Forget and Rewire: Enhancing the Resilience of Transformer-based Models against Bit-Flip Attacks

Najmeh Nazari, Hosein Makrani, Chongzhou Fang, Hossein Sayadi,
Setareh Rafatirad, Khaled N. Khasawneh, and Houman Homayoun

33rd USENIX Security
August 2024

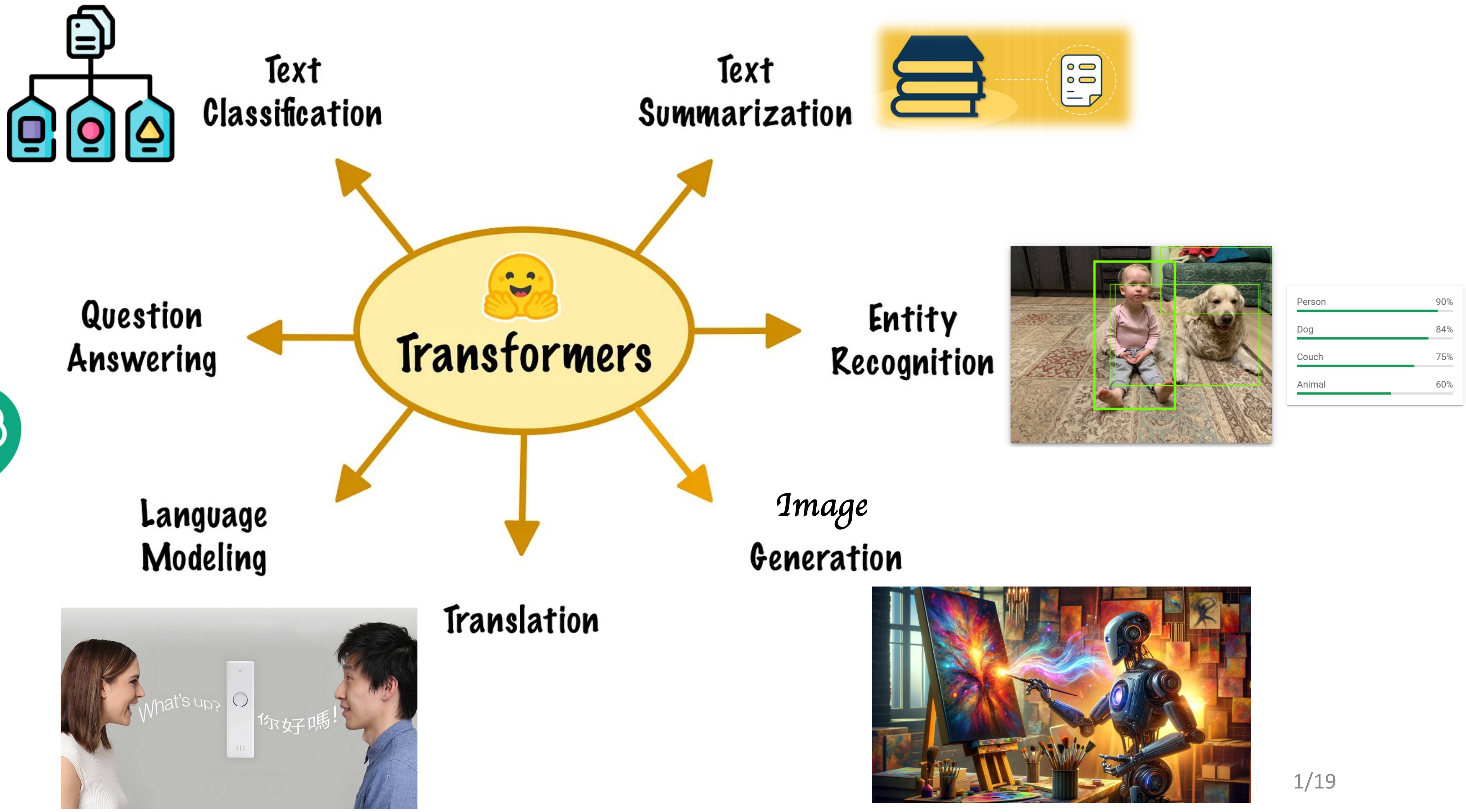
Outline

- Background and Problem Statement
- Threat Model
- Inspiration and Contribution
- Forget and Rewire
- Evaluation and Discussion
- Conclusion

Outline

- Background and Problem Statement
- Threat Model
- Inspiration and Contribution
- Forget and Rewire
- Evaluation and Discussion
- Conclusion

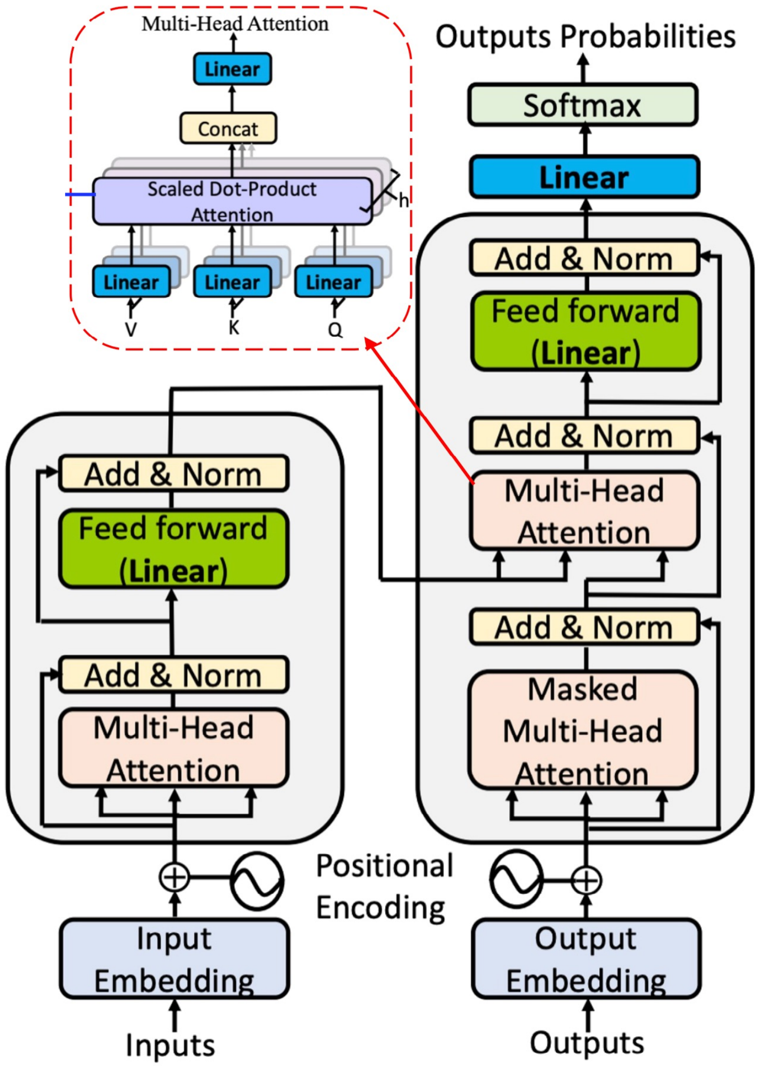
Transformer Models



ChatGPT

Transformer Models

- ❑ Linear layer
  - ❑ Efficient (training & inference)
  - ❑ Scalable



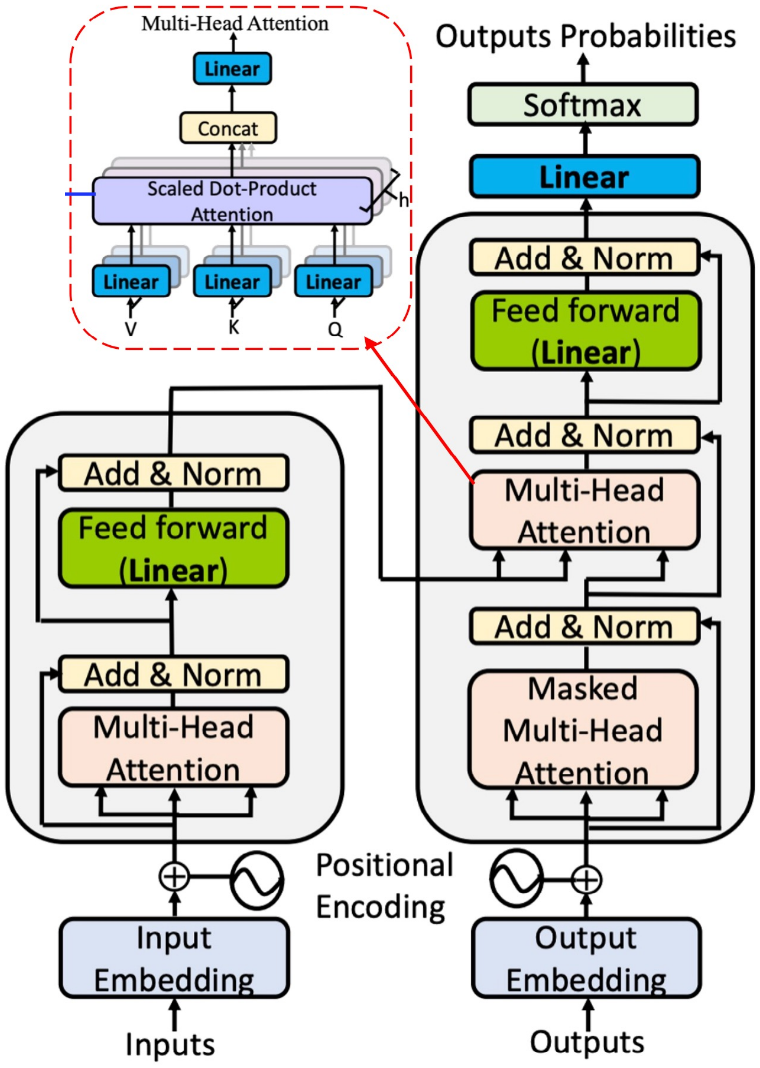
Vaswani, Ashish, and et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Transformer Models

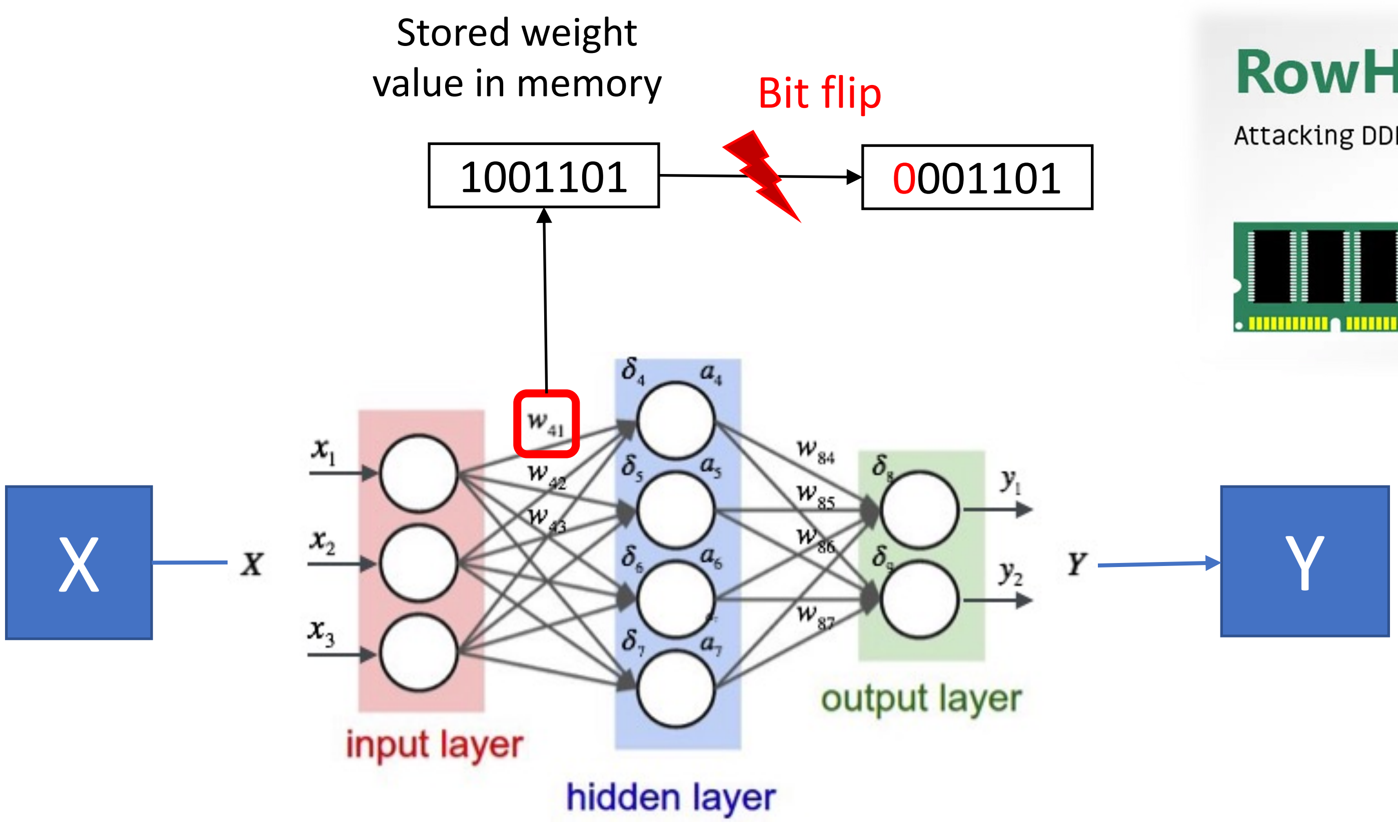
- Linear layer
  - Efficient (training & inference)
  - Scalable

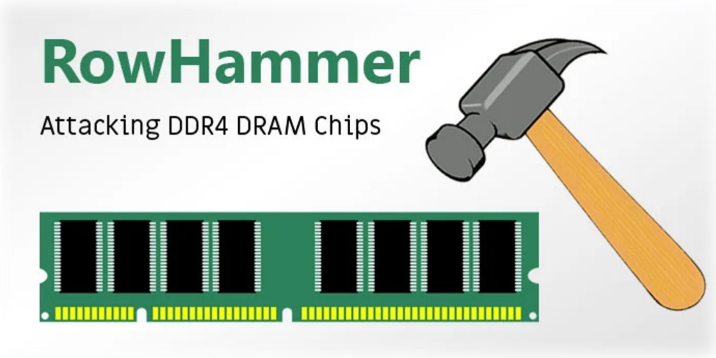
- Vulnerable to Bit Flip Attack

Can we use Linear layer to increase the resilience of Transformers against BFAs?



Bit Flip Attack

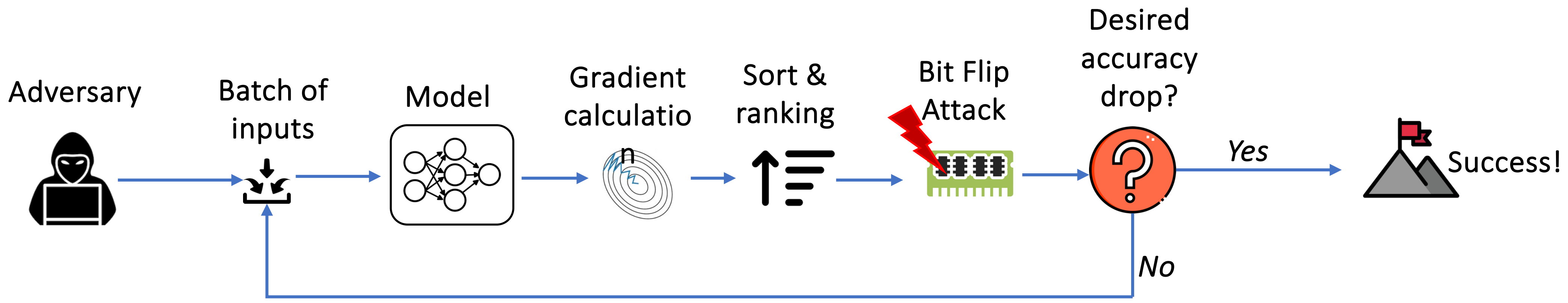




Neural network: Y = F(X . W + b)

Yao, Fan, and et all. "DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips." In *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

Flow of Bit Flip Attack



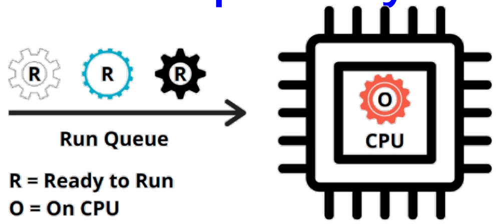
Attacker goal: use minimum number of BF for degrading the accuracy

Outline

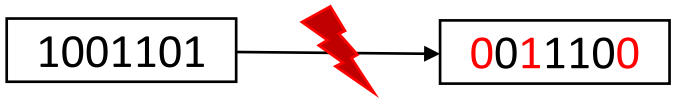
- Background and Problem Statement
- Threat Model
- Inspiration and Contribution
- Forget and Rewire
- Evaluation and Discussion
- Conclusion

Threat Model

- **Assumptions** about the Adversary
  - The adversary has the capability to execute BFAs after model deployment.



  - Adversary can manipulate multiple bits of multiple parameters.



  - Assumption of a white-box attack scenario (complete knowledge of the original model's architecture and parameters).

Types of Attackers

- Basic Adversary
- Expert Adversary
- Oracle Adversary

Types of Attackers

	Basic Adversary	Expert Adversary	Oracle Adversary
Knowledge of Defense	✗		
Access to defense configuration	✗		
Access to gradient on deployed model	✗		
Requires equivalent model for BFA testing	✓		

Types of Attackers

	Basic Adversary	Expert Adversary	Oracle Adversary
Knowledge of Defense	✗	✓	
Access to defense configuration	✗	✗	
Access to gradient on deployed model	✗	✓	
Requires equivalent model for BFA testing	✓	✗	

Types of Attackers

	Basic Adversary	Expert Adversary	Oracle Adversary
Knowledge of Defense	✗	✓	✓
Access to defense configuration	✗	✗	✓
Access to gradient on deployed model	✗	✓	✓
Requires equivalent model for BFA testing	✓	✗	✗

Outline

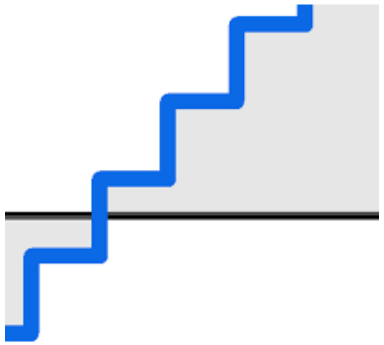
- ❑ Background and Problem Statement
- ❑ Threat Model
- ❑ Inspiration and Contribution
- ❑ Forget and Rewire
- ❑ Evaluation and Discussion
- ❑ Conclusion

Existing Defenses

- ❑ **Model Hardening**
- ❑ **Detection and Recovery**

Existing Defenses

- **Model Hardening**
  - **Quantization**
    - Require retraining
    - Degrades accuracy



He, Z., and et al. "Defending and harnessing the bit-flip based adversarial weight attack", CVPR 2020

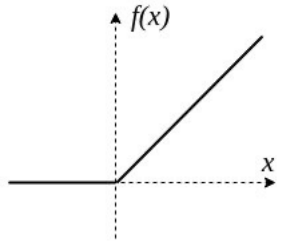
Existing Defenses

- ❑ **Model Hardening**
  - ❑ **Quantization**

    He, Z., and et al. “Defending and harnessing the bit-flip based adversarial weight attack”, CVPR 2020

  - ❑ **Activation optimization**
    - ❑ Like ReLu, Not completely mitigate the BFA

    Jinyu Zhan., and et al. “Improving fault tolerance for reliable dnn using boundary-aware activation”, IEEE TCAD 2021



Existing Defenses

- Model Hardening
  - Quantization

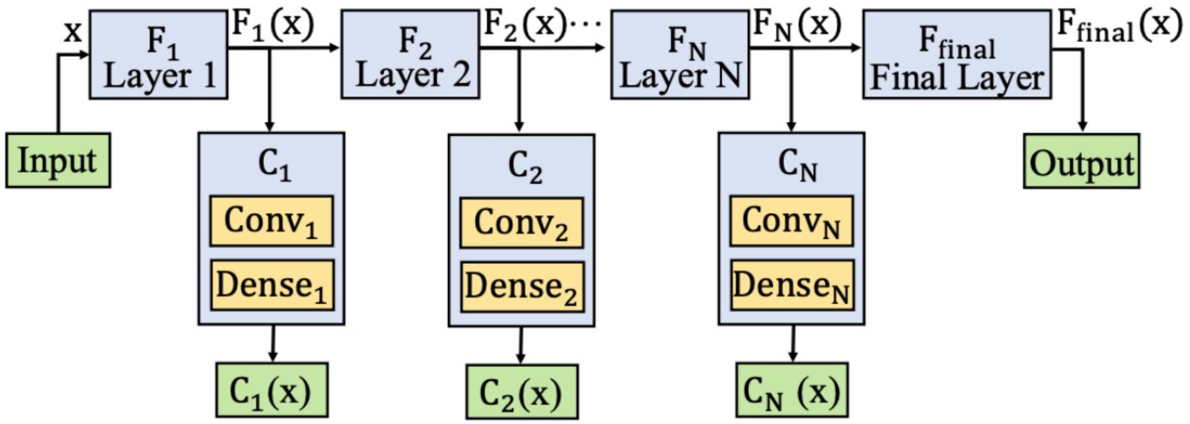
    He, Z., and et al. "Defending and harnessing the bit-flip based adversarial weight attack", CVPR 2020

  - Activation optimization

    Jinyu Zhan., and et al. "Improving fault tolerance for reliable dnn using boundary-aware activation", IEEE TCAD 2021

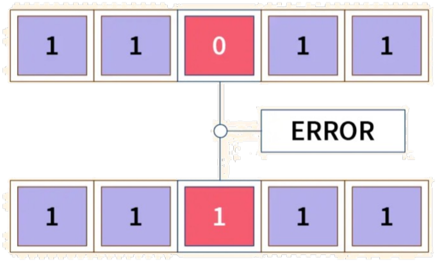
  - Randomization
    - Ageis: Additional classifier layers internally and dynamic exit

    Wang, J., and et al, "Aegis: Mitigating targeted bit-flip attacks against deep neural networks", *USENIX Security, 2023*



Existing Defenses

- ❑ **Detection and Recovery**
  - ❑ **ECC**
    - ❑ Limitation in recovery and detection



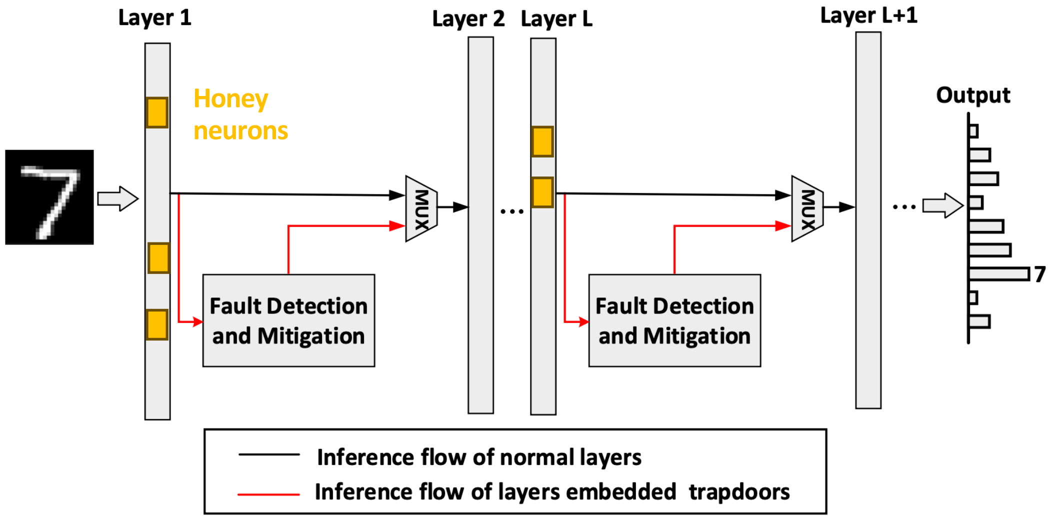
Li, Wei, and et al. "Improving DRAM Reliability Using a High Order Error Correction Code." *IEEE TCAD,* 2024.

Existing Defenses

- ❑ **Detection and Recovery**
  - ❑ **ECC**

    Li, Wei, and et al. "Improving DRAM Reliability Using a High Order Error Correction Code." *IEEE TCAD,* 2024.
  - ❑ **NeuroPots**
    - ❑ Expert attacker can bypass it by setting threshold

    Liu, Q., and et al, "NeuroPots: Realtime Proactive Defense against Bit-Flip Attacks in Neural Networks", *USENIX Security, 2023*.



Inspiration

- Brain Rewiring
  - Neurons that fire together wire together!

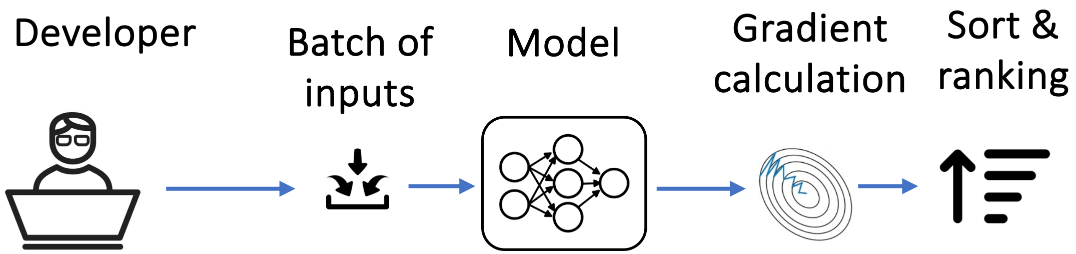
- **Forget** unimportant connections and **Rewire** them to robust the important ones

We call this operation, Forget and Rewire or FaR.

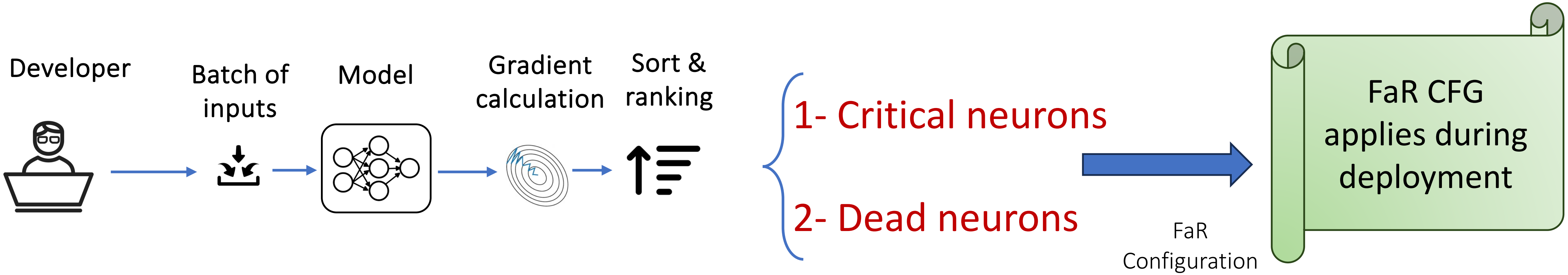
Outline

- Background and Problem Statement
- Threat Model
- Inspiration and Contribution
- Forget and Rewire
- Evaluation and Discussion
- Conclusion

Forget and Rewire Configuration



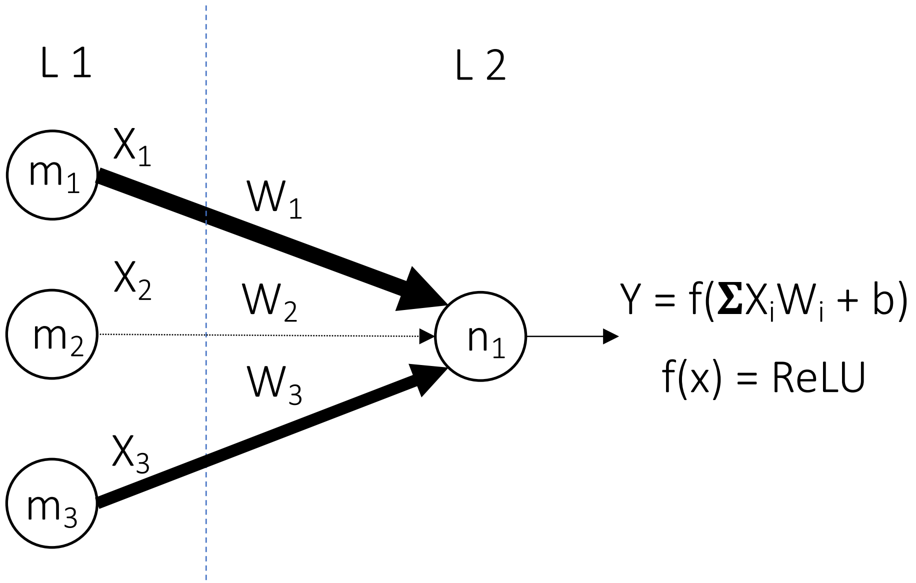
Forget and Rewire Configuration



Match dead neurons with critical ones

Applying FaR CFG

Normal linear layer



$X_1 > X_3 > X_2 = 0$

$Y = f(X_1W_1 + X_2W_2 + X_3W_3 + b)$

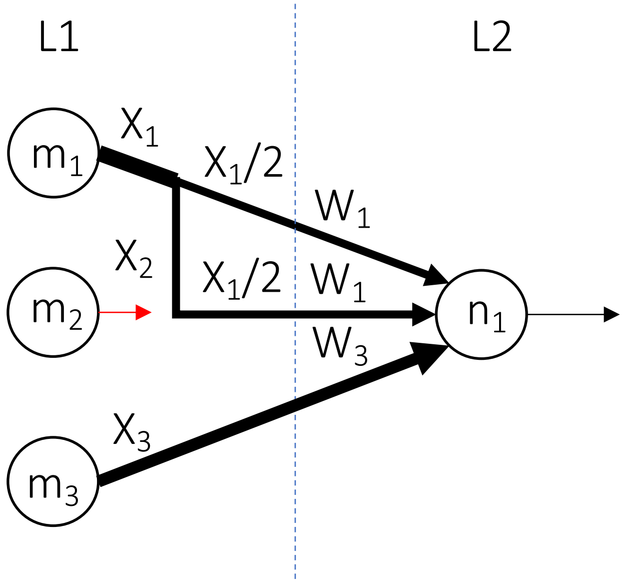
$\rightarrow Y = f(X_1W_1 + X_3W_3 + b)$

Sensitive weight : W_1

Note: Connections' thickness shows the gradient value

Applying FaR CFG

Forget & Rewire



$$Y = f(\boldsymbol{\Sigma} X_i W_i + b)$$

$$f(x) = \text{ReLU}$$

$$X_3 > (X_1/2) = (X_1/2)$$

$$Y = f((X_1/2)W1 + (X_1/2)W1 + X_3W_3 + b)$$

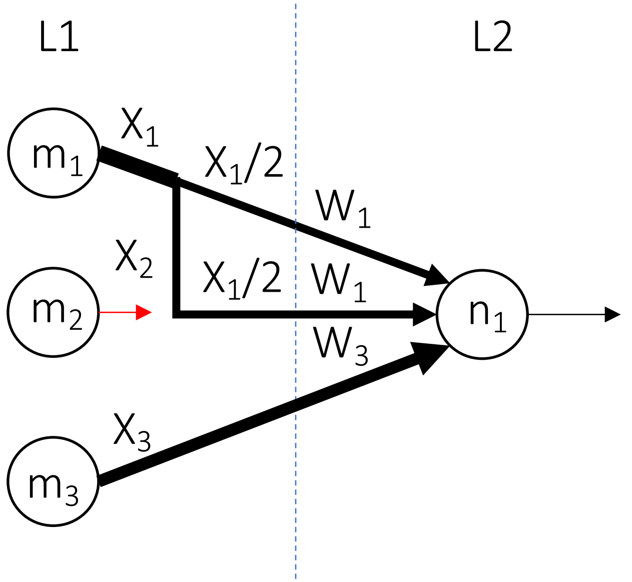
$$\rightarrow Y = f(X_1W_1 + X_3W_3 + b)$$

Sensitive weight : W_3

- ❑ Forget m2's connection
- ❑ Rewire W2 with W1
- ❑ Replace W2 value with W1
- ❑ Redistribute X1 activation to W2 and W1
- ❑ Preserve model's functionality

Applying FaR CFG

Forget & Rewire



$Y = f(\boldsymbol{\Sigma} X_i W_i + b)$

$f(x) = \text{ReLU}$

$X_3 > (X_1/2) = (X_1/2)$

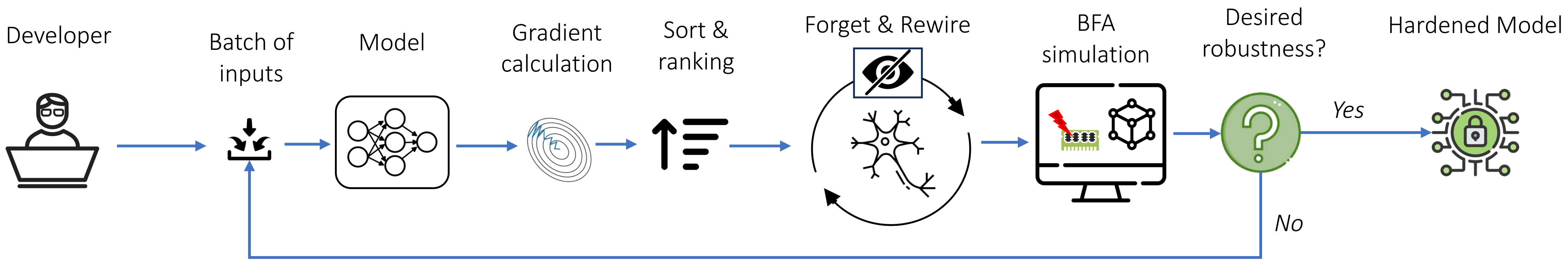
$Y = f((X_1/2)W1 + (X_1/2)W1 + X_3W_3 + b)$

$\rightarrow Y = f(X_1W_1 + X_3W_3 + b)$

Sensitive weight : W_3

- **Concealing Critical Parameters**
  - Reducing the gradient value
    - Redistributing task
- **Increasing robustness**
  - Both W1 and W2 must be attacked
  - Increases the cost of attack

FaR Flow



Developer
Batch of inputs
Model
Gradient calculation
Sort & ranking
Forget & Rewire
BFA simulation
Desired robustness?
Yes
No
Hardened Model

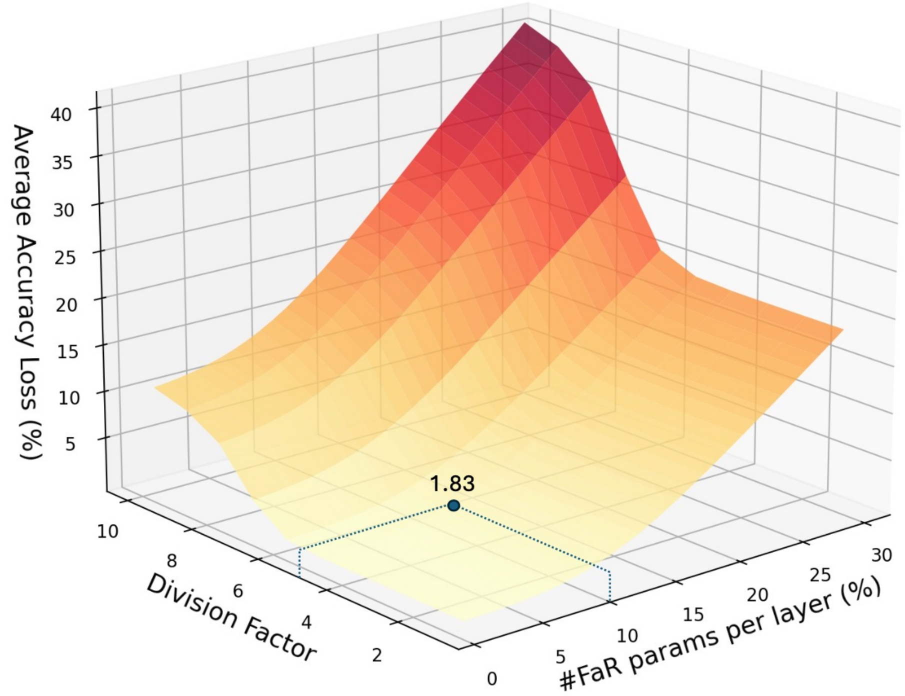
Outline

- ❑ Background and Problem Statement
- ❑ Threat Model
- ❑ Inspiration and Contribution
- ❑ Forget and Rewire
- ❑ Evaluation and Discussion
- ❑ Conclusion

Experimental Setup

- **Datasets used for evaluation**
  - ImageNet
  - MNIST
  - CIFAR-10/100
  - Yelp review
- **Models**
  - Custom ViTs (For MNIST, and CIFAR)
  - google/vit − base − patch16 − 224
  - dbmdz/bert − large − cased− finetuned−conll03−english
- **Evaluation metrics**
  - Accuracy
  - Robustness

Evaluation: Impact on Accuracy

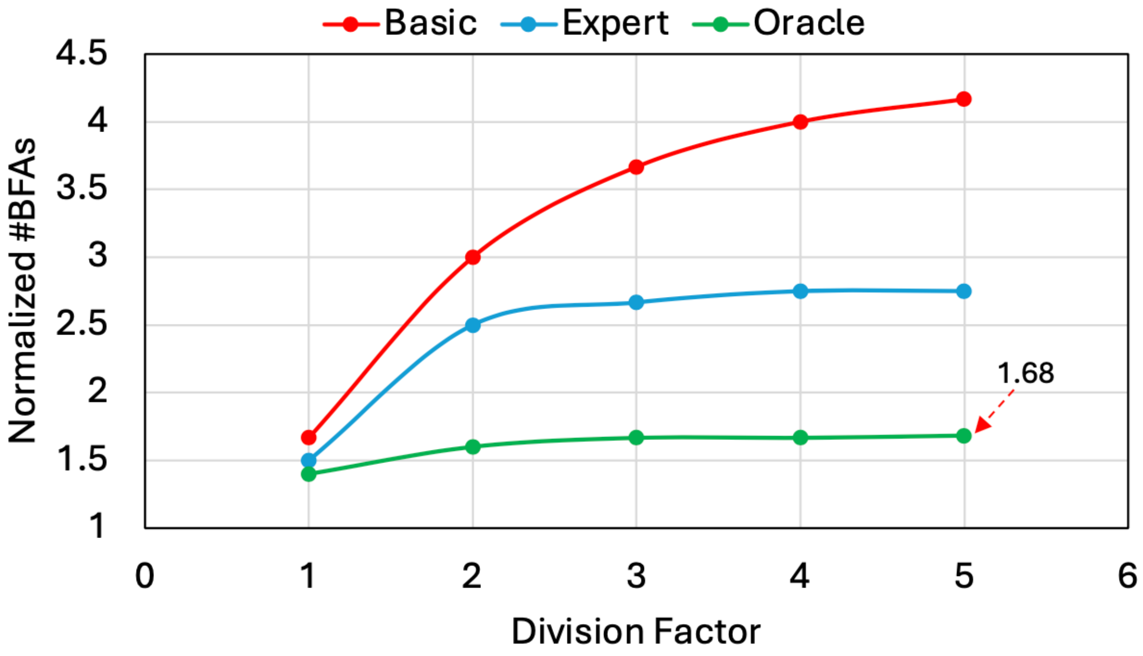


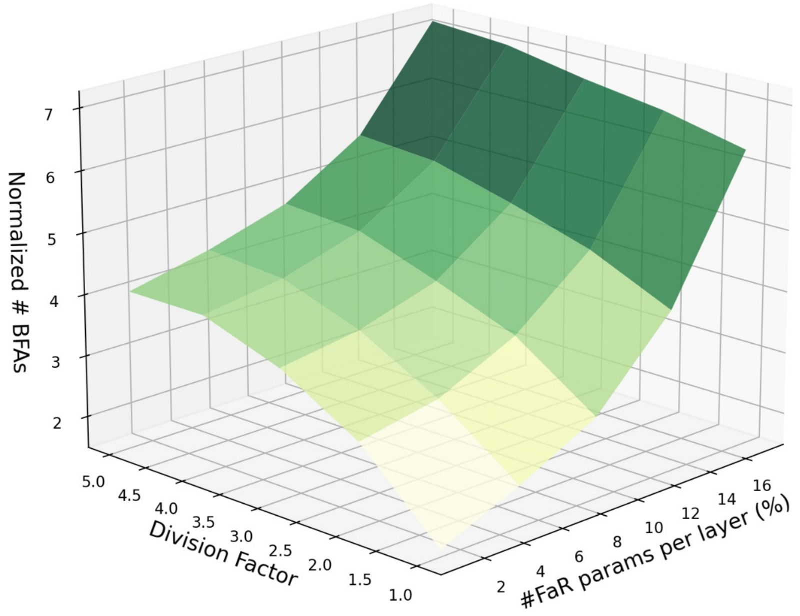
With 10% FaR per layer

Dataset	w/o FaR	w FaR	Ageis	NeuroPots
MNIST	98.3	-0.1	—	—
CIFAR-10	96.1	-1.14	-1.26	-1.0
CIFAR-100	92.8	-1.35	-1.96	—
ImaegNet	88.4	-1.97	—	-1.3
Yelp review	Base	-1.82	—	—

Trade off between Accuracy and Robustness

Evaluation: Robustness





With keeping same level of accuracy loss (2%)

Evaluation

- Storage and Time overhead
- Dropout and Pruning
- Adversarial example input attack

Please read the paper for detailed evaluation and analysis

Outline

- Background and Problem Statement
- Threat Model
- Inspiration and Contribution
- Forget and Rewire
- Evaluation and Discussion
- Conclusion

Conclusion

- Advantages of FaR
  - Redistribute task and **conceal critical neurons**
  - Making **redundant path** for critical information flow
  - Attackers needs **more bit flip** to degrade accuracy
  - **No retraining** is required
  - Reduction in BFA success with **minimal impact on accuracy**
  - **Compatibility** with other defenses

Thank you!