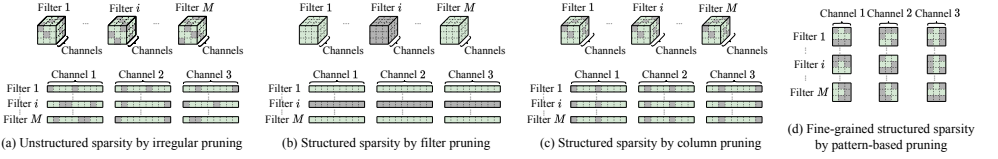
Appendices

A Weight pruning



(a) Unstructured sparsity by irregular pruning (b) Structured sparsity by filter pruning (c) Structured sparsity by column pruning (d) Fine-grained structured sparsity by pattern-based pruning

Figure A1: Various DNN sparsity schemes by weight pruning.

Figure A1 illustrates weight pruning methods using a CONV layer with M filters, each consisting of 3 channels with the kernel size of 3×3. "Kernels" and "channels" can be used interchangeably. The most common kernel size is 3×3, while the number of channels can go much beyond 3.

Figure A1 (a) shows the unstructured sparsity scheme by the irregular pruning method [19, 42, 72], where the upper part is directly in the weight tensor format and the lower part is in the General Matrix Multiplication (GEMM) matrix format. The GEMM routine is widely used for CONV layer operations on some computing platforms such as embedded systems and mobile devices [10]. Irregular pruning method prunes weights at arbitrary locations. It can achieve very high pruning rate, but the unstructured weight sparsity is not compatible with data parallel executions on the computing platforms.

Figure A1 (b) is a structured sparsity scheme by the filter pruning method [20, 41, 64], and Figure A1 (c) is a structured sparsity scheme by the column pruning method [37, 73]. Filter pruning, as suggested by the name, prunes whole filters from a layer.[1] Column pruning prunes weights for all filters in a layer, at the same locations. In the GEMM matrix format, filter pruning prunes whole rows of weights, and column pruning prunes whole columns of weights. The structured sparsity maintains the full matrix format with reduced dimensions, thus accelerating on-device inference for the resource-constrained computing platforms.

Figure A1 (d) shows a fine-grained structured sparsity scheme by the pattern-based pruning method [43, 49, 67], which is a combination of kernel pattern pruning and connectivity pruning. In kernel pattern pruning, for each kernel in a filter, a fixed number of weights are pruned, and the remaining weights form specific kernel patterns. The example in Figure A1 (d) is defined as 4-entry kernel pattern pruning, since every kernel reserves 4 non-zero weights out of the original 3×3 kernels. The connectivity pruning cuts the connections between some input and output channels, which is equivalent to removing corresponding kernels. Note that the pattern-based pruning is not based on the GEMM matrix format.

B Comparison with Adversarial Examples

Although our proposed intrinsic examples leverage similar algorithms (e.g., PGD) as those for generating adversarial examples, in Table A1 we demonstrate the stronger transferability

[1] Note that some references mention channel pruning [20], which by the name prunes some channels completely from the filters. Channel pruning is essentially equivalent to filter pruning, since pruning some filters in a layer invalidates the corresponding channels of the next layer.

of intrinsic examples than adversarial examples, from the pretrained model F_θ to the implemented model $F_{\theta'}$, which is derived by the unstructured pruning method on F_θ.

We use the (dense) CNN as the pretrained model and perform weight pruning with different pruning ratios. Intrinsic score (i.e., accuracy) by the intrinsic examples and attack success rate (i.e., accuracy) by the adversarial examples are shown in the table.

Algorithm	Dense Model	Models by Unstructured Pruning [19] with Various Pruning Ratios			
Pruning Ratio	/	80%	90%	95%	97%
Model Acc.	80.5%	80.3% (-0.2)	80.3% (-0.2)	79.7% (-0.8)	78.5% (-2.0)
Alg.1 (intrinsic score)	100%	98%	88%	64%	36%
Alg.2 (intrinsic score)	100%	**100%**	**100%**	**100%**	**73%**
Alg.3 (intrinsic score)	100%	**100%**	**100%**	83%	54%
Adversarial example (attack success rate)	100%	83%	76%	52%	33%

The experiment is evaluated on 200 examples generated from CIFAR-10 dataset.

Table A1: Intrinsic Score by Intrinsic Examples and Attack Success Rate of Adversarial Examples Using CIFAR-10 Dataset with Whole Model Weight Pruning. The second column is for pretrained model, and the third to sixth columns are for pruned models.

C Training Loss

Figure A2 shows the training loss during the min-max robust optimization process, where fluctuations due to the inner maximization steps are observed while following an overall decreasing trend.

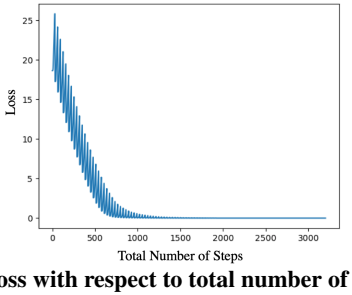


Figure A2: **Training Loss with respect to total number of steps for Algorithm 3.**

D Experiment Settings

In this section, we summarize the details of datasets and models used in our experiments in Table A2. We also conclude our experiment settings for model compression methods in Table A3.

Dataset	Task	Model	# Layers	# Conv	# FC
CIFAR-10	Image Classification	CNN	6	4	2
SVHN	Image Classification	CNN	6	4	2
ImageNet	Image Classification	VGG-16	16	13	3

Table A2: **Details of Datasets and Models Used.**

Dataset	Model	Test Accuracy of Pretrained Model	Generation Algorithms	Weight Pruning				Weight Quantization		
				Unstructured Pruning	Irregular Pruning	Column Pruning	Pattern Pruning	Decimal	Float16	Full Integer
CIFAR-10	CNN	80.5%	Algorithm 1, 2, 3	√	√	√	√	√	√	√
	VGG16	79.5%	Algorithm 1, 2, 3	√	√	√	√	√	√	√
SVHN	CNN	92.8%	Algorithm 1, 2, 3	√	√	√	√	√	√	√
ImageNet	VGG16	Top1: 74.6% Top5: 92.4%	Algorithm 1, 2, 3	√			√	√		

Table A3: Datasets, Models, and Model Compression Methods used for Evaluating Intrinsic Examples. Here the accuracy of intrinsic examples on the pretrained model is 100%.

E Experimental Results for Model Compression on SVHN dataset

Table A4 and A5 demonstrate our results using three intrinsic example generation algorithms to verify the functionality in two pruning modes: pruning the whole model or pruning only one layer. And Table A6 presents the verification results in three quantization modes: decimal quantization, float16 quantization and fullinteger quantization.

F Functionality Indication

We test and record the intrinsic score using different number of intrinsic examples for unstructured pruned models with various pruning ratios. Figure A3 demonstrate our results using Algorithm 1, 2, and 3. For each data point in Figure 4 we generate a number N of intrinsic examples and test the intrinsic score, and this process is conducted10 times to obtain the mean and variance of intrinsic score for each number N, as denoted by the solid line and shadow area in Figure A3. and obtain the mean and variance of intrinsic score, as denoted by the solid line and shadow area.

Algorithm	Dense Model	Unstructured Pruning [19]				Irregular Pruning [72]				Column Pruning[20]				Pattern Pruning[49]			
Pruning Ratio	/	80%	90%	95%	97%	70%	80%	90%	95%	50%	60%	70%	80%	70%	80%	90%	95%
Model Acc.	92.8%	92.9% (+0.1)	92.8% (0)	92.5% (-0.3)	92.7% (-0.1)	92.5% (-0.3)	92.4% (-0.4)	91.8% (-1.0)	89.8% (-2.0)	92.1% (-0.7)	91.0% (-1.8)	91.2% (-1.6)	82.4% (-10.4)	92.3% (-0.5)	92.4% (-0.4)	92.2% (-0.6)	92.2% (-0.6)
Baseline(0.025)	100%	68.5%	52%	21.5%	22.5%	4%	4%	6%	5%	6%	6.5%	8%	7%	5.5%	5%	7%	6%
Baseline(0.05)	100%	77%	63%	51%	39%	8%	8.5%	6.5%	8%	10%	10.5%	6.5%	8%	6.5%	6%	5.5%	5.5%
Baseline(0.1)	100%	70%	61.5%	46.5%	37.5%	5%	5.5%	4.5%	7%	5%	7.5%	7.5%	7.5%	5.5%	5.5%	10%	6.5%
Alg.1	100%	99.5%	98.5%	94%	93.5%	**100%**	**100%**	98.5%	86%	99.5%	95%	92%	50.5%	99%	98.5%	98%	99%
Alg.2	100%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**92%**	**100%**	**100%**	**100%**	**100%**
Alg.3	100%	**100%**	**100%**	99%	96%	**100%**	**100%**	**100%**	95.5%	**100%**	99%	98.5%	86.5%	**100%**	**100%**	99%	99%

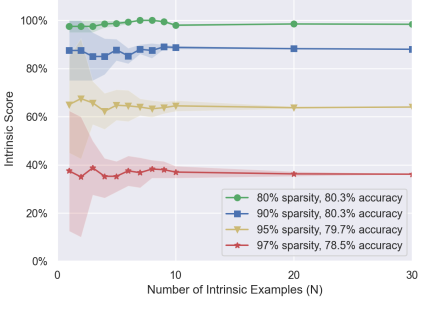
The intrinsic score by each algorithm is evaluated on 200 intrinsic examples generated from the CNN trained with SVHN Dataset.

Table A4: Intrinsic Score of Implemented Models by Different Weight Pruning Methods on the CNN with SVHN Dataset (whole model pruning): We use the (dense) CNN model as the pretrained model and perform different weight pruning methods to derive implemented models with various pruning ratios. Intrinsic score is the accuracy of intrinsic examples on each implemented model.

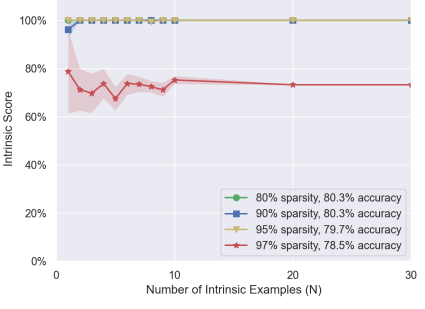
Algorithm	Dense Model	Unstructured Pruning [19]				Irregular Pruning [72]				Column Pruning [20]			
Pruning Ratio	/	80%	85%	90%	95%	90%	95%	97%	99%	90%	95%	97%	99%
Model Acc.	92.8%	92.5% (-0.3)	92.8% (0)	92.9% (+0.1)	92.8% (0)	92.6% (-0.2)	92.6% (-0.2)	92.5% (-0.3)	86.3% (-6.5)	92.6% (-0.2)	92.5% (-0.3)	92.4% (-0.4)	85.4% (-7.4)
Baseline (0.025)	100%	73.5%	69%	61%	62%	45%	40.5%	32%	18%	10.5%	9.5%	8%	6%
Baseline (0.05)	100%	78%	72.5%	70%	70.5%	51%	42%	40.5%	21.5%	16%	13%	12.5%	10%
Baseline (0.1)	77%	73%	72%	72%	68%	60%	53.5%	46%	20%	15%	10.5%	9%	9%
Alg.1	100%	99.5%	**100%**	99.5%	**100%**	99.5%	**100%**	**100%**	88.5%	**100%**	99.5%	99.5%	88%
Alg.2	100%	**100%**	**100%**	99.5%	**100%**	**100%**	**100%**	**100%**	91%	**100%**	**100%**	**100%**	90%
Alg.3	100%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**92.5%**	**100%**	**100%**	**100%**	**92.5%**

The intrinsic score by each algorithm is evaluated on 200 intrinsic examples generated from the CNN trained with SVHN Dataset.

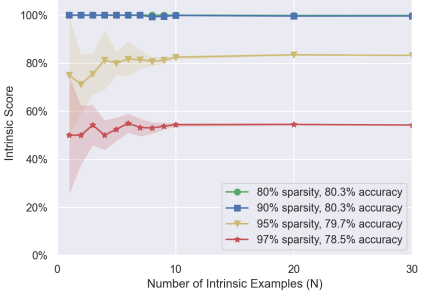
Table A5: **Intrinsic Score of Implemented Models by Different Weight Pruning Methods on the CNN with SVHN Dataset (single layer weight pruning):** We use the (dense) CNN model as the pretrained model and perform different weight pruning methods to derive implemented models with various pruning ratios. Intrinsic score is the accuracy of intrinsic examples on each implemented model.



(a) Algorithm 1



(b) Algorithm 2



(c) Algorithm 3

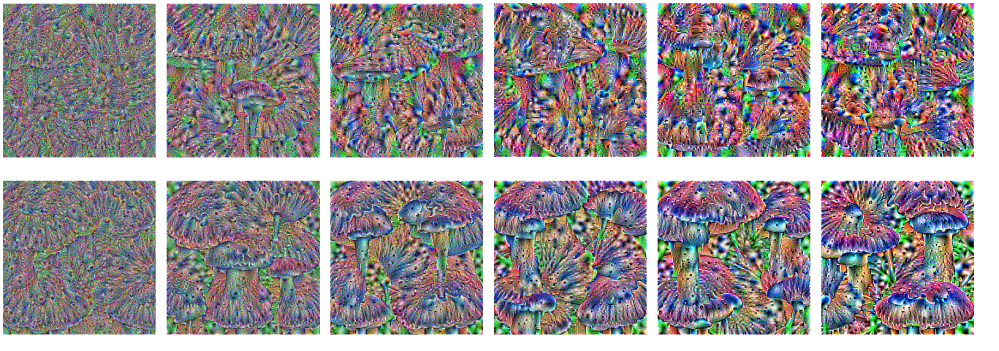
Figure A3: Intrinsic score w.r.t the number of examples on the CIFAR-10 dataset. (a) Intrinsic examples generated using Algorithm 1. (b) Intrinsic examples generated using Algorithm 2. (c) Intrinsic examples generated using Algorithm 3. Each line in the sub-figure represents an unstructured pruned neural network.

Algorithm	Pretrain Model	Decimal			Float16	Full Integer
		3 places	2 places	1 place		
Model Acc.	92.8%	92.8% (0)	92.7% (-0.1)	87.5% (-5.3)	92.8% (0)	92.2% (-0.5)
Alg.1	100%	100%	100%	94.5%	100%	100%
Alg.2	100%	100%	100%	100%	100%	100%
Alg.3	100%	100%	100%	98.5%	100%	100%

Table A6: Intrinsic Score of Implemented Models by Different Weight Quantizationn Methods on the CNN with SVHN Dataset.

G Results on ImageNet Dataset

Figure A4 visualizes intrinsic examples with the same label "mushroom" generated by Algorithm 1 and Algorithm 2, with different ε values. As we can observe, for images from Algorithm 1, it is difficult to figure out its expression based on the textures and pattern. However, image patterns of mushrooms from Algorithm 2 can be clearly observed, reflecting a matching between the high-level semantics and the low-level image features. We summarize the performance of intrinsic examples for functionality verification for different weight pruning and quantization methods as shown in Table A7. We can observe that intrinsic scores of Algorithm 2 outperforms Algorithm 1 for all models, showing its superiority of accurately verifying the acceptable modifications compared with Algorithm 1. Meanwhile, in the case of heavy compression with significant accuracy drop, a relatively low intrinsic score can be obtained (such as 34% from Algorithm 1 and 80% from Algorithm 2 in the case of decimal quantization with 1 place), indicating the unsatisfied performance from the compressed model. The high correlation between the intrinsic score and testing accuracy demonstrates that intrinsic examples can verify the model functionality for large scale image recognition task with a much fewer number of examples.



(a) 32/255 (b) 64/255 (c) 96/255 (d) 128/255 (e) 160/255 (f) 192/255

Figure A4: Generated intrinsic examples on ImageNet dataset. **First Row:** Intrinsic examples generated using Algorithm 1. **Second Row:** Intrinsic examples generated using Algorithm 2. Below we show the input clipping range ε for the two algorithms. For Algorithm 2, we also set the weight perturbation bound $\delta = 0.05$ during the generation process. The label assigned to all these image is “mushroom”.

Alg.	Dense Model	Unstructured Pruning		Pattern Pruning				Decimal Quantization		
		30%	60%	55.6%	77.4%	85.7%	87.5%	3 places	2 places	1 place
Acc.	Top1: 74.6% Top5: 92.4%	74.3% (-0.3)	74% (-0.6)	74.6% (0)	74.3% (-0.3)	73.9% (-0.7)	73.7% (-0.9)	74.5% (-0.1)	72.7% (-1.9)	67.3% (-7.3)
Alg.1	100%	85%	54.5%	**100%**	98%	96.5%	91%	67%	50%	34%
Alg.2	100%	**100%**	**97.5%**	**100%**	**100%**	**100%**	**100%**	**100%**	**89%**	**80%**

The experiment is evaluated on 200 intrinsic examples for each method generated from ImageNet Dataset.

Table A7: Intrinsic Score of Implemented Models by Different Model Compression Methods on VGG-16 with ImageNet Dataset.

H Limitation and Discussion

In our threat model, we assume the client (model owner) does not release the training data of the pretrained model to the system designer or anyone else, because the training data is the even more important intellectual property than the trained models. Such assumption also eliminates another threat scenario that some third-party entity performs the backdoor attack [17] on the pretrained model, because performing the backdoor attack does require access to the training dataset.

If we lift the constraint to allow the client to release the training dataset, our intrinsic examples are unable to detect the integrity breach by the backdoor attack, as the backdoored model has high normal testing accuracy and shows mis-behaviors only in presence of the specific trigger pattern, which is embedded into the training dataset and model. There is an existing work [59] focusing on identification and mitigation of DNN backdoor attacks, which can complement our intrinsic examples allowing for more general threat model.

If we would like to extend the intrinsic examples to incorporate the capability of model integrity breach detection, we may need to use Grad-CAM [54] for its visual and qualitative evaluation capability. Instead of generating intrinsic examples for the pretrained model and testing on the implemented model, we may need to generate intrinsic examples for the implemented model and exam the intrinsic examples with Grad-CAM. Because we only have black-box access to the implemented model, we may incorporate zero-th order optimization [38] when generating intrinsic examples for the black-box implemented model.

Furthermore, we may extend our intrinsic examples to other deep learning tasks such as object detection, speech recognition, and action recognition. We need to deal with large-scale datasets and models in those tasks, and therefore for our proposed Algorithms 2 and 3, we should identify the most effective layers to add weight perturbations.