Drop-Connect as a Fault-Tolerance Approach for RRAM-based Deep Neural Network Accelerators

Mingyuan Xiang[1*], Xuhan Xie[1*], Pedro Savarese[2], Xin Yuan[1], Michael Maire[1], Yanjing Li[1]

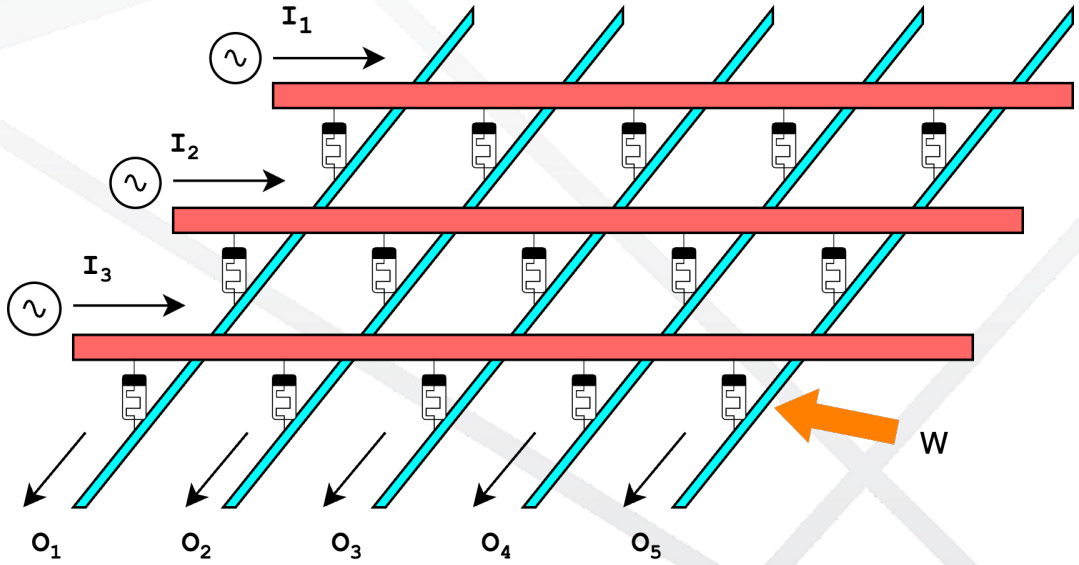
[1]: University of Chicago
[2]: Toyota Technological Institute at Chicago

THE UNIVERSITY OF CHICAGO | DEPARTMENT OF COMPUTER SCIENCE

psd

Background

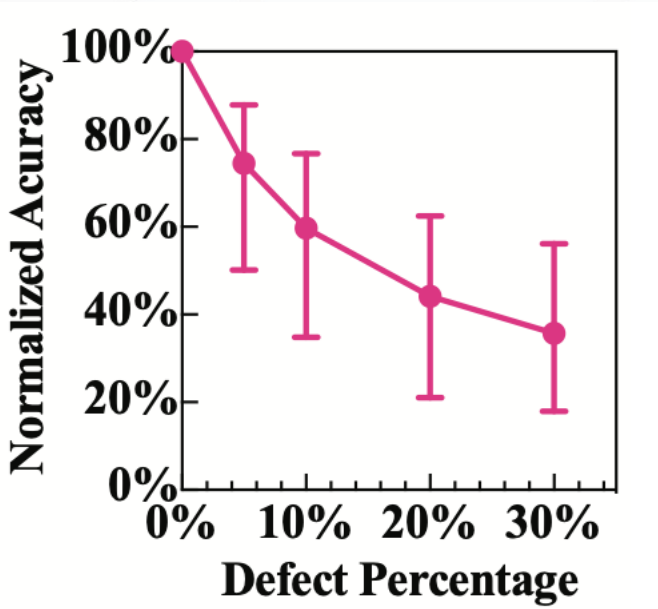
- RRAM accelerators for DNN
- RRAM defect
  - ~9% for SA1 faults for RRAM [Chen TC'15]



RRAM Defect Compensation

- Hardware solutions (ECC, etc.)
  - Additional area cost
- Software solutions
  - Deployment time cost
- Our solution:
  - Robustness of DNN
  - ~~Additional circuits~~ and ~~retraining~~

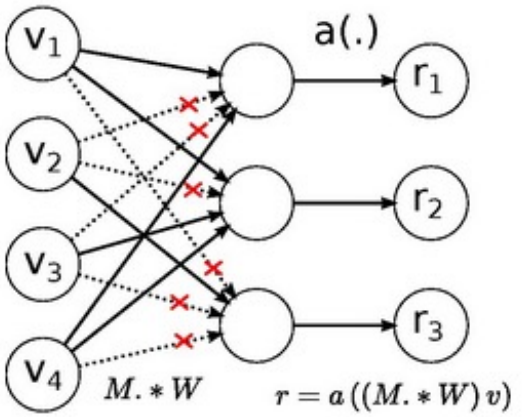
method	Additional Circuits	Retraining and Mapping
Liu et al. [5]	Defect Detection	required
Das et al. [7]	Checksums	None
Li et al. [9]	Refresh and Detection	required
Ours	**None**	**None**



[Liu DAC'17]

A Better Approach for Defect Compensation

- A solution with “no” additional cost
  - Fault-aware training
- How to inject faulty behavior during training?
  - Drop-connect [Wan ICML’13]
- Incorporate Drop-connect with modern DNN



DropConnect Network

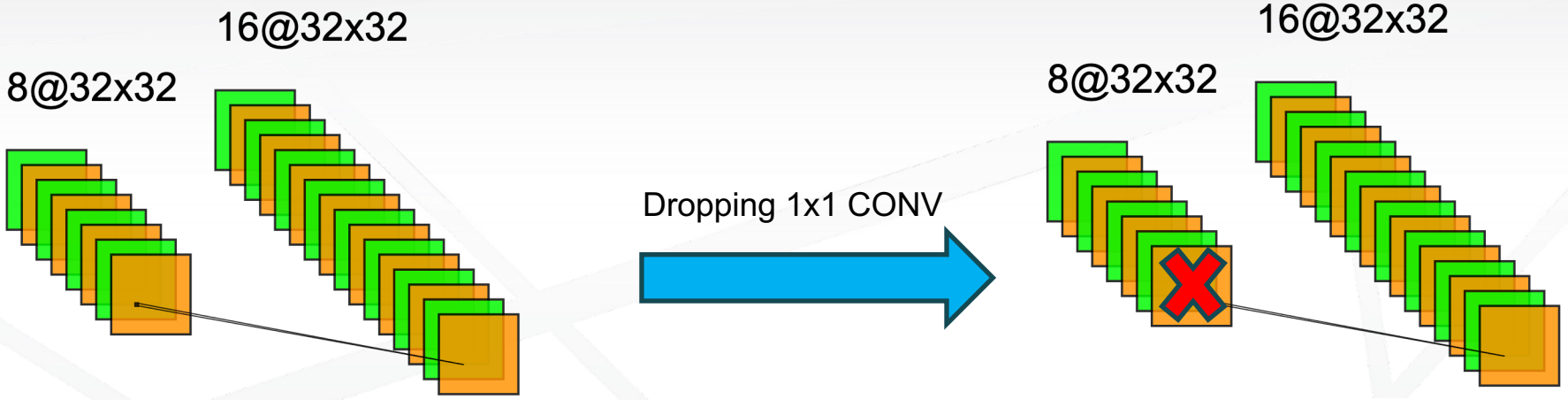
Drop-connect: https://cds.nyu.edu/projects/regularization-neural-networks-using-dropconnect/

Challenges: Incorporating Drop-Connect

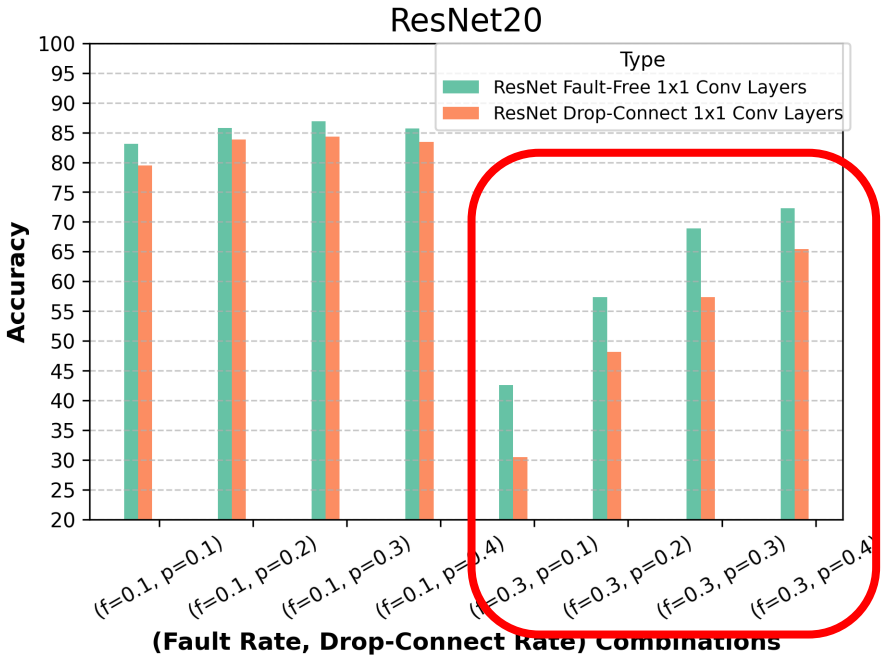
- RRAM defect distribution is changing
- Intuition: fault-aware training (SA1)
  - Having SA1 is the same as dropping weights

- Drop-connect with DNN training
- Special customization:
  - 1x1 convolutions
  - Batch normalization

Modify DNN Model Structure



- Dropping on 1x1 is effectively pruning
- -> Pad 1x1 CONV to 3x3 CONV



Adjust Batch Normalization

- Drop-connect will destroy running statistics
  - Moving mean and variance
- Additional epoch to align batch normalization

Algorithm 1 UpdateVar(MODEL)

```
// p': inference-time scaling factor, i.e., SA1 fault rate
for p' in [0%, 10%, 20%, 30%] do
    MODEL.train()
    for batch in TrainingSet do
        Freeze MODEL.weights
        MODEL.dropConnect(p')
        MODEL.forward(batch)
    end for
end for
```

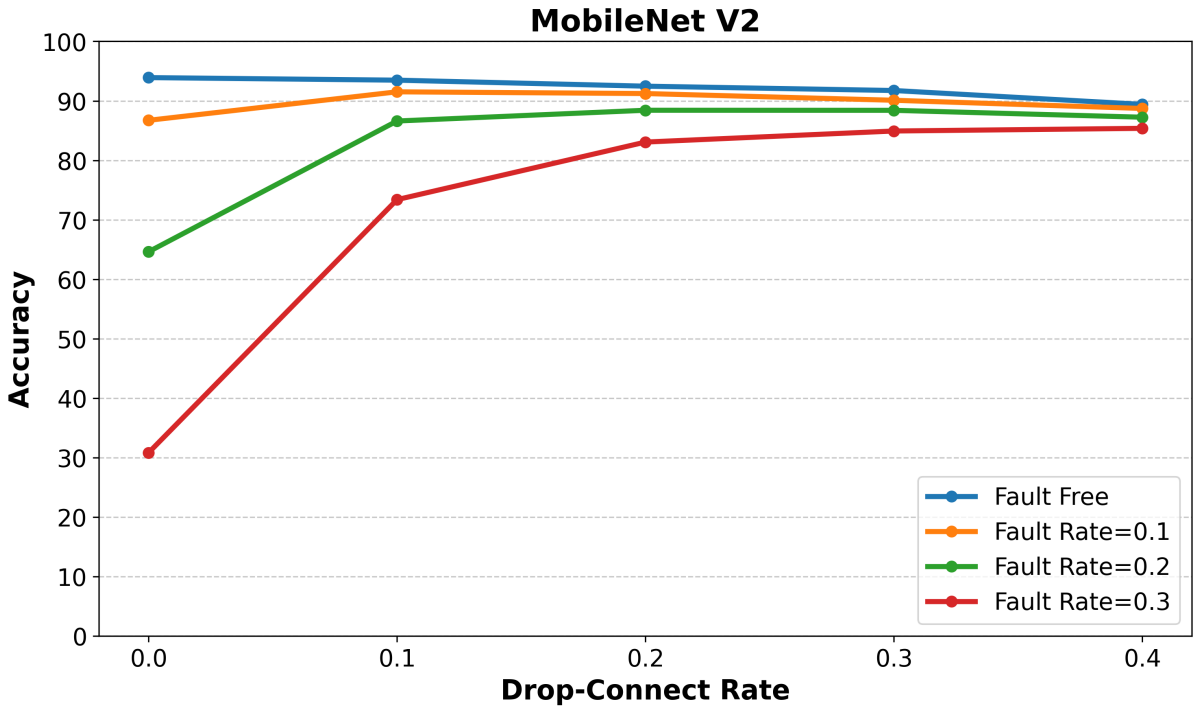
Experiments Set up

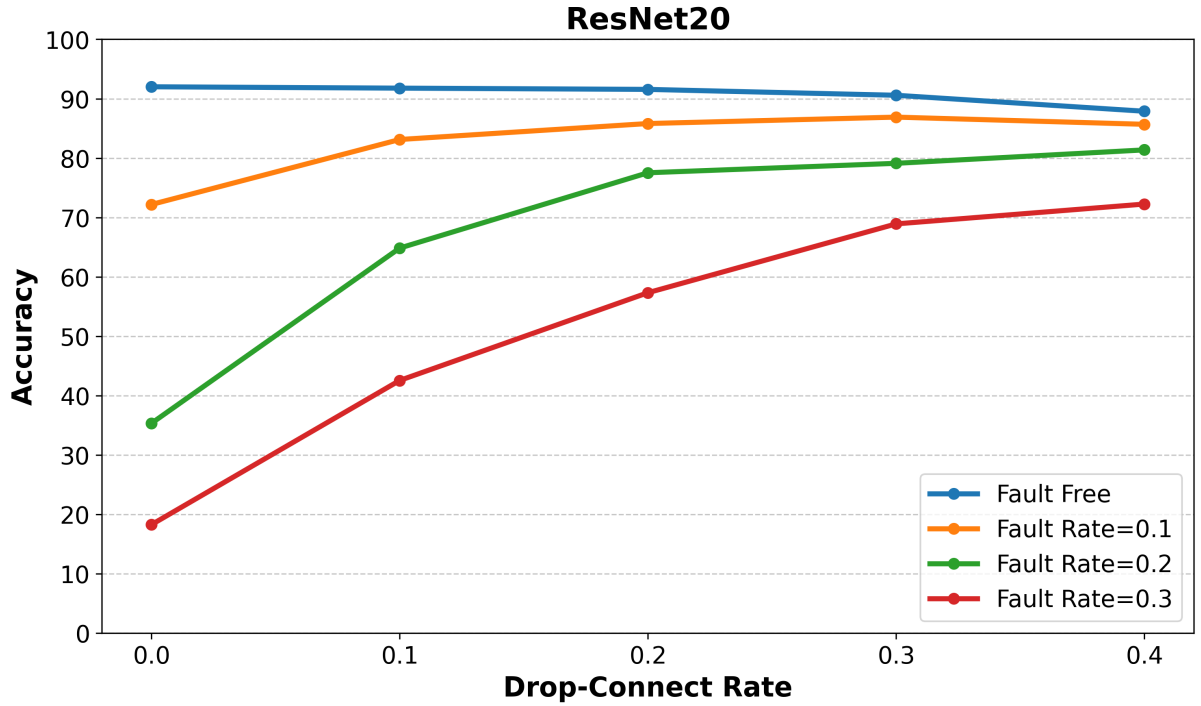
- Implement fault-aware training and simulate RRAM defect behavior using PyTorch
- Models (trained and tested on CIFAR-10[1]):
  - ResNet20 [He CVPR'16]
  - MobileNet V2 [Sandler CVPR'18]
  - VGG13 [Simonyan ICLR'15]
- Collect results for 100 runs (CIFAR-10 testset)
  - Randomly generate faulty RRAM crossbars each time

[1]:https://www.cs.toronto.edu/~kriz/cifar.html

Results

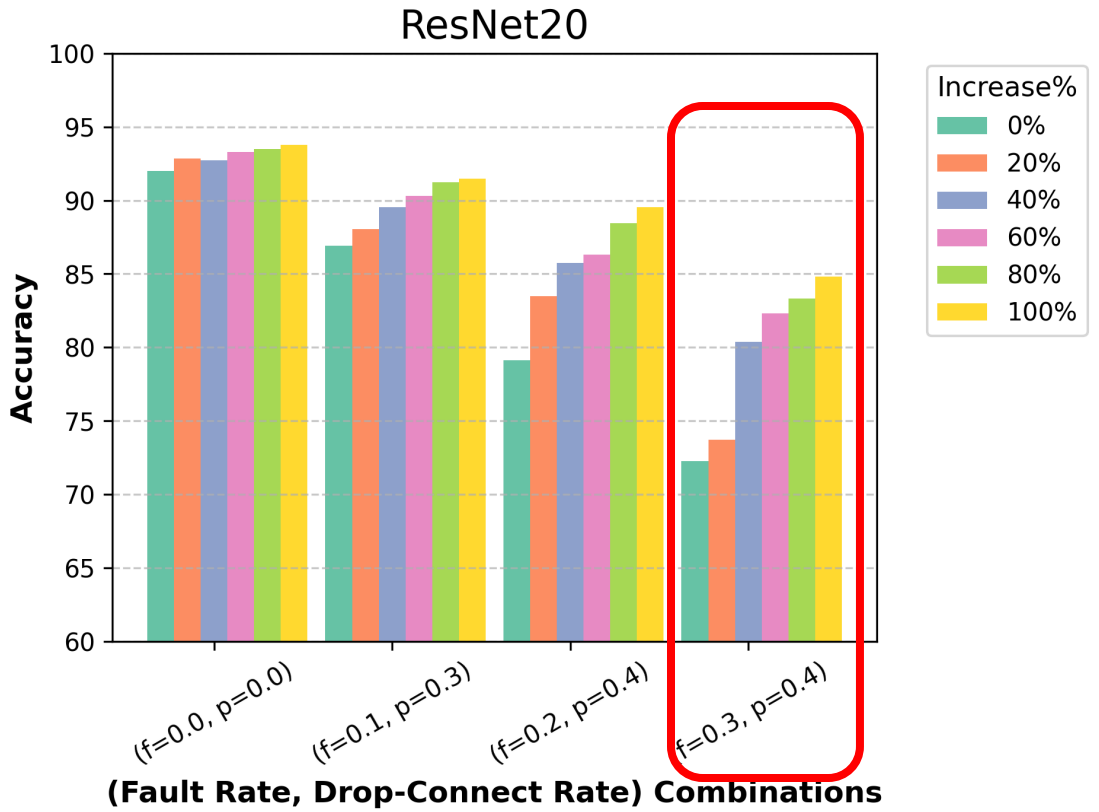
- ResNet20 : Drop-connect alone is not enough





Results

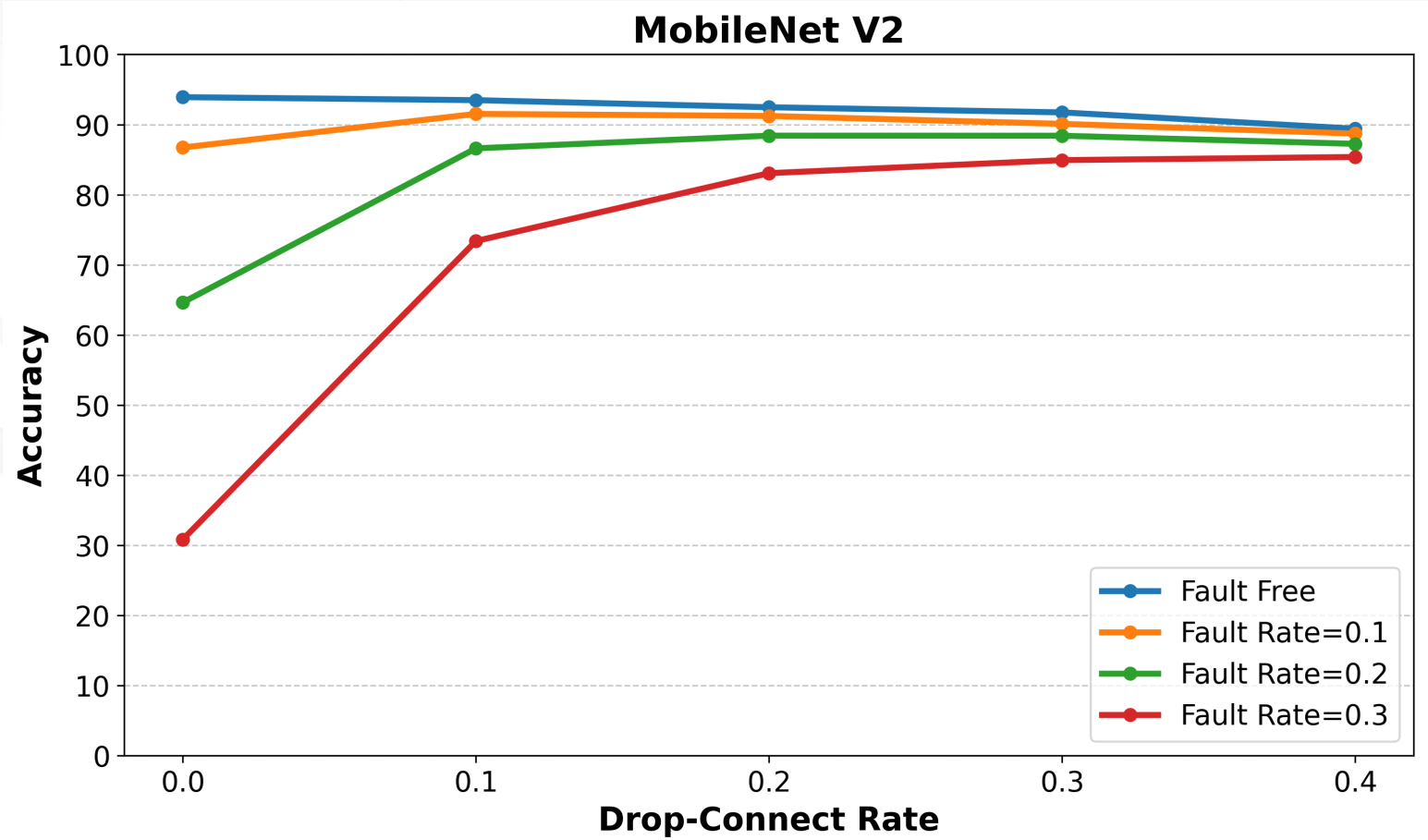
- Widen the models
  - Compensate the information loss because of high drop rates



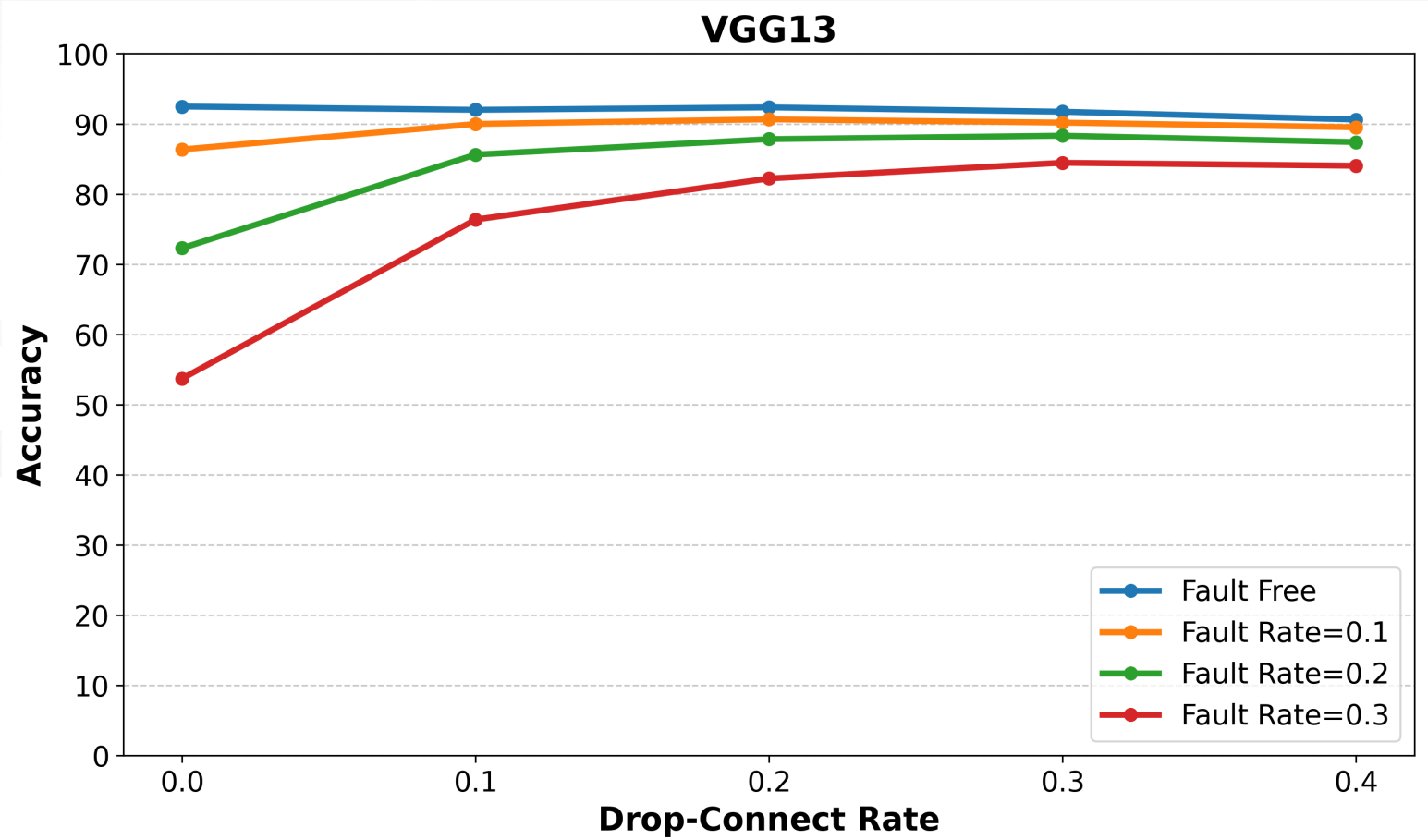
Conclusion

- Drop-connect as a fault tolerance solution
- No hardware/deployment overhead
  - Easy-to-deploy design
- Future work:
  - Why MobileNet has better performance than ResNet?
    - Bottleneck structure
  - Experiments with models in other domains
  - Other fault types

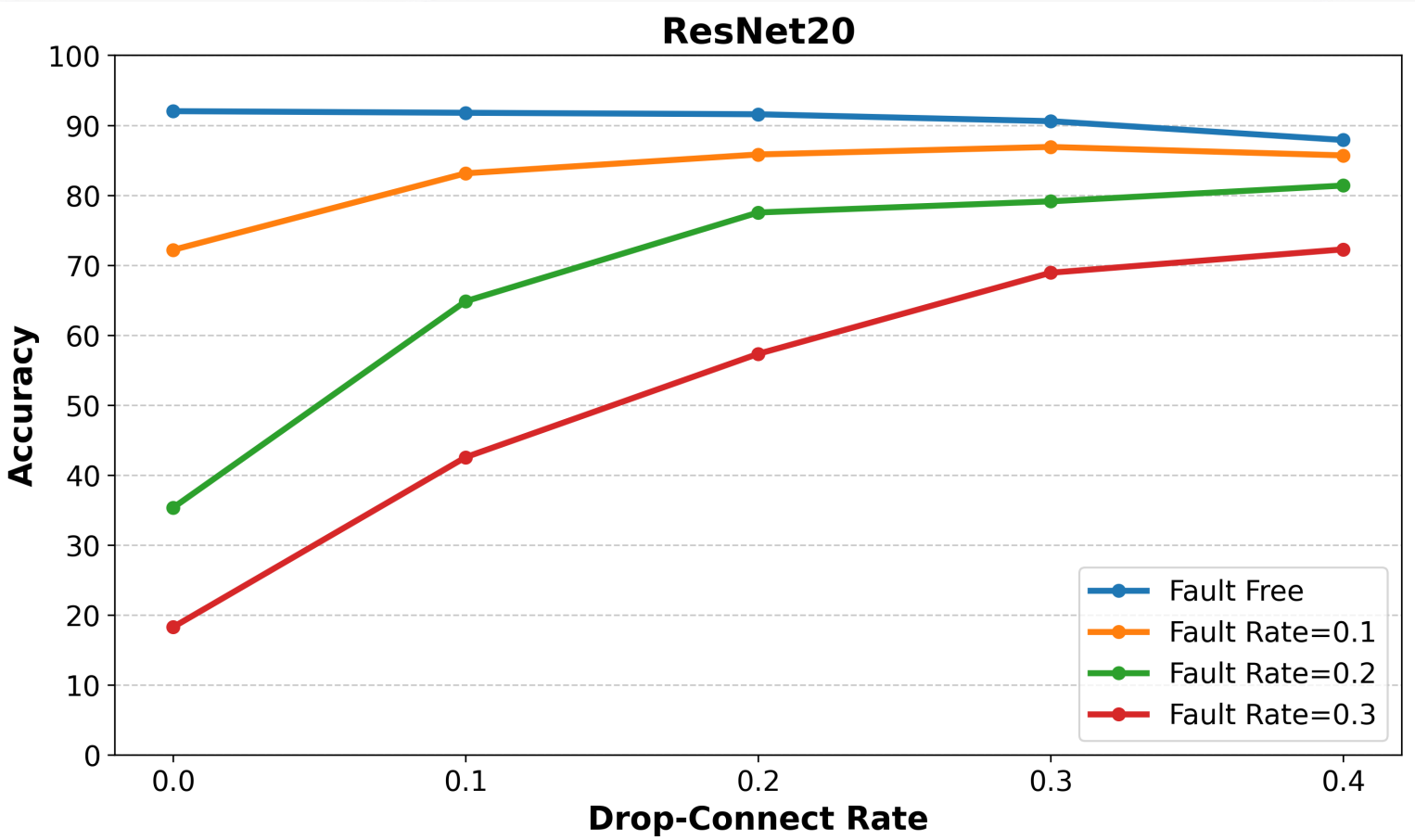
Appendix I (MobileNet V2 Results)



Appendix I (VGG Results)



Appendix I (ResNet20 results)



Appendix II (VGG Results)