Fine Grained Dataflow Tracking with Proximal Gradients

Gabriel Ryan[†], Abhishek Shah[†], Dongdong She[†], Koustubha Bhat[‡], Suman Jana[†]

[†]Columbia University

[‡]Vrije Universiteit Amsterdam

Dataflow Analysis

Summarize dataflows for **all possible inputs**

```
//input x

int x1 = 2 * x;
int x2 = x1 % 4;

return x2;
```

Dataflow Summary:

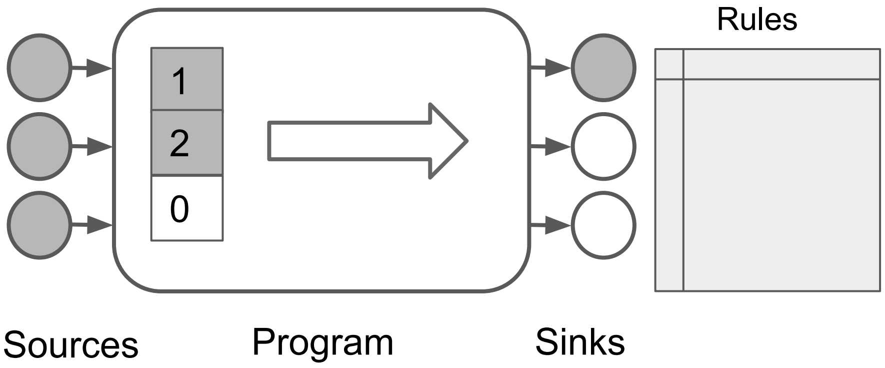
even inputs ⇨ return 0

odd inputs ⇨ return 2

Dynamic Taint Analysis (DTA)

Uses boolean **Taint Labels**

Applies **Propagation Rules** during Execution



Used in many security applications:

Exploit Detection

Greybox Fuzzing

Malware Analysis

Information Leak Identification

Dynamic Taint Analysis (DTA)

Uses boolean **Taint Labels**

Applies **Propagation Rules** during Execution

```
//input x

int x1 = 2 * x;
int x2 = x1 % 4;

return x2;
```

Dynamic Taint Analysis (DTA)

Uses boolean **Taint Labels**

Applies **Propagation Rules** during Execution

```
//input x

int x1 = 2 * x;
int x2 = x1 % 4;

return x2;
```

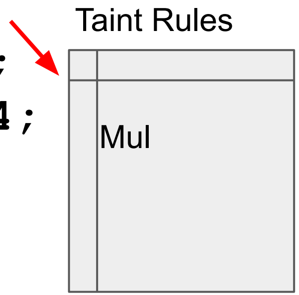
Label 1

Dynamic Taint Analysis (DTA)

Uses boolean **Taint Labels**

Applies **Propagation Rules** during Execution

```
//input x

int x1 = 2 * x;
int x2 = x1 % 4;

return x2;
```



Dynamic Taint Analysis (DTA)

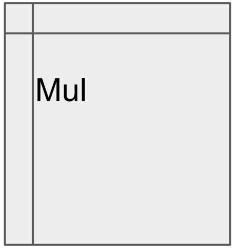
Uses boolean **Taint Labels**

Applies **Propagation Rules** during Execution

Execute

```
//input x

int x1 = 2 * x;
int x2 = x1 % 4;

return x2;
```

Taint Rules



Mul

Dataflow Summary:

input x ⇒ can change x2

Dynamic Taint Analysis (DTA)

Boolean Taint labels **cannot represent** all possible behavior

Causes taint **propagation errors**

Dynamic Taint Analysis (DTA)

Boolean Taint labels **cannot represent** all possible behavior

Causes taint **propagation errors**

```
//input x, k

int x1 = x - k;
int x2 = x - x1;

return x2;
```

Dynamic Taint Analysis (DTA)

Boolean Taint labels **cannot represent** all possible behavior

Causes taint **propagation errors**

```
//input x, k

int x1 = x - k;
int x2 = x - x1;  ←  x - x + k

return x2;
```

Dynamic Taint Analysis (DTA)

Boolean Taint labels **cannot represent** all possible behavior

Causes taint **propagation errors**

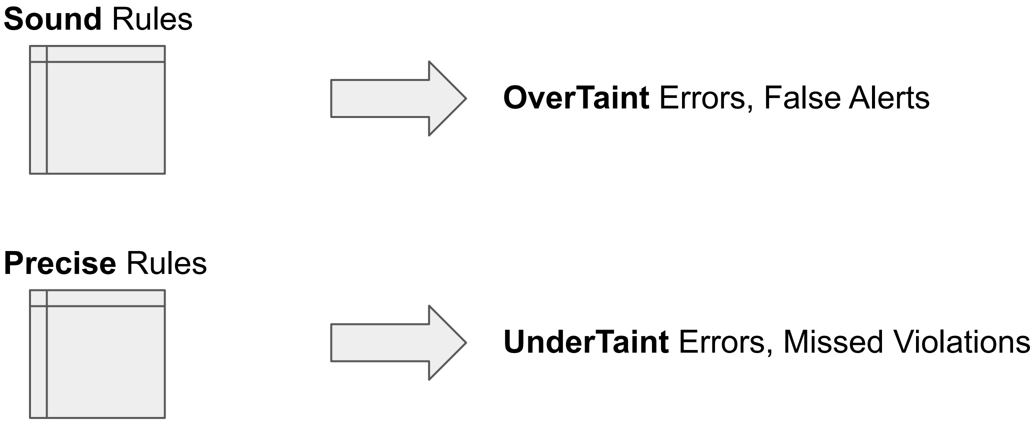
```
//input x, k

int x1 = x - k;
int x2 = x - x1;

return x2;
```

← taint error!

Dynamic Taint Analysis (DTA)

Boolean Taint labels **cannot represent** all possible behavior

Causes taint **propagation errors**



Sound Rules → **OverTaint** Errors, False Alerts

Precise Rules → **UnderTaint** Errors, Missed Violations

Chua, Zheng Leong, et al. "One Engine To Serve'em All: Inferring Taint Rules Without Architectural Semantics." *NDSS*. 2019.
Dalton, Michael, Hari Kannan, and Christos Kozyrakis. "Tainting is not pointless." *ACM SIGOPS Operating Systems Review* 44.2 (2010): 88-92.
Slowinska, Asia, and Herbert Bos. "Pointer tainting still pointless: (but we all see the point of tainting)." *ACM SIGOPS Operating Systems Review* 44.3 (2010): 88-92.
Yadegari, Babak, and Saumya Debray. "Bit-level taint analysis." *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*. IEEE, 2014.

Richer form of Dataflow?

Richer form of Dataflow?

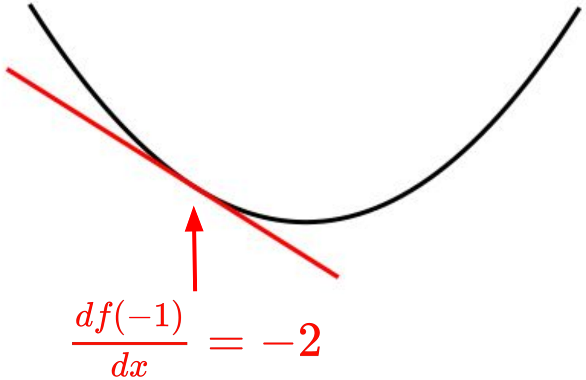
Gradient is a popular **Dataflow** measure in machine learning:

Richer form of Dataflow?

Gradient is a popular **Dataflow** measure in machine learning:

$$\frac{df(x)}{dx} = \frac{\text{change in output}}{\text{change in input}}$$

$$f(x) = x * x$$

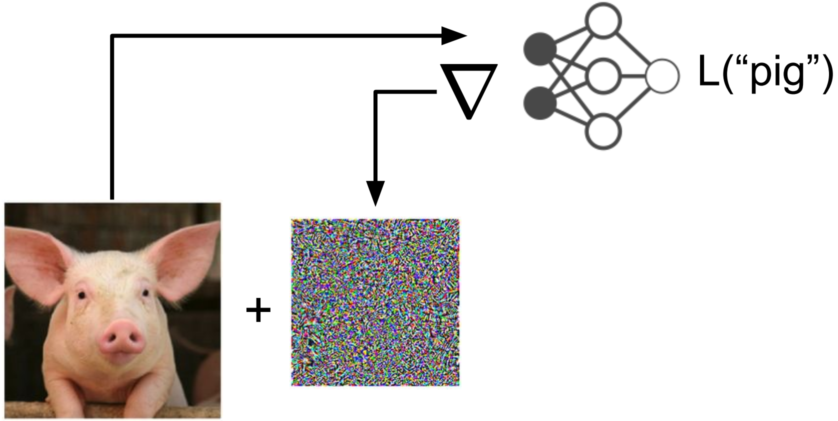


$$\frac{df(-1)}{dx} = -2$$

Richer form of Dataflow?

Gradient used for similar applications to Dataflow:

- Guiding adversarial testing



Vulnerability search:

```
int y = x1 + x2%2;

if (y > THRESHOLD) {
    // vulnerability
}
```

Gradient as Dataflow Measure

Gradient can be composed with **chain rule**

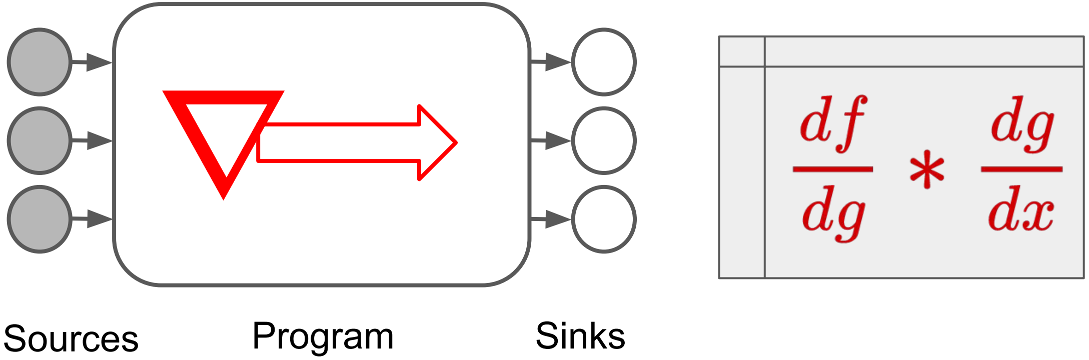
$$\frac{df(g(x))}{dx} = \frac{df}{dg} * \frac{dg}{dx}$$

Gradient as Dataflow Measure

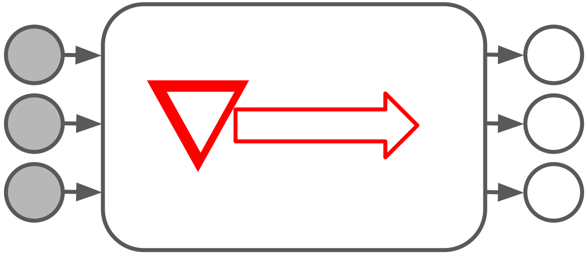
Gradient can be composed with **chain rule**

$$\frac{df(g(x))}{dx} = \frac{df}{dg} * \frac{dg}{dx}$$

Key Idea: Propagate Gradient directly over Program

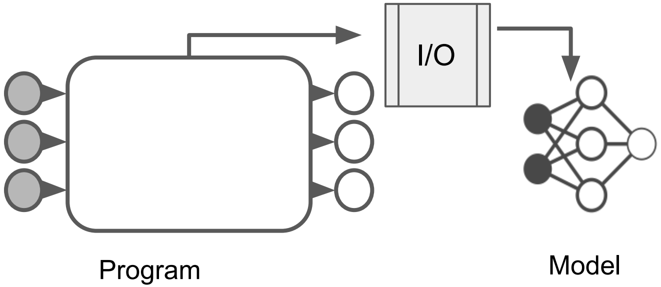


Gradient as Dataflow Measure

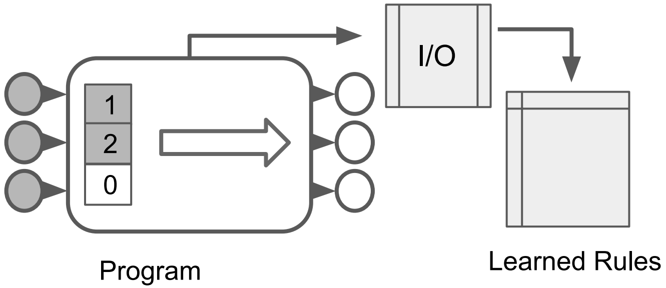


$$\frac{df}{dg} * \frac{dg}{dx}$$

Neutaint: Measures Gradient of Neural Network



TaintInduce: Learns taint rules from I/O samples



She, Dongdong, et al. "Neutaint: Efficient dynamic taint analysis with neural networks." *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020.
Chua, Zheng Leong, et al. "One Engine To Serve'em All: Inferring Taint Rules Without Architectural Semantics." *NDSS*. 2019.

Gradient Reduces Errors

Propagates dataflow correctly

```
//input x, k

int x1 = x - k;
int x2 = x - x1;  ←——  x - x + k

return x2;
```

Gradient Reduces Errors

Propagates dataflow correctly

```
//input x, k

int x1 = x - k;
int x2 = x - x1;

return x2;
```

$$\frac{dx1}{dx} = 1$$

$$\frac{dx2}{dx} = 1 - \frac{dx1}{dx} = 0$$

Gradient Reduces Errors

Propagates dataflow correctly

```
//input x, k

int x1 = x - k;
int x2 = x - x1;

return x2;
```

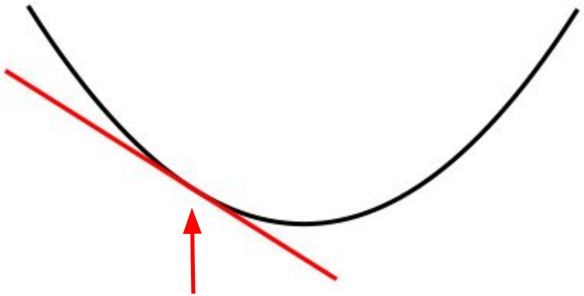
$$\frac{dx1}{dx} = 1$$

$$\frac{dx2}{dx} = 1 - \frac{dx1}{dx} = 0$$

← gradient to `x` is 0!

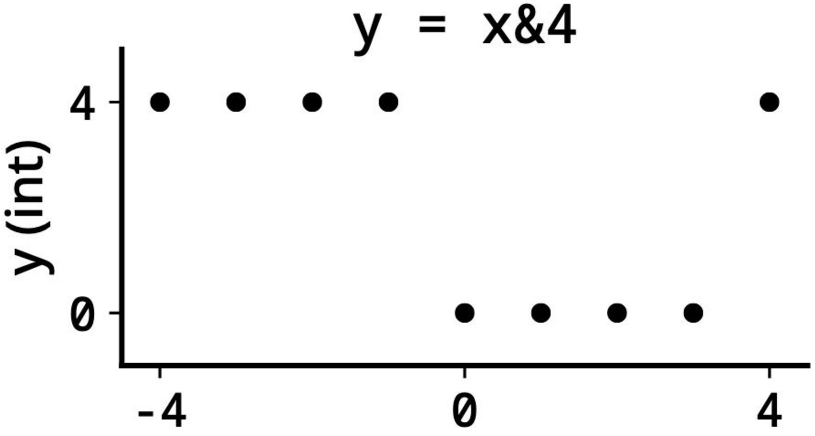
Problem: Programs are not differentiable!

$$f(x) = x * x$$

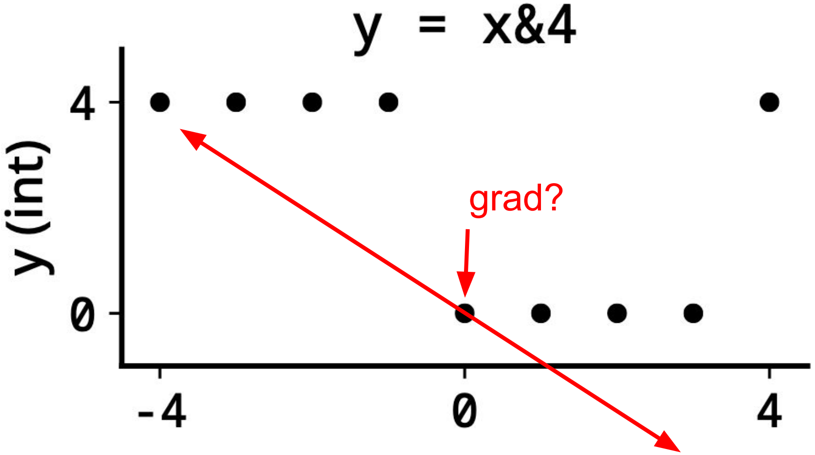


$$\frac{df(-1)}{dx} = -2$$

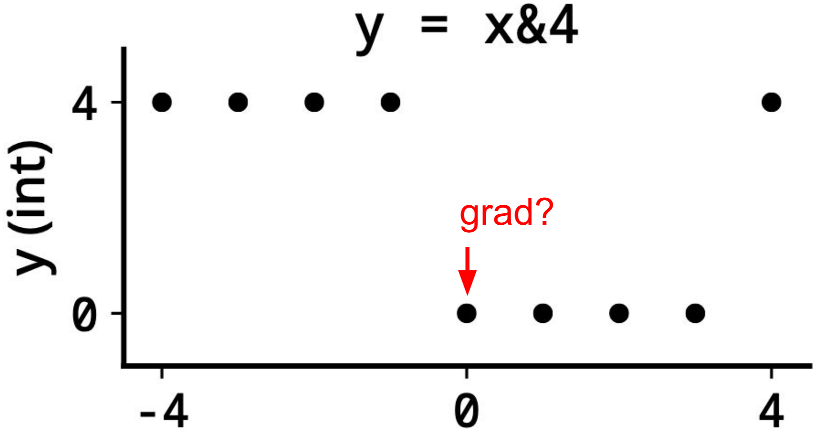
Problem: Programs are not differentiable!



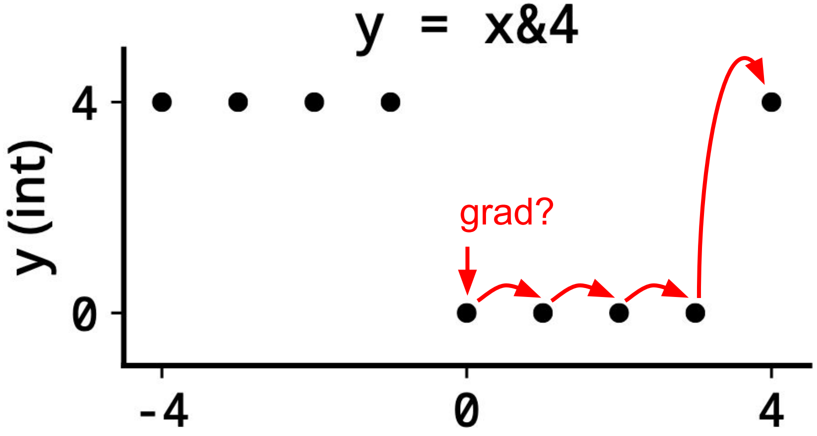
Problem: Programs are not differentiable!



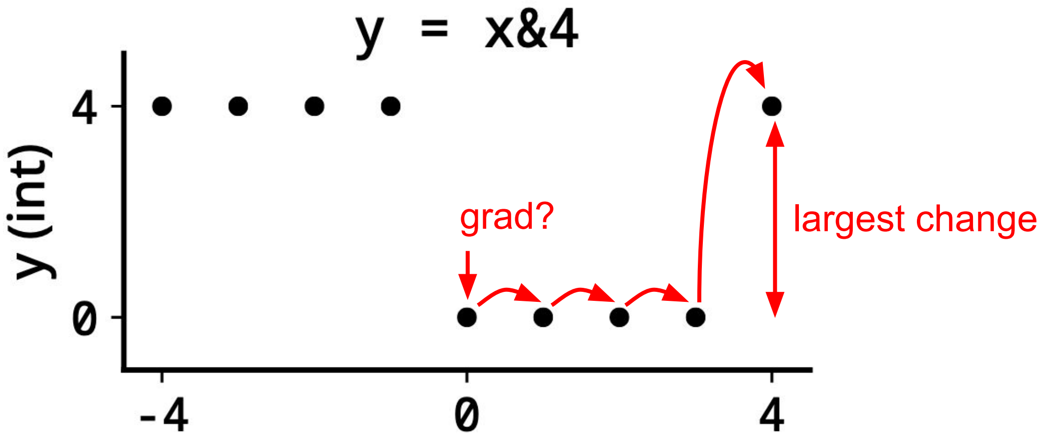
Proximal Gradient



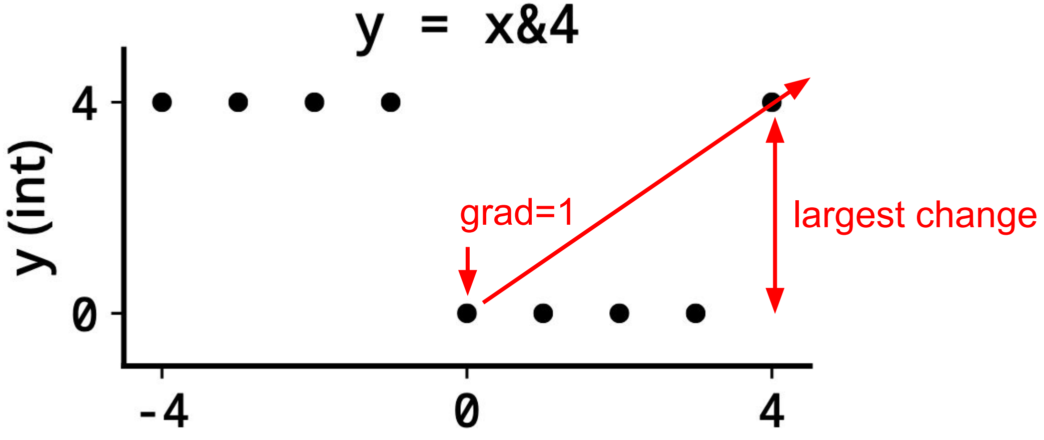
Proximal Gradient



Proximal Gradient



Proximal Gradient

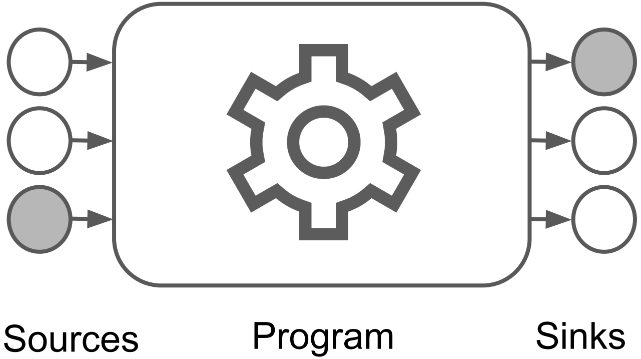


Proximal Gradient Analysis (PGA)

Implement as **LLVM Sanitizer Pass (`grsan`)**

Evaluate **Dataflow Accuracy**:

Ground Truth Dataflow:
For each input byte:
        flip each bit
        set to 0 and 255
        record changed sink variables



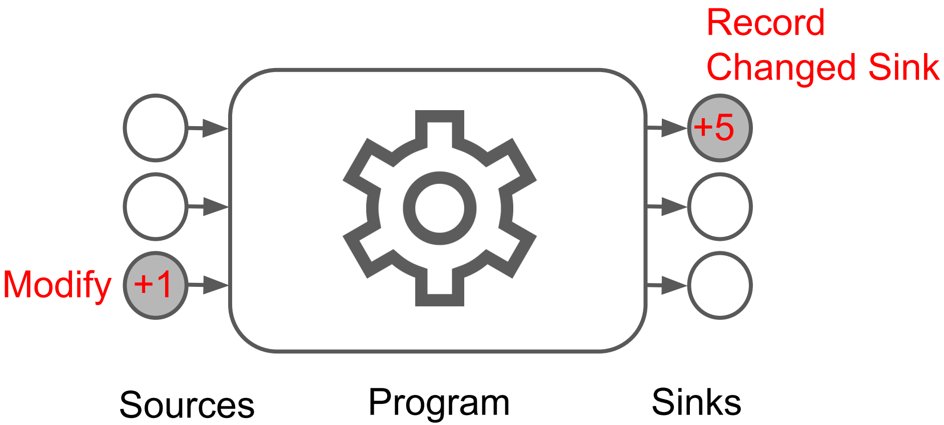
Proximal Gradient Analysis (PGA)

Implement as **LLVM Sanitizer Pass (`grsan`)**

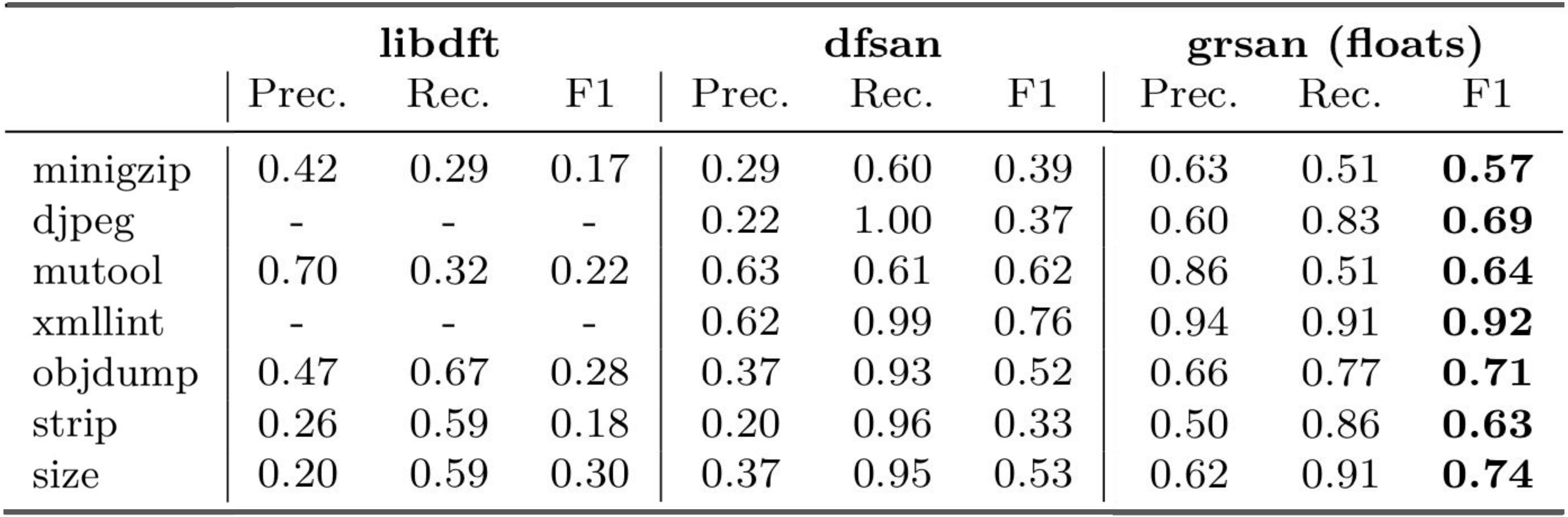
Evaluate **Dataflow Accuracy**:

Ground Truth Dataflow:

```
For each input byte:
        flip each bit
        set to 0 and 255
        record changed sink variables
```



Proximal Gradient Analysis (PGA)



	libdft			**dfsan**			**grsan (floats)**		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
minigzip	0.42	0.29	0.17	0.29	0.60	0.39	0.63	0.51	**0.57**
djpeg	-	-	-	0.22	1.00	0.37	0.60	0.83	**0.69**
mutool	0.70	0.32	0.22	0.63	0.61	0.62	0.86	0.51	**0.64**
xmllint	-	-	-	0.62	0.99	0.76	0.94	0.91	**0.92**
objdump	0.47	0.67	0.28	0.37	0.93	0.52	0.66	0.77	**0.71**
strip	0.26	0.59	0.18	0.20	0.96	0.33	0.50	0.86	**0.63**
size	0.20	0.59	0.30	0.37	0.95	0.53	0.62	0.91	**0.74**

F1 accuracy improvement: Up to **33% over SOTA** (20% better on average)

Proximal Gradient Analysis

Introduce **proximal gradients (PGA)** as new formulation of dataflow problem

Show PGA **improves dataflow accuracy**, and gradients are useful in dataflow applications.

To learn more about PGA please see our paper:

Paper: https://www.usenix.org/system/files/sec21fall-ryan.pdf

Code: https://github.com/gryan11/PGA