ImageNet, showing promising performance as compared to the existing methods and proving its efficiency for edge deployment.

Acknowledgement

This work has been funded in part by the National Science Foundation under grant CNS-2312875, by the Air Force Office of Scientific Research under contract number FA9550-23-1-0261, by the Office of Naval Research under award number N00014-23-1-2221.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ao Wang, Hui Chen, L. L. e. a. Yolov10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458*, 2024.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6541–6549, 2017.

Cai, Y., Yao, Z., Dong, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13169–13178, 2020.

Chen, Y., Ma, Z., Fang, W., Zheng, X., Yu, Z., and Tian, Y. A unified framework for soft threshold pruning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cCFqcrq0d8.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dong, R., Mao, Y., and Zhang, J. Resource-constrained edge ai with early exit prediction. *Journal of Communications and Information Networks*, 7(2):122–134, 2022. doi: 10.23919/JCIN.2022.9815196.

Gajjala, R. R., Banchhor, S., Abdelmoniem, A. M., Dutta, A., Canini, M., and Kalnis, P. Huffman coding based encoding techniques for fast distributed deep learning. In *Proceedings of the 1st Workshop on Distributed Machine Learning*, DistributedML'20, pp. 21–27, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381826. doi: 10.1145/3426745.3431334. URL https://doi.org/10.1145/3426745.3431334.

Garg, S. and Moschitti, A. Will this question be answered? question filtering via answer model distillation for efficient question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7329–7346, 2021.

Han, D.-J., Park, J., Ham, S., Lee, N., and Moon, J. Improving low-latency predictions in multi-exit neural networks via block-dependent losses. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–9, 2023. doi: 10.1109/TNNLS.2023.3282249.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pp. 1135–1143, 2015b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Hu, H., Peng, R., Tai, Y., and Tang, C. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *CoRR*, abs/1607.03250, 2016. URL http://arxiv.org/abs/1607.03250.

Krizhevsky, A. and Hinton, G. Learning Multiple Layers of Features from Tiny Images. 2009.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rJqFGTslg.

Li, H., Yan, C., Lin, S., Zheng, X., Li, Y., Zhang, B., Yang, F., and Ji, R. Pams: Quantized super-resolution via parameterized max scale. *arXiv preprint arXiv:2011.04212*, 2020.

Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., and Shao, L. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1529–1538, 2020.

Matsubara, Y., Levorato, M., and Restuccia, F. Split Computing and Early Exiting for Deep Learning Applications:

Survey and Research Challenges. *ACM Computing Surveys (CSUR)*, 2021.

Mittal, D., Bhardwaj, S., Khapra, M. M., and Ravindran, B. Studying the plasticity in deep convolutional neural networks using random pruning. *Machine Vision and Applications*, 30(2):203–216, 2019.

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., and Kautz, J. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Murti, C., Narshana, T., and Bhattacharyya, C. TVSPrune - pruning non-discriminative filters via total variation separability of intermediate representations without fine tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sZI1Oj9KBKy.

Narayan, H., Hanawal, M. K., and Bhardwaj, A. Unsupervised early exit in dnns with multiple exits. In *Proceedings of the Second International Conference on AI-ML Systems*, AIMLSystems '22, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450398473. doi: 10.1145/3564121.3564137. URL https://doi.org/10.1145/3564121.3564137.

Olah, C., Mordvintsev, A., and Schubert, L. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Park, E., Kim, D., Kim, S., Kim, Y.-D., Kim, G., Yoon, S., and Yoo, S. Big/little deep neural network for ultra low power inference. In *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 124–132, 2015. doi: 10.1109/CODESISSS.2015.7331375.

Phuong, M. and Lampert, C. Distillation-based training for multi-exit architectures. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1355–1364, 2019. doi: 10.1109/ICCV.2019.00144.

Pomponi, J., Scardapane, S., and Uncini, A. A probabilistic re-intepretation of confidence scores in multi-exit models. *Entropy*, 24(1), 2022. ISSN 1099-4300. doi: 10.3390/e24010001. URL https://www.mdpi.com/1099-4300/24/1/1.

Qin, H., Ding, Y., Zhang, M., YAN, Q., Liu, A., Dang, Q., Liu, Z., and Liu, X. BiBERT: Accurate fully binarized BERT. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=5xEgrl_5FAJ.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sui, Y., Yin, M., Xie, Y., Phan, H., Zonouz, S. A., and Yuan, B. CHIP: CHannel independence-based pruning for compact neural networks. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=EmeWbcWORRg.

Teerapittayanon, S., McDanel, B., and Kung, H. Branchynet: Fast inference via early exiting from deep neural networks.

Tu, Z., Hu, J., Chen, H., and Wang, Y. Toward accurate post-training quantization for image super resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5856–5865, 2023.

Verdú, S. Total variation distance and the distribution of relative information. In *2014 Information Theory and Applications Workshop (ITA)*, pp. 1–3. IEEE, 2014.

Wang, M., Mo, J., Lin, J., Wang, Z., and Du, L. Dynexit: A dynamic early-exit strategy for deep residual networks. In *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 178–183, 2019. doi: 10.1109/SiPS47522.2019.9020551.

Zhong, Y., Lin, M., Li, X., Li, K., Shen, Y., Chao, F., Wu, Y., and Ji, R. Dynamic dual trainable bounds for ultra-low precision super-resolution networks. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVIII*, pp. 1–18, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-19796-3. doi: 10.1007/978-3-031-19797-0_1. URL https://doi.org/10.1007/978-3-031-19797-0_1.

A. Details of the Experiment Motivating SINF

To obtain the results in the top portion of Figure 2, we have observed the activation pattern of the filters of a specific layer of the DNN for a given class. To obtain the activation pattern of the filters of layer L for a specific class c, we feed the input samples of that class and obtain the activation vectors $A_L \in \mathbb{R}^{N_c \times H \times W}$ from layer L, where N_c, H, W are number of channels, height and width of the activation. Next, we take the mean value of the activations for each filter giving us activation vectors of length N_c for each sample. For future reference, we will denote this with $\tilde{A}_L$. Next, we take the expected values of the activation to obtain the filter activation pattern of layer L for class c. For ease of visualization, we standardize the activation pattern using min-max standardization.

To obtain the filter activation frequency in the bottom side of Figure 2, we take the $\tilde{A}_L$ for each sample and stack them along the sample axis. This gives us a matrix of size $\mathbb{R}^{N_samples \times N_c}$. For each filter f (each column represents a filter of layer L now), we calculate the maximum value of activation max_f and minimum value of activation min_f. We consider the filter to be activated if its activation value exceeds 70% of the range $max_f - min_f$. Next, we obtain class-wise activation frequency $freq_c$ for each filter where c denotes the class for which the activation frequency is being calculated. Next, we assign each filter with the top 20 classes for which it gets most activated.

B. DCS Algorithm

Algorithm 1 DCS: Discriminative Capability Scoring for Filters in Layer l and Semantic Cluster γ

Require: Dataset $\mathcal{D}_\gamma = \{(X^j, t^j)\}_{j=1}^{|\mathcal{D}_\gamma|}$ for semantic cluster γ
Require: Pretrained L-layer DNN $\mathcal{F} = \mathcal{F}_{L-1} \circ \ldots \circ \mathcal{F}_0$
Require: Objective function $\mathcal{L}_{DOF}$
Ensure: Discriminative Capability Score DCS^l for filters in layer l

Initialize score list $\mathbf{s}_l \leftarrow []$
for each (X^j, t^j) in $\mathcal{D}_\gamma$ **do**
  $\mathbf{A}_l^j \leftarrow \mathcal{F}_l \circ \ldots \circ \mathcal{F}_0(X^j)$
  $\tilde{\mathbf{A}}_l^j \leftarrow \mathcal{P}(\mathbf{A}_l^j)$ {adaptive pooling to $k \times k$}
  $\mathbf{F}_l^j \leftarrow$ Flatten$(\tilde{\mathbf{A}}_l^j)$
end for
$\mathbf{W}_l^* \leftarrow \arg\min_{\mathbf{W}} \frac{1}{|\mathcal{D}_\gamma|} \sum_j \mathcal{L}_{DOF}(\mathbf{W} \cdot \mathbf{F}_l^j, t^j)$
$\mathbf{I}_l \leftarrow \mathbf{W}_l^* \odot \nabla_{\mathbf{W}_l^*} \mathcal{L}_{DOF}$
for $i = 0$ to $C_{out}^l - 1$ **do**
  Append $\|\mathbf{I}_l[:, i]\|_2$ to $\mathbf{s}_l$
end for
DCS$_i^l \leftarrow \sqrt{\sum_j |s_j|}$ {j indexes features of filter i}

C. Algorithm for Subgraph Extraction

Algorithm 2 Subgraph Extraction for Semantic Clusters

Require: Partitioned Dataset $\mathcal{D} = \{\mathcal{D}_{\gamma_1}, \mathcal{D}_{\gamma_2}, \ldots, \mathcal{D}_{\gamma_K}\}$
Require: Pretrained DNN $\mathcal{F} = \mathcal{F}_{L-1} \circ \ldots \circ \mathcal{F}_0$
Require: Discriminative Objective Function $\mathcal{L}_{DOF}$
Require: Filter retention percentages r_L at layer L, r_M at layer M
Require: Accuracy threshold τ_{acc}
Ensure: Filter annotations $\mathbf{SA}$[] for extracted subgraphs

1: Initialize empty dictionary $\mathbf{SA} \leftarrow \{\}$
2: **for** each $\mathcal{D}_{\gamma_i}$ in $\mathcal{D}$ **do**
3: $\mathbf{SA}_{\gamma_i} \leftarrow \{\}$
4: **for** $l = L$ to M decreasing **do**
5: $r_l \leftarrow r_M + \frac{(l-M)(r_L - r_M)}{L-M}$ {Linear interpolation of retention rate}
6: DCS$^l \leftarrow DCS(\mathcal{F}, \mathcal{D}_{\gamma_i}, \mathcal{L}_{DOF})$
7: Rank filters in layer l by DCS^l
8: Save indices of top $r_l\%$ filters in $\mathbf{SA}_{\gamma_i}[l]$
9: **end for**
10: $acc_{avg} \leftarrow \frac{1}{K} \sum_{i=1}^{K}$ accuracy$(\mathcal{F}_{\gamma_i})$
11: **if** $acc_{avg} \geq \tau_{acc}$ **then**
12: Save $\mathbf{SA}_{\gamma_i}$ in $\mathbf{SA}[\gamma_i]$
13: **end if**
14: **end for**

D. Experimental Setup

Hyperparameters: We set r_L between 90% and 10%, with steps of 10, while r_M is set between 10% and 1%, with steps of 2. This way, we vary the number of retained filters in different layers allowing us to find multiple sub-graphs satisfying our constraint set discussed in Section 3. Based on the application-level performance constraint, we can choose the optimum model based on additional requirements (e.g. subgraph size, latency). We linearly decrease the percentage of filters retained from layer M to layer L according to the Equation presented in Line 5 of Algorithm 2. Since we are considering classification tasks, categorical cross entropy is used for $\mathcal{L}_{DOF}$.

Dataset, Base DNNs, and Baselines. We have used the well-known CIFAR100 (Krizhevsky & Hinton, 2009) and ImageNet (Deng et al., 2009) for image classification. CIFAR100 has 100 classes and the entire dataset is labeled into 20 super-classes corresponding to 20 coarse labels corresponding to our semantic clusters. With ImageNet we form 6 semantic clusters each consisting of 5 fine-grained classes. The semantic clusters formed are *fish*, *bird*, *lizards*, *animal*, *insects*, and *seafish*. A summary of the semantic clusters and their member classes are provided in Table 3. We have considered VGG16 and VGG19 (Simonyan & Zisserman, 2014) as well as ResNet-50 (He et al., 2016) for base DNNs.

To the best of our knowledge, there is no prior work on semantic clustering. As such, we adapt pruning methods to use them without fine-tuning, i.e., (Molchanov et al., 2019), (Hu et al., 2016), (Mittal et al., 2019), (Sui et al., 2021), and (Lin et al., 2020).

Semantic Cluster	Class Label (Wordnet Label)
Fishes	2 (great white shark), 3 (tiger shark), 4 (hammerhead shark), 5 (electric ray), 6 (sting ray)
Birds	10 (brambling), 11 (goldfinch), 12 (house finch), 13 (junco), 14 (indigo bunting)
Lizards	37 (box turtle), 38 (banded gecko), 39 (common iguana), 40 (American chameleon), 41 (whiptail)
Animals	253 (basenji), 261 (keeshond), 276 (hyena), 283 (persian cat), 298 (mongoose)
Insects	305 (dung beetle), 308 (fly), 309 (bee), 310 (ant), 311 (grasshopper)
Sea fish	393 (anemone fish), 394 (sturgeon), 395 (gar), 396 (lionfish), 397 (puffer)

Table 3: Summary of the semantic clusters formed from ImageNet dataset.