APPENDIX

In the appendix, we provide more implementation details for our method, experimental results on more datasets and settings, ablation studies, and qualitative analysis. The appendices cover the following:

- **Implementation Details:** NeuralFuse Training Algorithm (Sec. A), NeuralFuse Generator (Sec. B), SCALE-SIM (Sec. C)
- **Qualitative Studies:** Energy-Accuracy Tradeoff (Sec. D), Model Parameters and MAC Values (Sec. E), Data Embeddings Visualization (Sec. J), Transformed Inputs Visualization (Sec. K), Latency Reports (Sec. N)
- **Additional Experimental Results:** Ablation Studies (Sec. F), Relaxed Access (Sec. G), Restricted Access (Sec. H), Reduced-precision Quantization (Sec. I), Adversarial Training (Sec. L), Adversarial Weight Perturbation (Sec. M)

Our code can be found at https://anonymous.4open.science/r/neuralfuse/.

A TRAINING ALGORITHM OF NEURALFUSE

Algorithm 1 Training steps for NeuralFuse

Input: Base model M_0; Generator $\mathcal{G}$; Training data samples $\mathcal{X}$; Distribution of the perturbed models $\mathcal{M}_p$; Number of perturbed models N; Total training iterations T
Output: Optimized parameters $\mathcal{W}_\mathcal{G}$ for the Generator $\mathcal{G}$

for $t = 0, ..., T-1$ **do**
  for all mini-batches $\{\mathbf{x}, y\}_{b=1}^{B} \sim \mathcal{X}$ **do**
    Create transformed inputs $\mathbf{x}_t = \mathcal{F}(\mathbf{x}) = \text{clip}_{[-1,1]}\big(\mathbf{x} + \mathcal{G}(\mathbf{x})\big)$.
    Sample N perturbed models $\{M_{p_1}, ..., M_{p_N}\}$ from $\mathcal{M}_p$ under $p\%$ random bit error rate.
    for all $M_{p_i} \sim \{M_{p_1}, ..., M_{p_N}\}$ **do**
      Calculate the loss $loss_{p_i}$ based on the output of the perturbed model M_{p_i}. Then calculate the gradients g_{p_i} for $\mathcal{W}_\mathcal{G}$ based on $loss_{p_i}$.
    end for
    Calculate the loss $loss_0$ based on the output of the clean model M_0. Then calculate the gradients g_0 for $\mathcal{W}_\mathcal{G}$ based on $loss_0$.
    Calculate the final gradient g_{final} using (5) based on g_0 and $g_{p_1}, ..., g_{p_N}$.
    Update $\mathcal{W}_\mathcal{G}$ using g_{final}.
  end for
end for

B IMPLEMENTATION DETAILS OF NEURALFUSE GENERATOR

We consider two main goals in designing the NeuralFuse Generator: 1) efficiency (so the overall energy overhead is decreased) and 2) robustness (so that it can generate robust patterns on the input image and overcome the random bit flipping in subsequent models). Accordingly, we choose to utilize an encode-decoder architecture in implementing the generator. The design of ConvL is inspired by Nguyen & Tran (2020), in which the authors utilize a similar architecture to design an input-aware trigger generator, and have demonstrated its efficiency and effectiveness. Furthermore, we attempted to enhance it by replacing the *Upsampling* layer with a *Deconvolution* layer, leading to the creation of DeConvL. The UNetL-based NeuralFuse draws inspiration from Ronneberger et al. (2015), known for its robust performance in image segmentation, and thus, we incorporated it as one of our architectures. Lastly, ConvS, DeConvS, and UNetS are *scaled-down* versions of the model designed to reduce computational costs and total parameters. The architectures of Convolutional-based and Deconvolutional-based are shown in Table 4, and the architecture of UNet-based generators is in Table 5. For the abbreviation used in the table, ConvBlock means the Convolution block, Conv means a single Convolution layer, DeConvBlock means the Deconvolution block, DeConv means a single Deconvolution layer, and BN means a Batch Normalization layer. We use learning rate $= 0.001$, $\lambda = 5$, and Adam optimizer. For CIFAR-10, GTSRB, and CIFAR-100, we set batch size $b = 25$ for each base model. For ImageNet-10, we set $b = 64$ for ResNet18, ResNet50 and VGG11, and $b = 32$ for both VGG16 and VGG19.

Table 4: Model architecture for both Convolution-based and Deconvolution-based generators. Each ConvBlock consists of a Convolution (kernel $= 3 \times 3$, padding $= 1$, stride $= 1$), a Batch Normalization, and a ReLU layer. Each DeConvBlock consists of a Deconvolution (kernel $= 4 \times 4$, padding $= 1$, stride $= 2$), a Batch Normalization, and a ReLU layer.

ConvL		ConvS		DeConvL		DeConvS	
Layers	#CHs	Layers	#CHs	Layers	#CHs	Layers	#CHs
(ConvBlock)×2, MaxPool	32	ConvBlock, Maxpool	32	(ConvBlock)×2, MaxPool	32	ConvBlock, Maxpool	32
(ConvBlock)×2, MaxPool	64	ConvBlock, Maxpool	64	(ConvBlock)×2, MaxPool	64	ConvBlock, Maxpool	64
(ConvBlock)×2, MaxPool	128	ConvBlock, Maxpool	64	(ConvBlock)×2, MaxPool,	128	ConvBlock, Maxpool	64
ConvBlock, UpSample, ConvBlock	128	ConvBlock, UpSample	64	ConvBlock	128	DeConvBlock	64
ConvBlock, UpSample, ConvBlock	64	ConvBlock, UpSample	32	DeConvBlock, ConvBlock	64	DeConvBlock	32
ConvBlock, UpSample, ConvBlock	32	ConvBlock, UpSample	3	DeConvBlock, ConvBlock	32	DeConv, BN, Tanh	3
Conv, BN, Tanh	32	Conv, BN, Tanh	3	Conv, BN, Tanh	3		

[Note] #CHs: *number of channels*.

Table 5: Model architecture for UNet-based generators. Each ConvBlock consists of a Convolution (kernel $= 3 \times 3$, padding $= 1$, stride $= 1$), a Batch Normalization, and a ReLU layer. Other layers, such as the Deconvolutional layer (kernel $= 2 \times 2$, padding $= 1$, stride $= 2$), are used in UNet-based models. For the final Convolution layer, the kernel size is set to 1.

UNetL		UNetS	
Layers	#Channels	Layers	#Channels
L1: (ConvBlock)×2	16	L1: (ConvBlock)×2	8
L2: Maxpool, (ConvBlock)×2	32	L2: Maxpool, (ConvBlock)×2	16
L3: Maxpool, (ConvBlock)×2	64	L3: Maxpool, (ConvBlock)×2	32
L4: Maxpool, (ConvBlock)×2	128	L4: Maxpool, (ConvBlock)×2	64
L5: DeConv	64	L5: DeConv	32
L6: Concat[L3, L5]	128	L6: Concat[L3, L5]	64
L7: (ConvBlock)×2	64	L7: (ConvBlock)×2	32
L8: DeConv	32	L8: DeConv	16
L9: Concat[L2, L8]	64	L9: Concat[L2, L8]	32
L10: (ConvBlock)×2	32	L10: (ConvBlock)×2	16
L11: DeConv	16	L11: DeConv	8
L12: Concat[L1, L11]	32	L12: Concat[L1, L11]	16
L13: (ConvBlock)×2	16	L13: (ConvBlock)×2	8
L14: Conv	3	L14: Conv	3

C Implementation Details of SCALE-SIM

SCALE-SIM (Samajdar et al., 2020) is a systolic array based CNN simulator that can calculate the number of memory accesses and the total time in execution cycles by giving the specific model architecture and accelerator architectural configuration as inputs. In this paper, we use SCALE-SIM to calculate the weights memory access of 5 based models (ResNet18, ResNet50, VGG11, VGG16, VGG19), and 6 generators (ConvL, ConvS, DeConvL, DeConvS, UNetL, UNetS). While SCALE-SIM supports both Convolutional and Linear layers, it does not yet support Deconvolution layers. Instead, we try to approximate the memory costs of Deconvolution layers by Convolution layers. We change the input and output from Deconvolution into the output and input of the Convolution layers. Besides, we also change the stride into 1 when we approximate it. We also add padding for the convolution layers while generating input files for SCALE-SIM. In this paper, we only consider the energy saving on weights accesses, so we only take the value "SRAM Filter Reads" from the output of SCALE-SIM as the *total weights memory accesses* (T.W.M.A.) for further energy calculation.

D THE ENERGY-ACCURACY TRADEOFF UNDER 1% BIT ERROR RATE

In Table 6, we report the total weight memory access (T.W.M.A.) calculated by SCALE-SIM. We then showed the energy-accuracy tradeoff between all of the combinations of NeuralFuse and base models under a 1% of bit error rate in Figure 5.

Table 6: The total weights memory access calculated by SCALE-SIM.

Base Model	ResNet18	ResNet50	VGG11	VGG16	VGG19	-
T.W.M.A.	2,755,968	6,182,144	1,334,656	2,366,848	3,104,128	-
NeuralFuse	ConvL	ConvS	DeConvL	DeConvS	UNetL	UNetS
T.W.M.A.	320,256	41,508	259,264	86,208	180,894	45,711

[Note] T.W.M.A.: *total weight memory access*.

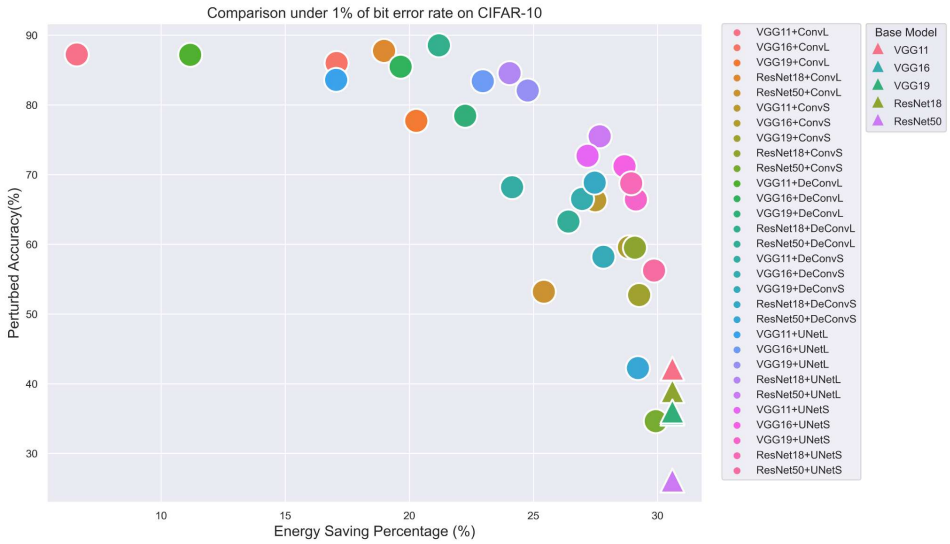


Figure 5: The energy-accuracy tradeoff of different NeuralFuse implementations with all CIFAR-10 pre-trained based models. X-axis represents the percentage reduction in dynamic memory access energy at low-voltage settings (base model protected by NeuralFuse), compared to the bit-error-free (nominal) voltage. Y-axis represents the perturbed accuracy (evaluated at low voltage) with a 1% bit error rate.

E MODEL PARAMETERS AND MAC VALUES

In addition to T.W.M.A., the model's parameters and MACs (multiply–accumulate operations) are common metrics in measuring the energy consumption of machine learning models. Yang et al. (2017) have also shown that the *energy consumption of computation* and *memory accesses* are both proportional to MACs, allowing us to estimate the overall (or end-to-end) energy consumption.

Here, we use the open-source package `ptflops` (Sovrasov, 2018-2023) to calculate the parameters and MAC values of all the base models and the NeuralFuse generators, in the same units as Bejnordi et al. (2020) used. The results are shown in Table 7. Note that we modified the base model architectures for ImageNet-10, as it has larger input sizes. For example, we use a larger kernel size $= 7$ instead of 3 in the first Convolution layer in ResNet-based models to enhance the learning abilities. Therefore, the parameters of base models are different between different datasets. For NeuralFuse generators, we utilize the same architectures for implementation (including ImageNet-10). As a result, our proposed NeuralFuse generators are generally smaller than base models, either on total model parameters or MAC values.

Table 7: Parameter counts and MACs for all base models and generators in this paper.

		Base Model					
		ResNet18	ResNet50	VGG11	VGG16	VGG19	-
Parameter	CIFAR-10	11,173,962	23,520,842	9,231,114	14,728,266	20,040,522	-
	ImageNet-10	11,181,642	23,528,522	128,812,810	134,309,962	139,622,218	
MACs	CIFAR-10	557.14M	1.31G	153.5M	314.43M	399.47M	-
	ImageNet-10	1.82G	4.12G	7.64G	15.53G	19.69G	
		NeuralFuse					
		ConvL	ConvS	DeConvL	DeConvS	UNetL	UNetS
Parameter	CIFAR-10	723,273	113,187	647,785	156,777	482,771	121,195
	ImageNet-10						
MACs	CIFAR-10	80.5M	10.34M	64.69M	22.44M	41.41M	10.58M
	ImageNet-10	3.94G	506.78M	3.17G	1.1G	2.03G	518.47M

MACs-Based Energy Saving Calculation. We can then use the MAC values to further approximate the end-to-end energy consumption of the whole model. Assume that all values are stored on SRAM and that a MAC represents single memory access. The corresponding MACs-based energy saving percentage (MAC-ES, %) can be derived from Eq. 6 (c.f. Sec. 4.4), and results can be found in Table 8. We can observe that most combinations can save a large amount of energy, except that VGG11 with two larger NeuralFuse (ConvL and DeConvL) may increase the total energy. These results are consistent with the results reported in Table 2. In addition, we also showed the MACs-based energy-accuracy tradeoff between all of the combinations of NeuralFuse and base models under a 1% of bit error rate in Figure 6.

$$\text{MAC-ES} = \frac{\text{MACs}_{\text{base model}} \cdot \text{Energy}_{\text{nominal voltage}} - \left(\text{MACs}_{\text{base model}} \cdot \text{Energy}_{\text{low-voltage-regime}} + \text{MACs}_{\text{NeuralFuse}} \cdot \text{Energy}_{\text{NeuralFuse at nominal voltage}}\right)}{\text{MACs}_{\text{base model}} \cdot \text{Energy}_{\text{nominal voltage}}} \times 100\% \quad (6)$$

Table 8: The MACs-Based energy saving percentage (%) for different combinations of base models and NeuralFuse.

Base Model	ConvL	ConvS	DeConvL	DeConvS	UNetL	UNetS
ResNet18	16.2	28.7	19.0	26.6	23.2	28.7
ResNet50	24.5	29.8	25.7	28.9	27.4	29.8
VGG11	-21.8	23.9	-11.5	16.0	3.6	23.7
VGG16	5.0	27.3	10.0	23.5	17.4	27.2
VGG19	10.4	28.0	14.4	25.0	20.2	28.0

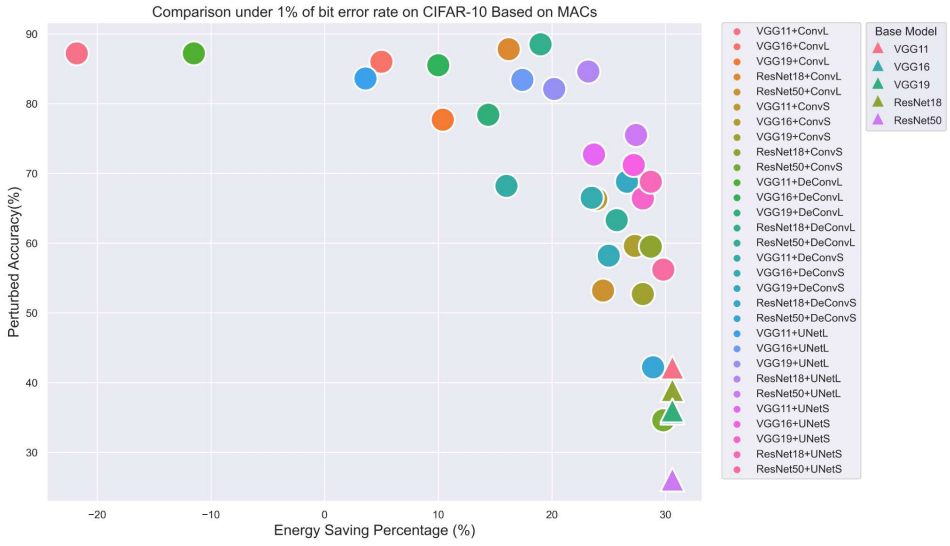
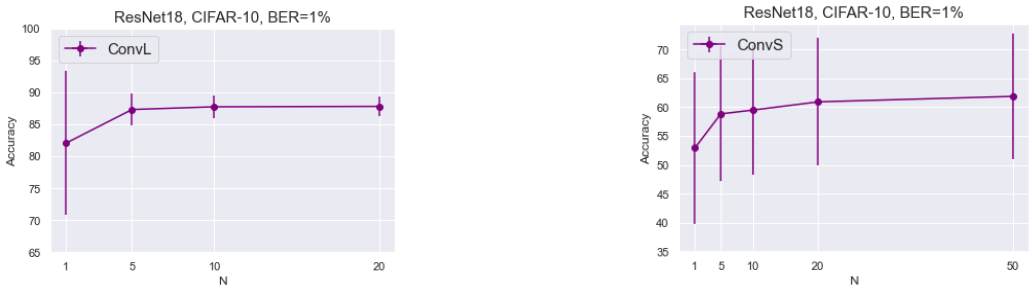


Figure 6: The MAC-Based energy-accuracy tradeoff of different NeuralFuse implementations with all CIFAR-10 pre-trained based models. X-axis represents the percentage reduction in dynamic memory access energy at low-voltage settings (base model protected by NeuralFuse), compared to the bit-error-free (nominal) voltage. Y-axis represents the perturbed accuracy (evaluated at low voltage) with a 1% bit error rate.

F Ablation Studies

Study for N in EOPM. Here, we study the effect of N used in EOPM (Eq. 5). In Figure 7, we report the results for ConvL and ConvS on CIFAR-10 pre-trained ResNet18, under a 1% bit error rate (B.E.R.). The results demonstrate that if we apply larger N, the performance increases until convergence. Specifically, for ConvL (Figure 7a), larger N empirically has a smaller standard deviation; this means larger N gives better stability but at the cost of time-consuming training. In contrast, for the small generator ConvS (Figure 7b), we can find that the standard deviation is still large even trained by larger N; the reason might be that small generators are not as capable of learning representations as larger ones. Therefore, there exists a trade-off between the *stability* of the generator performance and the total training time. In our implementation, choosing $N = 5$ or 10 is a good balance.



(a) Using ConvL (b) Using ConvS

Figure 7: The experimental results on using different sizes of N for EOPM.

Tradeoff Between Clean Accuracy (C.A.) and Perturbed Accuracy (P.A.). We conducted an experiment to study the effect of different λ values, which balance the ratio of clean accuracy and perturbed accuracy. In Table 9, the experimental results showed that a smaller λ can preserve clean accuracy, but result in poor perturbed accuracy. On the contrary, larger λ can deliver higher perturbed accuracy, but with more clean accuracy drop. This phenomenon has also been observed in adversarial training (Zhang et al., 2019).

Table 9: Experimental results based on λ value choosing. The results show that $\lambda = 5$ can balance the tradeoff between clean accuracy and perturbed accuracy.

Base Model	λ	C.A.	P.A.	ConvL C.A. (NF)	ConvL P.A. (NF)	ConvL R.P.
ResNet18	10	92.6	38.9 ± 12.4	90.1	88.0 ± 1.7	49.1
	5			89.8	87.8 ± 1.7	48.8
	1			90.0	84.2 ± 3.8	45.3
	0.1			91.6	65.7 ± 9.3	26.8
	0.01			92.2	43.6 ± 13	4.7
VGG19	10	90.5	36.0 ± 12.0	89.6	77.9 ± 19	41.9
	5			89.8	77.7 ± 19	41.7
	1			89.9	73.1 ± 19	37.1
	0.1			89.1	51.2 ± 16	15.2
	0.01			90.2	36.8 ± 12	0.8

[Note] C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*

Comparison to Universal Input Perturbation (UIP). Moosavi-Dezfooli et al. (2017) has shown that there exists a universal adversarial perturbation to the input data such that the model will make

wrong predictions on a majority of the perturbed images. In our NeuralFuse framework, the universal perturbation is a special case when we set $\mathcal{G}(\mathbf{x}) = \tanh(\textbf{UIP})$ for any data sample $\mathbf{x}$. The transformed data sample then becomes $\mathbf{x}_t = \text{clip}_{[-1,1]}\big(\mathbf{x} + \tanh(\textbf{UIP})\big)$, where $\mathbf{x}_t \in [-1,1]^d$ and **UIP** is a trainable universal input perturbation that has the same size as the input data. The experimental results with the universal input perturbation are shown in Table 10. We observe that its performance is much worse than our proposed NeuralFuse. This result validates the necessity of adopting input-aware transformation for learning error-resistant data representations in low-voltage scenarios.

Table 10: Performance of the universal input perturbation (UIP) trained by EOPM on CIFAR-10 pre-trained ResNet18.

Base Model	B.E.R.	C.A.	P.A.	C.A. (**UIP**)	P.A. (**UIP**)	R.P.
ResNet18	1%	92.6	38.9 ± 12.4	91.8	37.9 ± 11	-1.0
	0.5%		70.1 ± 11.6	92.5	70.6 ± 11	0.5
ResNet50	1%	92.6	26.1 ± 9.4	80.7	21.0 ± 5.9	-5.1
	0.5%		61.0 ± 10.3	91.9	62.4 ± 12	1.4
VGG11	1%	88.4	42.2 ± 11.6	86.9	43.0 ± 11	0.8
	0.5%		63.6 ± 9.3	88.2	64.2 ± 8.8	0.6
VGG16	1%	90.3	35.7 ± 7.9	90.1	37.1 ± 8.5	1.4
	0.5%		66.6 ± 8.1	90.4	67.3 ± 8.1	0.7
VGG19	1%	90.5	36.0 ± 12.0	89.9	35.3 ± 12	-0.7
	0.5%		64.2 ± 12.4	90.1	64.4 ± 12	0.2

[Note] B.E.R.: *the bit error rate of the base model*, C.A. (%): *clean accuracy*, **UIP**: *universal input transformation parameter*, P.A.(%): *perturbed accuracy*, and R.P.: *total recover percentage of P.A. (UIP) v.s. P.A.*

G Additional Experimental Results on Relaxed Access Settings

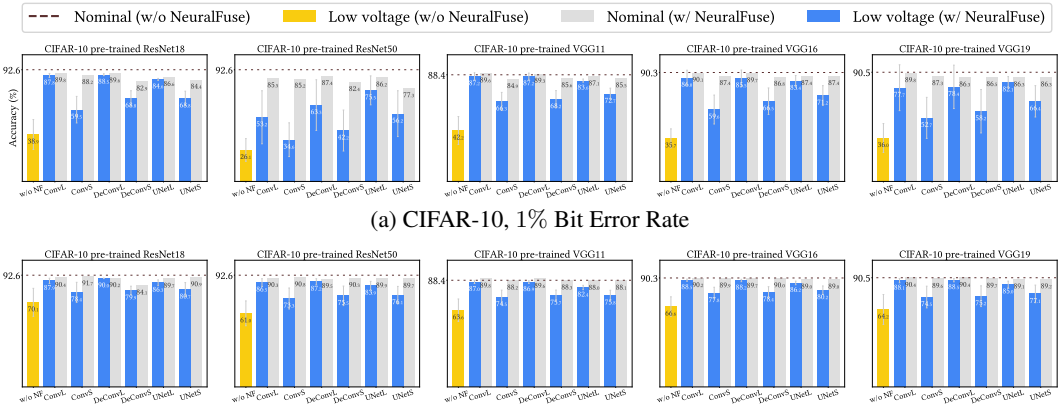
We conducted more experiments on *Relaxed Access* settings to show that our NeuralFuse can protect the models under different B.E.R. The results can be found in Sec. G.1 (CIFAR-10), Sec. G.2 (GTSRB), Sec. G.3 (ImageNet-10), and Sec. G.4 (CIFAR-100). For comparison, we also visualize the experimental results in the figures below each table.

G.1 CIFAR-10

Table 11: Testing accuracy (%) under 1% and 0.5% of random bit error rate on CIFAR-10.

Base Model	NF	C.A.	1% B.E.R. P.A.	1% B.E.R. C.A. (NF)	1% B.E.R. P.A. (NF)	1% B.E.R. R.P.	0.5% B.E.R. P.A.	0.5% B.E.R. C.A. (NF)	0.5% B.E.R. P.A. (NF)	0.5% B.E.R. R.P.
ResNet18	ConvL	92.6	38.9 ± 12.4	89.8	87.8 ± 1.7	48.8	70.1 ± 11.6	90.4	87.9 ± 2.2	17.8
	ConvS			88.2	59.5 ± 11	20.6		91.7	78.4 ± 8.3	8.3
	DeConvL			89.6	88.5 ± 0.8	49.6		90.2	90.0 ± 0.2	19.9
	DeConvS			82.9	68.8 ± 6.4	29.9		84.1	79.9 ± 3.6	9.8
	UNetL			86.6	84.6 ± 0.8	45.6		89.7	86.3 ± 2.4	16.2
	UNetS			84.4	68.8 ± 6.0	29.8		90.9	80.7 ± 5.8	10.7
ResNet50	ConvL	92.6	26.1 ± 9.4	85.5	53.2 ± 22	27.1	61.0 ± 10.3	90.3	86.5 ± 3.2	25.5
	ConvS			85.2	34.6 ± 14	8.5		90.8	73.3 ± 8.7	12.3
	DeConvL			87.4	63.3 ± 21	37.2		89.5	87.2 ± 2.5	26.2
	DeConvS			82.4	42.2 ± 17	16.1		90.3	75.5 ± 8.1	14.5
	UNetL			86.2	75.5 ± 12	49.4		89.9	83.9 ± 3.6	22.9
	UNetS			77.3	56.2 ± 19	30.1		89.7	76.1 ± 7.2	15.1
VGG11	ConvL	88.4	42.2 ± 11.6	89.6	87.2 ± 2.9	45.1	63.6 ± 9.3	89.8	87.0 ± 1.3	23.3
	ConvS			84.9	66.3 ± 7.5	24.1		88.2	74.5 ± 5.7	10.9
	DeConvL			89.3	87.2 ± 2.6	45.0		89.6	86.9 ± 1.1	23.2
	DeConvS			85.6	68.2 ± 7.1	26.0		88.3	75.7 ± 4.6	12.1
	UNetL			87.1	83.6 ± 1.3	41.4		88.0	82.4 ± 1.8	18.8
	UNetS			85.5	72.7 ± 4.6	30.5		88.1	75.8 ± 4.3	12.2
VGG16	ConvL	90.3	35.7 ± 7.9	90.1	86.0 ± 6.2	50.3	66.6 ± 8.1	90.2	88.5 ± 0.9	21.9
	ConvS			87.4	59.6 ± 12	23.9		89.9	77.8 ± 4.8	11.1
	DeConvL			89.7	85.5 ± 6.8	49.8		89.7	88.2 ± 1.0	21.4
	DeConvS			86.8	66.5 ± 11	30.8		90.0	78.4 ± 4.7	11.8
	UNetL			87.4	83.4 ± 4.4	47.7		89.0	86.2 ± 1.5	19.6
	UNetS			87.4	71.2 ± 8.2	35.5		89.0	80.2 ± 3.5	13.7
VGG19	ConvL	90.5	36.0 ± 12.0	89.8	77.7 ± 19	41.7	64.2 ± 12.4	90.4	88.1 ± 1.8	23.9
	ConvS			87.3	52.7 ± 17	16.7		89.6	74.5 ± 9.0	10.3
	DeConvL			86.3	78.4 ± 18	42.4		90.4	88.5 ± 1.4	24.3
	DeConvS			86.5	58.2 ± 18	22.2		89.7	75.2 ± 8.6	11.0
	UNetL			86.3	82.1 ± 4.8	46.0		89.1	85.0 ± 2.7	20.8
	UNetS			86.3	66.4 ± 13	30.4		89.2	77.1 ± 7.3	12.9

[Note] C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*



(a) CIFAR-10, 1% Bit Error Rate

(b) CIFAR-10, 0.5% Bit Error Rate

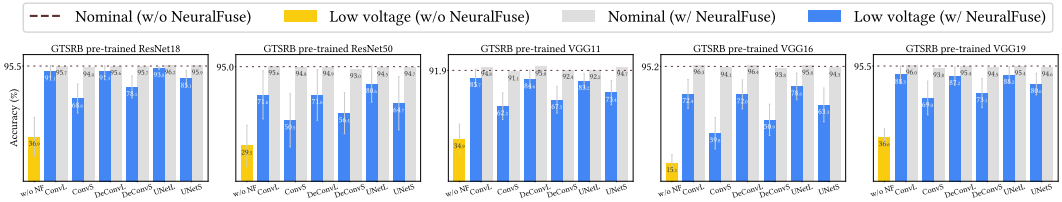
Figure 8: Experimental results on CIFAR-10

G.2 GTSRB

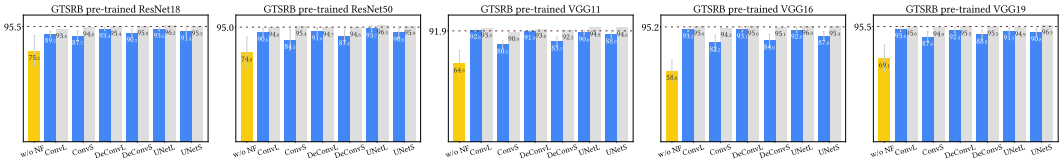
Table 12: Testing accuracy (%) under 1% and 0.5% of random bit error rate on GTSRB.

Base Model	NF	C.A.	1% B.E.R. P.A.	1% B.E.R. C.A. (NF)	1% B.E.R. P.A. (NF)	1% B.E.R. R.P.	0.5% B.E.R. P.A.	0.5% B.E.R. C.A. (NF)	0.5% B.E.R. P.A. (NF)	0.5% B.E.R. R.P.
ResNet18	ConvL	95.5	36.9 ± 16.0	95.7	91.1 ± 4.7	54.2	75.2 ± 12.7	93.4	89.5 ± 1.9	14.3
	ConvS			94.4	68.6 ± 12	31.7		94.8	87.7 ± 4.2	12.4
	DeConvL			95.6	91.3 ± 4.3	54.4		95.4	93.4 ± 1.1	18.1
	DeConvS			95.7	78.1 ± 9.1	41.2		95.8	90.1 ± 3.3	14.9
	UNetL			96.2	93.8 ± 1.0	56.9		96.2	93.5 ± 1.6	18.3
	UNetS			95.9	85.1 ± 6.9	48.2		95.5	91.4 ± 2.8	16.2
ResNet50	ConvL	95.0	29.5 ± 16.9	95.6	71.6 ± 20	42.1	74.0 ± 13.0	94.6	90.6 ± 3.7	16.6
	ConvS			94.8	50.5 ± 22	21.0		95.4	84.5 ± 8.5	10.5
	DeConvL			94.9	71.6 ± 21	42.0		94.7	91.6 ± 2.9	17.6
	DeConvS			93.0	56.4 ± 17	26.9		94.6	87.4 ± 5.9	13.5
	UNetL			94.5	80.6 ± 15	51.1		96.5	93.7 ± 2.3	19.7
	UNetS			94.7	64.7 ± 22	35.2		95.9	90.6 ± 4.8	16.7
VGG11	ConvL	91.9	34.9 ± 12.4	94.8	85.7 ± 7.2	50.9	64.9 ± 10.8	93.9	92.6 ± 0.7	27.7
	ConvS			91.1	62.2 ± 11	27.3		90.9	80.5 ± 3.5	15.7
	DeConvL			95.0	84.6 ± 7.6	49.7		93.6	91.9 ± 0.6	27.1
	DeConvS			92.4	67.5 ± 11	32.6		92.3	83.1 ± 3.7	18.2
	UNetL			92.2	83.2 ± 6.0	48.3		94.8	90.6 ± 1.7	25.7
	UNetS			94.7	73.4 ± 10	38.5		94.6	88.9 ± 2.2	24.1
VGG16	ConvL	95.2	15.1 ± 6.8	96.3	72.4 ± 12	57.3	58.8 ± 8.9	95.6	93.2 ± 1.8	34.4
	ConvS			94.1	39.8 ± 13	24.6		94.3	82.2 ± 6.2	23.4
	DeConvL			96.4	72.0 ± 12	56.9		95.6	93.1 ± 2.0	34.3
	DeConvS			93.8	50.9 ± 13	35.8		95.1	84.0 ± 5.3	25.2
	UNetL			95.8	78.6 ± 11	63.5		96.0	92.8 ± 2.0	34.0
	UNetS			94.3	63.3 ± 14	48.1		95.4	87.8 ± 3.6	29.0
VGG19	ConvL	95.5	36.6 ± 6.8	96.0	88.3 ± 7.2	51.7	69.1 ± 11.1	95.6	93.4 ± 2.1	24.2
	ConvS			93.8	69.0 ± 14	32.4		94.9	87.0 ± 4.4	17.8
	DeConvL			95.4	87.2 ± 7.5	50.6		95.5	92.4 ± 2.2	23.3
	DeConvS			94.5	73.1 ± 12	36.5		95.5	88.8 ± 3.7	19.7
	UNetL			95.4	88.2 ± 6.7	51.7		94.9	91.7 ± 2.5	22.6
	UNetS			94.6	80.6 ± 9.0	44.1		96.5	90.8 ± 3.4	21.6

[Note] C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*



(a) GTSRB, 1% Bit Error Rate



(b) GTSRB, 0.5% Bit Error Rate

Figure 9: Experimental results on GTSRB.

G.3 IMAGENET-10

Table 13: Testing accuracy under 0.5% of random bit error rate on ImageNet-10.

Base Model	NF	C.A.	0.5% B.E.R. P.A.	C.A. (NF)	P.A. (NF)	R.P.
ResNet18	ConvL	92.2	72.3 ± 7.0	94.0	88.0 ± 2.0	15.7
	ConvS			91.8	83.6 ± 4.1	11.3
	DeConvL			94.0	89.2 ± 1.3	16.9
	DeConvS			92.8	87.5 ± 2.3	15.2
	UNetL			94.0	88.1 ± 1.4	15.8
	UNetS			93.2	86.4 ± 2.2	14.1
ResNet50	ConvL	89.8	39.4 ± 11	92.2	80.0 ± 5.8	40.6
	ConvS			91.8	65.0 ± 11	25.6
	DeConvL			93.0	79.4 ± 5.9	40.0
	DeConvS			93.2	70.9 ± 9.1	31.5
	UNetL			92.2	80.5 ± 5.8	41.1
	UNetS			92.4	73.6 ± 8.9	34.2
VGG11	ConvL	91.6	47.8 ± 13	92.0	86.1 ± 3.7	38.3
	ConvS			89.4	66.4 ± 7.1	18.6
	DeConvL			91.0	86.0 ± 3.0	38.2
	DeConvS			89.0	72.5 ± 7.8	24.7
	UNetL			92.4	83.0 ± 3.5	35.2
	UNetS			86.2	73.5 ± 6.0	25.7
VGG16	ConvL	94.6	38.4 ± 17	90.8	77.1 ± 11	38.7
	ConvS			90.2	60.2 ± 14	21.8
	DeConvL			91.2	77.2 ± 11	38.8
	DeConvS			90.0	62.3 ± 14	23.9
	UNetL			90.6	81.1 ± 5.9	42.7
	UNetS			86.4	72.3 ± 8.8	33.9
VGG19	ConvL	92.4	37.2 ± 11	91.4	75.5 ± 8.8	38.3
	ConvS			88.8	56.5 ± 13	19.3
	DeConvL			91.0	75.9 ± 8.9	38.7
	DeConvS			88.8	64.0 ± 11	26.8
	UNetL			89.4	77.9 ± 6.1	40.7
	UNetS			87.6	65.9 ± 10	28.7

[Note] C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*

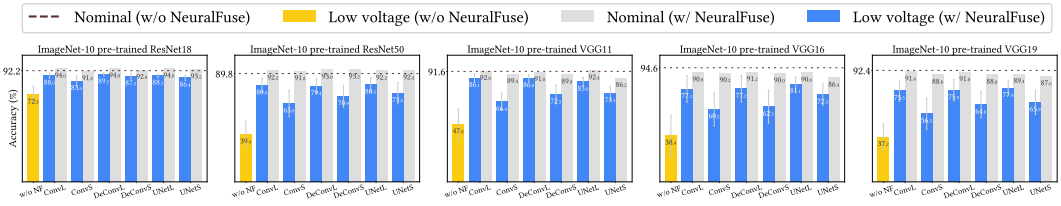


Figure 10: Experimental results on ImageNet-10, 0.5% Bit Error Rate.

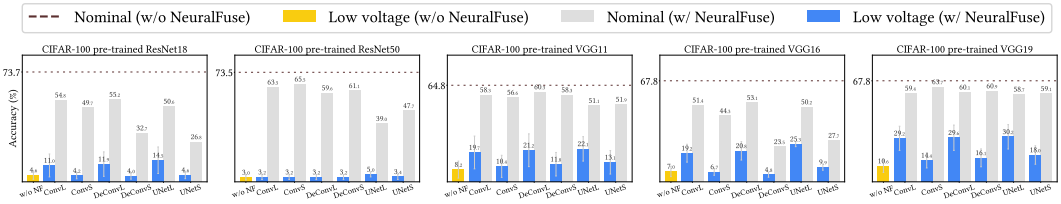
G.4 CIFAR-100

As mentioned in the previous section, larger generators like ConvL, DeConvL, and UNetL have better performance than small generators. For CIFAR-100, we find that the gains of utilizing NeuralFuse are less compared to the other datasets. We believe this is because CIFAR-100 is a more challenging dataset (more classes) for the generators to learn to protect the base models. Nevertheless, NeuralFuse can still function to restore some degraded accuracy; these results also demonstrate that our NeuralFuse is applicable to different datasets. In addition, although the recover percentage is less obvious on CIFAR-100 (the more difficult dataset), we can still conclude that our NeuralFuse is applicable to different datasets.

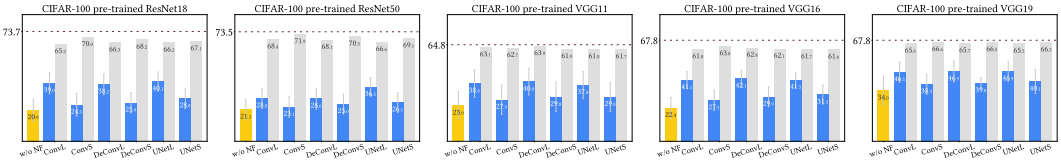
Table 14: Testing accuracy (%) under 1%, 0.5% and 0.35% of random bit error rate on CIFAR-100.

Base Model	NF	C.A.	1% B.E.R.				0.5% B.E.R.				0.35% B.E.R.			
			P.A.	C.A. (NF)	P.A. (NF)	R.P.	P.A.	C.A. (NF)	P.A. (NF)	R.P.	P.A.	C.A. (NF)	P.A. (NF)	R.P.
ResNet18	ConvL	73.7	4.6 ± 2.9	54.8	11.0 ± 7.7	6.4	20.9 ± 7.4	65.2	39.0 ± 7.1	18.1	31.4 ± 7.6	69.4	42.9 ± 6.2	11.4
	ConvS			49.7	4.2 ± 2.2	-0.4		70.0	24.5 ± 7.6	3.6		72.1	35.1 ± 7.3	3.7
	DeConvL			55.2	11.9 ± 8.2	7.3		66.3	38.2 ± 6.9	17.3		69.2	42.9 ± 5.5	11.4
	DeConvS			32.7	4.0 ± 2.2	-0.6		68.2	25.9 ± 6.8	5		71.6	35.8 ± 5.5	4.4
	UNetL			50.6	14.5 ± 8.9	10.0		66.2	40.1 ± 6.4	19.2		70.3	46.3 ± 5.5	14.9
	UNetS			26.8	4.6 ± 2.5	-0.0		67.1	28.8 ± 6.8	7.9		70.9	38.3 ± 6.4	6.9
ResNet50	ConvL	73.5	3.0 ± 1.8	63.5	3.2 ± 1.7	0.1	21.3 ± 7.0	68.4	28.8 ± 6.7	7.6	35.7 ± 8.6	72.0	40.8 ± 7.5	5.1
	ConvS			65.5	3.2 ± 1.6	0.1		71.9	23.1 ± 6.9	1.9		73.0	37.4 ± 8.0	1.7
	DeConvL			59.6	3.2 ± 1.7	0.2		68.1	28.6 ± 7.0	7.4		71.7	41.7 ± 7.7	6.1
	DeConvS			61.1	3.2 ± 1.7	0.1		70.3	25.0 ± 6.7	3.7		72.8	38.9 ± 7.9	3.3
	UNetL			39.0	5.0 ± 1.7	1.9		66.6	36.5 ± 6.2	15.3		70.8	45.3 ± 6.7	9.6
	UNetS			47.7	3.4 ± 1.8	0.3		69.1	26.1 ± 6.6	4.8		72.6	39.6 ± 7.8	3.9
VGG11	ConvL	64.8	8.2 ± 5.7	58.3	19.7 ± 11	11.5	23.9 ± 9.4	63.1	38.8 ± 9.3	15.0	31.3 ± 10	63.9	42.4 ± 9.0	11.1
	ConvS			56.6	10.4 ± 7.4	2.2		62.7	27.9 ± 10	4.0		63.9	41.8 ± 8.3	10.5
	DeConvL			60.3	21.2 ± 11	13.0		63.9	40.0 ± 9.0	16.2		64.0	42.8 ± 9.1	11.5
	DeConvS			58.3	11.8 ± 7.9	3.5		61.9	29.8 ± 9.9	5.9		63.5	36.1 ± 10	4.8
	UNetL			51.1	22.1 ± 8.2	13.9		61.8	37.8 ± 9.0	13.9		63.5	40.9 ± 9.3	9.6
	UNetS			51.9	13.1 ± 7.9	4.9		61.7	29.8 ± 9.7	6.0		63.8	35.7 ± 9.9	4.5
VGG16	ConvL	67.8	7.0 ± 3.5	51.4	19.2 ± 6.0	12.6	22.4 ± 7.0	61.8	41.1 ± 5.6	18.7	31.1 ± 7.2	64.9	44.9 ± 5.3	13.8
	ConvS			44.3	6.7 ± 2.3	0.1		63.8	27.5 ± 6.8	5.1		66.0	36.3 ± 6.1	5.1
	DeConvL			53.1	20.8 ± 6.2	14.2		62.8	42.1 ± 5.5	19.8		65.0	46.6 ± 5.2	15.5
	DeConvS			23.5	4.8 ± 1.7	-1.8		62.1	29.9 ± 6.7	7.5		64.9	38.1 ± 6.3	7.0
	UNetL			50.2	25.3 ± 1.7	18.7		61.7	41.3 ± 5.0	18.9		64.8	46.8 ± 4.6	15.7
	UNetS			27.7	9.9 ± 2.1	3.3		61.6	31.3 ± 6.3	8.9		65.0	39.8 ± 5.9	8.7
VGG19	ConvL	67.8	10.6 ± 4.3	59.4	29.2 ± 8.1	18.6	34.0 ± 9.6	65.6	46.5 ± 6.8	12.5	42.1 ± 9.4	66.9	49.2 ± 7.4	7.0
	ConvS			63.7	14.4 ± 5.1	3.8		66.6	38.3 ± 6.8	4.2		67.7	45.3 ± 8.5	3.2
	DeConvL			60.1	29.6 ± 8.5	19.0		65.7	46.9 ± 7.1	12.9		67.3	49.8 ± 7.6	7.6
	DeConvS			60.9	16.1 ± 6.0	5.6		66.5	39.0 ± 3.7	5.0		67.7	45.7 ± 8.4	3.6
	UNetL			58.7	30.2 ± 8.2	19.6		65.5	46.9 ± 6.5	12.9		67.4	50.0 ± 7.5	7.9
	UNetS			59.1	18.0 ± 6.2	7.4		66.3	40.1 ± 8.0	6.1		67.5	46.6 ± 8.4	4.5

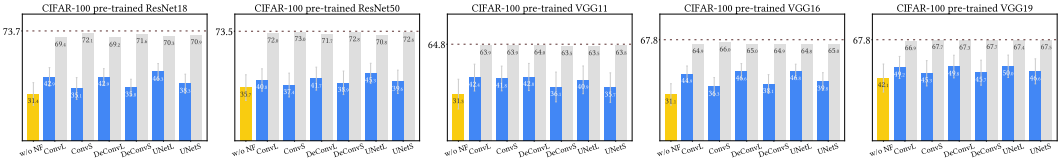
[Note] C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*



(a) CIFAR-100, 1% Bit Error Rate



(b) CIFAR-100, 0.5% Bit Error Rate



(c) CIFAR-100, 0.35% Bit Error Rate

Figure 11: Experimental results on CIFAR-100.

H Additional Experimental Results on Restricted Access Settings (Transferability)

We conduct more experiments with *Restricted Access* settings to show that our NeuralFuse can be transferred to protect various black-box models. The experimental results are shown in Sec. H.1 (CIFAR-10), Sec. H.2 (GTSRB), and Sec. H.3 (CIFAR-100).

We find that using VGG19 as a white-box surrogate model has better *transferability* than ResNet18 for all datasets. In addition, we can observe that some NeuralFuse generators have *downward applicability* if base models have a similar architecture. In other words, if we try to transfer a generator trained on a large B.E.R. (e.g., 1%) to a model with a small B.E.R. (e.g., 0.5%), the performance will be better than that of a generator trained with the original B.E.R. (e.g., 0.5%). For example, in Table 15, we could find that if we use VGG19 as a source model to train the generator ConvL (1%), the generator could deliver better performance (in terms of P.A. (NF)) when applied to similar base models (e.g., VGG11, VGG16, or VGG19) under a 0.5% B.E.R., compared to using itself as a source model (shown in Table 11). We conjecture that this is because the generators trained on a larger B.E.R. can also cover the error patterns of a smaller B.E.R., and thus they have better generalizability across smaller B.E.Rs.

To further improve the transferability to cross-architecture target models, we also conduct an experiment in Sec. H.4 to show that using ensemble-based training can help the generator to achieve this feature.

H.1 CIFAR-10

The results of CIFAR-10 in which NeuralFuse is trained at 1% B.E.R. are shown in Table 15.

Table 15: Transfer results on CIFAR-10: NeuralFuse trained on S.M. with 1% B.E.R.

S.M.	T.M.	B.E.R.	C.A.	P.A.	ConvL (1%)			UNetL (1%)		
					C.A. (NF)	P.A. (NF)	R.P.	C.A. (NF)	P.A. (NF)	R.P.
ResNet18	ResNet18	0.5%	92.6	70.1 ± 11.6	89.8	89.5 ± 0.2	19.4	86.6	86.2 ± 0.3	16.1
	ResNet50	1%	92.6	26.1 ± 9.4	89.5	36.0 ± 19	9.9	85.2	38.8 ± 19	12.7
		0.5%		61.0 ± 10.3		75.1 ± 10	14.1		77.1 ± 5.0	16.1
	VGG11	1%	88.4	42.2 ± 11.6	88.4	62.5 ± 8.4	20.3	76.8	61.1 ± 8.5	18.9
		0.5%		63.6 ± 9.3		81.0 ± 4.6	17.4		73.7 ± 3.0	10.1
	VGG16	1%	90.3	35.7 ± 7.9	89.6	63.3 ± 18	27.6	85.2	59.9 ± 16	24.2
		0.5%		66.6 ± 8.1		85.0 ± 3.4	18.4		80.2 ± 4.5	13.6
	VGG19	1%	90.5	36.0 ± 12.0	89.6	50.7 ± 22	14.7	85.3	51.1 ± 16	15.1
		0.5%		64.2 ± 12.4		80.2 ± 8.7	16.0		76.5 ± 7.8	12.3
VGG19	ResNet18	1%	92.6	38.9 ± 12.4	89.8	61.0 ± 17	22.1	87.0	69.7 ± 11	30.8
		0.5%		70.1 ± 11.6		86.1 ± 6.9	16.0		84.2 ± 3.0	14.1
	ResNet50	1%	92.6	26.1 ± 9.4	89.9	34.0 ± 19	7.9	87.0	44.2 ± 17	18.1
		0.5%		61.0 ± 10.3		76.5 ± 10	15.5		80.7 ± 4.2	19.7
	VGG11	1%	88.4	42.2 ± 11.6	89.7	76.5 ± 7.0	34.3	87.1	79.9 ± 5.6	37.7
		0.5%		63.6 ± 9.3		88.0 ± 2.1	24.4		85.4 ± 0.8	21.8
	VGG16	1%	90.3	35.7 ± 7.9	89.6	75.5 ± 12	39.8	87.2	78.9 ± 7.8	43.2
		0.5%		66.6 ± 8.1		88.9 ± 0.6	22.3		86.2 ± 0.3	19.6
	VGG19	0.5%	90.5	64.2 ± 12.4	89.8	89.6 ± 8.7	25.4	87.4	86.8 ± 0.4	22.6

[Note] S.M.: *source model, used for training generators*, T.M.: *target model, used for testing generators*, B.E.R.: *the bit error rate of the target model*, C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*

H.2 GTSRB

In Tables 16 and 17, we show the results on GTSRB in which NeuralFuse is trained at 1.5% and 1% B.E.R., respectively.

Table 16: Transfer results on GTSRB: NeuralFuse trained on S.M. with 1.5% B.E.R.

S.M.	T.M.	B.E.R.	C.A.	P.A.	ConvL (1.5%)			UNetL (1.5%)		
					C.A. (NF)	P.A. (NF)	R.P.	C.A. (NF)	P.A. (NF)	R.P.
ResNet18	ResNet18	1%	95.5	36.9 ± 16.0	95.7	93.9 ± 1.9	57.0	94.9	94.4 ± 0.4	57.5
		0.5%		75.2 ± 12.7		95.7 ± 0.2	20.5		94.8 ± 0.2	19.6
	ResNet50	1%	95.0	29.5 ± 16.9	94.4	37.0 ± 22	7.5	94.4	47.1 ± 23	17.6
		0.5%		74.0 ± 13.0		77.5 ± 13	3.5		84.8 ± 9.5	10.8
	VGG11	1%	91.9	34.9 ± 12.4	92.8	45.2 ± 10	10.3	91.4	50.5 ± 13	15.6
		0.5%		64.9 ± 10.8		79.4 ± 5.8	14.5		83.9 ± 4.2	19.0
	VGG16	1%	95.2	15.1 ± 6.8	95.4	31.1 ± 13	15.9	94.6	36.8 ± 12	21.7
		0.5%		58.8 ± 8.9		84.5 ± 8.3	25.8		86.0 ± 8.6	27.2
	VGG19	1%	95.5	36.6 ± 6.8	95.0	56.4 ± 15	19.8	94.3	60.8 ± 15	24.2
		0.5%		69.1 ± 11.1		86.9 ± 3.4	17.8		87.7 ± 3.8	18.6
VGG19	ResNet18	1%	95.5	36.9 ± 16.0	88.4	50.3 ± 12	13.4	92.8	63.7 ± 16	26.8
		0.5%		75.2 ± 12.7		77.9 ± 7.4	2.7		87.5 ± 3.9	12.3
	ResNet50	1%	95.0	29.5 ± 16.9	87.5	29.7 ± 17	0.2	92.5	40.4 ± 21	10.9
		0.5%		74.0 ± 13.0		67.9 ± 17	-6.1		77.5 ± 15	3.5
	VGG11	1%	91.9	34.9 ± 12.4	89.7	47.1 ± 11	12.2	93.5	60.0 ± 12	25.1
		0.5%		64.9 ± 10.8		76.3 ± 5.1	11.4		86.0 ± 3.8	21.1
	VGG16	1%	95.2	15.1 ± 6.8	93.0	29.2 ± 15	14.1	93.0	38.5 ± 16	23.4
		0.5%		58.8 ± 8.9		75.7 ± 12	16.9		79.9 ± 8.3	21.1
	VGG19	1%	95.5	36.6 ± 6.8	95.1	87.4 ± 6.0	50.8	94.6	88.7 ± 5.0	52.1
		0.5%		69.1 ± 11.1		92.4 ± 2.4	23.3		92.4 ± 2.2	23.3

[Note] S.M.: *source model, used for training generators*, T.M.: *target model, used for testing generators*, B.E.R.: *the bit error rate of the target model*, C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*

Table 17: Transfer results on GTSRB: NeuralFuse trained on S.M. with 1% B.E.R.

S.M.	T.M.	B.E.R.	C.A.	P.A.	ConvL (1%)			UNetL (1%)		
					C.A. (NF)	P.A. (NF)	R.P.	C.A. (NF)	P.A. (NF)	R.P.
ResNet18	ResNet18	0.5%	95.5	75.2 ± 12.7	95.7	95.3 ± 0.5	20.1	96.2	95.7 ± 0.3	20.5
	ResNet50	1%	95.0	29.5 ± 16.9	94.5	35.6 ± 21	6.1	95.6	42.6 ± 23	13.1
		0.5%		74.0 ± 13.0		78.8 ± 13	4.8		87.3 ± 9.0	13.3
	VGG11	1%	91.9	34.9 ± 12.4	93.1	45.8 ± 11	10.9	94.0	47.1 ± 14	12.2
		0.5%		64.9 ± 10.8		81.8 ± 5.0	16.9		84.2 ± 4.8	19.3
	VGG16	1%	95.2	15.1 ± 6.8	95.5	26.5 ± 12	11.4	95.5	32.4 ± 11	17.3
		0.5%		58.8 ± 8.9		82.2 ± 9.0	23.4		85.4 ± 6.7	26.6
	VGG19	1%	95.5	36.6 ± 6.8	94.9	53.2 ± 14	16.6	95.6	60.9 ± 15	24.3
		0.5%		69.1 ± 11.1		85.4 ± 4.5	16.3		87.5 ± 3.7	18.4
VGG19	ResNet18	1%	95.5	36.9 ± 16.0	93.7	53.1 ± 16	16.2	95.0	63.4 ± 18	26.5
		0.5%		75.2 ± 12.7		83.9 ± 7.6	8.7		89.7 ± 4.8	14.5
	ResNet50	1%	95.0	29.5 ± 16.9	92.8	30.6 ± 18	1.1	95.4	38.9 ± 22	9.4
		0.5%		74.0 ± 13.0		74.7 ± 18	0.7		81.5 ± 16	7.5
	VGG11	1%	91.9	34.9 ± 12.4	93.7	50.6 ± 11	15.7	95.1	58.9 ± 15	24.0
		0.5%		64.9 ± 10.8		82.3 ± 5.1	17.4		87.5 ± 3.7	22.6
	VGG16	1%	95.2	15.1 ± 6.8	95.2	27.8 ± 15	12.7	95.2	33.5 ± 14	18.4
		0.5%		58.8 ± 8.9		79.0 ± 12	20.2		81.8 ± 7.8	23.0
	VGG19	0.5%	95.5	69.1 ± 11.1	96.0	94.0 ± 2.2	24.9	95.4	93.9 ± 2.1	24.8

[Note] S.M.: *source model, used for training generators*, T.M.: *target model, used for testing generators*, B.E.R.: *the bit error rate of the target model*, C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*

H.3 CIFAR-100

In Tables 18 and 19, we show results on CIFAR-100 with NeuralFuse trained at 1% and 0.5% B.E.R., respectively.

Table 18: Transfer results on CIFAR-100: NeuralFuse trained on S.M. with 1% B.E.R.

S.M.	T.M.	B.E.R.	C.A.	P.A.	ConvL (1%)			UNetL (1%)		
					C.A. (NF)	P.A. (NF)	R.P.	C.A. (NF)	P.A. (NF)	R.P.
ResNet18	ResNet18	0.5%	73.7	20.9 ± 7.4	54.8	35.8 ± 5.2	14.9	50.6	39.3 ± 2.8	18.4
		0.35%		31.4 ± 7.6		41.7 ± 3.7	10.3		43.3 ± 1.4	11.9
	ResNet50	1%	73.5	3.0 ± 1.8	44.9	2.2 ± 2.0	-0.8	41.5	2.4 ± 1.9	-0.6
		0.5%		21.3 ± 7.0		15.9 ± 8.2	-5.4		17.1 ± 7.1	-4.2
		0.35%		35.7 ± 8.6		23.7 ± 7.1	-12.0		26.2 ± 5.6	-9.5
	VGG11	1%	64.8	8.2 ± 5.7	41.2	9.8 ± 5.6	1.6	37.5	10.2 ± 5.1	2.0
		0.5%		23.9 ± 9.4		24.2 ± 5.9	0.3		24.5 ± 4.7	0.6
		0.35%		31.3 ± 10.0		29.0 ± 5.4	-2.3		28.2 ± 4.5	-3.1
	VGG16	1%	67.8	7.0 ± 3.5	44.0	7.9 ± 3.7	0.9	39.5	10.1 ± 4.5	3.1
		0.5%		22.4 ± 7.0		22.4 ± 7.6	0.0		26.3 ± 5.3	3.9
		0.35%		31.1 ± 7.2		28.1 ± 5.9	-3.0		30.6 ± 3.6	-0.5
	VGG19	1%	67.8	10.6 ± 4.3	44.2	13.5 ± 6.1	2.9	40.7	15.6 ± 6.2	5.0
		0.5%		34.0 ± 9.6		27.9 ± 4.8	-6.1		29.3 ± 4.6	-4.7
		0.35%		42.1 ± 9.4		33.2 ± 48	-8.9		32.8 ± 3.9	-9.3
VGG19	ResNet18	1%	73.7	4.6 ± 2.9	55.5	5.8 ± 3.7	1.2	57.3	6.8 ± 4.4	2.2
		0.5%		20.9 ± 7.4		24.6 ± 6.3	3.7		28.1 ± 5.9	7.2
		0.35%		31.4 ± 7.6		31.1 ± 5.0	-0.3		36.4 ± 4.5	5.0
	ResNet50	1%	73.5	3.0 ± 1.8	56.1	2.8 ± 2.1	-0.2	56.1	3.7 ± 2.4	0.7
		0.5%		21.3 ± 7.0		18.9 ± 8.6	-2.4		22.8 ± 8.5	1.5
		0.35%		35.7 ± 8.6		28.7 ± 8.2	-7.0		33.7 ± 7.0	-2.0
	VGG11	1%	64.8	8.2 ± 5.7	52.8	12.3 ± 8.4	4.1	53.9	15.4 ± 9.4	7.2
		0.5%		23.9 ± 9.4		30.0 ± 9.3	6.1		33.3 ± 7.2	9.4
		0.35%		31.3 ± 10.0		36.5 ± 7.7	5.2		38.8 ± 6.5	7.5
	VGG16	1%	67.8	7.0 ± 3.5	53.6	11.2 ± 4.4	4.2	55.2	13.6 ± 5.2	6.6
		0.5%		22.4 ± 7.0		32.4 ± 7.3	10.0		35.9 ± 6.2	13.5
		0.35%		31.1 ± 7.2		39.4 ± 6.3	8.3		42.4 ± 4.9	11.3
	VGG19	0.5%	67.8	34.0 ± 9.6	59.4	50.2 ± 3.1	16.2	58.7	49.1 ± 3.5	15.1
		0.35%		42.1 ± 9.4		53.1 ± 2.3	11.0		52.0 ± 3.1	9.9

[Note] S.M.: *source model, used for training generators*, T.M.: *target model, used for testing generators*, B.E.R.: *the bit error rate of the target model*, C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*

Table 19: Transfer results on CIFAR-100: NeuralFuse trained on S.M. with 0.5% B.E.R.

S.M.	T.M.	B.E.R.	C.A.	P.A.	ConvL (0.5%)			UNetL (0.5%)		
					C.A. (NF)	P.A. (NF)	R.P.	C.A. (NF)	P.A. (NF)	R.P.
ResNet18	ResNet18	0.35%	73.7	31.4 ± 7.6	65.2	47.7 ± 4.9	16.3	66.2	49.2 ± 4.1	17.8
	ResNet50	0.5%	73.5	21.3 ± 7.0	62.5	24.0 ± 9.9	2.8	63.5	26.4 ± 9.1	5.1
		0.35%		35.7 ± 8.6		36.3 ± 8.9	0.6		39.4 ± 8.1	3.7
	VGG11	0.5%	64.8	23.9 ± 9.4	59.2	33.0 ± 9.8	9.2	61.1	34.2 ± 9.8	10.3
		0.35%		31.3 ± 10.0		40.4 ± 8.7	9.1		41.4 ± 9.0	10.1
	VGG16	0.5%	67.8	22.4 ± 7.0	59.5	34.7 ± 8.0	12.3	61.4	37.5 ± 6.8	15.2
		0.35%		31.1 ± 7.2		42.9 ± 6.0	11.8		45.3 ± 4.9	14.2
	VGG19	0.5%	67.8	34.0 ± 9.6	61.6	43.7 ± 6.2	9.6	62.0	45.0 ± 6.3	11.0
		0.35%		42.1 ± 9.4		49.0 ± 5.5	6.8		50.5 ± 5.3	8.3
VGG19	ResNet18	0.5%	73.7	20.9 ± 7.4	66.1	24.9 ± 6.7	4.0	67.8	27.7 ± 6.8	6.8
		0.35%		31.4 ± 7.6		34.4 ± 5.4	3.0		38.1 ± 5.6	6.7
	ResNet50	0.5%	73.5	21.3 ± 7.0	66.2	22.7 ± 7.8	1.4	66.7	25.4 ± 8.0	4.2
		0.35%		35.7 ± 8.6		35.5 ± 7.7	-0.2		38.8 ± 7.5	3.2
	VGG11	0.5%	64.8	23.9 ± 9.4	59.9	29.3 ± 10	5.4	61.0	31.2 ± 9.8	7.4
		0.35%		31.3 ± 10.0		36.6 ± 9.5	5.3		38.1 ± 9.0	6.8
	VGG16	0.5%	67.8	22.4 ± 7.0	62.5	30.8 ± 7.3	8.4	62.6	33.0 ± 7.3	10.7
		0.35%		31.1 ± 7.2		40.0 ± 6.5	8.9		42.5 ± 5.9	11.3
	VGG19	0.35%	67.8	42.1 ± 9.4	65.6	52.0 ± 6.2	9.8	65.5	52.6 ± 6.1	10.4

[Note] S.M.: *source model, used for training generators*, T.M.: *target model, used for testing generators*, B.E.R.: *the bit error rate of the target model*, C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*

H.4 GENERATOR ENSEMBLING

To improve the transferability performance on cross-architecture cases (e.g., using ResNet-based models as surrogate models to train NeuralFuse and then transfer NeuralFuse to VGG-based target models), we try to adopt ensemble surrogate models to train our NeuralFuse. The experimental results are shown in Table 20. We use the same experimental settings mentioned in Table 1 but change one source model (e.g., ResNet18 or VGG19) into two (ResNet18 with VGG19) for training. The results show that the overall performance is better than the results shown in Table 1, which means ensemble-based training can easily solve the performance degradation on cross-architecture target models.

Table 20: Transfer results on CIFAR-10: NeuralFuse trained on two S.M. with 1.5% B.E.R.

S.M.	T.M.	B.E.R.	C.A.	P.A.	ConvL (1.5%)			UNetL (1.5%)		
					C.A. (NF)	P.A. (NF)	R.P.	C.A. (NF)	P.A. (NF)	R.P.
ResNet18 + VGG19	ResNet18	1%	92.6	38.9 ± 12.4	89.4	88.1 ± 1.0	49.2	86.3	85.4 ± 0.5	46.5
		0.5%		70.1 ± 11.6		89.2 ± 0.2	19.1		86.1 ± 0.2	16.0
	ResNet50	1%	92.6	26.1 ± 9.4	89.3	44.0 ± 22	17.9	86.1	50.9 ± 20	24.8
		0.5%		61.0 ± 10.3		80.3 ± 6.7	19.3		78.6 ± 3.9	17.6
	VGG11	1%	88.4	42.2 ± 11.6	89.1	77.0 ± 5.6	34.8	85.9	82.3 ± 4.1	40.1
		0.5%		63.6 ± 9.3		87.5 ± 1.6	23.9		85.0 ± 0.6	21.4
	VGG16	1%	90.3	35.7 ± 7.9	89.1	80.5 ± 8.6	44.8	85.7	81.4 ± 5.5	45.7
		0.5%		66.6 ± 8.1		88.2 ± 0.7	21.6		85.0 ± 0.7	18.4
	VGG19	1%	90.5	36.0 ± 12.0	89.2	75.1 ± 17	39.1	86.1	83.0 ± 3.4	47.0
		0.5%		64.2 ± 12.4		89.0 ± 0.2	24.8		85.9 ± 0.4	21.7

[Note] S.M.: *source model, used for training generators*, T.M.: *target model, used for testing generators*, B.E.R.: *the bit error rate of the target model*, C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*

I NEURALFUSE ON REDUCED-PRECISION QUANTIZATION AND RANDOM BIT ERRORS

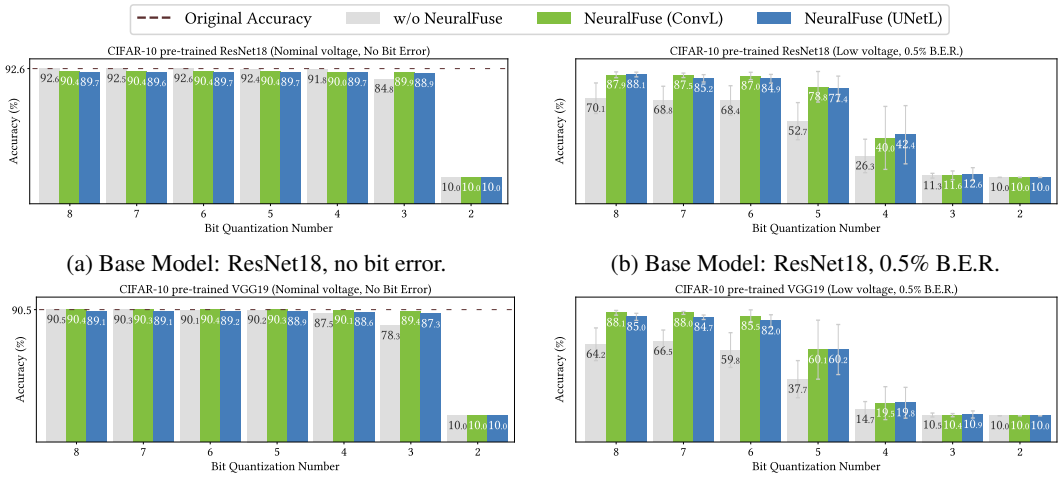
As mentioned in Sec. 4.6, we explore the robustness of NeuralFuse to low-precision quantization of model weights and consider the case of random bit errors. Here, we demonstrate that NeuralFuse can recover not only the accuracy drop due to reduced precision, but also the drop caused by low-voltage-induced bit errors (0.5% B.E.R.) under low precision. We selected two NeuralFuse generators (ConvL and UNetL) for our experiments, and these generators were trained with the corresponding base models (ResNet18 and VGG19) at 1% B.E.R. (CIFAR-10, GTSRB) and 0.5% B.E.R. (ImageNet-10). The experimental results are shown as follows: CIFAR-10 (Sec. I.1), GTSRB (Sec. I.2), and ImageNet-10 (Sec. I.3). Similarly, for ease of comparison, we visualize the experimental results in the figures below each table. Our results show that NeuralFuse can consistently perform well in low-precision regimes as well as recover the low-voltage-induced accuracy drop.

I.1 CIFAR-10

Table 21: Reduced-precision Quantization and with 0.5% B.E.R. on CIFAR-10 pre-trained models.

Base Model	#Bits	C.A.	P.A.	ConvL (1%)			UNetL (1%)		
				C.A. (NF)	P.A. (NF)	R.P.	C.A. (NF)	P.A. (NF)	R.P.
ResNet18	8	92.6	70.1 ± 11.6	89.8	89.5 ± 0.2	19.4	86.6	86.2 ± 0.3	16.1
	7	92.5	68.8 ± 10.4	89.8	89.5 ± 1.7	20.7	86.5	86.0 ± 0.5	17.2
	6	92.6	68.4 ± 11.2	89.7	89.5 ± 0.2	21.1	86.6	85.9 ± 0.3	17.5
	5	92.4	52.7 ± 14.1	89.7	90.0 ± 0.7	37.3	86.5	85.5 ± 0.8	32.8
	4	91.8	26.3 ± 12.7	89.8	58.7 ± 24.5	32.4	86.6	64.9 ± 22.5	38.6
	3	84.8	11.3 ± 1.8	89.8	12.8 ± 5.8	1.5	86.0	14.8 ± 10.0	3.5
	2	10.0	10.0 ± 0.0	10.0	10.0 ± 0.0	0.0	10.0	10.0 ± 0.0	0.0
VGG19	8	90.5	64.2 ± 12.4	89.8	89.6 ± 8.7	25.4	87.4	86.8 ± 0.4	22.6
	7	90.3	66.5 ± 8.5	89.8	89.6 ± 0.2	23.1	87.4	86.7 ± 0.3	20.2
	6	90.1	59.8 ± 13.2	89.9	89.4 ± 3.8	29.6	87.4	86.4 ± 0.7	26.6
	5	90.2	37.7 ± 14.1	89.8	78.0 ± 15.8	40.3	87.2	79.8 ± 0.8	42.1
	4	87.5	14.7 ± 6.0	89.8	27.8 ± 18.9	13.1	87.2	34.4 ± 20.5	19.7
	3	78.3	10.5 ± 1.5	89.7	10.9 ± 2.6	0.4	86.8	11.0 ± 2.9	0.5
	2	10.0	10.0 ± 0.0	10.0	10.0 ± 0.0	0.0	10.0	10.0 ± 0.0	0.0

[Note] C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*



(a) Base Model: ResNet18, no bit error.

(b) Base Model: ResNet18, 0.5% B.E.R.

(c) Base Model: VGG19, no bit error.

(d) Base Model: VGG19, 0.5% B.E.R.

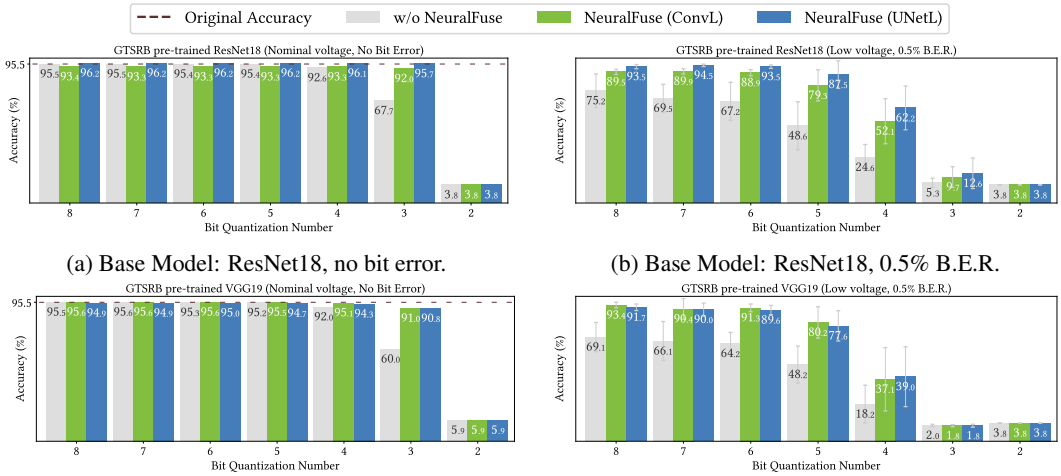
Figure 12: Results of Reduced-precision and bit errors (0.5%) on CIFAR-10 pre-trained base models.

I.2 GTSRB

Table 22: Reduced-precision Quantization and with 0.5% B.E.R. on GTSRB pre-trained models.

Base Model	#Bits	C.A.	P.A.	ConvL (1%) C.A. (NF)	ConvL (1%) P.A. (NF)	ConvL (1%) R.P.	UNetL (1%) C.A. (NF)	UNetL (1%) P.A. (NF)	UNetL (1%) R.P.
ResNet18	8	95.5	75.2 ± 12.7	95.7	95.3 ± 0.5	20.1	96.2	95.7 ± 0.3	20.5
	7	95.5	69.5 ± 10.6	95.7	95.3 ± 0.3	25.8	96.2	95.9 ± 0.3	26.4
	6	95.4	67.2 ± 14.4	95.7	95.2 ± 0.5	28.0	96.2	95.7 ± 0.5	28.5
	5	95.4	48.6 ± 18.2	95.8	92.6 ± 5.1	44.0	96.2	94.8 ± 2.5	46.2
	4	92.6	24.6 ± 9.8	95.9	75.6 ± 16.2	51.0	96.2	86.6 ± 9.5	62.0
	3	67.7	5.3 ± 3.5	95.4	18.4 ± 15.3	13.1	96.2	25.3 ± 22.5	20.0
	2	3.8	3.8 ± 0.0	4.1	3.8 ± 0.0	0.0	3.8	3.8 ± 0.0	0.0
VGG19	8	95.5	69.1 ± 11.1	96.0	94.0 ± 2.2	24.9	95.4	93.9 ± 2.1	24.8
	7	95.6	66.1 ± 14.8	96.0	92.2 ± 5.7	26.1	95.4	92.6 ± 3.7	26.5
	6	95.3	64.2 ± 8.4	96.0	92.2 ± 5.7	28.0	95.4	92.3 ± 2.3	28.1
	5	95.2	48.2 ± 14.0	96.0	92.2 ± 5.7	44.0	95.4	86.2 ± 8.4	38.0
	4	92.0	18.2 ± 14.3	93.0	92.2 ± 5.7	74.0	95.0	49.6 ± 22.8	31.4
	3	60.0	2.0 ± 0.9	87.3	92.2 ± 5.7	90.2	87.2	1.7 ± 0.9	-0.3
	2	5.9	3.8 ± 0.0	5.9	3.8 ± 0.0	0.0	5.9	3.8 ± 0.0	0.0

[Note] C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*



(a) Base Model: ResNet18, no bit error.

(b) Base Model: ResNet18, 0.5% B.E.R.

(c) Base Model: VGG19, no bit error.

(d) Base Model: VGG19, 0.5% B.E.R.

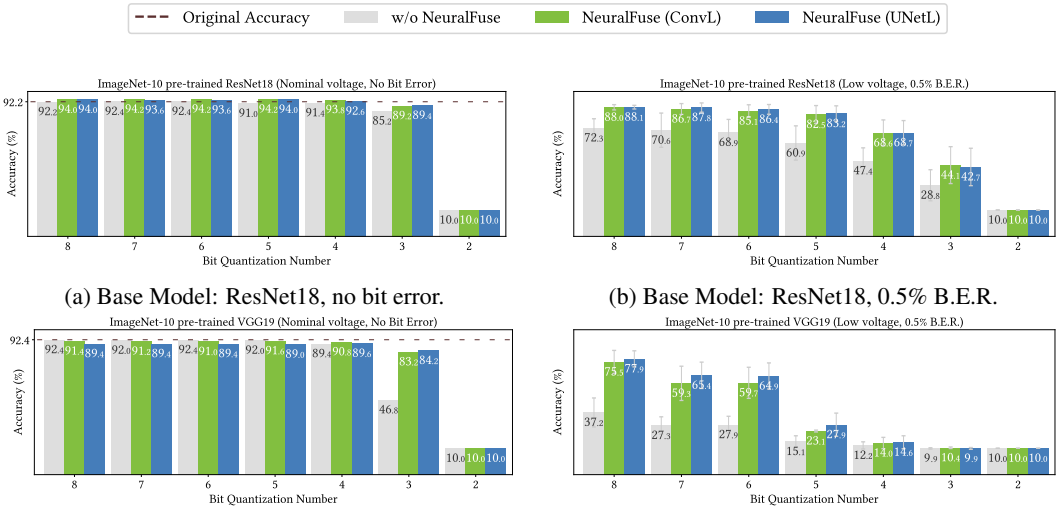
Figure 13: Results of Reduced-precision and bit errors (0.5%) on GTSRB pre-trained base models.

I.3 IMAGENET-10

Table 23: Reduced-precision Quantization and with 0.5% B.E.R. on ImageNet-10 pre-trained models.

Base Model	#Bits	C.A.	P.A.	ConvL (0.5%) C.A. (NF)	ConvL (0.5%) P.A. (NF)	ConvL (0.5%) R.P.	UNetL (0.5%) C.A. (NF)	UNetL (0.5%) P.A. (NF)	UNetL (0.5%) R.P.
ResNet18	8	92.2	72.3 ± 7.0	94.0	88.0 ± 2.0	15.7	94.0	88.1 ± 1.4	15.8
	7	92.4	70.6 ± 13.0	94.2	86.7 ± 4.1	16.1	93.6	87.8 ± 3.5	17.2
	6	92.4	68.9 ± 9.9	94.2	85.1 ± 4.8	16.2	93.6	86.4 ± 3.7	17.5
	5	91.0	60.9 ± 13.0	94.2	82.5 ± 6.8	21.6	94.0	83.2 ± 5.9	22.3
	4	91.4	47.4 ± 9.8	93.8	68.6 ± 9.8	21.2	92.6	68.7 ± 9.2	21.3
	3	85.2	28.8 ± 11.8	89.2	44.1 ± 14.0	15.3	89.4	42.7 ± 14.2	13.9
	2	10.0	10.0 ± 0.0	10.0	10.0 ± 0.0	0.0	10.0	10.0 ± 0.0	0.0
VGG19	8	92.4	37.2 ± 11.0	91.4	75.5 ± 8.8	38.3	89.4	77.9 ± 6.1	40.7
	7	92.0	27.3 ± 6.6	91.2	59.3 ± 13.0	32.0	89.4	65.4 ± 10.0	38.1
	6	92.4	27.9 ± 6.4	91.0	59.7 ± 11.8	31.8	89.4	64.9 ± 9.9	37.0
	5	92.0	15.1 ± 4.4	91.6	23.1 ± 0.7	8.0	89.0	27.9 ± 8.8	12.8
	4	89.4	12.2 ± 2.7	90.8	14.0 ± 4.3	1.8	89.6	14.6 ± 4.9	2.4
	3	46.8	9.9 ± 0.5	83.2	10.4 ± 0.6	0.5	84.2	9.9 ± 0.7	0.0
	2	10.0	10.0 ± 0.0	10.0	10.0 ± 0.0	0.0	10.0	10.0 ± 0.0	0.0

[Note] C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*



(a) Base Model: ResNet18, no bit error.

(b) Base Model: ResNet18, 0.5% B.E.R.

(c) Base Model: VGG19, no bit error.

(d) Base Model: VGG19, 0.5% B.E.R.

Figure 14: Results of Reduced-precision and bit errors (0.5%) on ImageNet-10 pre-trained base models.

J Data Embeddings Visualization

To further understand how our proposed NeuralFuse works, we visualize the output distribution from the final linear layer of the base models and project the results onto the 2D space using t-SNE (van der Maaten & Hinton, 2008). Figure 15 shows the output distribution from ResNet18 (trained on CIFAR-10) under a 1% bit error rate. We chose two generators that have similar architecture: ConvL and ConvS, for this experiment. We can observe that: (a) The output distribution of the clean model without NeuralFuse can be grouped into 10 classes denoted by different colors. (b) The output distribution of the perturbed model under a 1% bit error rate without NeuralFuse shows mixed representations and therefore degraded accuracy. (c) The output distribution of the clean model with ConvL shows that applying NeuralFuse will not hurt the prediction of the clean model too much (i.e., it retains high accuracy in the regular voltage setting). (d) The output distribution of the perturbed model with ConvL shows high separability (and therefore high perturbed accuracy) as opposed to (b). (e)/(f) shows the output distribution of the clean/perturbed model with ConvS. For both (e) and (f), we can see nosier clustering when compared to (c) and (d), which means the degraded performance of ConvS compared to ConvL. The visualization validates that NeuralFuse can help retain good data representations under random bit errors and that larger generators in NeuralFuse have better performance than smaller ones.

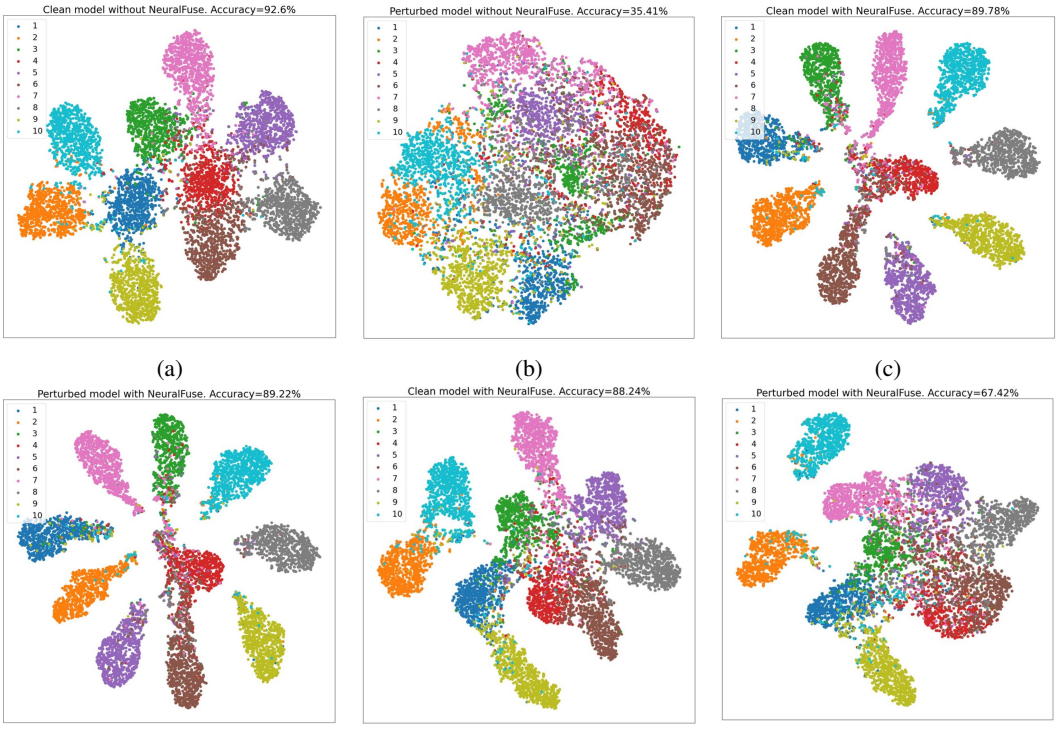


Figure 15: t-SNE results for ResNet18 trained by CIFAR-10 under 1% of bit error rate. (a) Clean model. (b) Perturbed model. (c) Clean model with ConvL. (d) Perturbed model with ConvL. (e) Clean model with ConvS. (f) Perturbed model with ConvS.

K Qualitative Analysis of Transformed Inputs

In this section, we conduct a qualitative study to visualize the images which are transformed by NeuralFuse, and then present some properties of these images. We adopt six different architectures of NeuralFuse generators trained with ResNet18 under a 1% bit error rate. In Figure 16(a), we show several images from the truck class in CIFAR-10. We observe that different images in the same class transformed by the same NeuralFuse will exhibit a similar pattern. For example, the

patterns contain several circles, which may symbolize the wheels of the trucks. In Figure 16(b), we show several images of a traffic sign category (No Overtaking) in GTSRB. We also oversee that the transformed images contain similar patterns. In particular, in GTSRB, NeuralFuse will generate patterns that highlight the shape of the sign with a green background, even if the original images are of a dark background and under different lighting conditions.

In Figure 17, we show the images from ten different classes in CIFAR-10 and GTSRB separately. The transformed images have distinct patterns for each class. Therefore, we speculate that NeuralFuse effectively transforms images to some class-specific patterns such that the associated features are robust to random bit errors and can be easily recognizable by the base model in low-voltage settings.

Figure 16: Visualization results of the transformed images from six different NeuralFuse generators trained with ResNet18 under 1% bit error rate. (a) *Truck* class in CIFAR-10. (b) *No Overtaking (general)* sign in GTSRB.

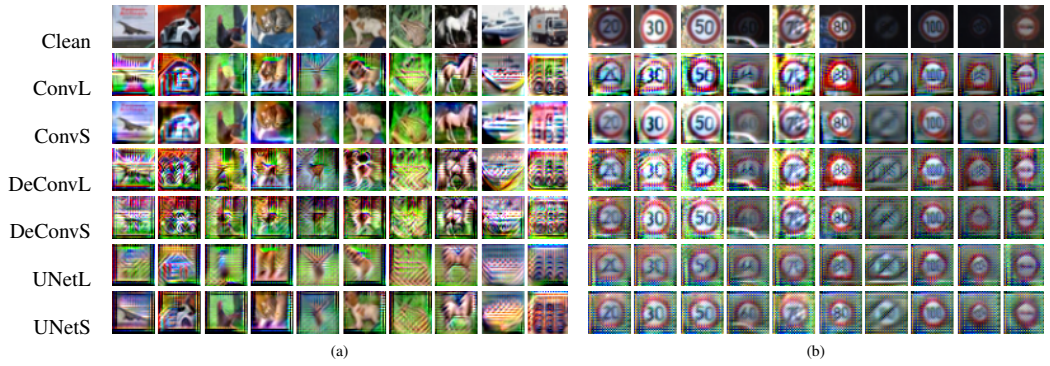


Figure 17: Visualization results of the transformed images from six different NeuralFuse generators trained by ResNet18 with 1% bit error rate. (a) Ten different classes sampled from CIFAR-10. (b) Ten different traffic signs sampled from GTSRB.

In Figure 18, we show several images from the apple class in CIFAR-100. We observe that the different images transformed by the same NeuralFuse will provide the similar patterns. This observation is similar to CIFAR-10 and GTSRB mentioned above. In Figure 19, we show more different classes and their corresponding transformed results.

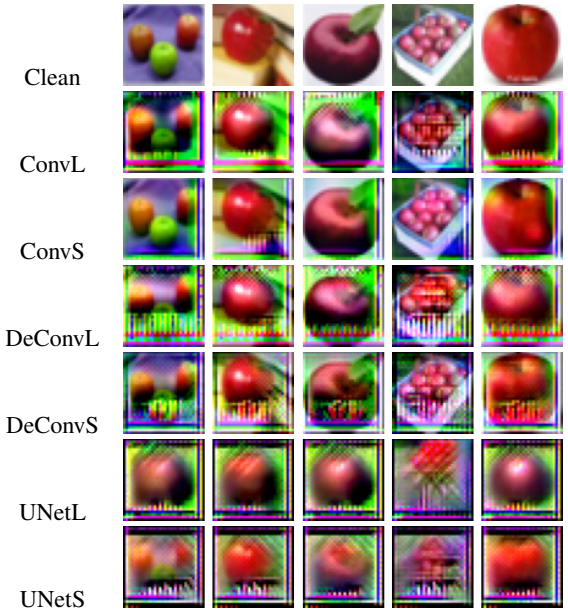


Figure 18: Visualization results of the transformed images on CIFAR-100 from six different NeuralFuse generators trained with ResNet18 under 1% of bit error rate.

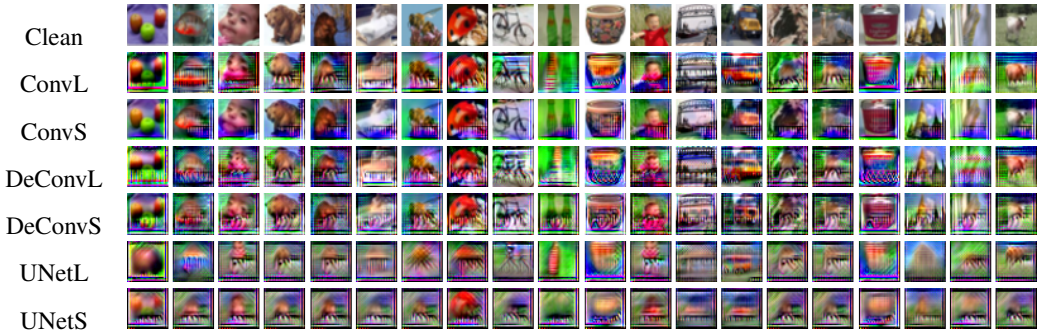


Figure 19: Visualization results of twenty different classes of the transformed images from CIFAR-100 made by six different NeuralFuse generators, which are trained with ResNet18 under 1% of bit error rate.

L ADDITIONAL EXPERIMENTS ON ADVERSARIAL TRAINING

Adversarial training is a common strategy to derive a robust neural network against certain perturbations. By training the generator using adversarial training proposed in Stutz et al. (2021), we report its performance against low voltage-induced bit errors. We use ConvL as the generator and ResNet18 as the base model, trained on CIFAR-10. Furthermore, we explore different K flip bits as the perturbation on weights of the base model during adversarial training, and then for evaluation, the trained-generator will be applied against 1% of bit errors rate on the base model. The results are shown in Table 24. After careful tuning of hyperparameters, we find that we are not able to obtain satisfactory recovery when adopting adversarial training. Empirically, we argue that adversarial training may not be suitable for training generator-based methods.

Table 24: Performance of the generator trained by adversarial training under K flip bits on ResNet18 with CIFAR-10. The results show that the generator trained by adversarial training cannot achieve high accuracy against bit errors under 1% bit error rate.

K-bits	C.A.	P.A.	C.A. (NF)	P.A. (NF)	R.P.
100	92.6	38.9 ± 12.4	92.4	38.3 ± 12.1	-0.6
500			92.1	38.7 ± 12.5	-0.2
5,000			92.6	38.9 ± 12.5	0
20,000			60.1	23.0 ± 8.1	-16
100,000			71.1	23.6 ± 6.6	-16

[Note] C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, NF: *NeuralFuse*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*

M ADDITIONAL EXPERIMENTS ON ROBUST MODEL TRAINED WITH ADVERSARIAL WEIGHT PERTURBATION WITH NEURALFUSE

Previously, Wu et al. proposed that one could obtain a more robust model via adversarial weight perturbation (Wu et al., 2020). To seek whether such models could also be robust to random bit errors, we conducted an experiment on CIFAR-10 with the proposed adversarially trained PreAct ResNet18. The experimental results are shown in Table 25. We find that the average perturbed accuracy is 23% and 63.2% for PreAct ResNet18 under 1% and 0.5% B.E.R., respectively. This result is lower than 38.9% and 70.1% from ResNet18 in Table 11, indicating their poor generalization ability against random bit errors. Nevertheless, when equipped NeuralFuse on the perturbed model, we could still witness a significant recover percentage under both 1% and 0.5% B.E.R. This result further demonstrates that NeuralFuse could be adapted to various models (i.e., trained in different learning algorithms).

Table 25: Performance of NeuralFuse trained with rubust CIFAR-10 pre-trained PreAct ResNet18. The results show that NeuralFuse can be used together with a robust model and further improve perturbed accuracy under both 1% and 0.5% B.E.R.

Base Model	B.E.R.	NF	C.A.	P.A.	C.A. (NF)	P.A. (NF)	R.P.
PreAct ResNet18	1%	ConvL	89.7	23.0 ± 9.3	87.6	53.7 ± 26	30.7
		ConvS			83.1	34.6 ± 15	11.6
		DeConvL			87.7	55.4 ± 27	32.4
		DeConvS			82.9	32.4 ± 14	9.4
		UNetL			86.1	60.4 ± 28	37.4
		UNetS			80.4	51.9 ± 24	28.9
	0.5%	ConvL	89.7	63.2 ± 8.7	89.2	87.8 ± 1.1	24.6
		ConvS			89.2	74.0 ± 6.5	10.8
		DeConvL			89.0	87.4 ± 1.1	24.2
		DeConvS			89.9	74.4 ± 7.0	11.2
		UNetL			87.5	85.9 ± 0.8	22.7
		UNetS			88.2	80.4 ± 3.9	17.2

[Note] B.E.R.: *the bit error rate of the base model*, NF: *NeuralFuse*, C.A. (%): *clean accuracy*, P.A. (%): *perturbed accuracy*, and R.P.: *total recover percentage of P.A. (NF) v.s. P.A.*

N Inference Latency of NeuralFuse

In Table 26, we report the latency (batch_size=1, CIFAR-10/ImageNet-10 testing dataset) of utilizing the different NeuralFuse generators with two different base models, ResNet18 and VGG19. We can see that although NeuralFuse indeed brings a certain degree of extra latency, we argue that this is an unavoidable factor; however, since the latency is measured on a general-purpose GPU (i.e. V100), when the base model and NeuralFuse are deployed on a custom accelerator, we believe this delay will be further reduced.

Table 26: The Inference Latency of base model and base model with NeuralFuse.

	ResNet18 (CIFAR-10)	VGG19 (CIFAR-10)	ResNet18 (ImageNet-10)	VGG19 (ImageNet-10)
Base Model	5.84 ms	5.32 ms	6.21 ms	14.34 ms
+ ConvL	9.37 ms (+3.53)	8.96 ms (+3.64)	10.51 ms (+4.3)	17.66 ms (+3.32)
+ ConvS	7.86 ms (+2.02)	7.40 ms (+2.08)	8.28 ms (+2.07)	16.72 ms (+2.38)
+ DeConvL	9.18 ms (+3.34)	8.59 ms (+3.27)	10.07 ms (+3.86)	17.24 ms (+2.90)
+ DeConvS	7.49 ms (+1.65)	7.04 ms (+1.72)	7.79 ms (+1.58)	15.67 ms (+1.33)
+ UNetL	10.69 ms (+4.85)	10.06 ms (+4.74)	11.14 ms (+4.93)	18.54 ms (+4.20)
+ UNetS	10.63 ms (+4.79)	10.13 ms (+4.81)	11.36 ms (+5.15)	18.60 ms (+4.26)

O Discussion for Real-World Application or Potential use cases

In this section, we provide some possible real-world applications for using NeuralFuse under low-voltage regimes. Previous works have pointed out some possible scenarios that suffer from energy concerns and hence need some strategies to reduce energy consumption. For example, in Yang et al. (2017; 2019a), the authors mention that due to the high computation cost of CNN processing and some DNN-based vision algorithms, they will incur high energy consumption. This will significantly reduce the battery life of battery-powered devices, indirectly impacting the user experience of the devices. Therefore, to avoid the aforementioned issues, we can mitigate the device's energy consumption by lowering the operating voltage and then incorporating NeuralFuse to recover model performance, reducing the side effects caused by low voltage.