Australian National University

Microsoft Research

Uncertain<*T*>

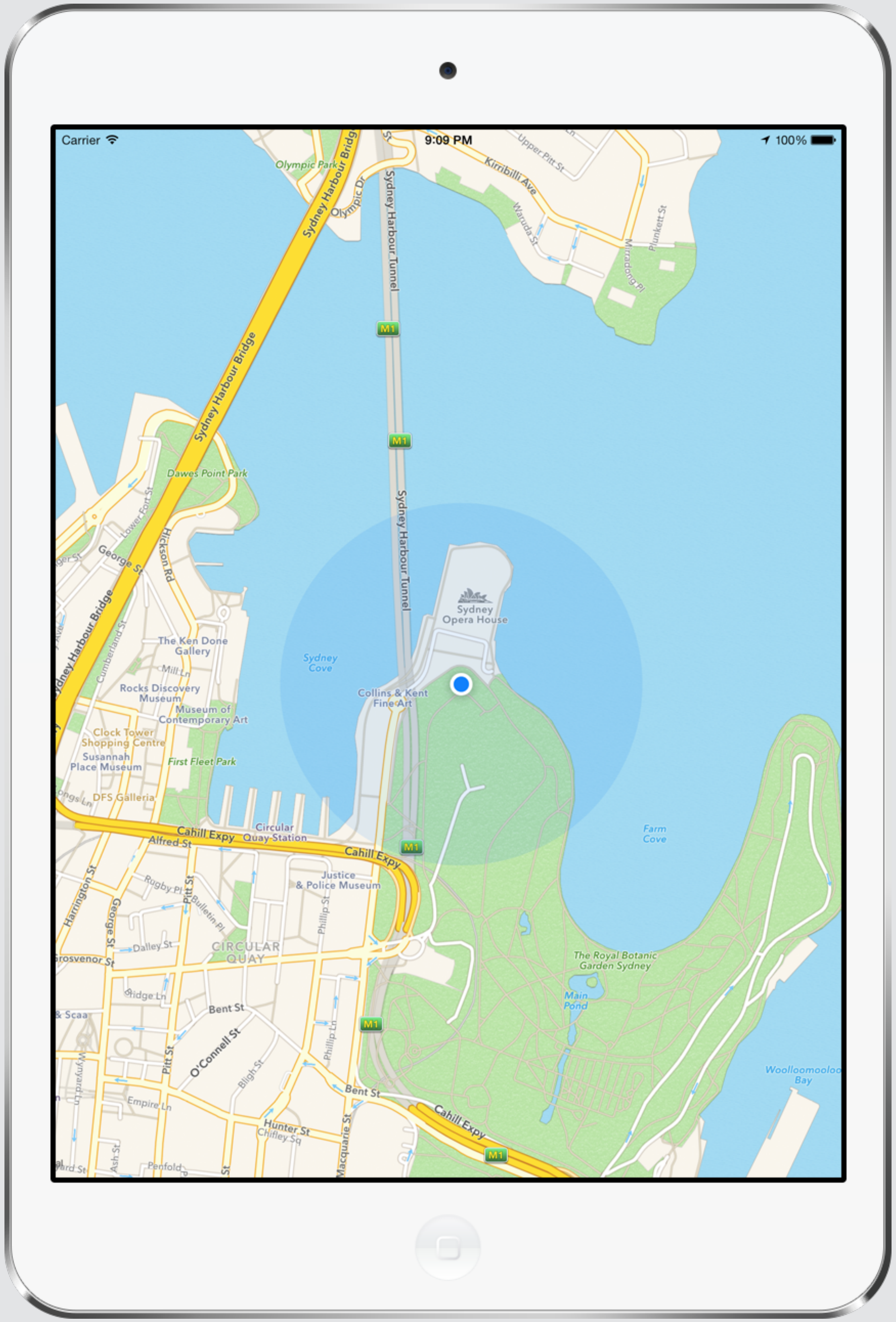
A First-Order Type for Uncertain Data

James Bornholt Australian National University

Todd Mytkowicz Microsoft Research

Kathryn S. McKinley Microsoft Research

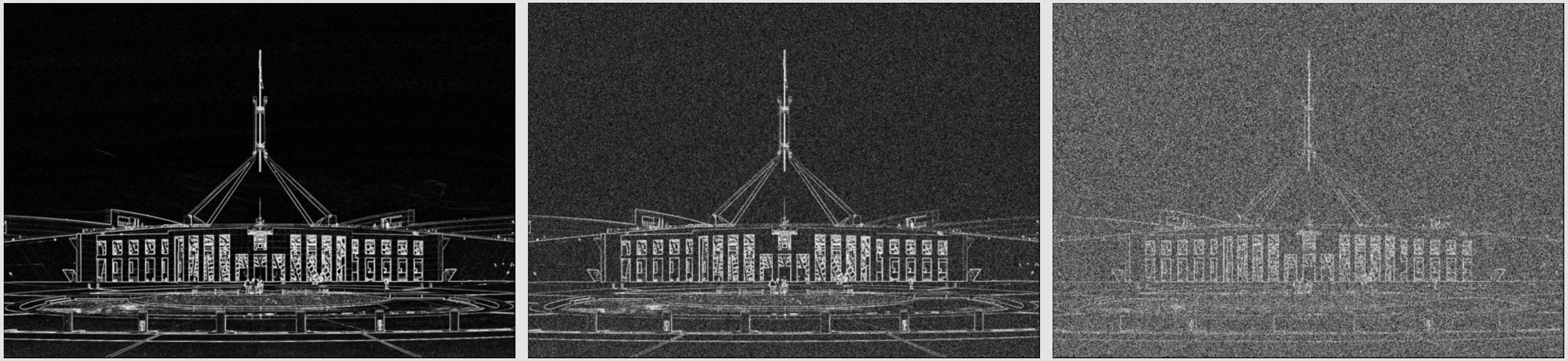
Sensors



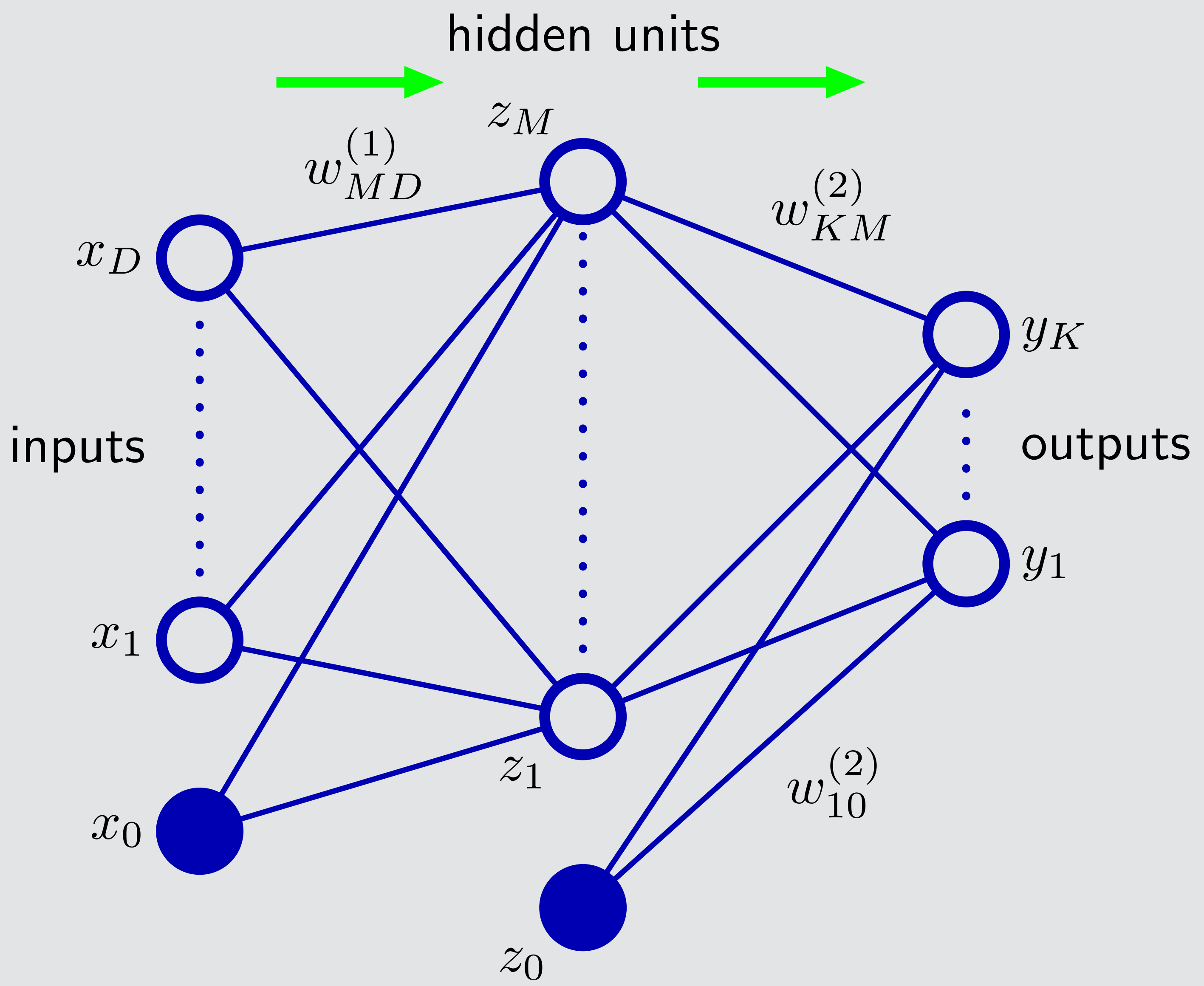
Carrier
9:09 PM
100%
Sydney Harbour Bridge
Olympic Park
Olympic Dr
Sydney Harbour Tunnel
Kirribilli Ave
M1
Dawes Point Park
George St
Hickson Rd
Sydney Opera House
Sydney Cove
Collins & Kent Fine Art
The Ken Done Gallery
Rocks Discovery Museum
Museum of Contemporary Art
Clock Tower Shopping Centre
Susannah Place Museum
First Fleet Park
DFS Galleria
Circular Quay Station
Cahill Expy
Alfred St
Justice & Police Museum
Farm Cove
CIRCULAR QUAY
Dalley St
Harrington St
Pitt St
Phillip St
Bent St
O'Connell St
Empire Ln
Hunter St
Chifley Sq
Macquarie St
The Royal Botanic Garden Sydney
Main Pond
Woolloomooloo Bay

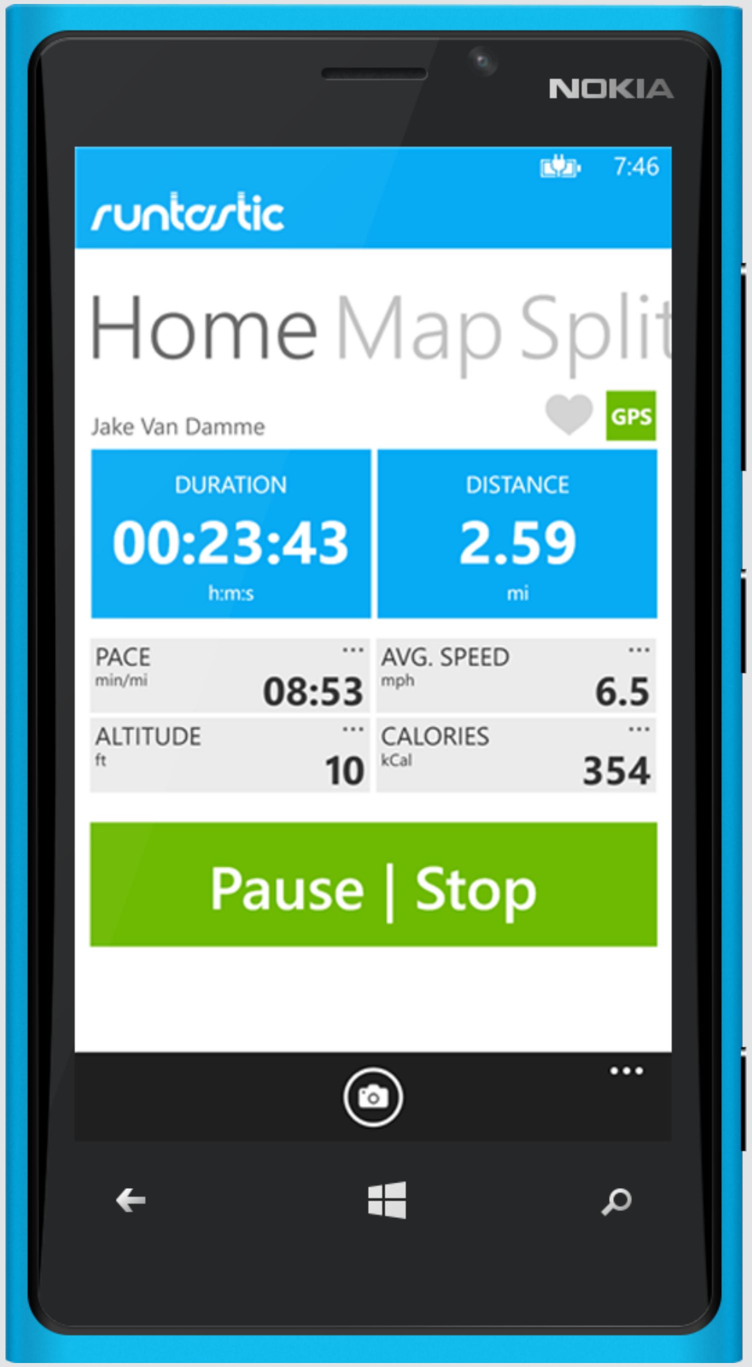
Approximate computing

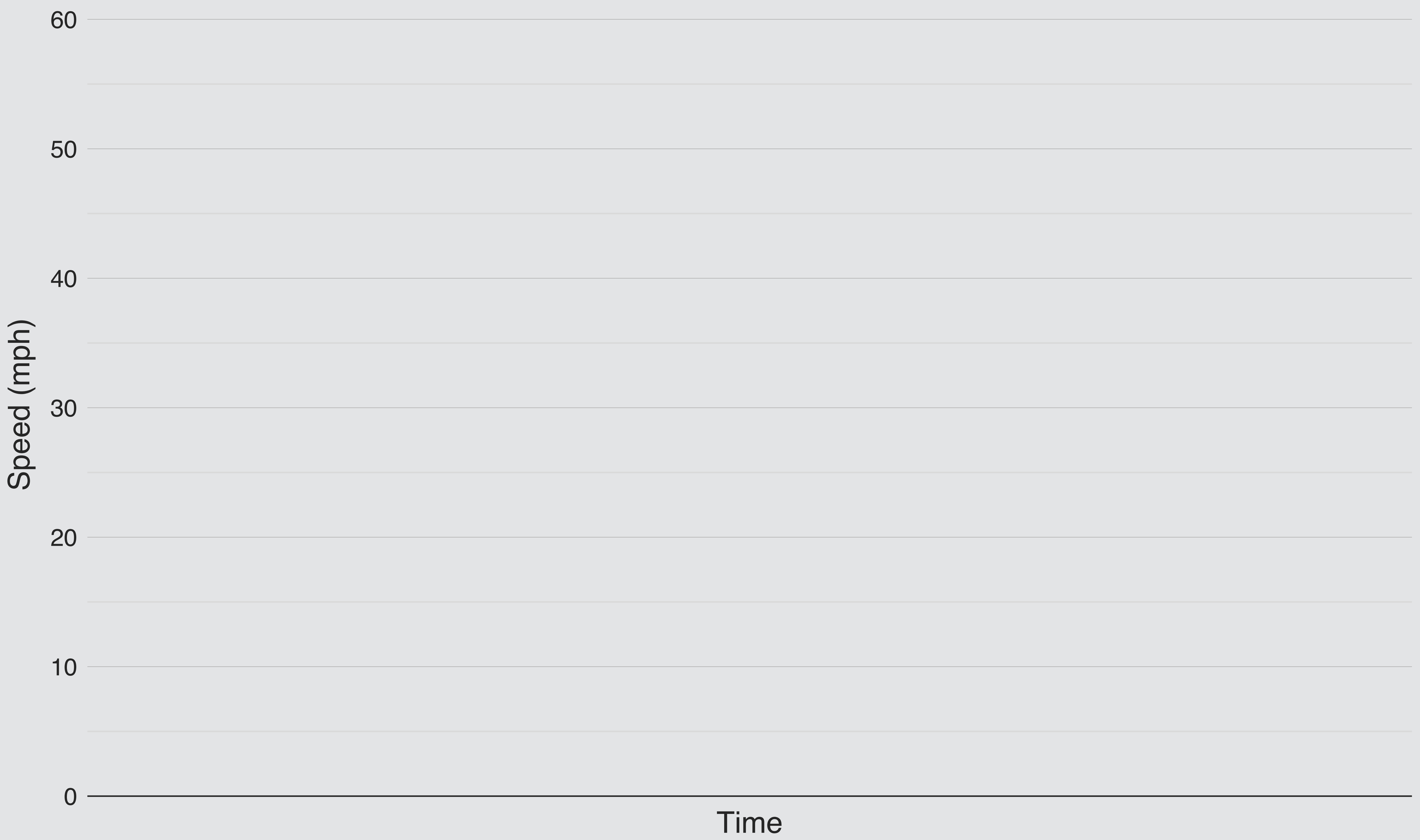
approximate edge detection



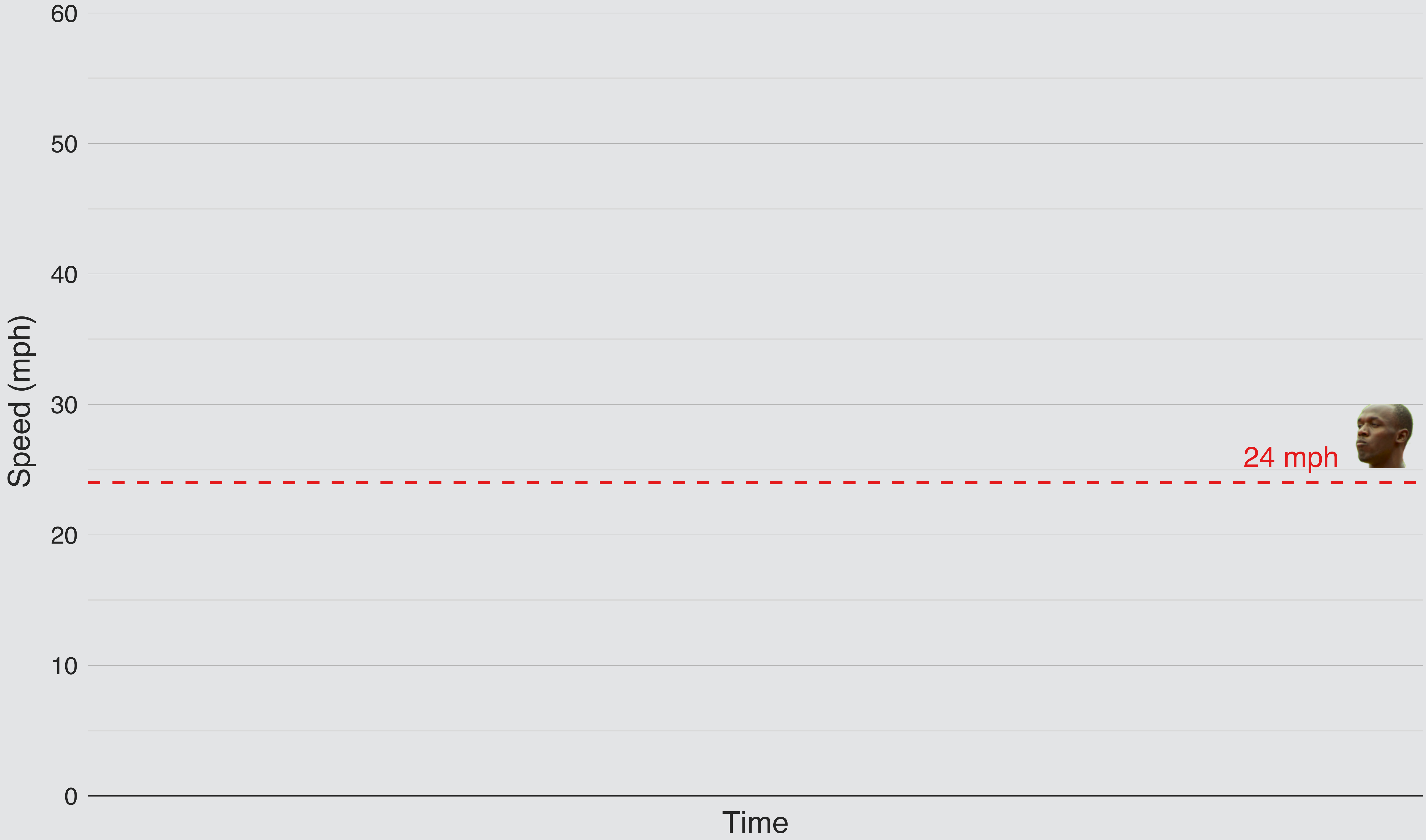
Machine learning



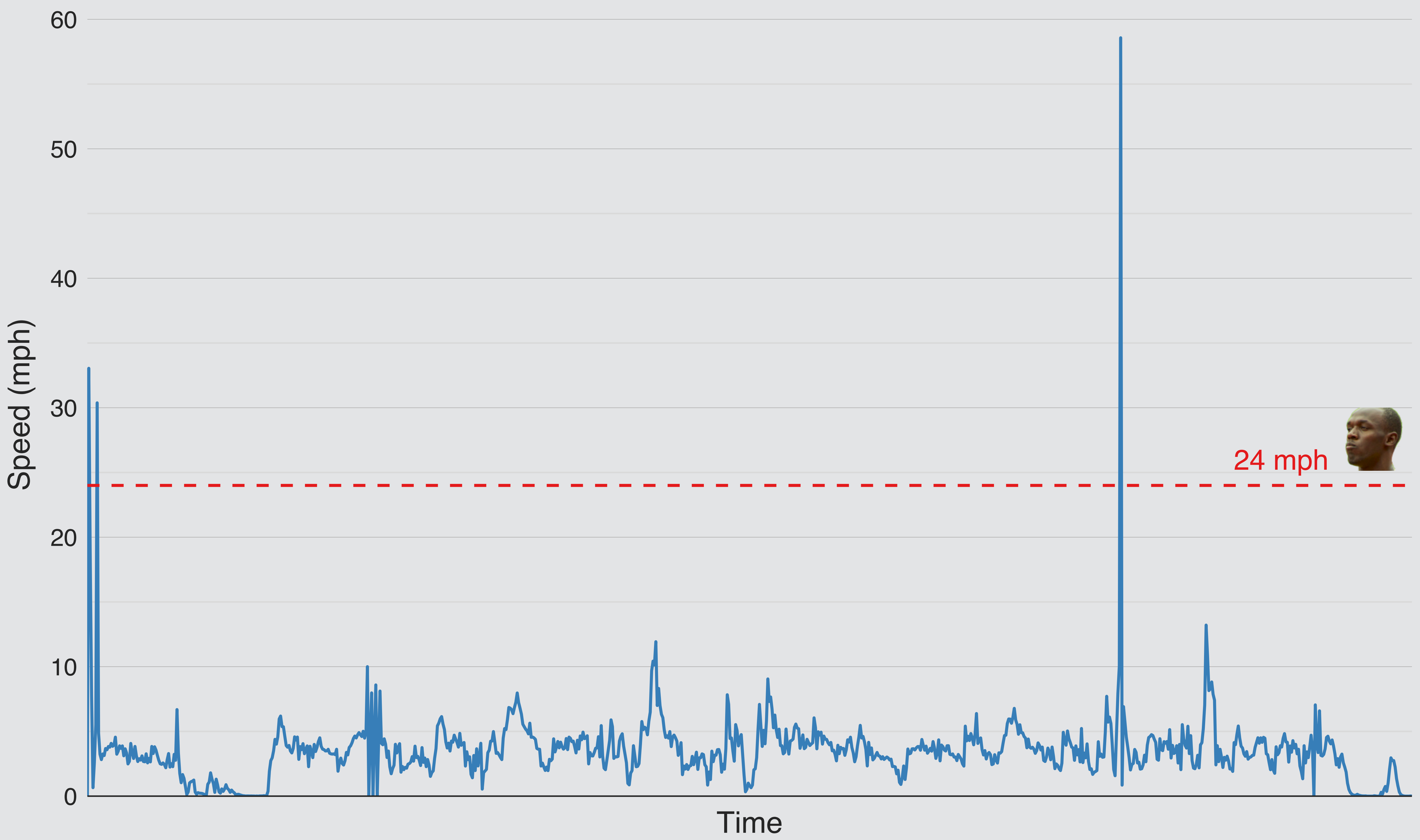

NOKIA
7:46
runtastic
Home Map Split
Jake Van Damme
GPS
DURATION
00:23:43
h:m:s
DISTANCE
2.59
mi
PACE
min/mi
08:53
AVG. SPEED
mph
6.5
ALTITUDE
ft
10
CALORIES
kCal
354
Pause | Stop



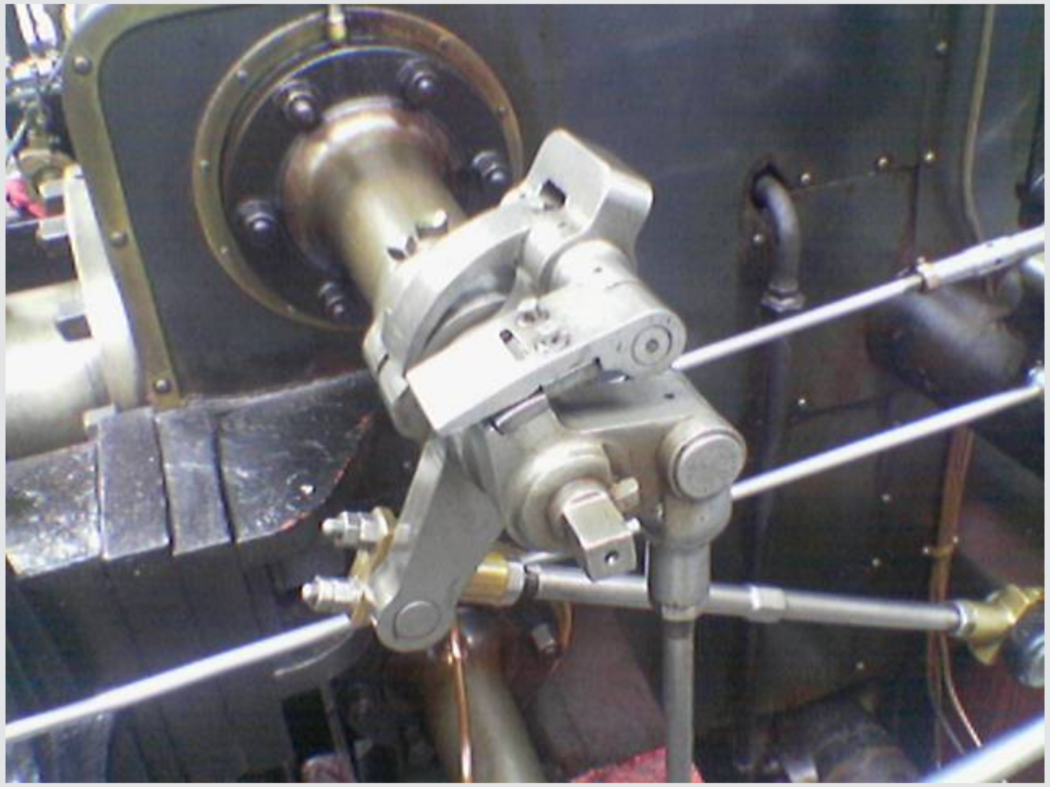
60
50
40
30
20
10
0
Speed (mph)
Time

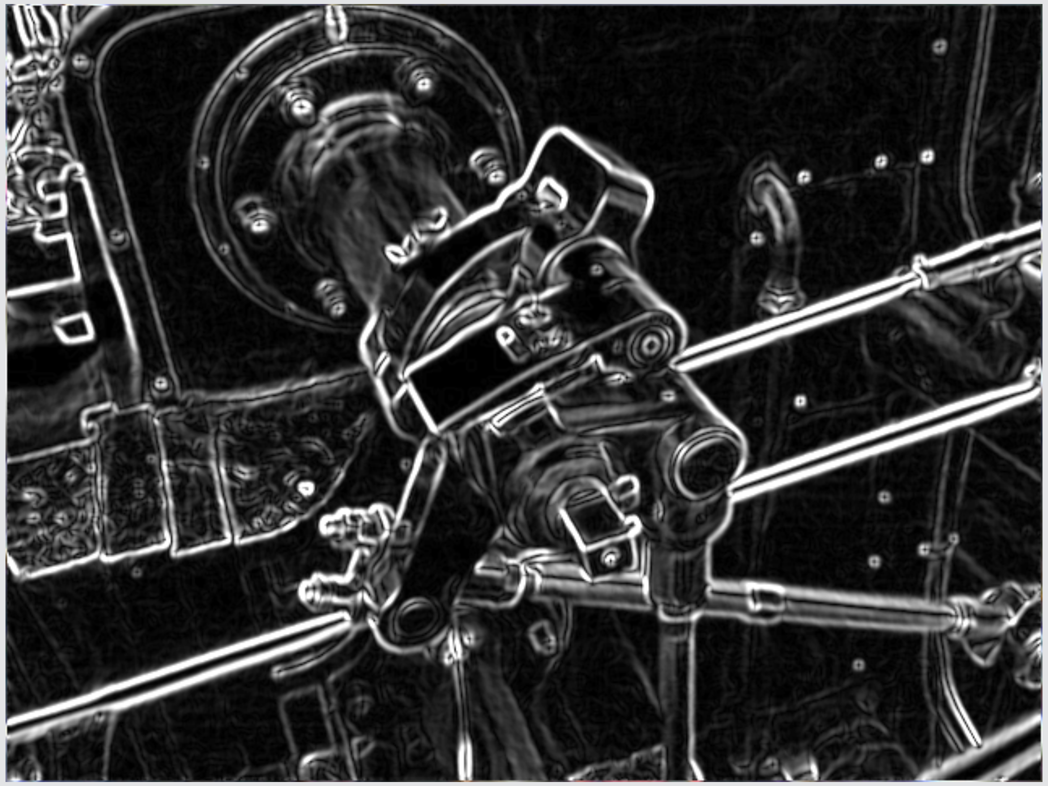


Speed (mph)
60
50
40
30
20
10
0
24 mph
Time

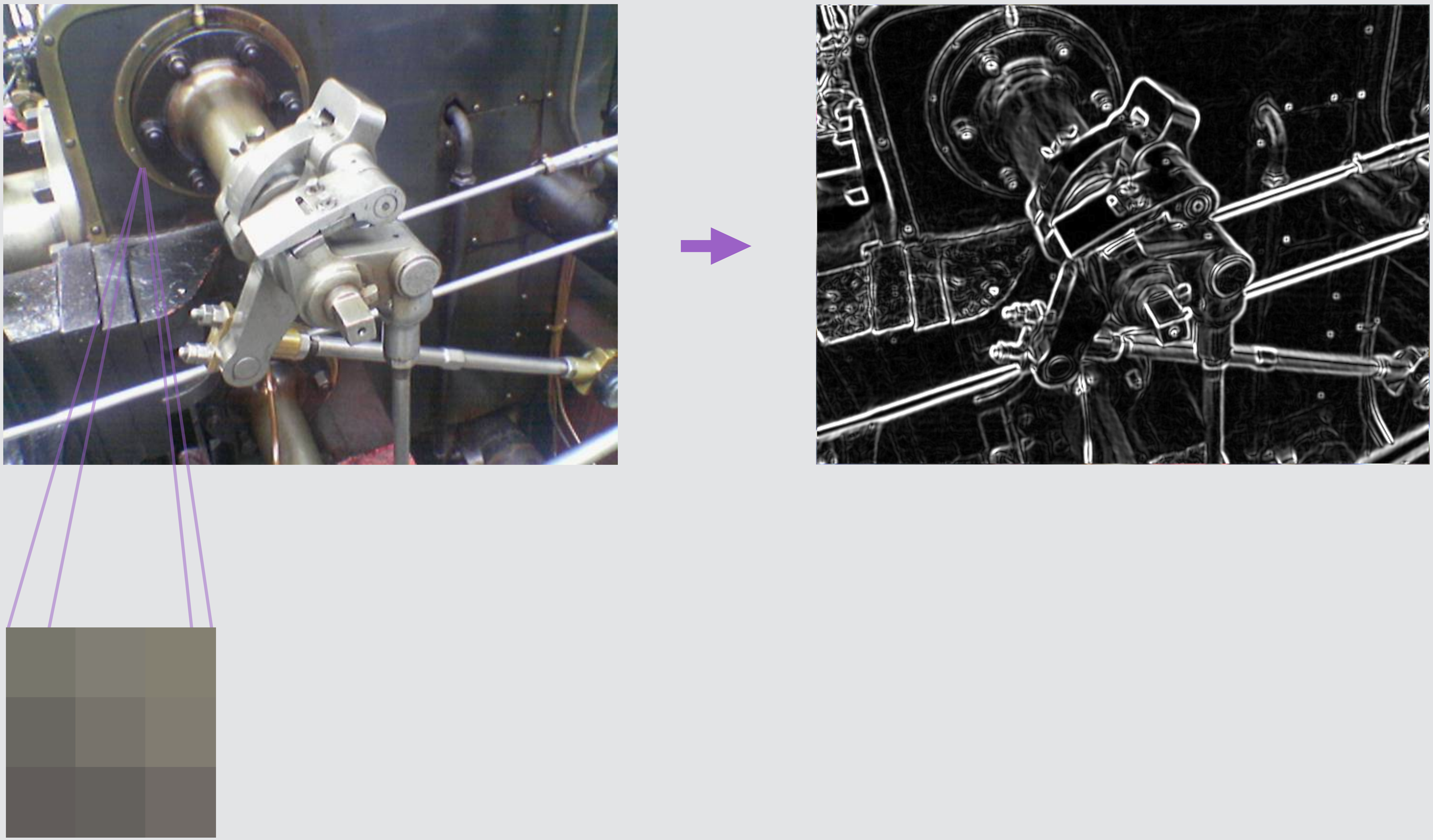

Speed (mph)
60
50
40
30
20
10
0
24 mph
Time

Edge detection

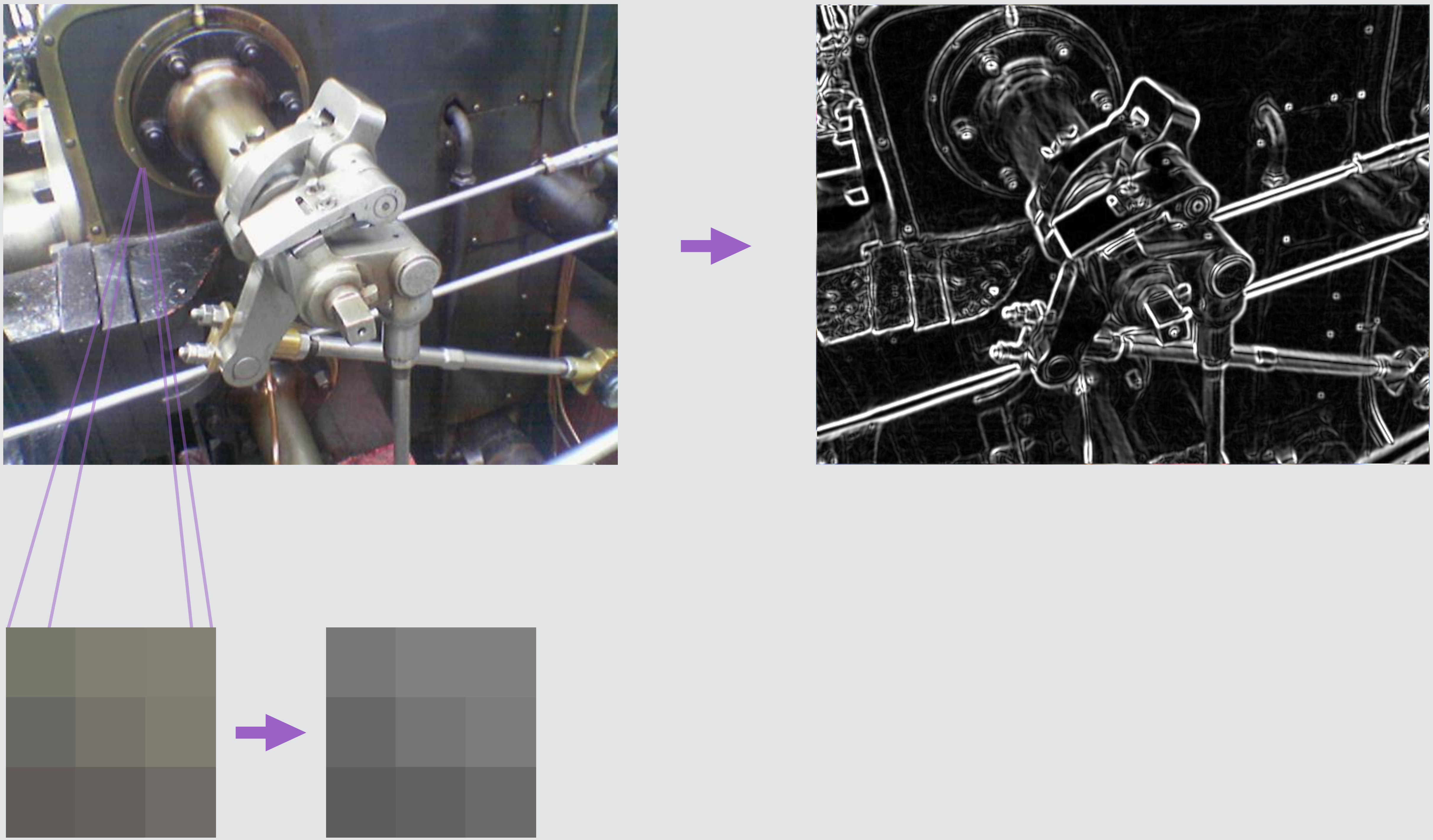




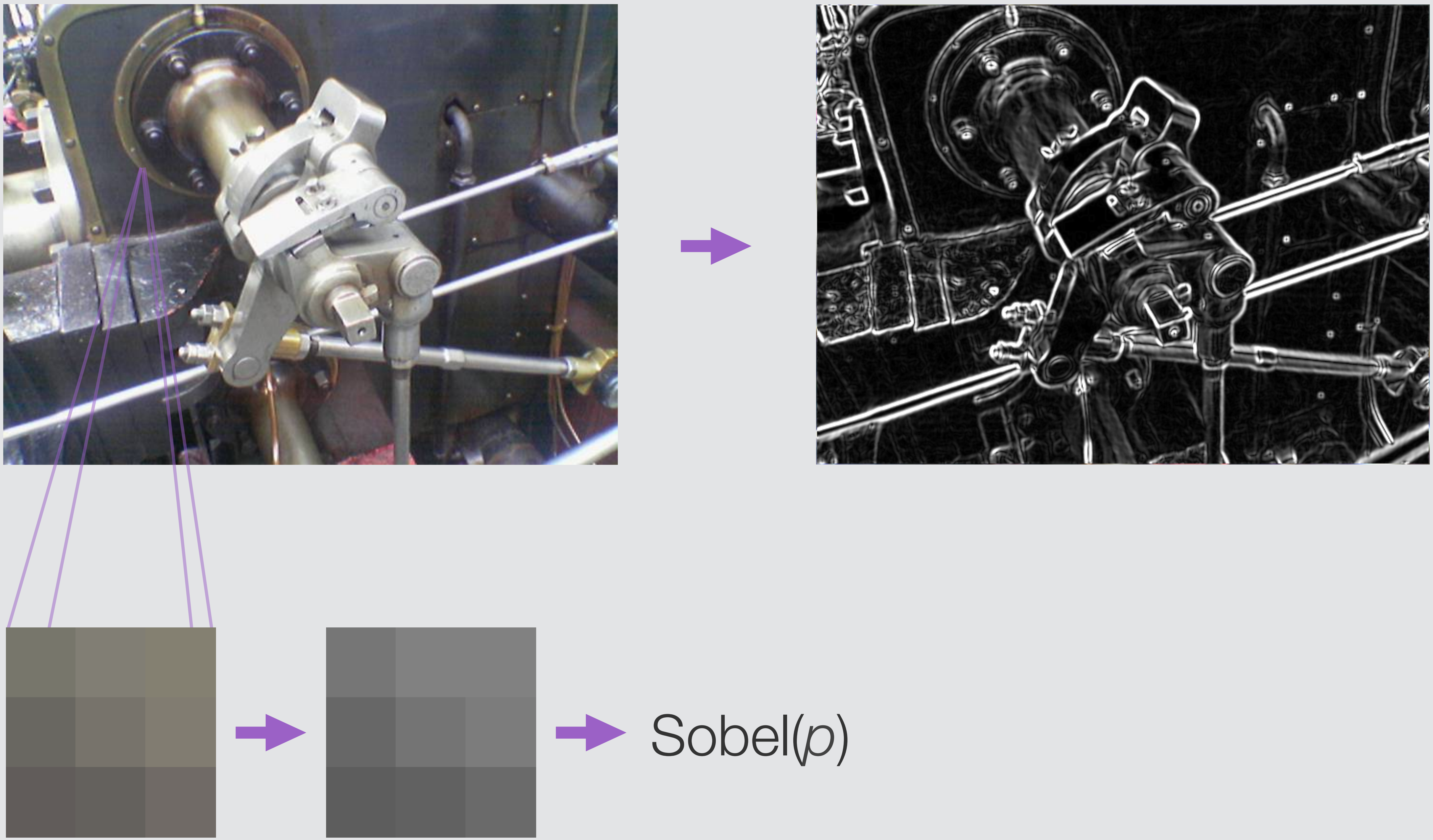
Edge detection



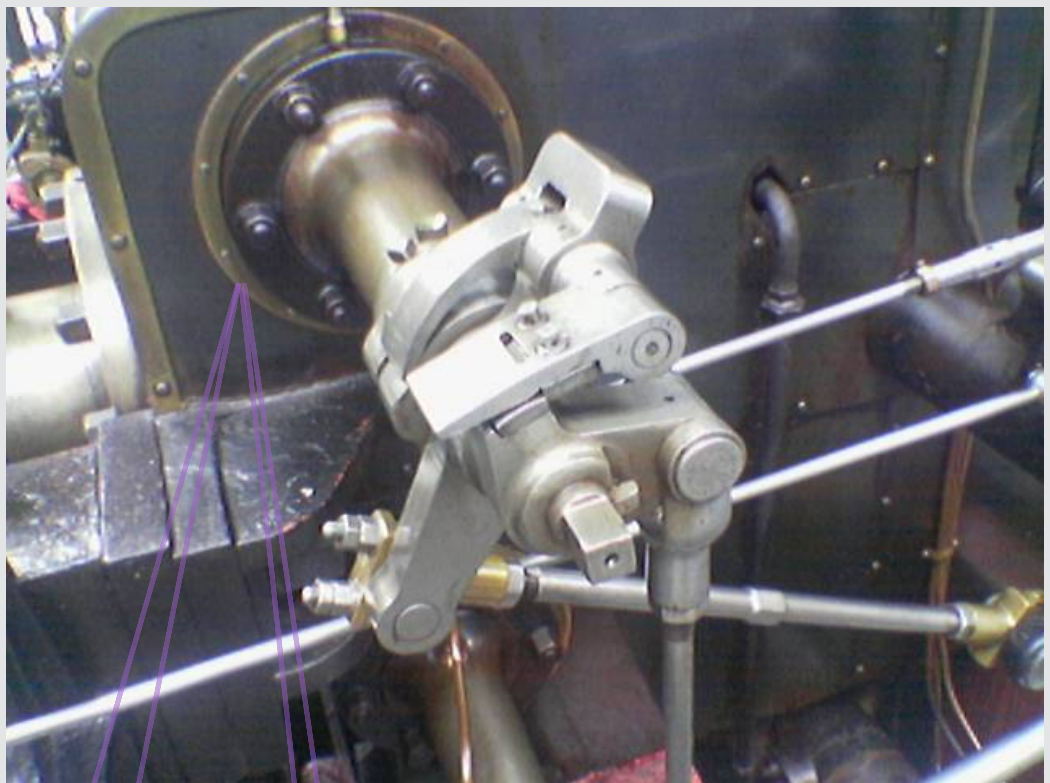
Edge detection

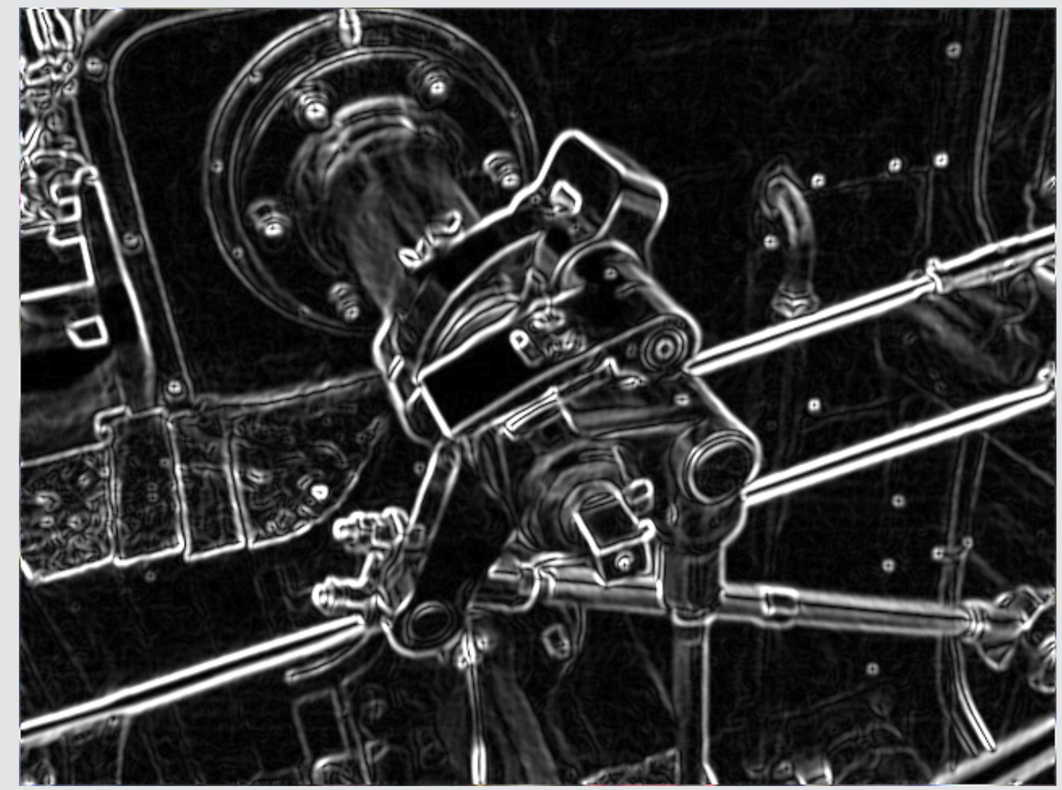


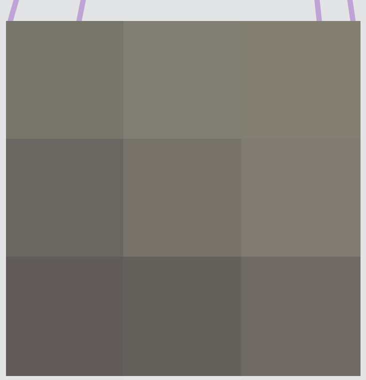
Edge detection

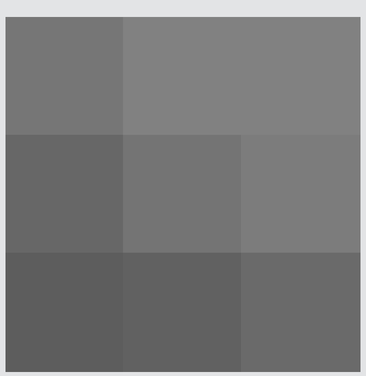


Edge detection



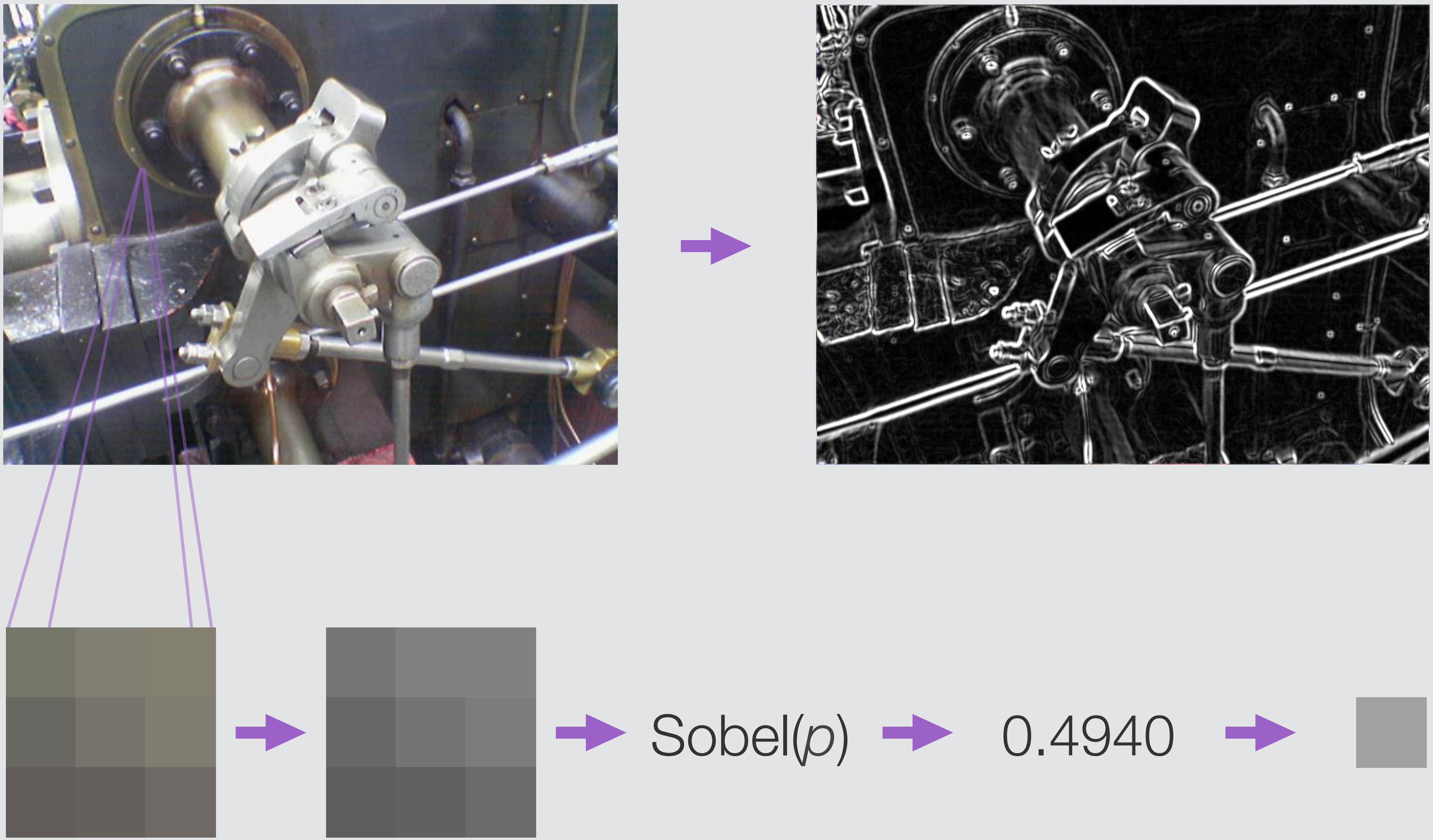




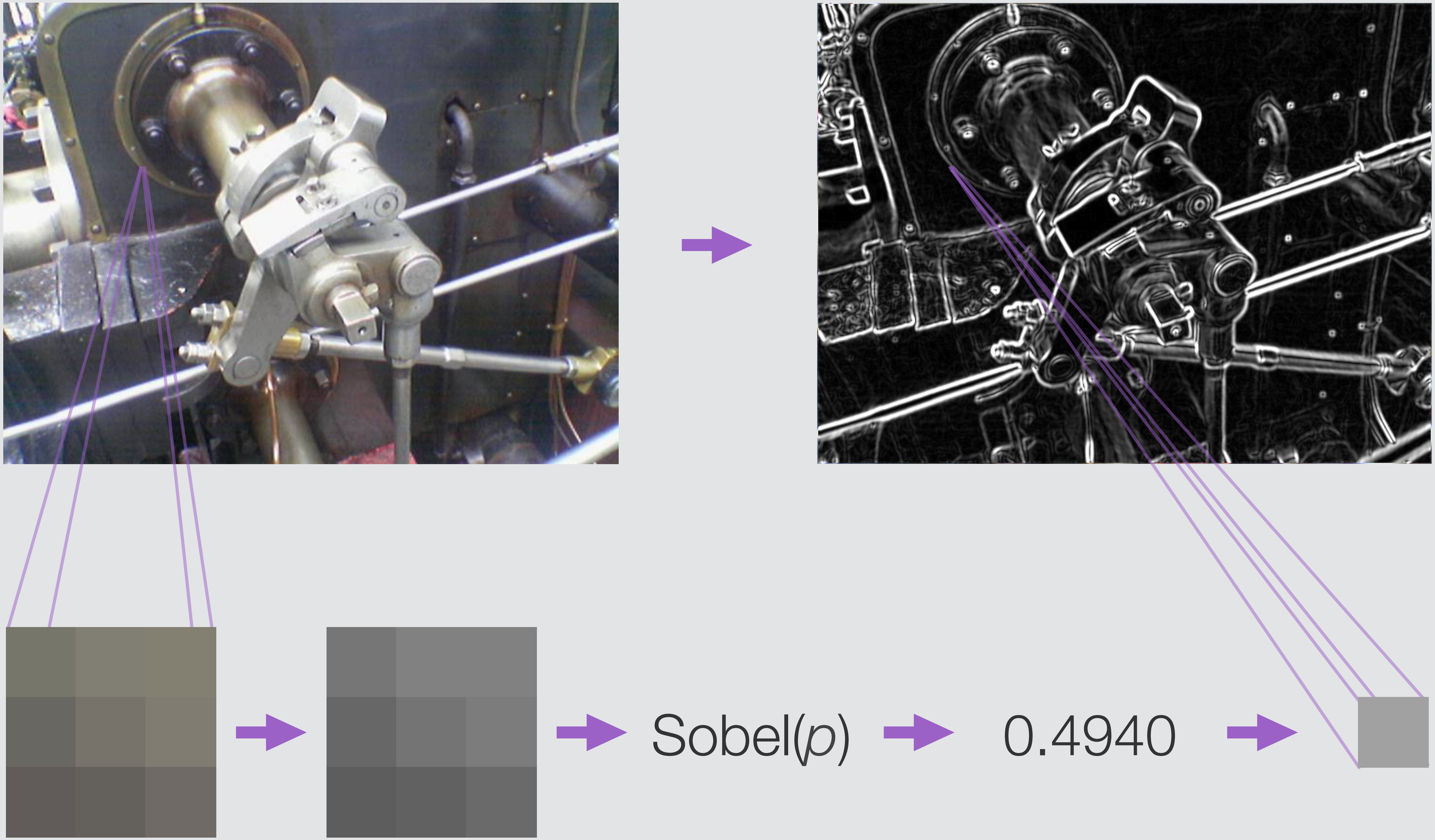


Sobel(p) → 0.4940

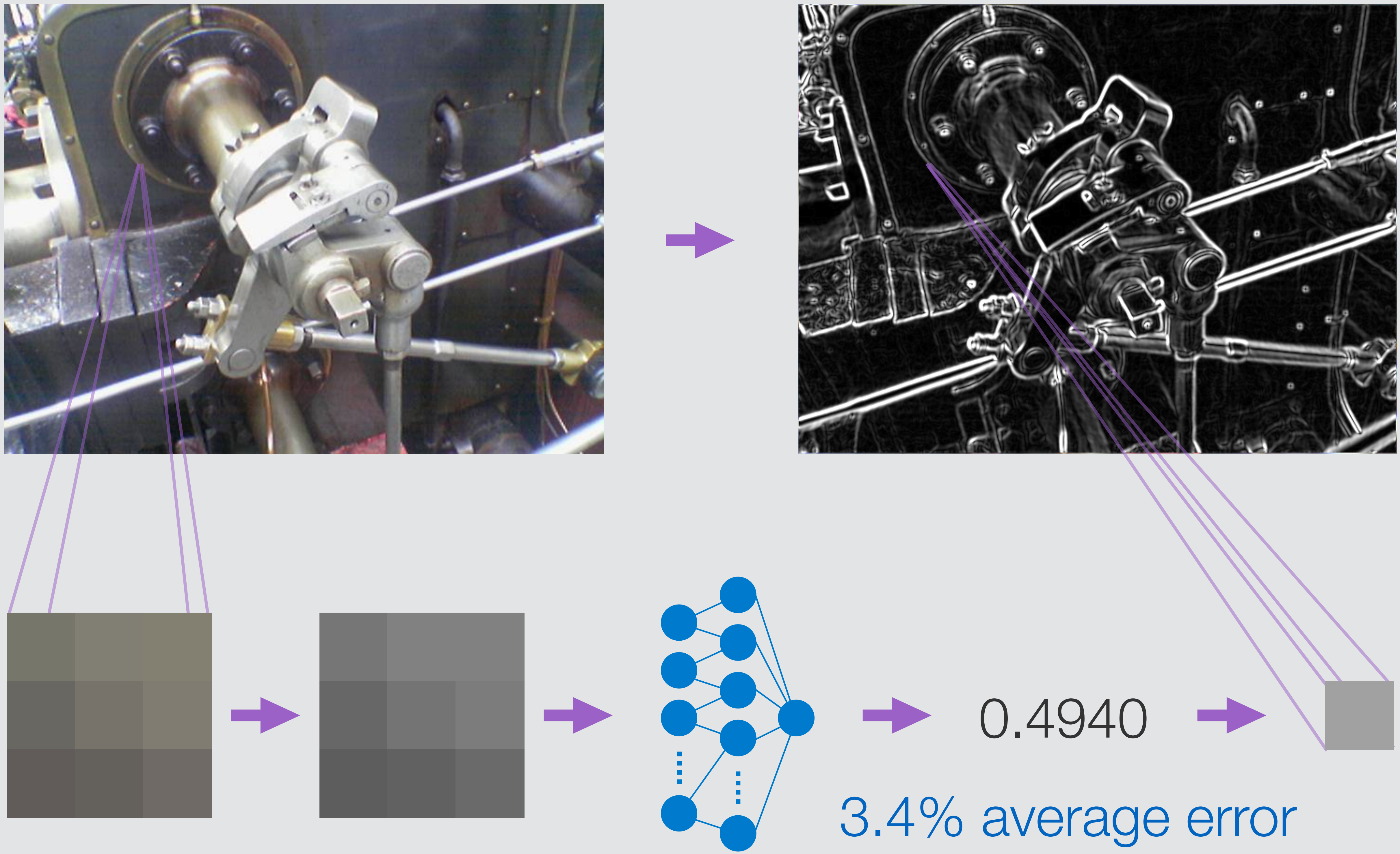
Edge detection



Edge detection



Approximate edge detection



Approximate edge detection

What is the gradient at pixel *p*?

`Sobel(p)`

3.4% average training error

Approximate edge detection

What is the gradient at pixel *p*?

```
Sobel(p)
```

3.4% average training error

Is there an edge at pixel *p*?

```
if (Sobel(p) > 0.1)
    EdgeFound();
```

Approximate edge detection

What is the gradient at pixel *p*?

```
Sobel(p)
```

3.4% average training error

Is there an edge at pixel *p*?

```
if (Sobel(p) > 0.1)
    EdgeFound();
```

36% false positives on the same data!

Approximate edge detection

What is the gradient at pixel *p*?

```
Sobel(p)
```

3.4% average training error

Is there an edge at pixel *p*?

```
if (Sobel(p) > 0.1)
    EdgeFound();
```

36% false positives on the same data!

Computation compounds uncertainty!

```
GeoCoordinate  Location;

double  Grad = Sobel(p);
```

```
Uncertain<GeoCoordinate> Location;

Uncertain<double> Grad = Sobel(p);
```

Uncertain<T> is an uncertain type abstraction.

It encourages non-expert developers to explicitly reason about uncertainty.

```
Uncertain<GeoCoordinate> LastLoc =
                              GPS.GetLocation();
Sleep(5);
Uncertain<GeoCoordinate> Loc =
                              GPS.GetLocation();
```

```
Uncertain<GeoCoordinate> LastLoc =
                      GPS.GetLocation();
Sleep(5);
Uncertain<GeoCoordinate> Loc =
                      GPS.GetLocation();

Uncertain<double> Dist =
             GPS.Distance(Loc, LastLoc);
Uncertain<double> Speed = Dist / 5;
```

```
Uncertain<GeoCoordinate> LastLoc =
                      GPS.GetLocation();
Sleep(5);
Uncertain<GeoCoordinate> Loc =
                      GPS.GetLocation();

Uncertain<double> Dist =
             GPS.Distance(Loc, LastLoc);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4) print("Great job!");
```

```
Uncertain<GeoCoordinate> LastLoc =
                      GPS.GetLocation();
Sleep(5);
Uncertain<GeoCoordinate> Loc =
                      GPS.GetLocation();

Uncertain<double> Dist =
             GPS.Distance(Loc, LastLoc);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4) print("Great job!");

print("Your speed: " + Speed.E());
```

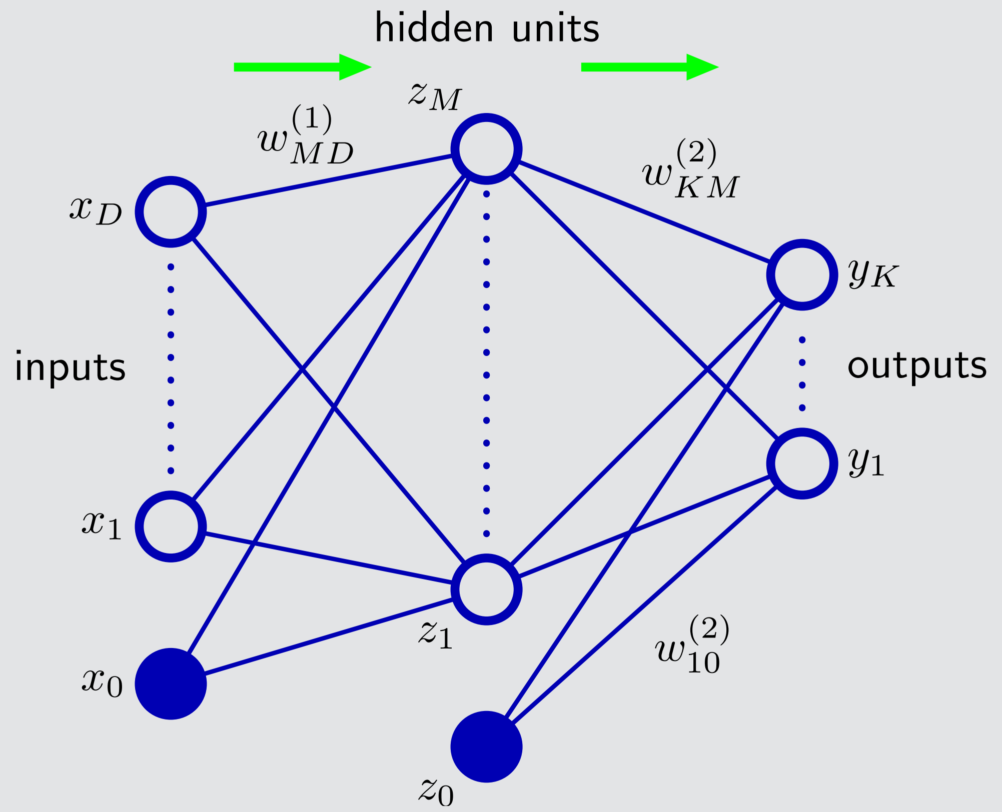
```
Uncertain<GeoCoordinate> LastLoc =
                         GPS.GetLocation();
Sleep(5);
Uncertain<GeoCoordinate> Loc =
                         GPS.GetLocation();

Uncertain<double> Dist =
               GPS.Distance(Loc, LastLoc);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4) print("Great job!");

print("Your speed: " + Speed.E());
```

Probabilistic programming

BUGS, Church, Infer.NET, …

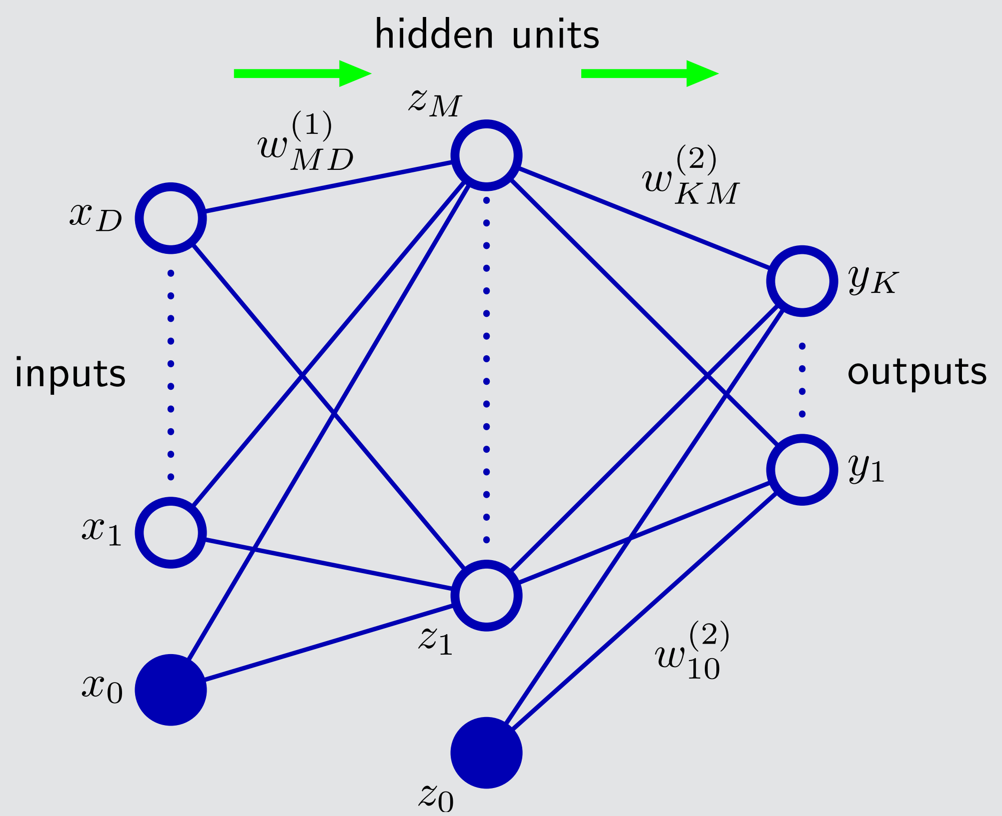
Probabilistic programming

BUGS, Church, Infer.NET, …



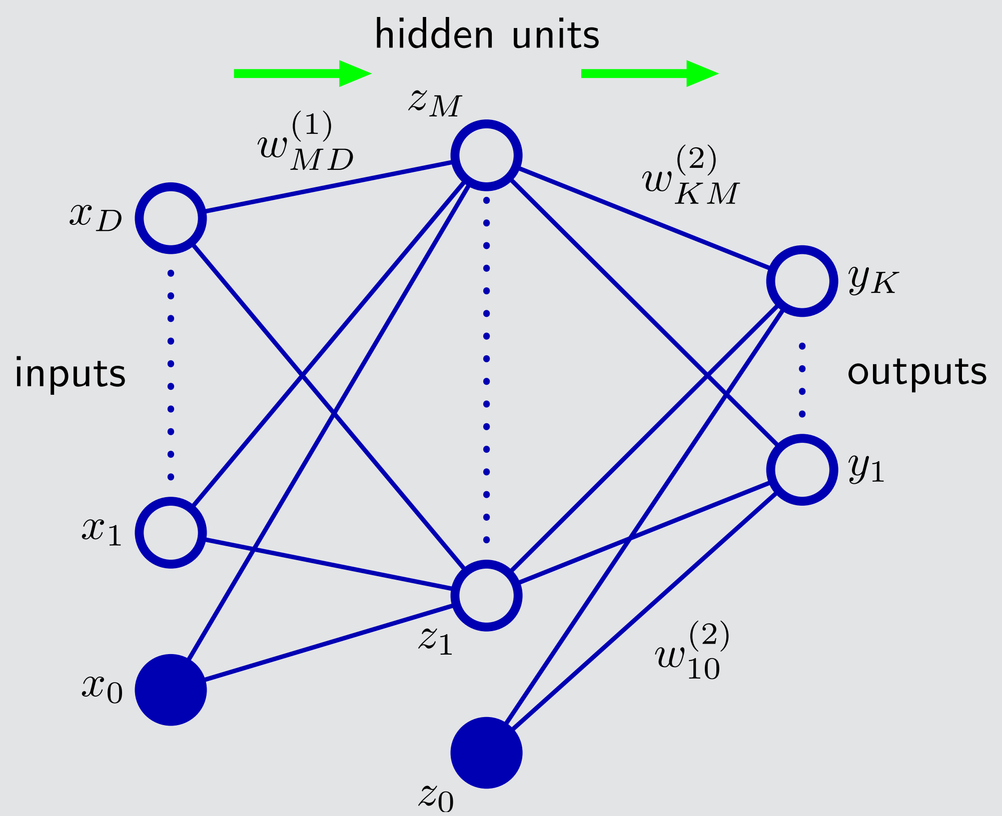
Probabilistic programming

BUGS, Church, Infer.NET, …



Probabilistic programming

BUGS, Church, Infer.NET, …



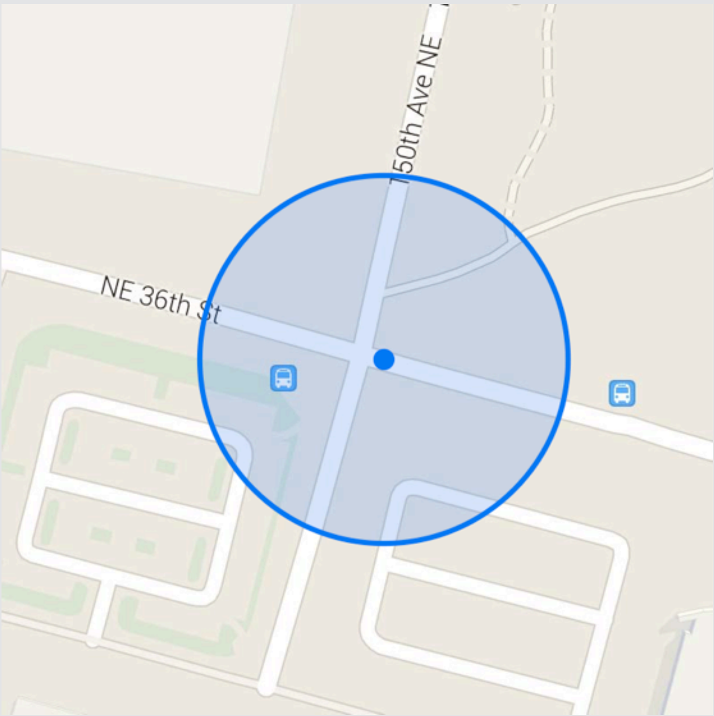
Uncertain<*T*> helps developers *without* statistics PhDs.

```
Uncertain<GeoCoordinate> LastLoc =
                            GPS.GetLocation();
```

A variable of type Uncertain<*T*> is a random variable, represented by a distribution.

```
Uncertain<GeoCoordinate> LastLoc =
                                GPS.GetLocation();
```

A variable of type Uncertain<*T*> is a random variable, represented by a distribution.



"We define accuracy as the radius of 68% confidence [of a] normal distribution."
—Android

Sampling functions return random samples.

Sampling functions return random samples.

✓ Simple computations.

Sampling functions return random samples.

✓ Simple computations.

✓ Represent many distributions.

Sampling functions return random samples.

- ✓ Simple computations.
- ✓ Represent many distributions.
- ✗ Sampling is approximate.

(Later: how Uncertain<*T*> learned to love approximation, and you can too)

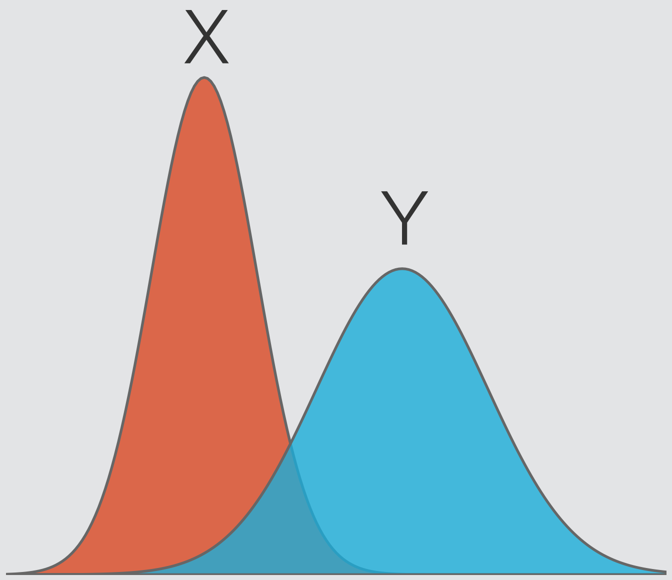
```
Uncertain<double> Speed = Dist / 5;
```

```
Uncertain<double> Speed = Dist / 5;
```

Or more generally, `Z = X + Y`, if `X` and `Y` are distributions.

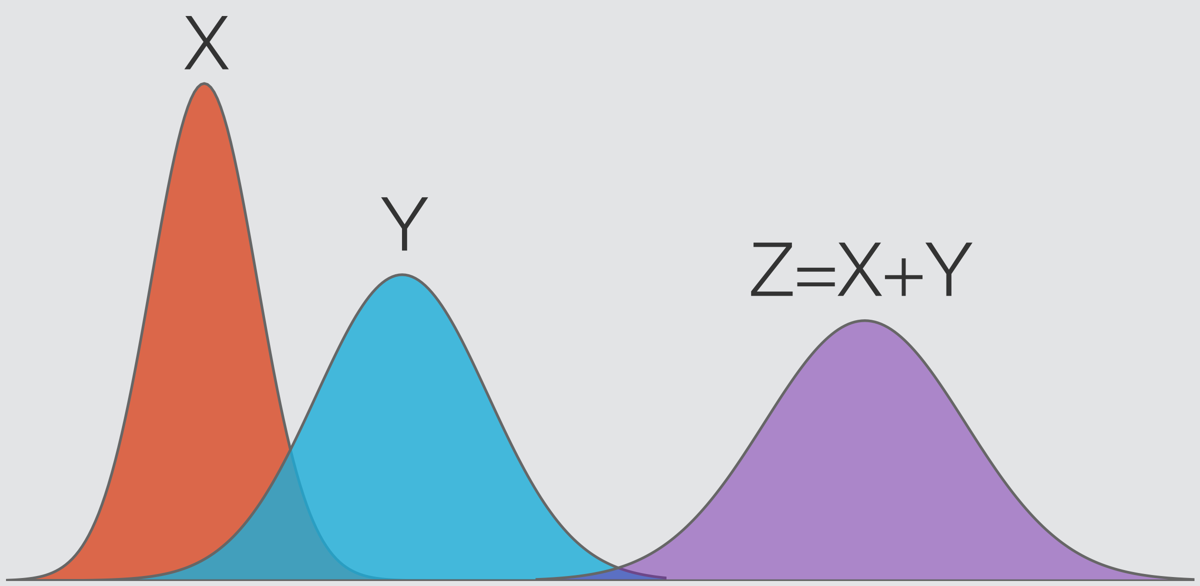
```
Uncertain<double> Speed = Dist / 5;
```

Or more generally, `Z = X + Y`, if `X` and `Y` are distributions.



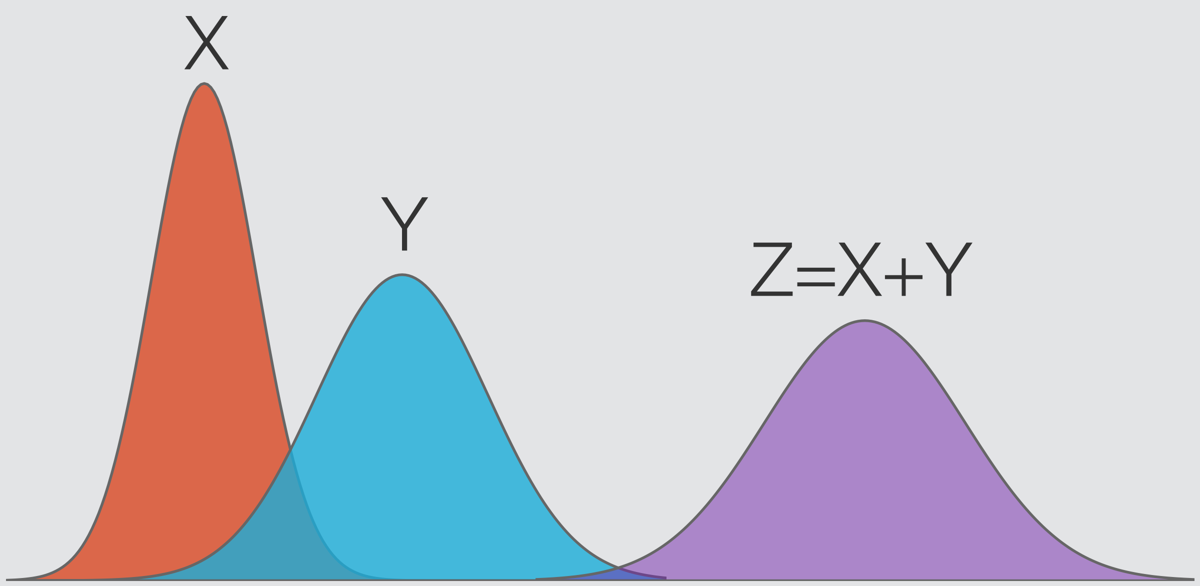
```
Uncertain<double> Speed = Dist / 5;
```

Or more generally, `Z = X + Y`, if `X` and `Y` are distributions.



```
Uncertain<double> Speed = Dist / 5;
```

Or more generally, `Z = X + Y`, if `X` and `Y` are distributions.

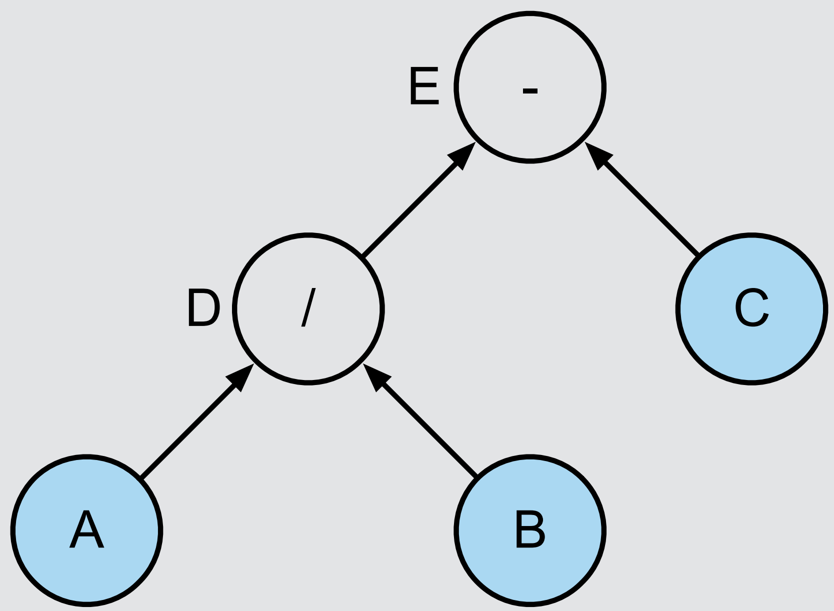


If x is a sample of X
and y is a sample of Y
then $x+y$ is a sample of $X+Y$ *

* if X and Y are independent

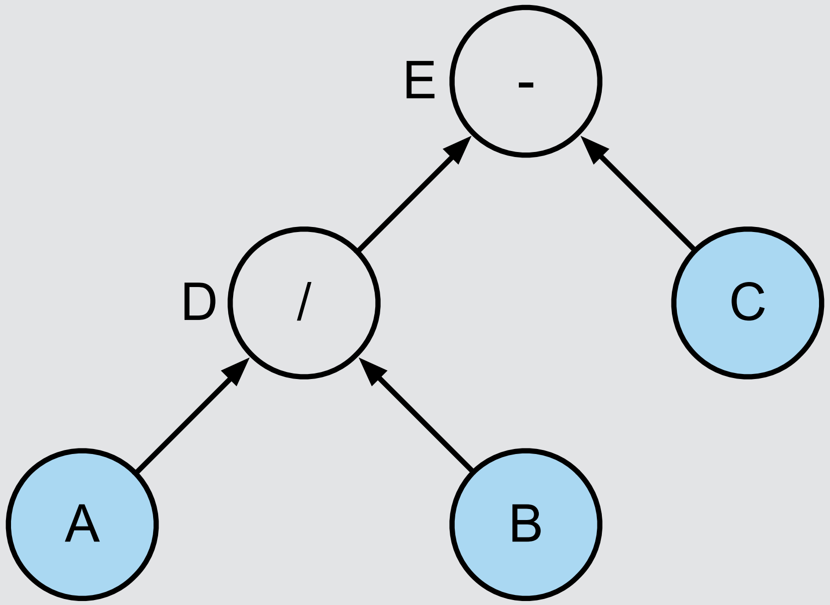
```
D = A / B
E = D - C
```

Bayesian network representation:



```
D = A / B
E = D - C
```

Bayesian network representation:



Sampling function for **E** recursively samples children.

If x is a sample of X
and y is a sample of Y
then $x+y$ is a sample of $X+Y$ *

* Only if X and Y are independent.

If x is a sample of X
and y is a sample of Y
then $x+y$ is a sample of $X+Y$ *

* Only if X and Y are independent.

```
A = X + Y
B = A + X
```

(X,Y independent)

If x is a sample of X
and y is a sample of Y
then $x+y$ is a sample of $X+Y$ *

* Only if X and Y are independent.

```
A = X + Y
B = A + X
```
(X,Y independent)

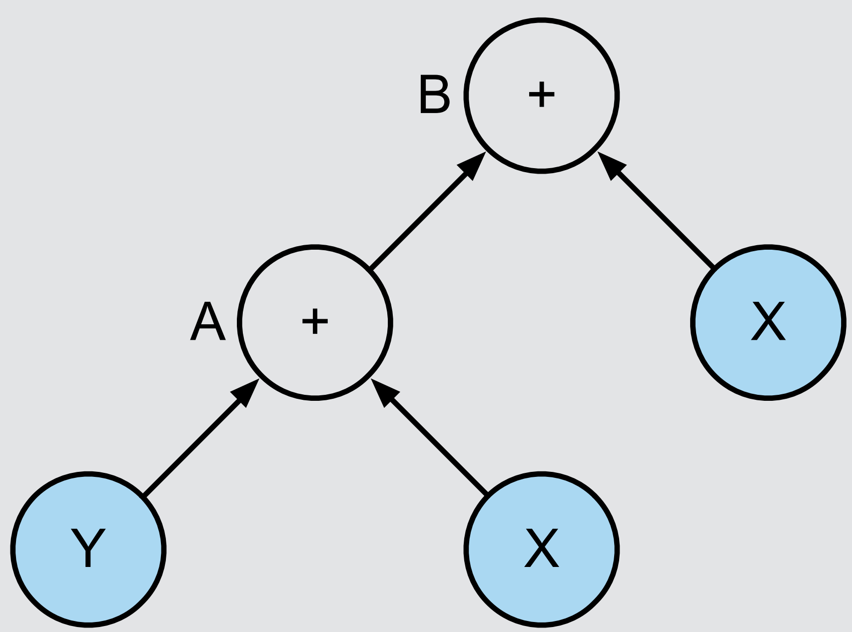
A and B depend on X – not independent!

If x is a sample of X
and y is a sample of Y
then $x+y$ is a sample of $X+Y$ *

* Only if X and Y are independent.

```
A = X + Y
B = A + X
```
(X,Y independent)

A and B depend on X – not independent!

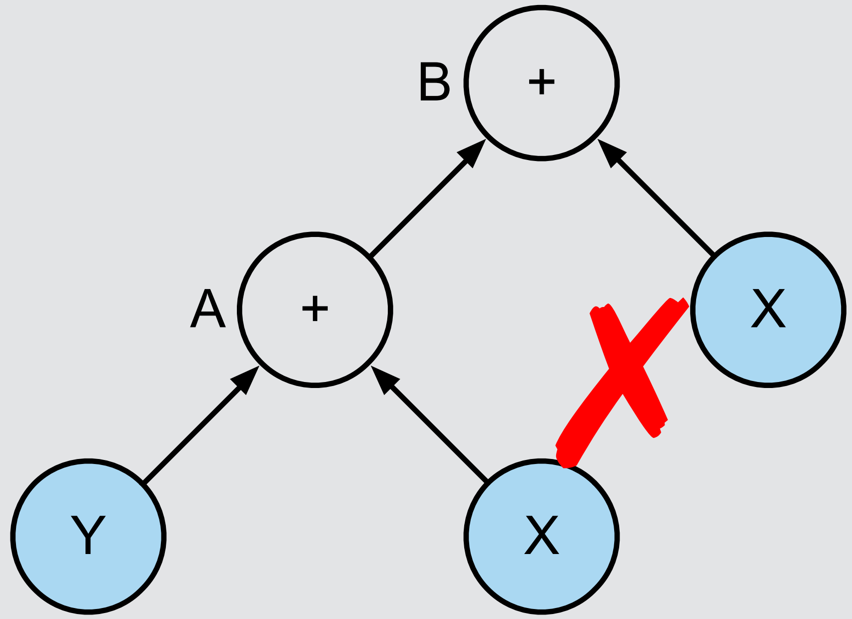


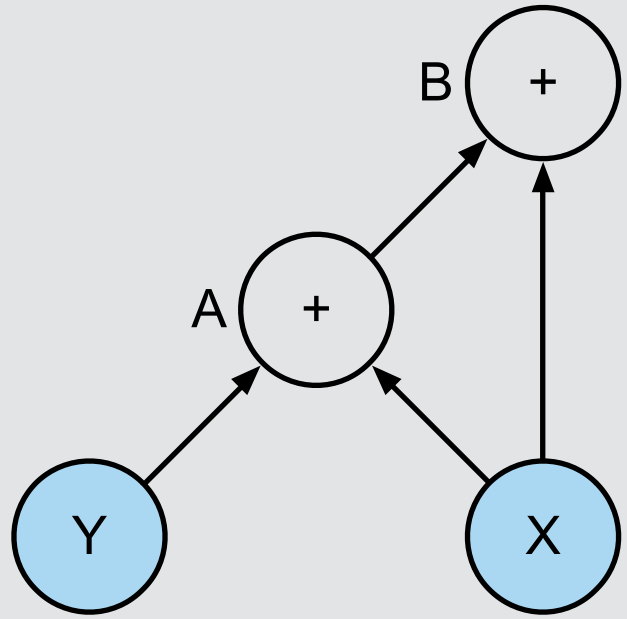
If x is a sample of X
and y is a sample of Y
then $x+y$ is a sample of $X+Y$ *

* Only if X and Y are independent.

```
A = X + Y
B = A + X
```
(X,Y independent)

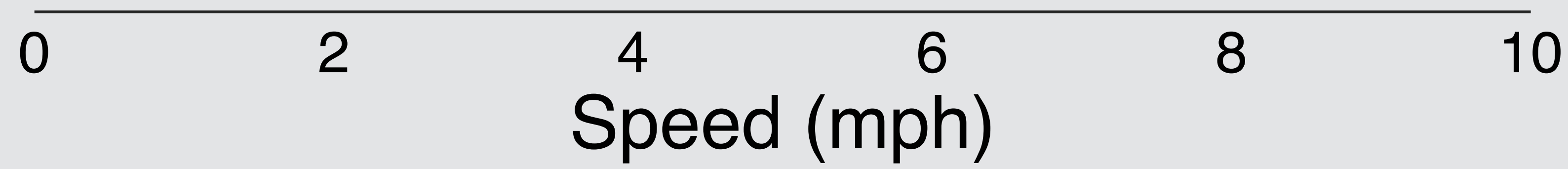
A and B depend on X – not independent!



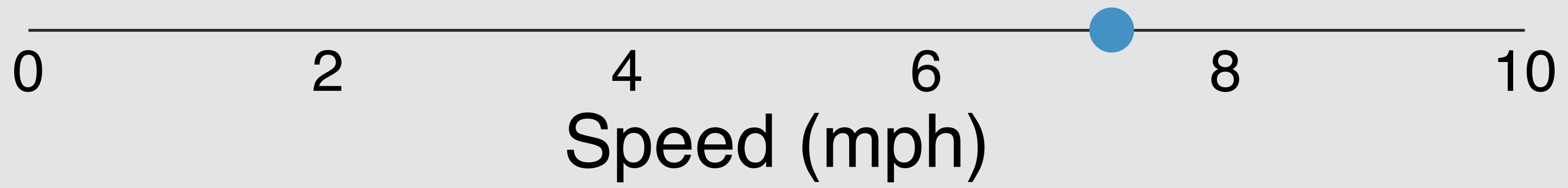


```
if (Speed > 4) print("Great job!");
```

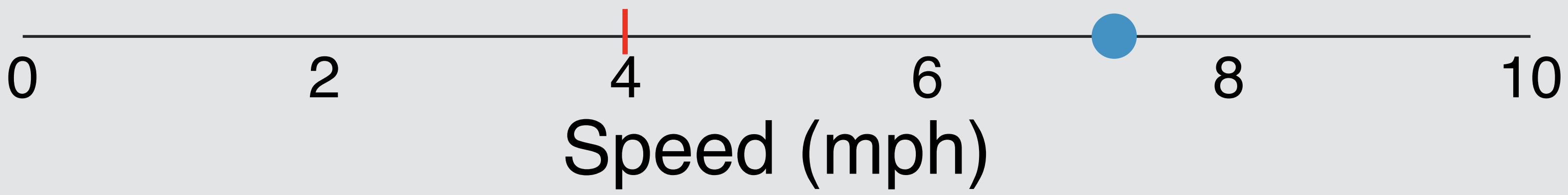
```
if (Speed > 4) print("Great job!");
```



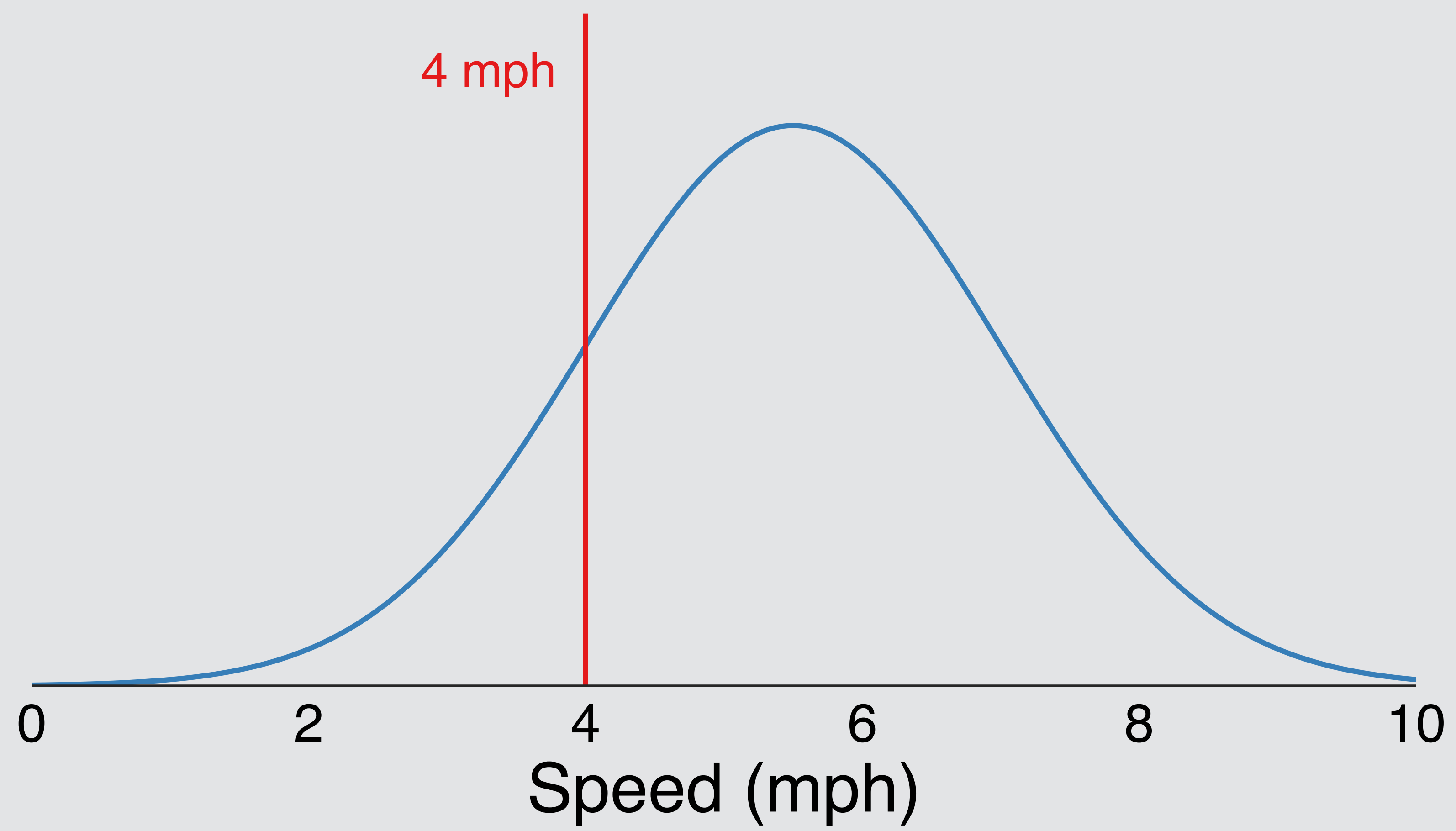
```
if (Speed > 4) print("Great job!");
```



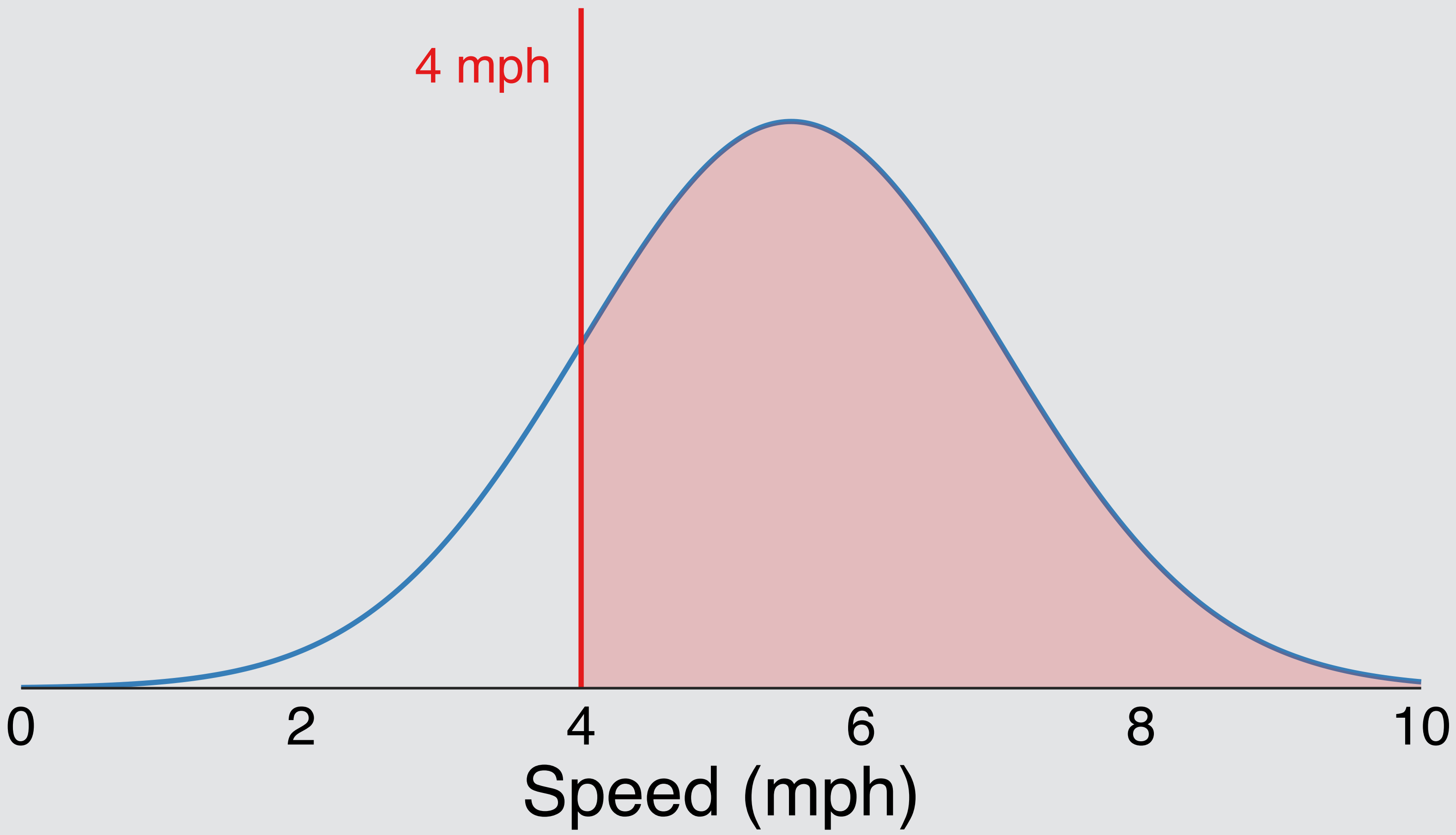
```
if (Speed > 4) print("Great job!");
```



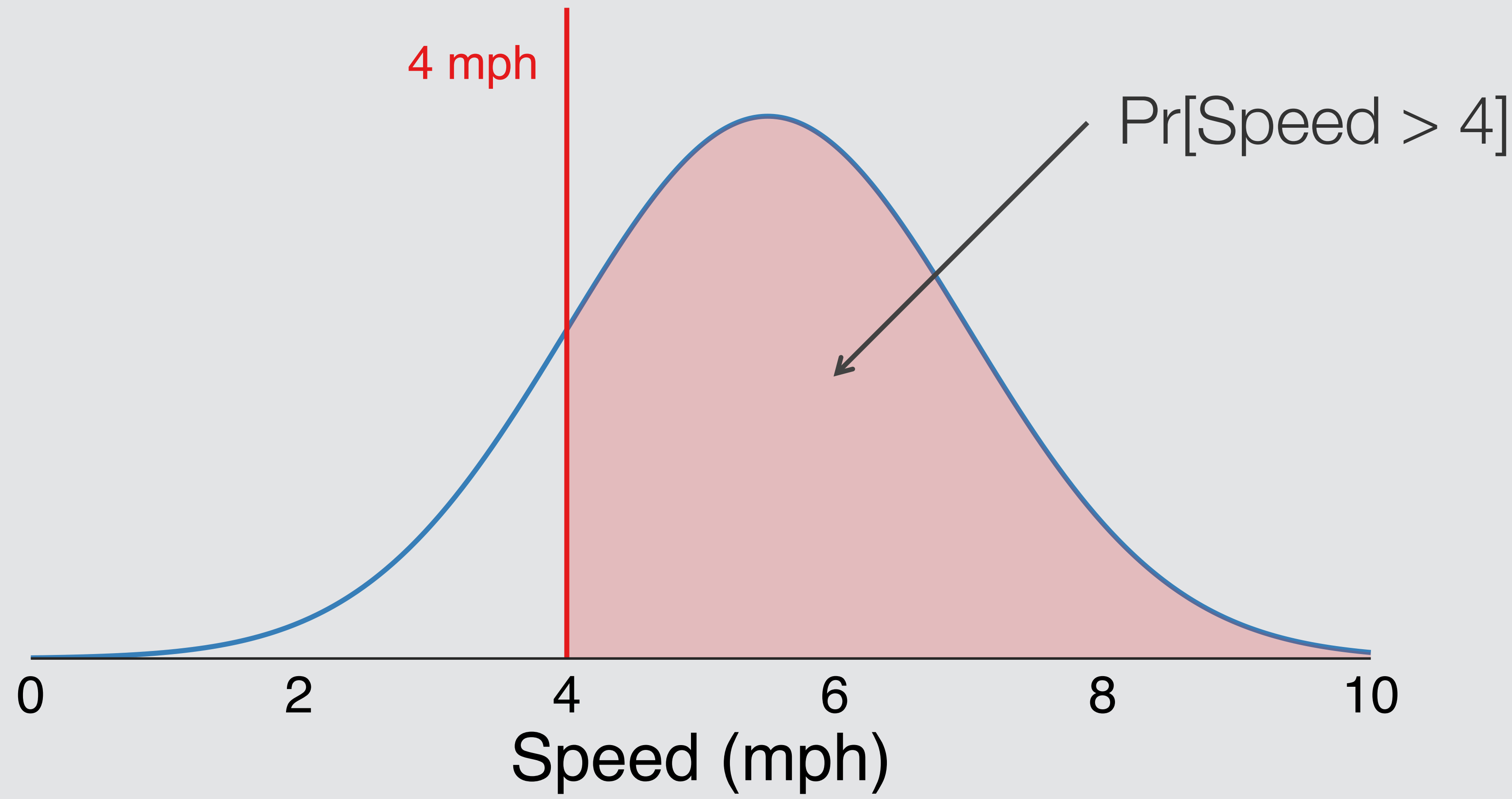
```
if (Speed > 4) print("Great job!");
```



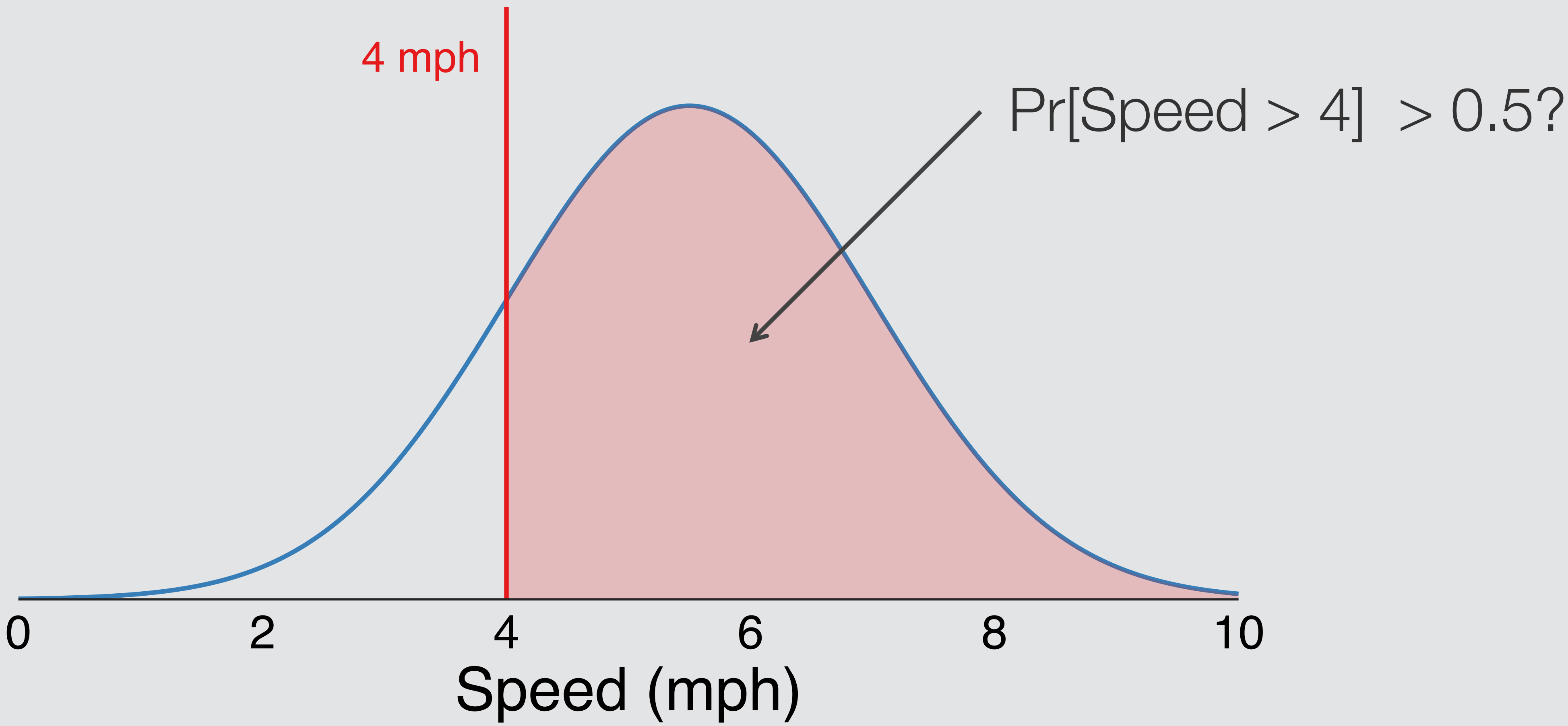
```
if (Speed > 4) print("Great job!");
```



```
if (Speed > 4) print("Great job!");
```

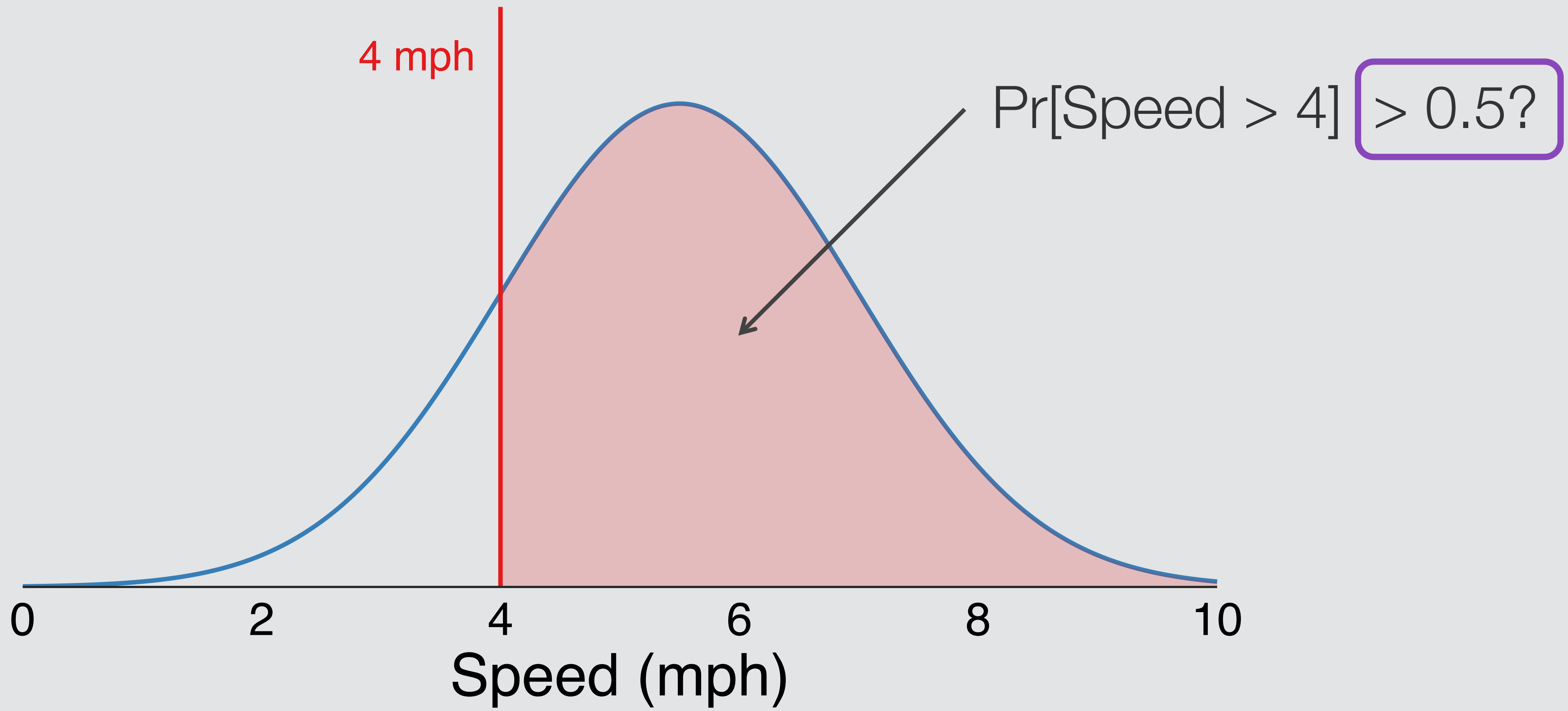


```
if (Speed > 4) print("Great job!");
```



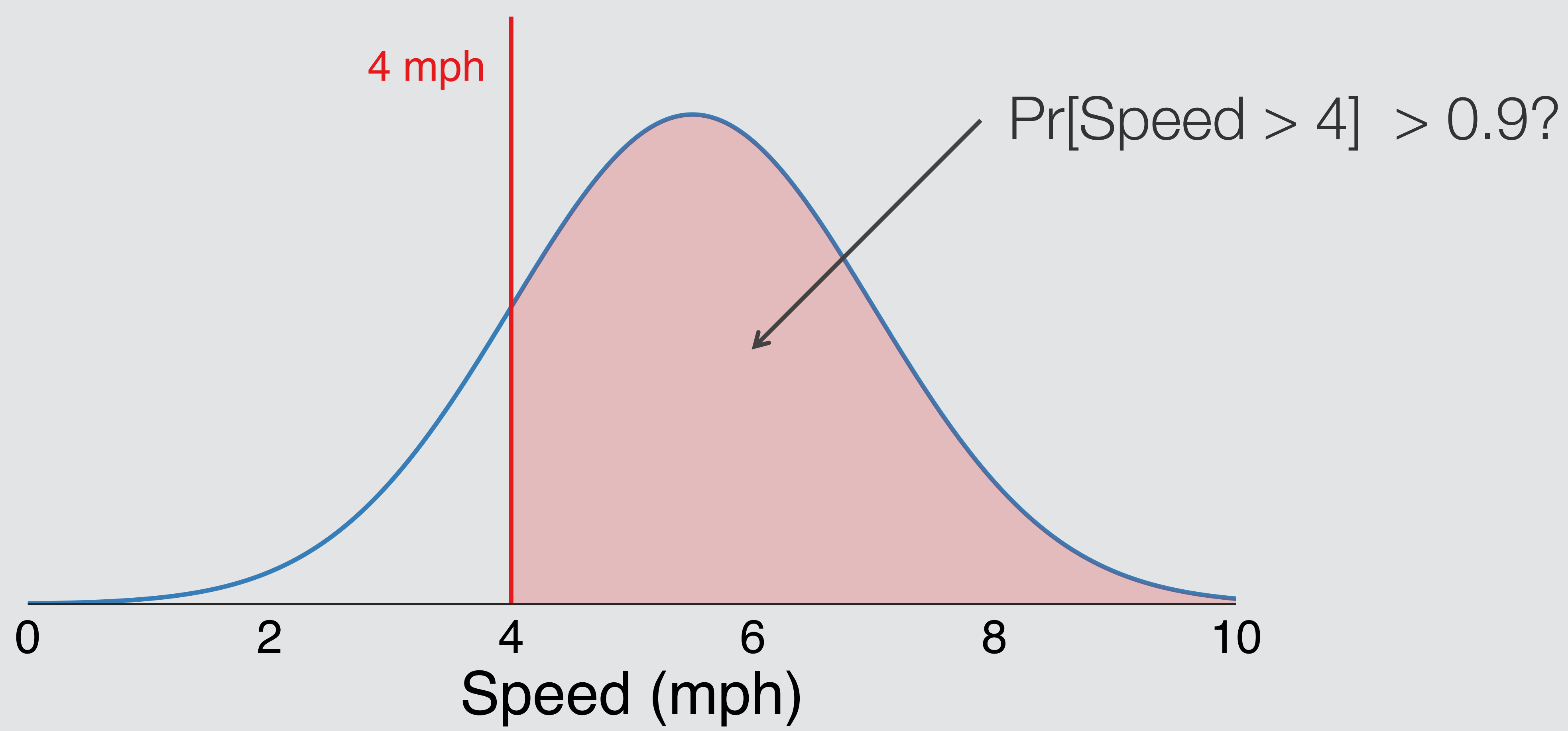
More likely than not that Speed > 4?

```
if (Speed > 4) print("Great job!");
```



More likely than not that Speed > 4?

```
if (Speed > 4).Pr(0.9) print("Great job!");
```



```
if (Speed > 4).Pr(0.9) print("Great job!");
```

At least 90% likely that Speed > 4?

```
if (Speed > 4).Pr(0.9) print("Great job!");
```

$$\Pr[\text{Speed} > 4] > 0.9$$

```
if (Speed > 4).Pr(0.9) print("Great job!");
```

$$\Pr[\underbrace{\text{Speed}}_{\text{approximate!}} > 4] > 0.9$$

```
if (Speed > 4).Pr(0.9) print("Great job!");
```

$$\underbrace{\Pr[\text{Speed} > 4]}_{\text{approximate!}} > 0.9$$

```
if (Speed > 4).Pr(0.9) print("Great job!");
```

null hypothesis H_0: $\Pr[\text{Speed} > 4] \leq 0.9$

$\underbrace{\Pr[\text{Speed} > 4]}_{\text{approximate!}} > 0.9$

```
if (Speed > 4).Pr(0.9) print("Great job!");
```

null hypothesis H_0: $\Pr[\text{Speed} > 4] \leq 0.9$

alternate hypothesis H_A: $\underbrace{\Pr[\text{Speed} > 4]}_{\text{approximate!}} > 0.9$

```
if (Speed > 4).Pr(0.9) print("Great job!");
```

null hypothesis H_0: $\Pr[\text{Speed} > 4] \leq 0.9$

alternate hypothesis H_A: $\underbrace{\Pr[\text{Speed} > 4]}_{\text{approximate!}} > 0.9$

How many samples?

```
if (Speed > 4).Pr(0.9) print("Great job!");
```

null hypothesis H_0: $\Pr[\text{Speed} > 4] \leq 0.9$

alternate hypothesis H_A: $\underbrace{\Pr[\text{Speed} > 4]}_{\text{approximate!}} > 0.9$

How many samples?

Too many = too slow

Too few = too noisy

```
if (Speed > 4).Pr(0.9) print("Great job!");
```

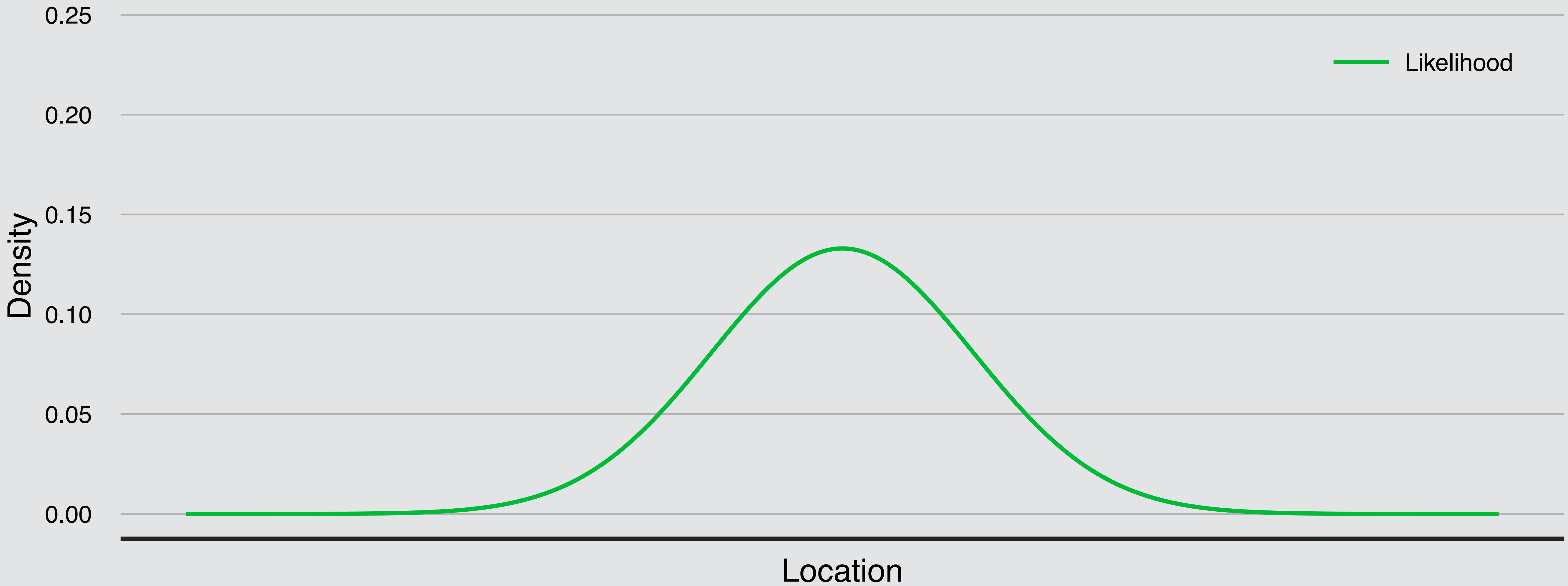
null hypothesis H_0: $\Pr[\text{Speed} > 4] \leq 0.9$

alternate hypothesis H_A: $\underbrace{\Pr[\text{Speed} > 4]}_{\text{approximate!}} > 0.9$

How many samples? Too many = too slow

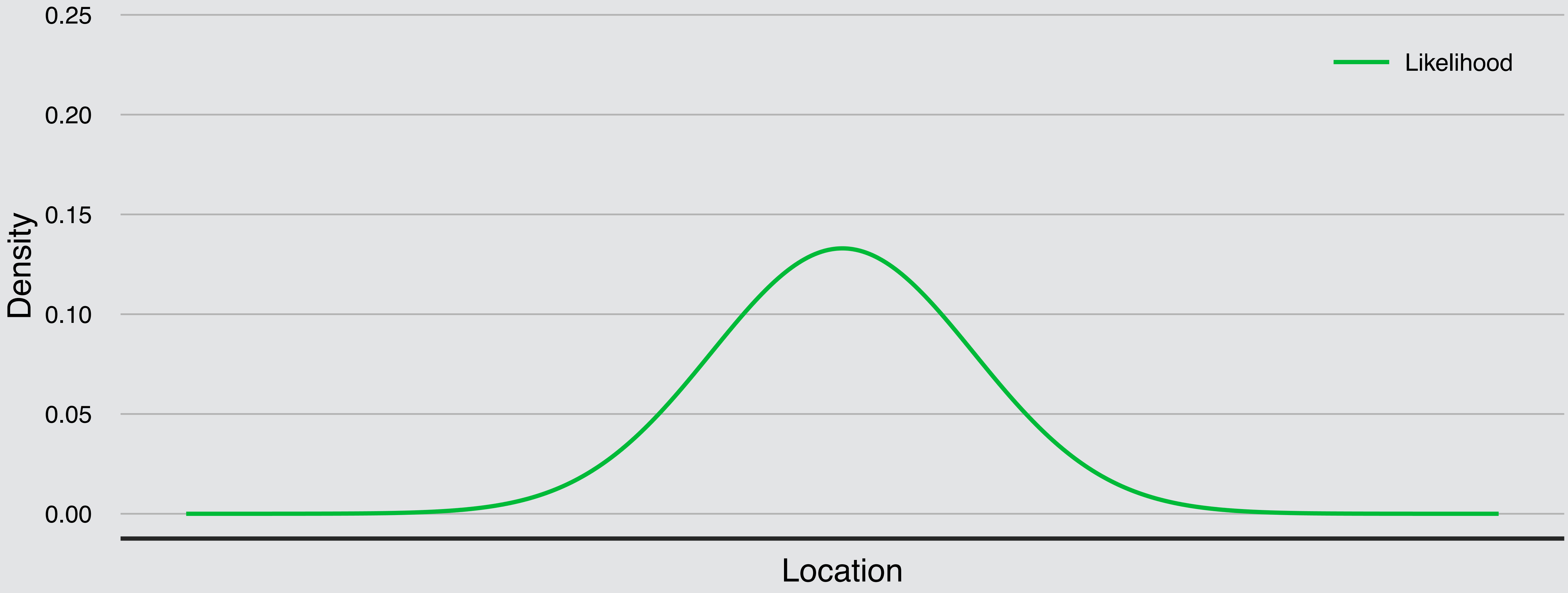
Too few = too noisy

Sequential sampling: sample size depends on progress

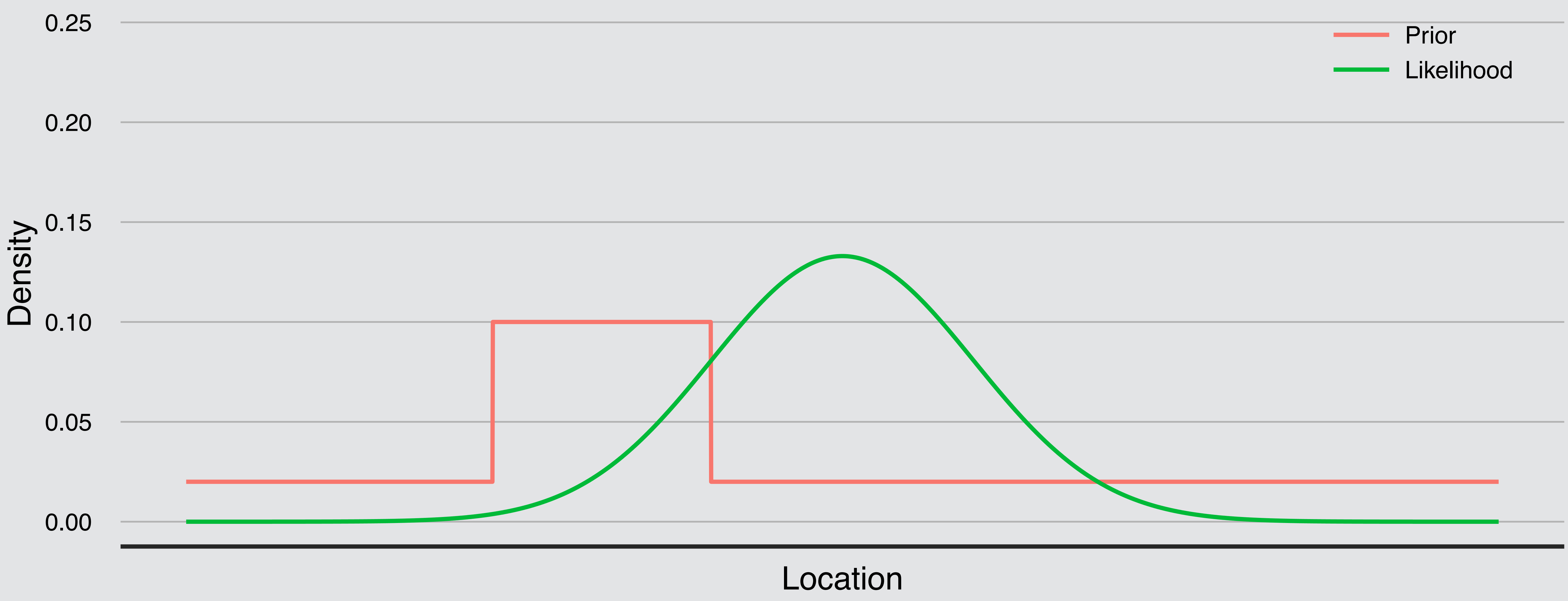


0.25
0.20
0.15
0.10
0.05
0.00
Density
Likelihood
Location

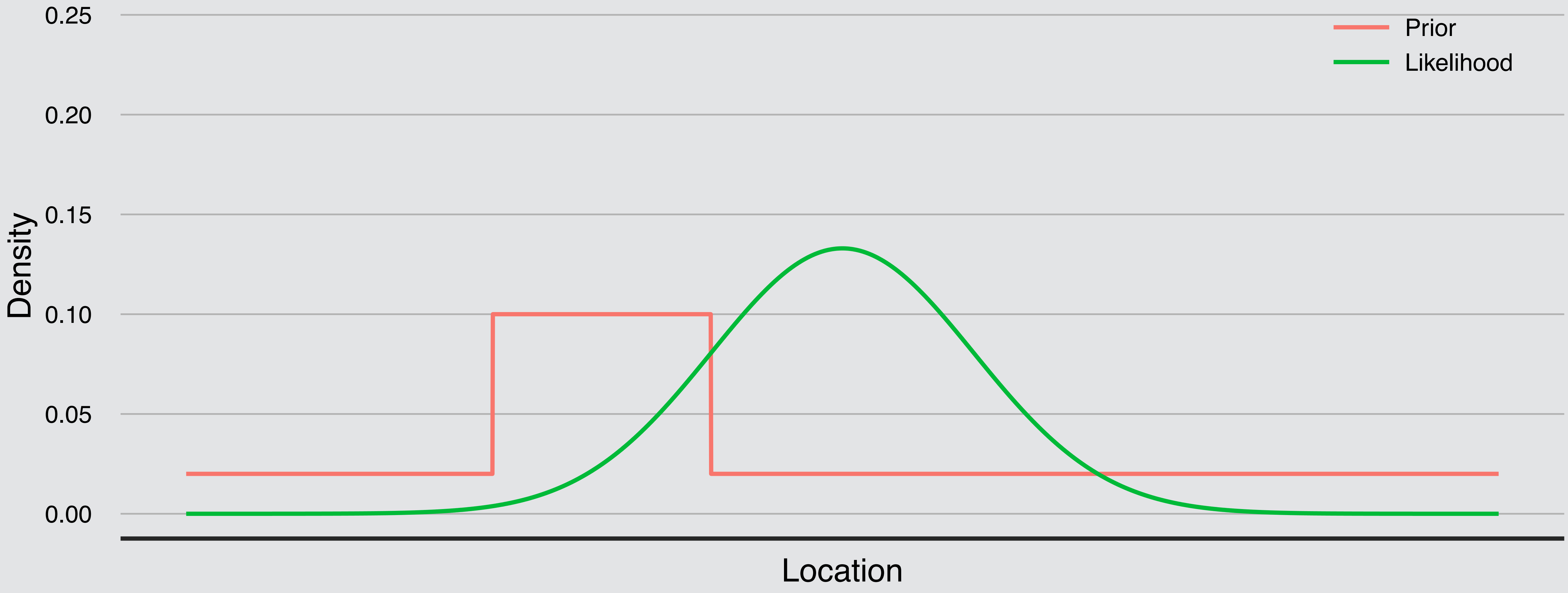
Incorporate domain knowledge: “I’m on a road”



Incorporate domain knowledge: “I’m on a road”



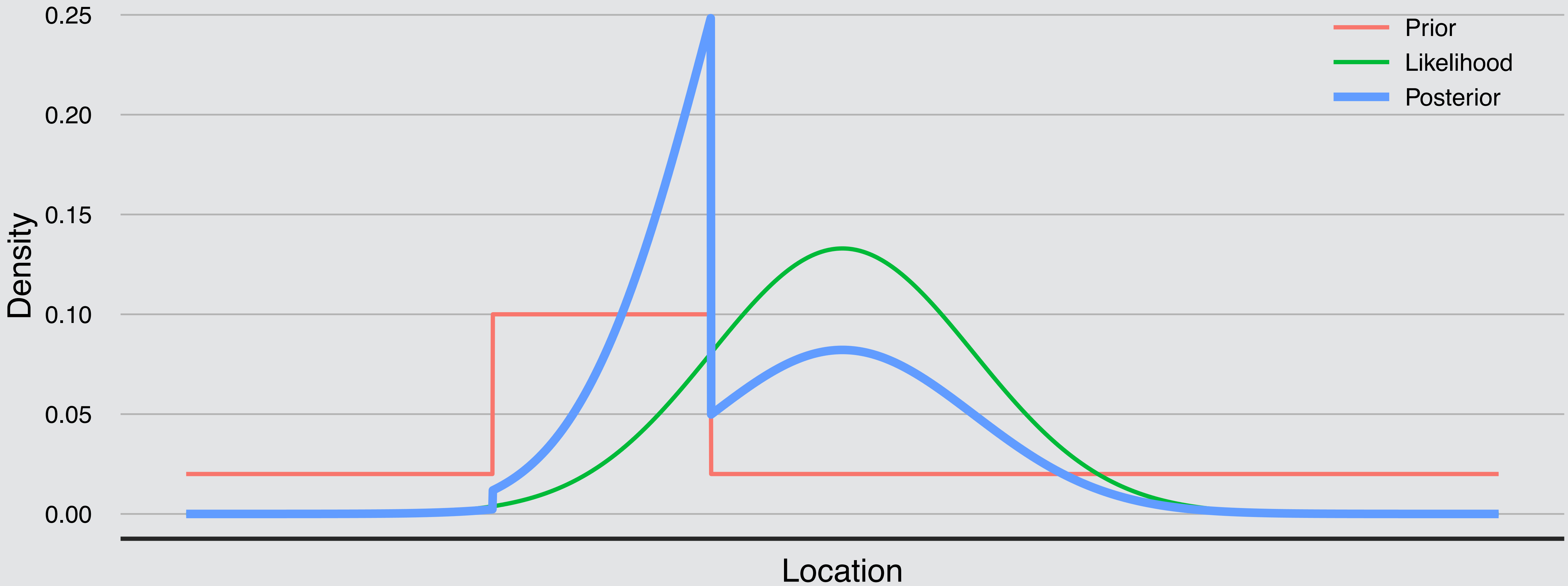
Incorporate domain knowledge: “I’m on a road”



posterior likelihood prior

$$\Pr[H|E] = \frac{\Pr[E|H]\,\Pr[H]}{\Pr[E]}$$

Incorporate domain knowledge: "I'm on a road"



posterior

likelihood

prior

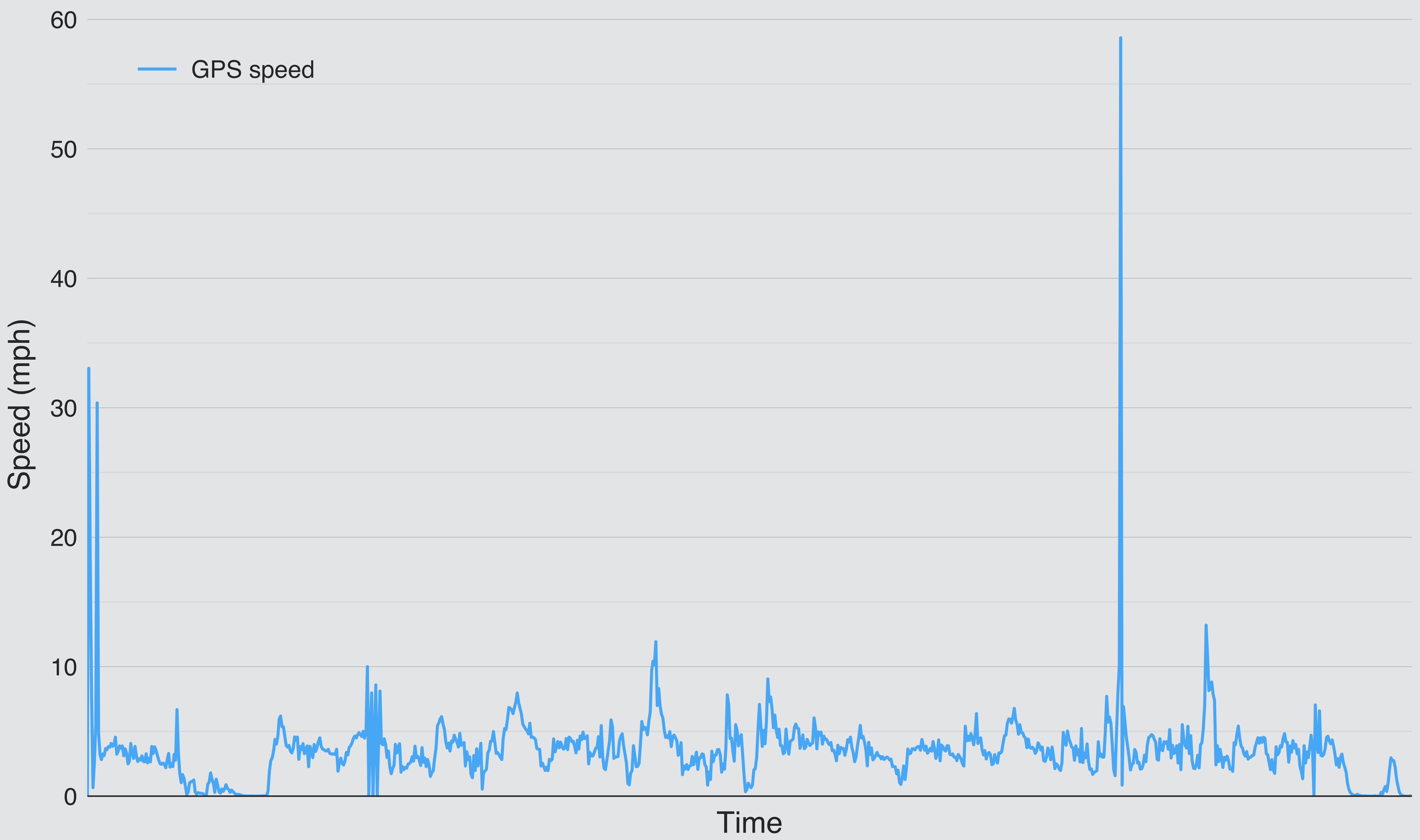
$$\Pr[H|E] = \frac{\Pr[E|H]\,\Pr[H]}{\Pr[E]}$$

Case studies

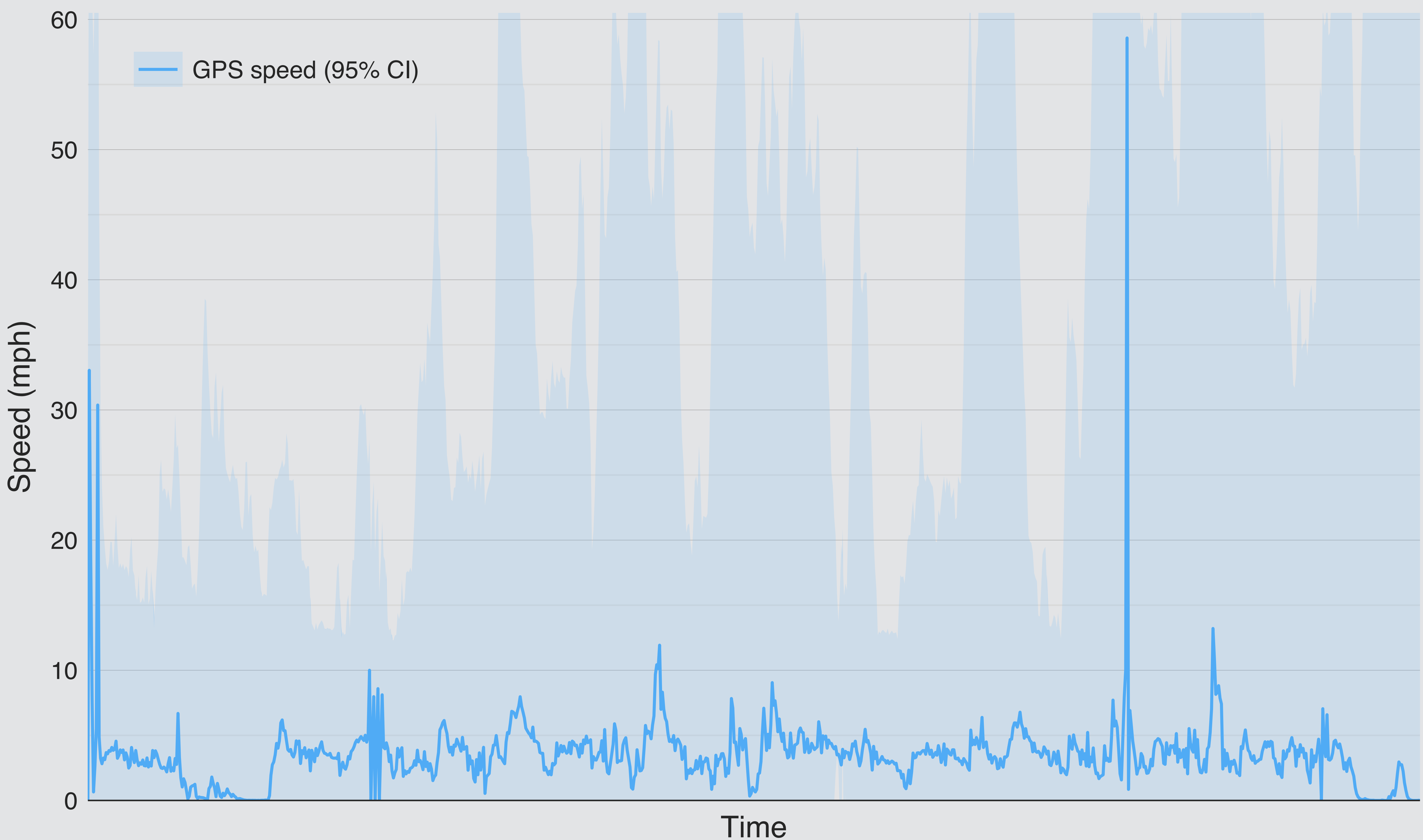
Smartphone GPS sensors

Noisy Game of Life (see the paper)

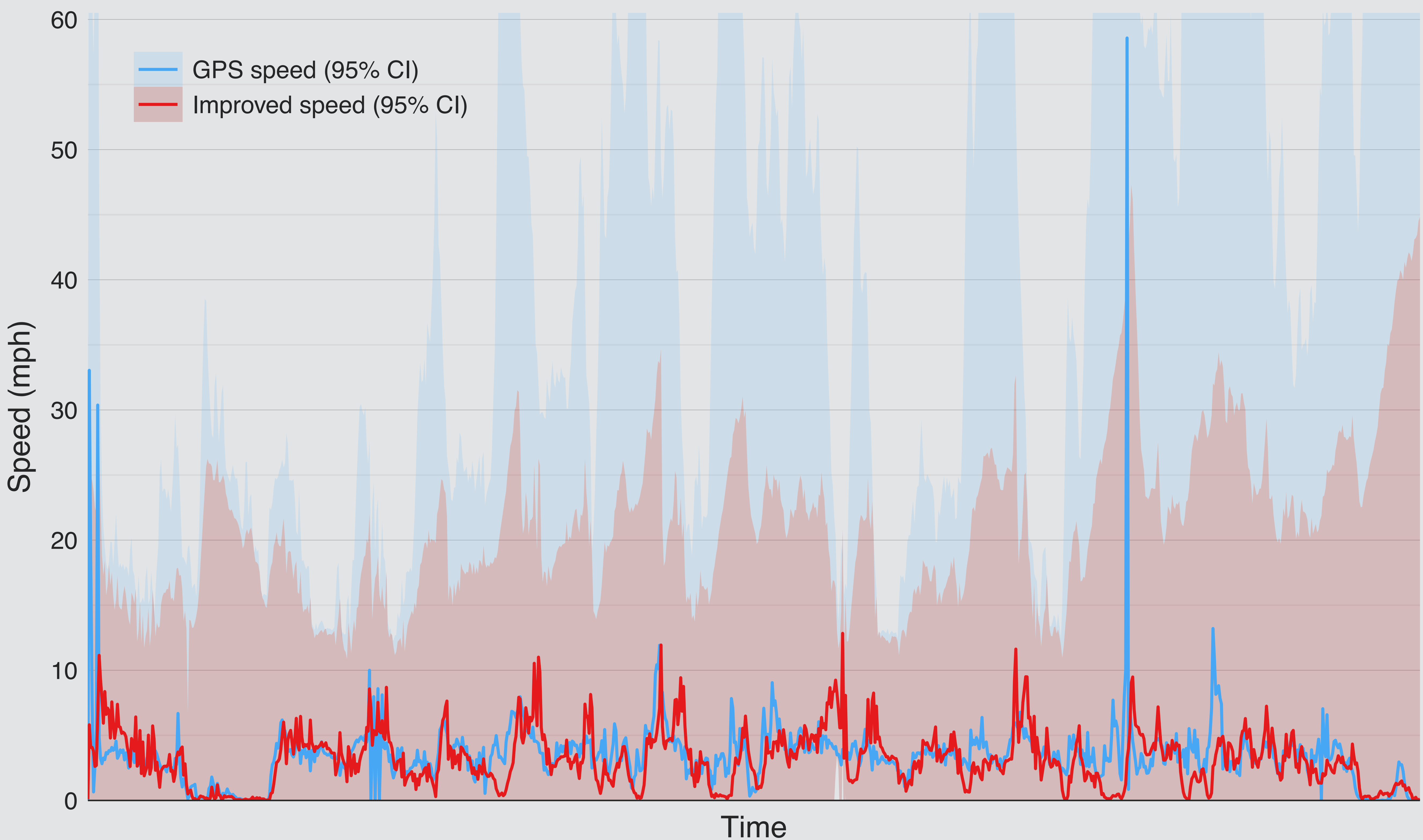
Neural networks/approximate computing



GPS speed
Speed (mph)
60
50
40
30
20
10
0
Time



GPS speed (95% CI)
Speed (mph)
Time
0
10
20
30
40
50
60



GPS speed (95% CI)
Improved speed (95% CI)
Speed (mph)
Time
0
10
20
30
40
50
60

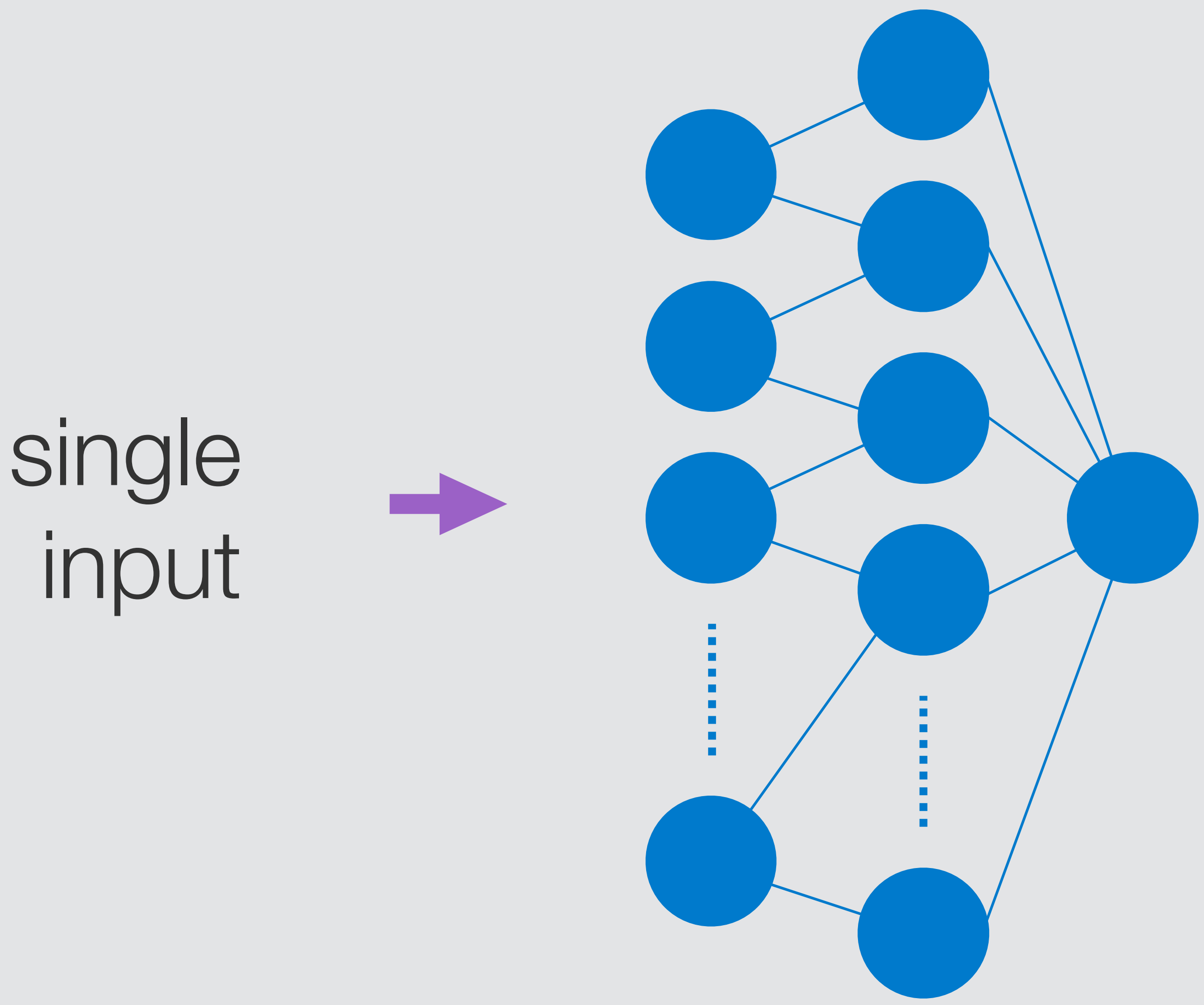
What is the gradient at pixel *p*?

`Sobel(p)` 3.4% average error

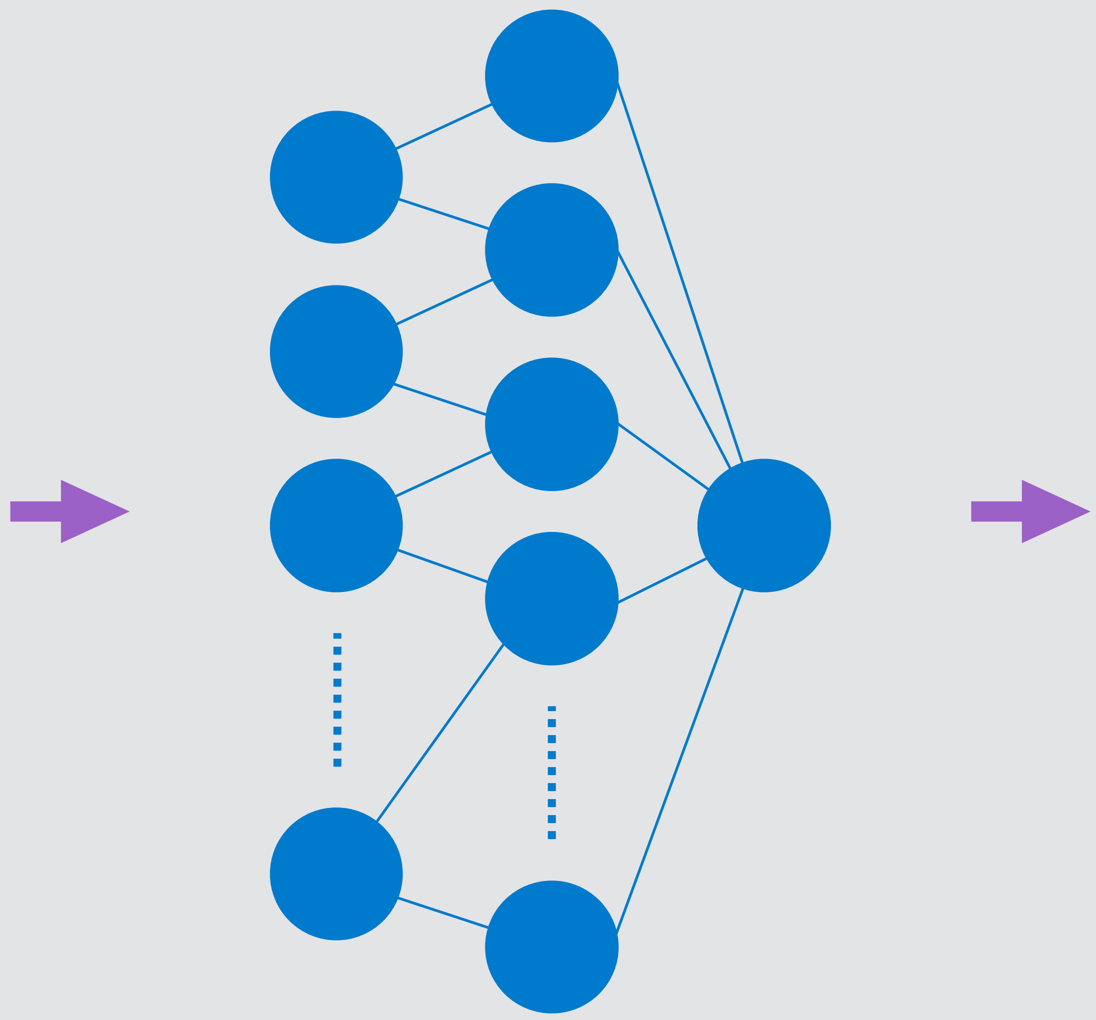
Is there an edge at pixel *p*?

`Sobel(p) > 0.1` 36% false positives!

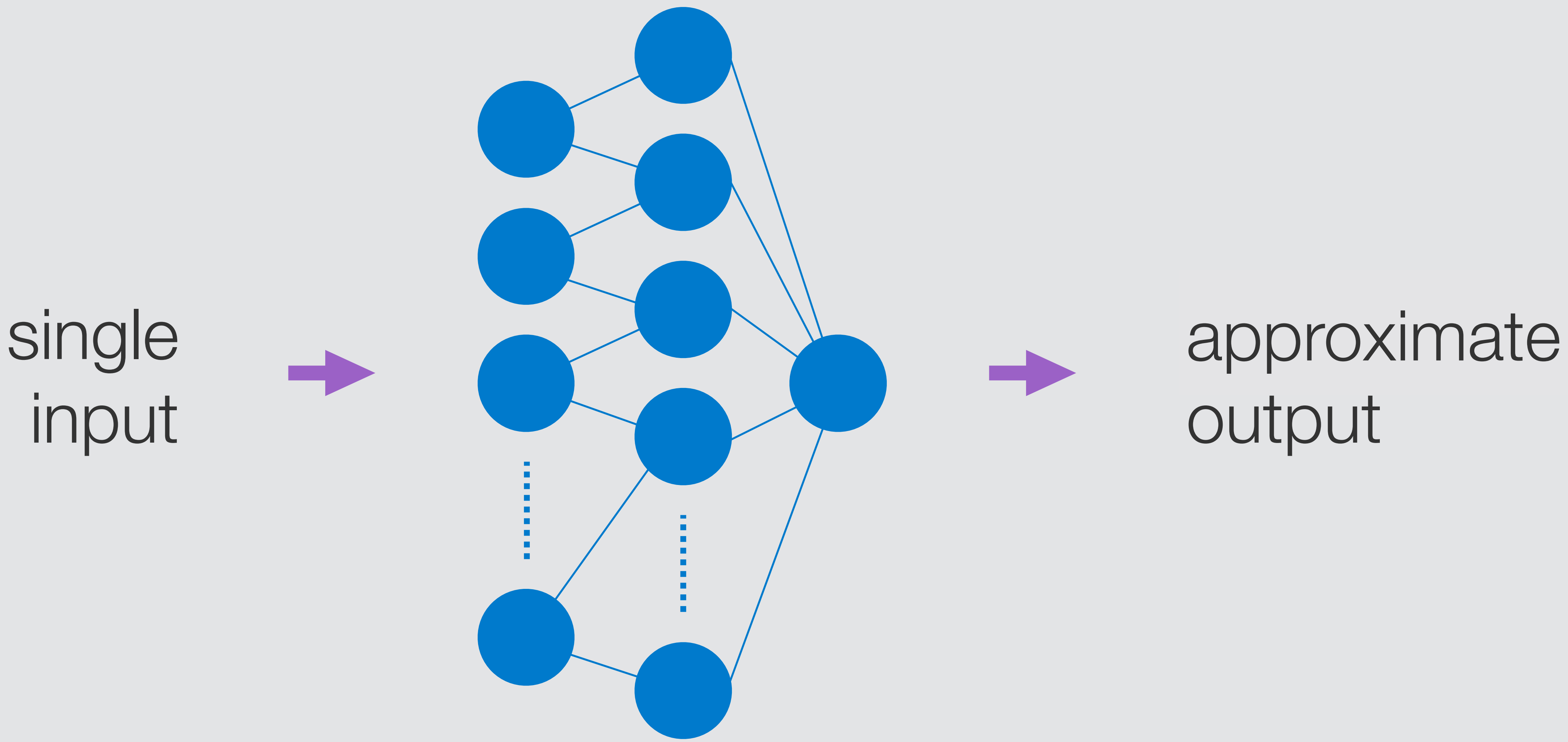
single
input


single
input

single
input



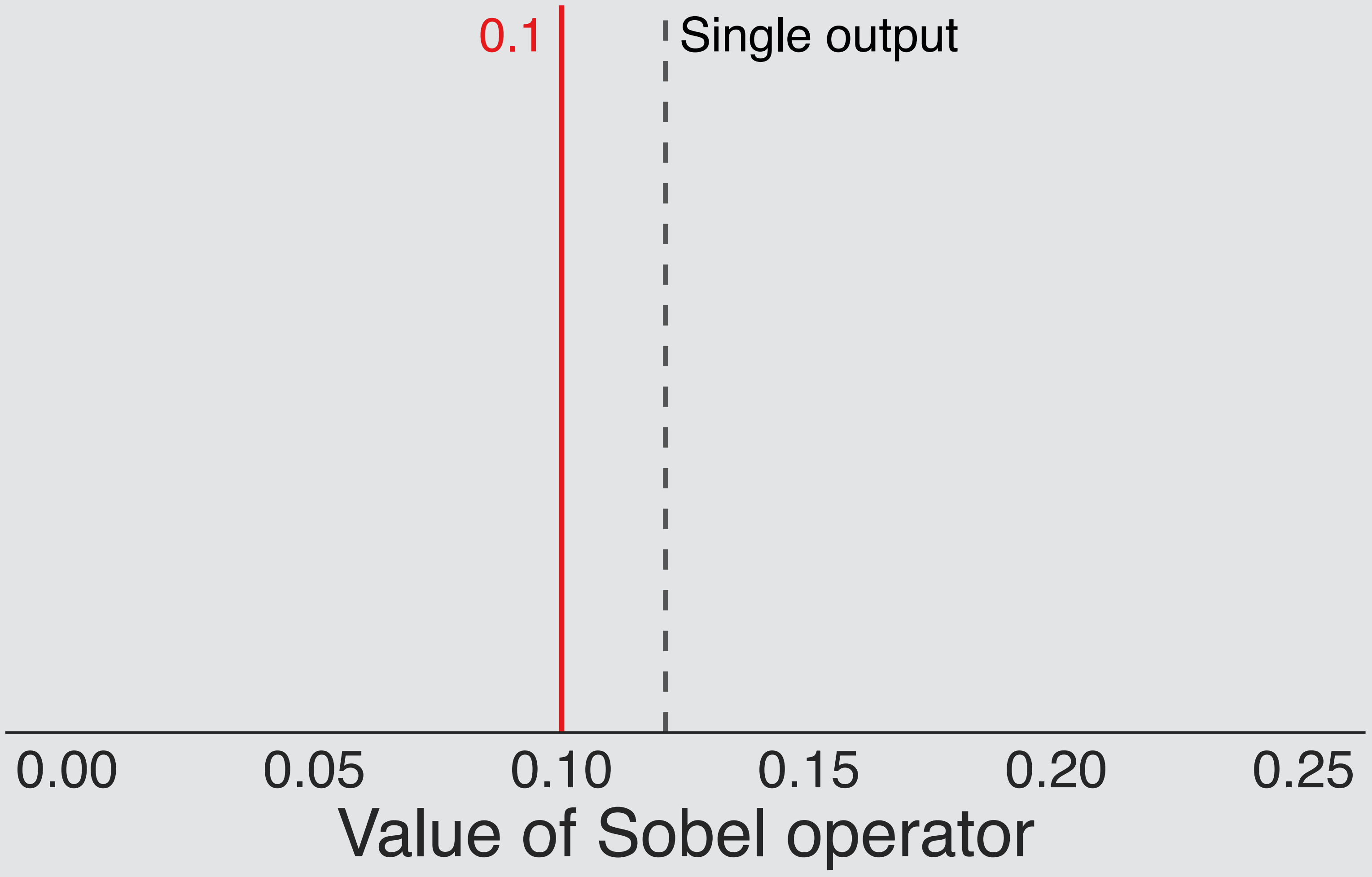
single
output


single
input
approximate
output

Is there an edge at pixel *p*?

`Sobel(p) > 0.1`

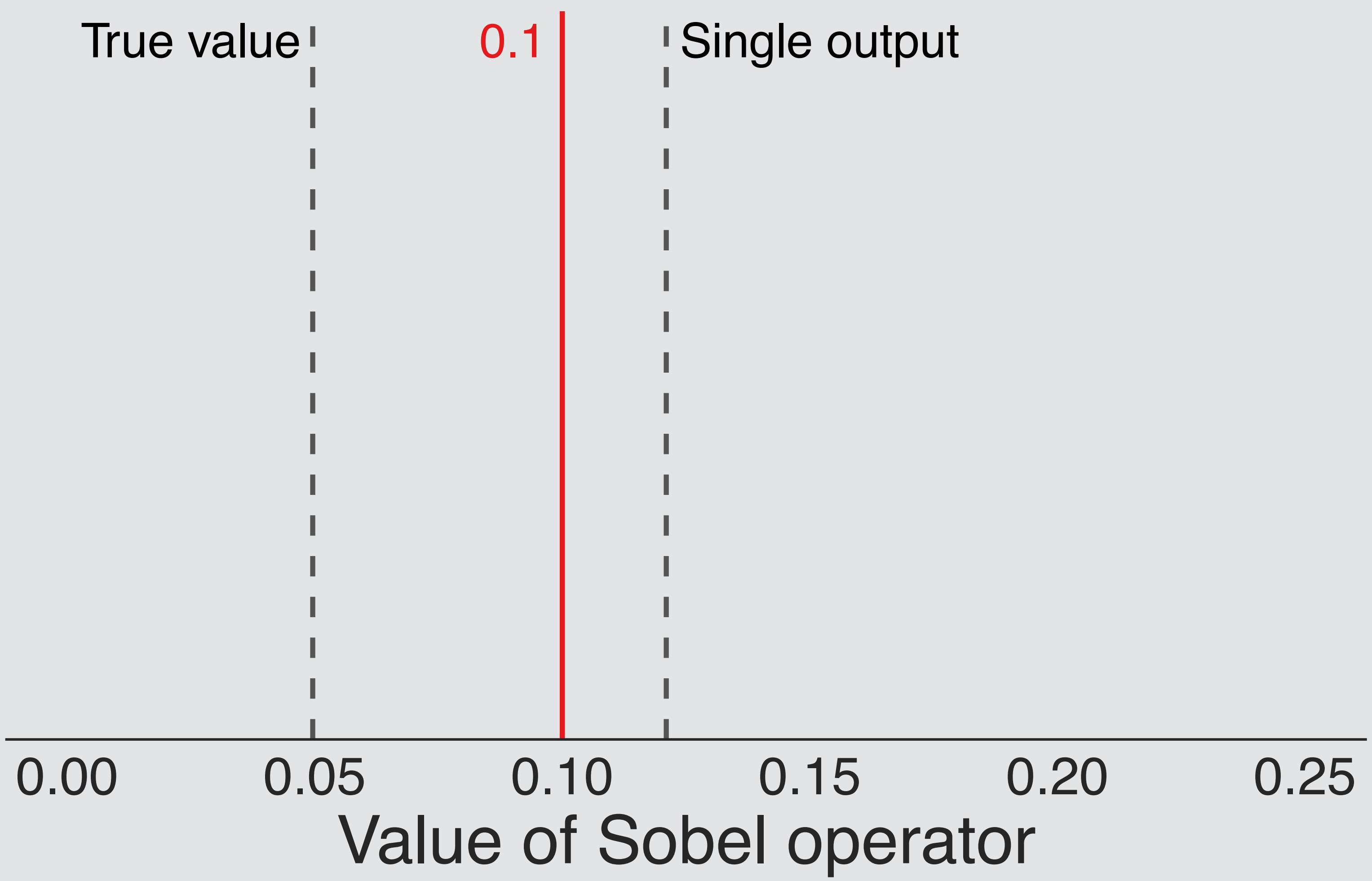
36% false positives!



Is there an edge at pixel *p*?

`Sobel(p) > 0.1`

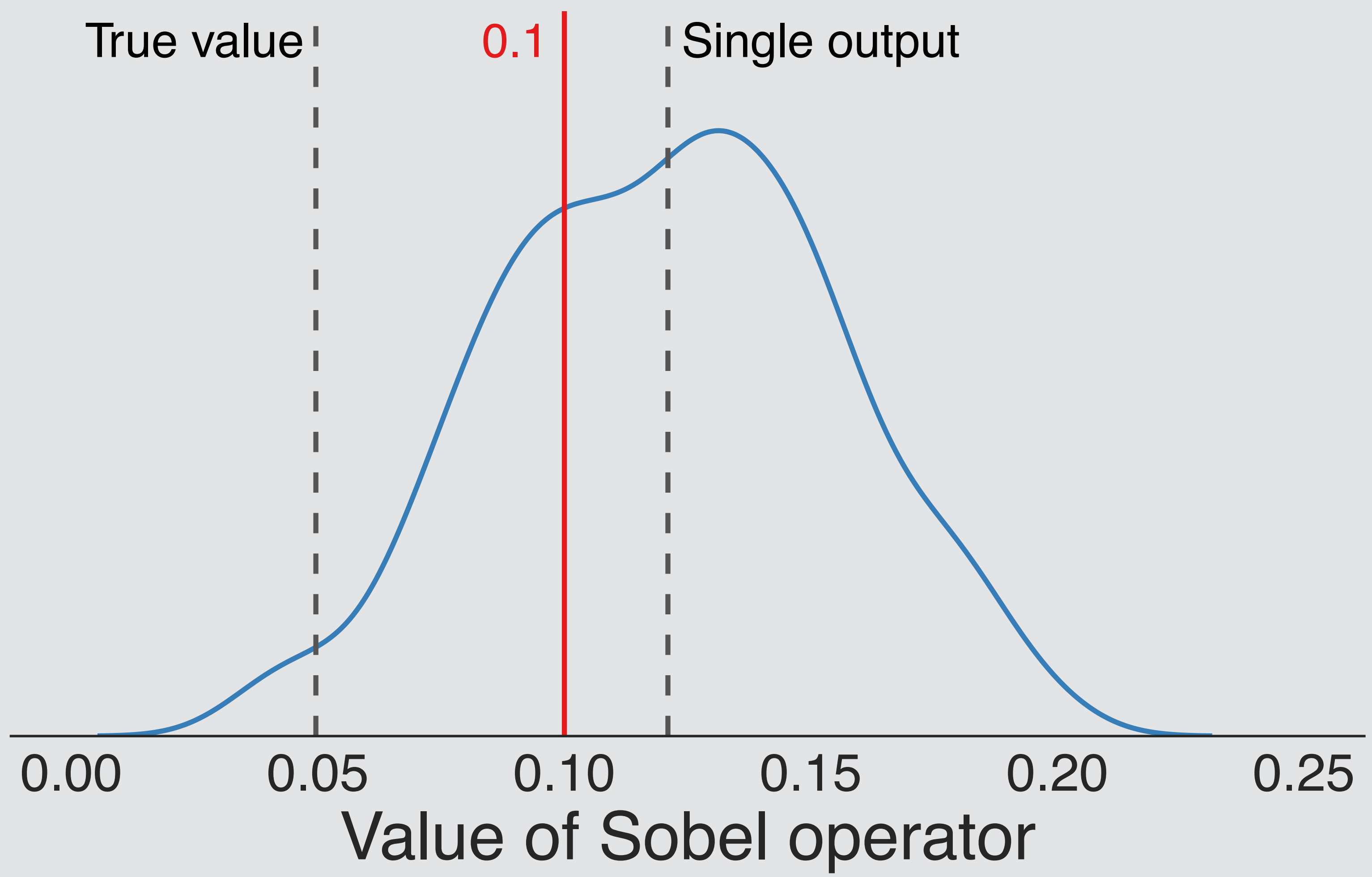
36% false positives!



Is there an edge at pixel *p*?

`Sobel(p) > 0.1`

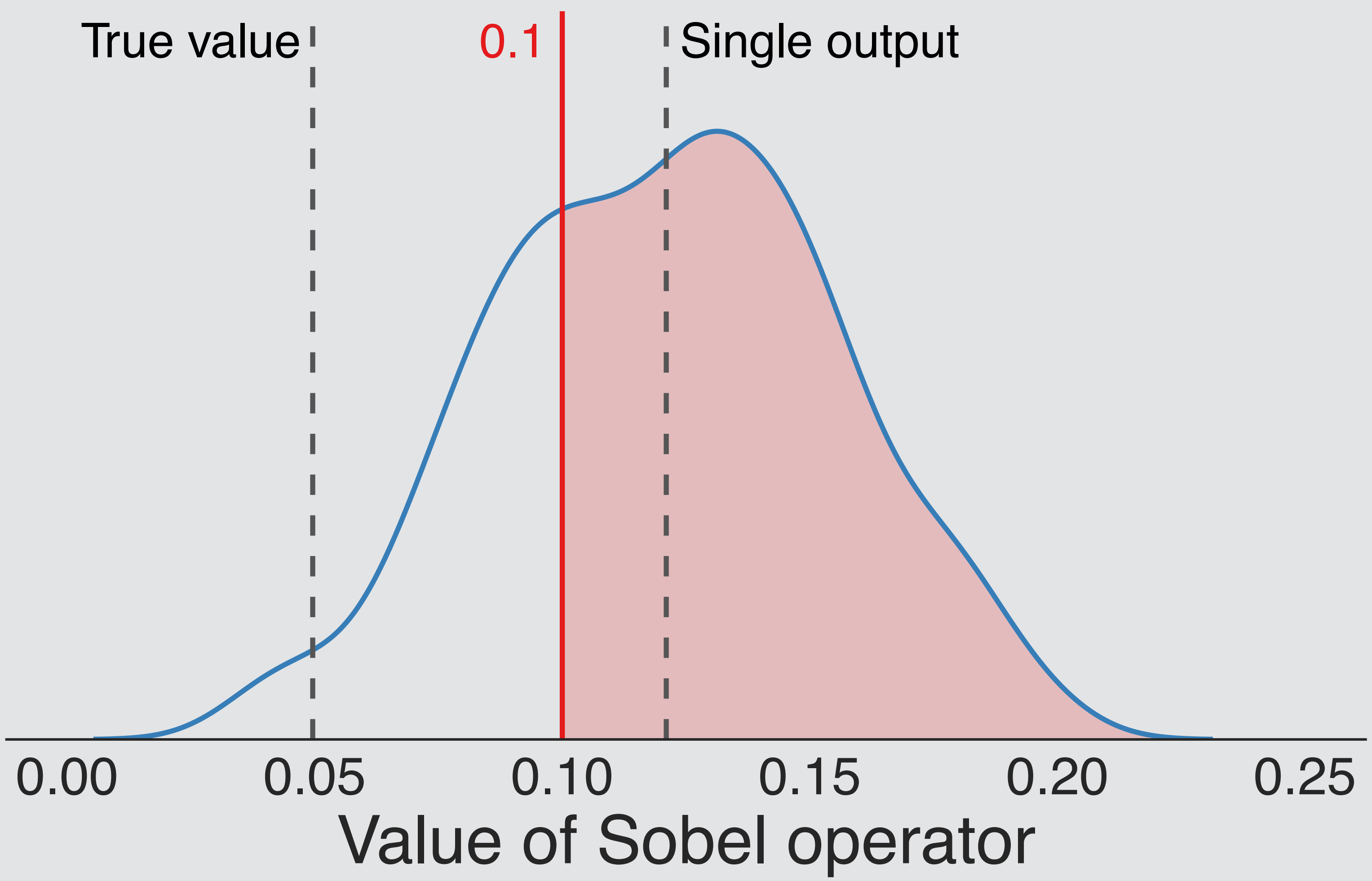
36% false positives!



Is there an edge at pixel p?

`Sobel(p) > 0.1`

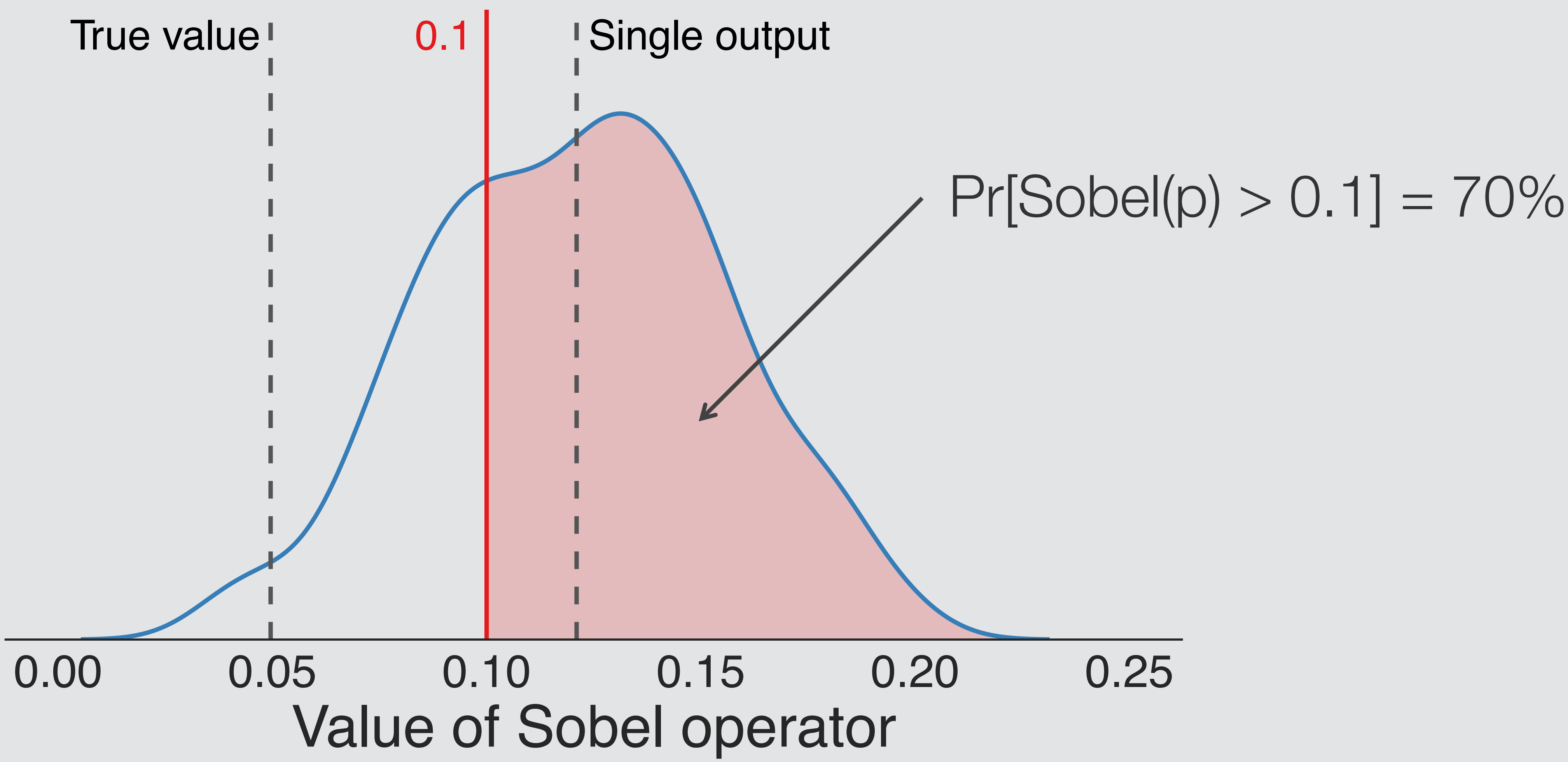
36% false positives!

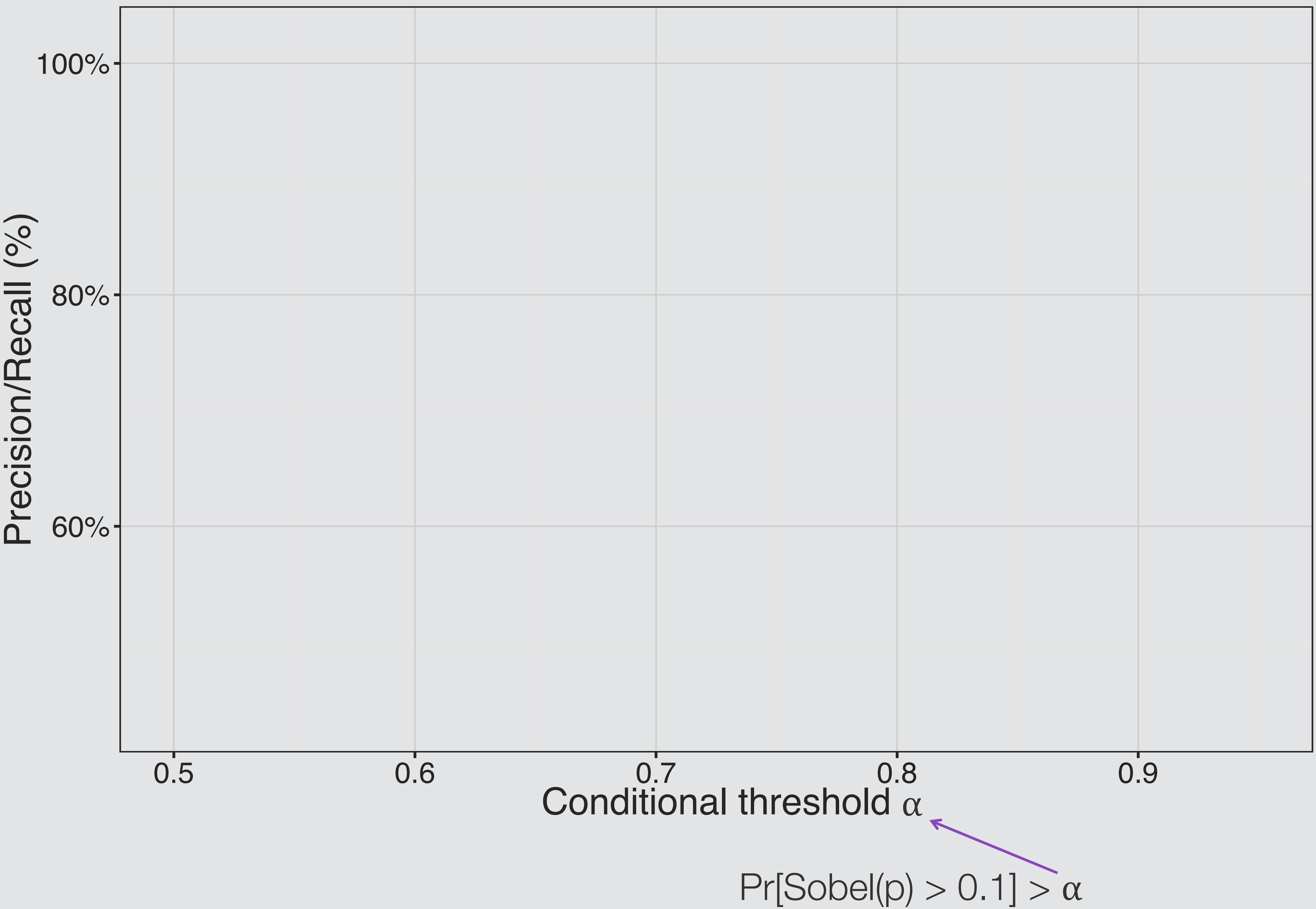


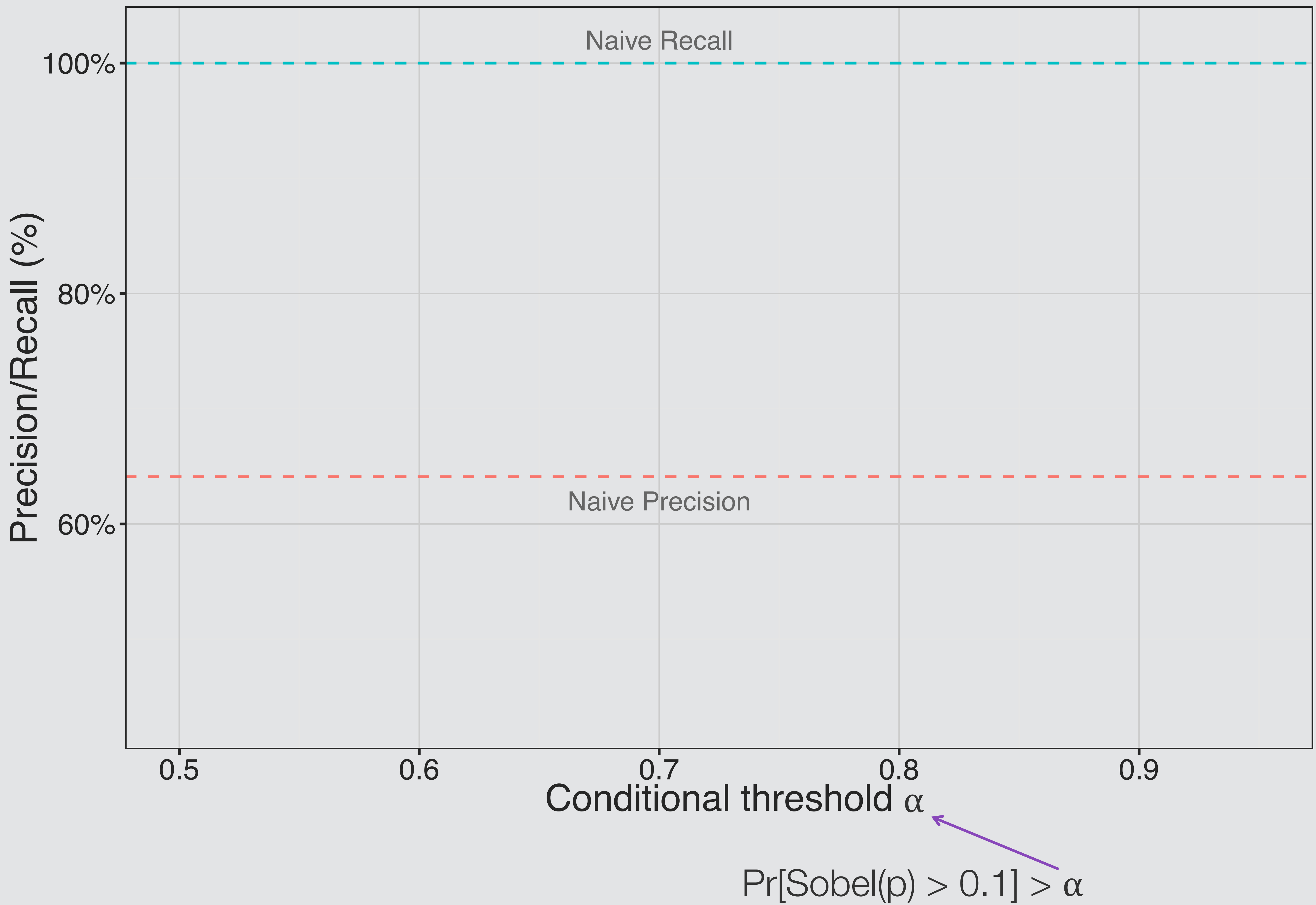
Is there an edge at pixel *p*?

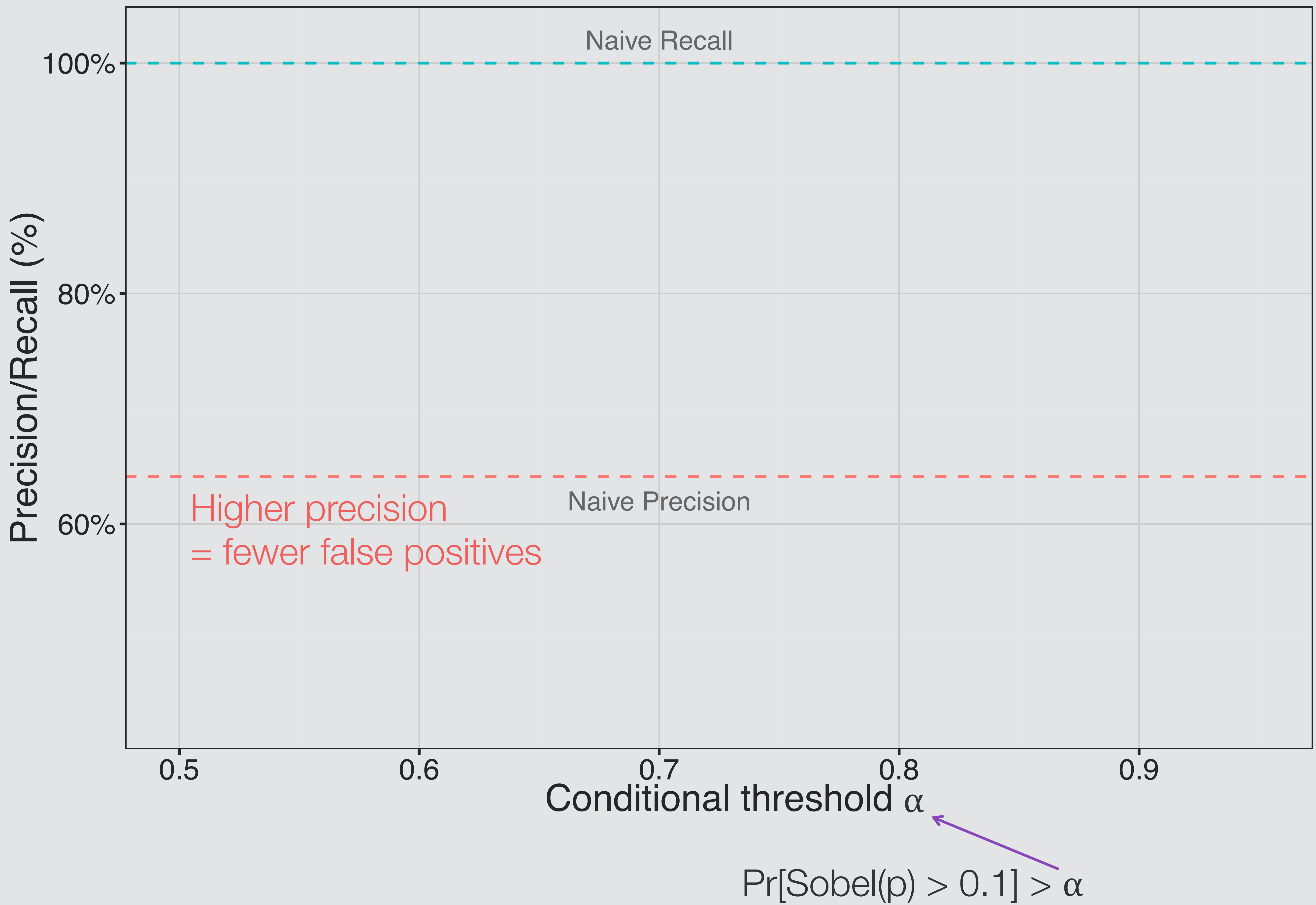
`Sobel(p) > 0.1`

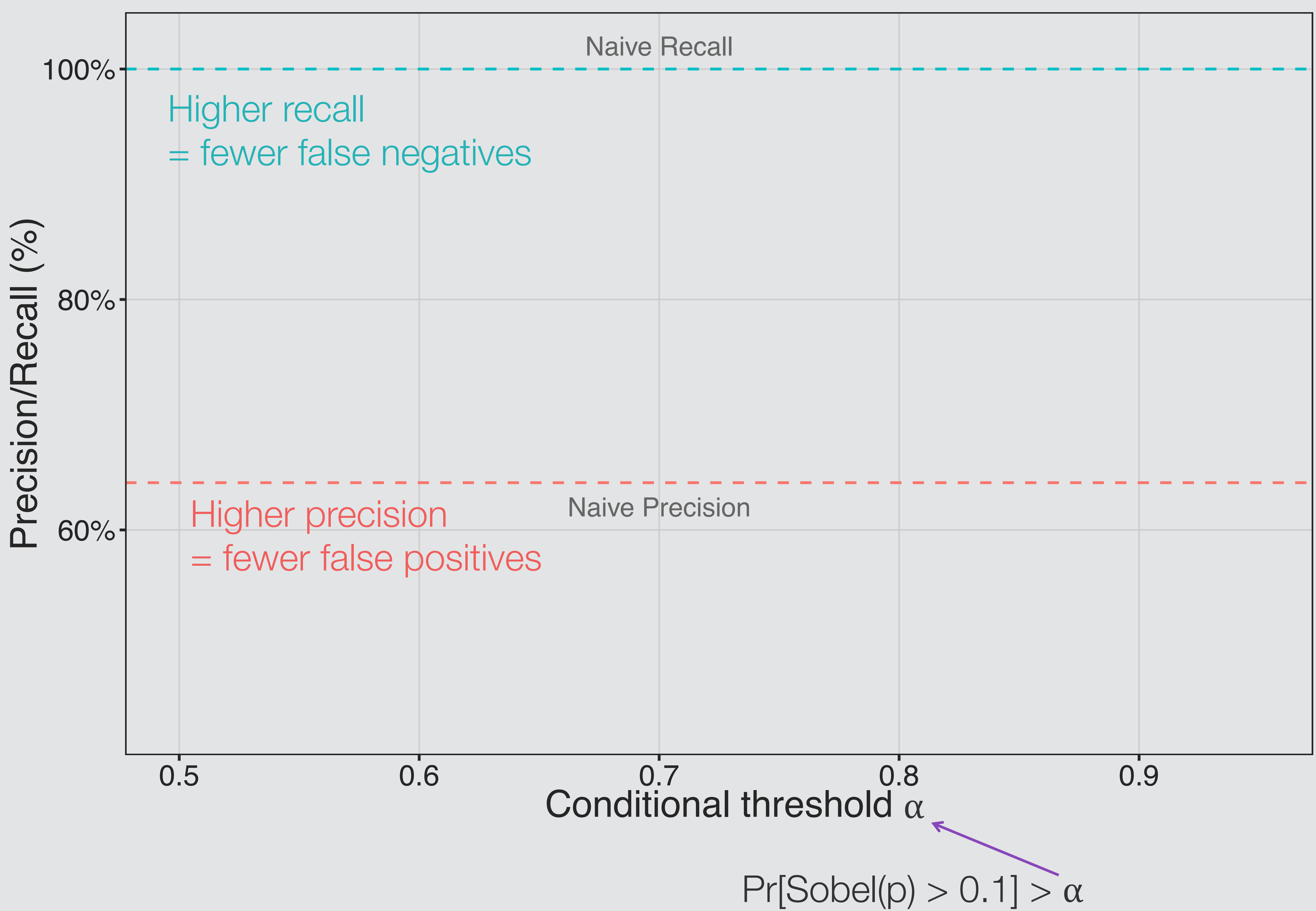
36% false positives!



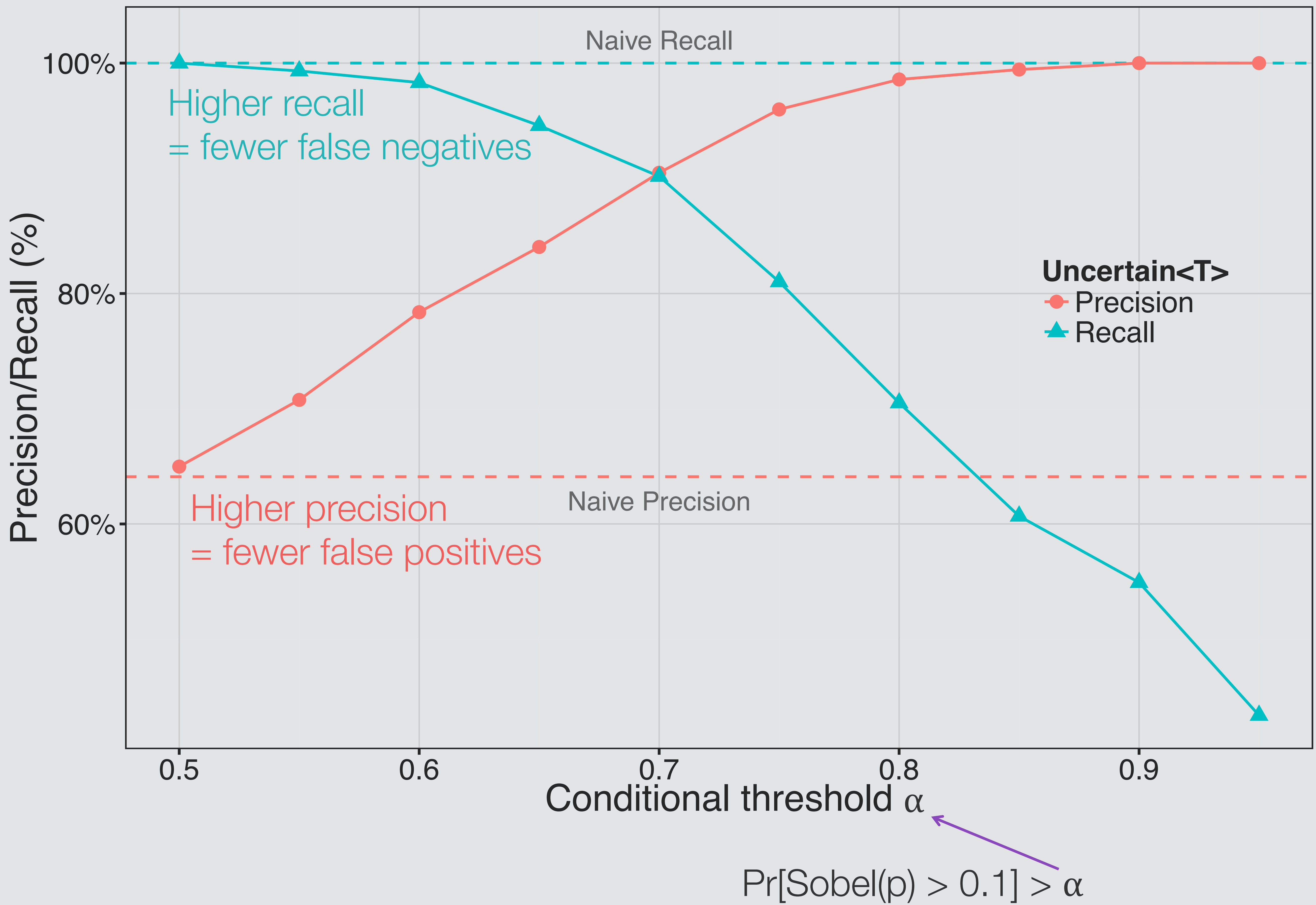

100%
80%
60%
Precision/Recall (%)
0.5
0.6
0.7
0.8
0.9
Conditional threshold α
Pr[Sobel(p) > 0.1] > α


Naive Recall
Naive Precision
100%
80%
60%
Precision/Recall (%)
0.5
0.6
0.7
0.8
0.9
Conditional threshold α
Pr[Sobel(p) > 0.1] > α


Naive Recall
100%
80%
60%
Precision/Recall (%)
Higher precision
= fewer false positives
Naive Precision
0.5
0.6
0.7
0.8
0.9
Conditional threshold α
Pr[Sobel(p) > 0.1] > α



Naive Recall
100%
Higher recall
= fewer false negatives
Precision/Recall (%)
80%
60%
Naive Precision
Higher precision
= fewer false positives
0.5
0.6
0.7
0.8
0.9
Conditional threshold α
Pr[Sobel(p) > 0.1] > α


Naive Recall
Higher recall
= fewer false negatives
Higher precision
= fewer false positives
Naive Precision
Uncertain<T>
Precision
Recall
Precision/Recall (%)
100%
80%
60%
0.5
0.6
0.7
0.8
0.9
Conditional threshold α
Pr[Sobel(p) > 0.1] > α

Uncertain<T> is an uncertain type abstraction.

It encourages non-expert developers to explicitly reason about uncertainty.

Uncertain<T> is an uncertain type abstraction.

It encourages non-expert developers to explicitly reason about uncertainty.

Thank you!