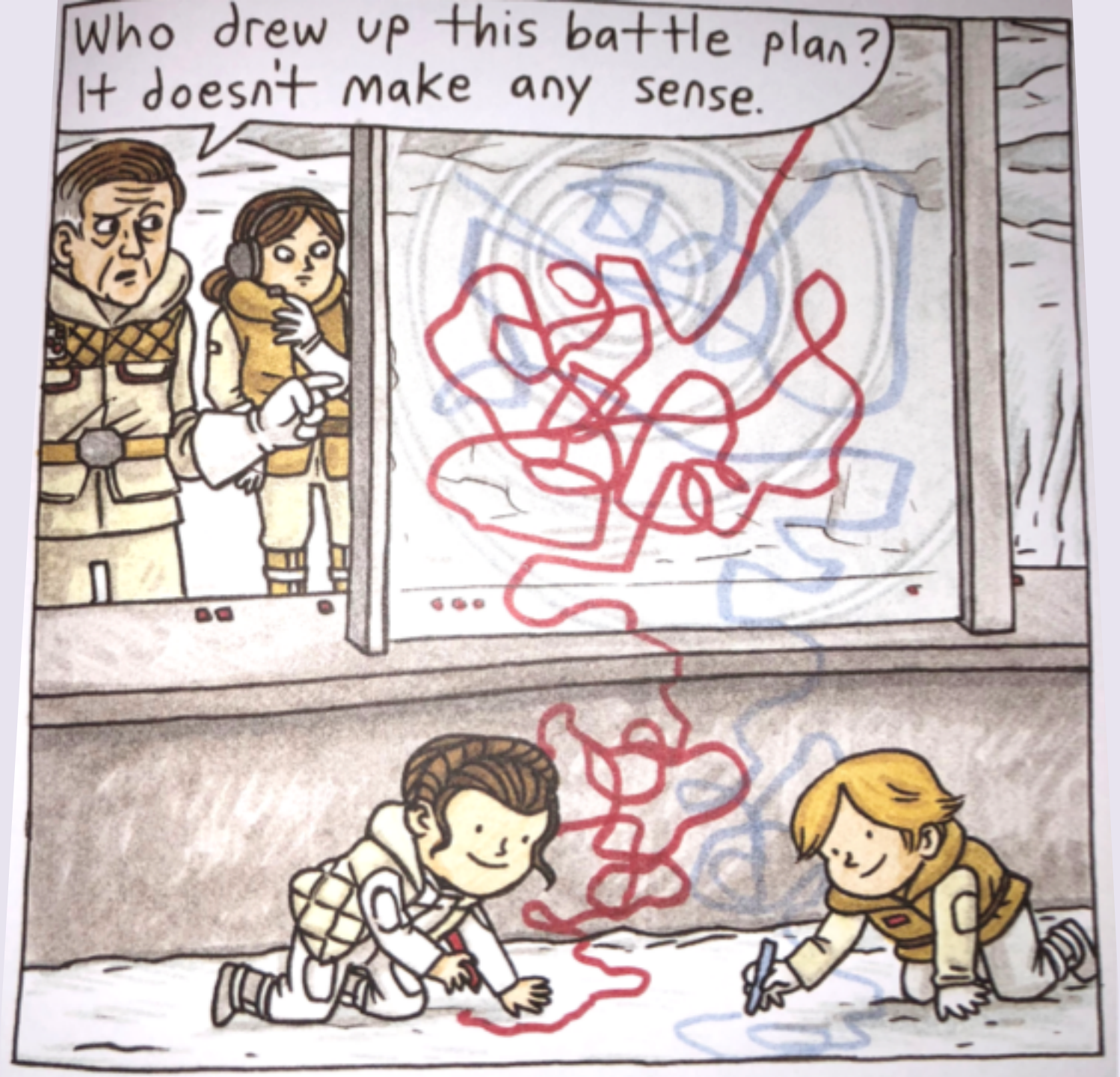
Artificial Intelligence in

@ray_deck


E = mc2
ax + 2x = b - 4c
(a + 2)x = b - 4c
π = 3,14159265


Who drew up this battle plan? It doesn't make any sense.

Statistical Computing

Machine Learning

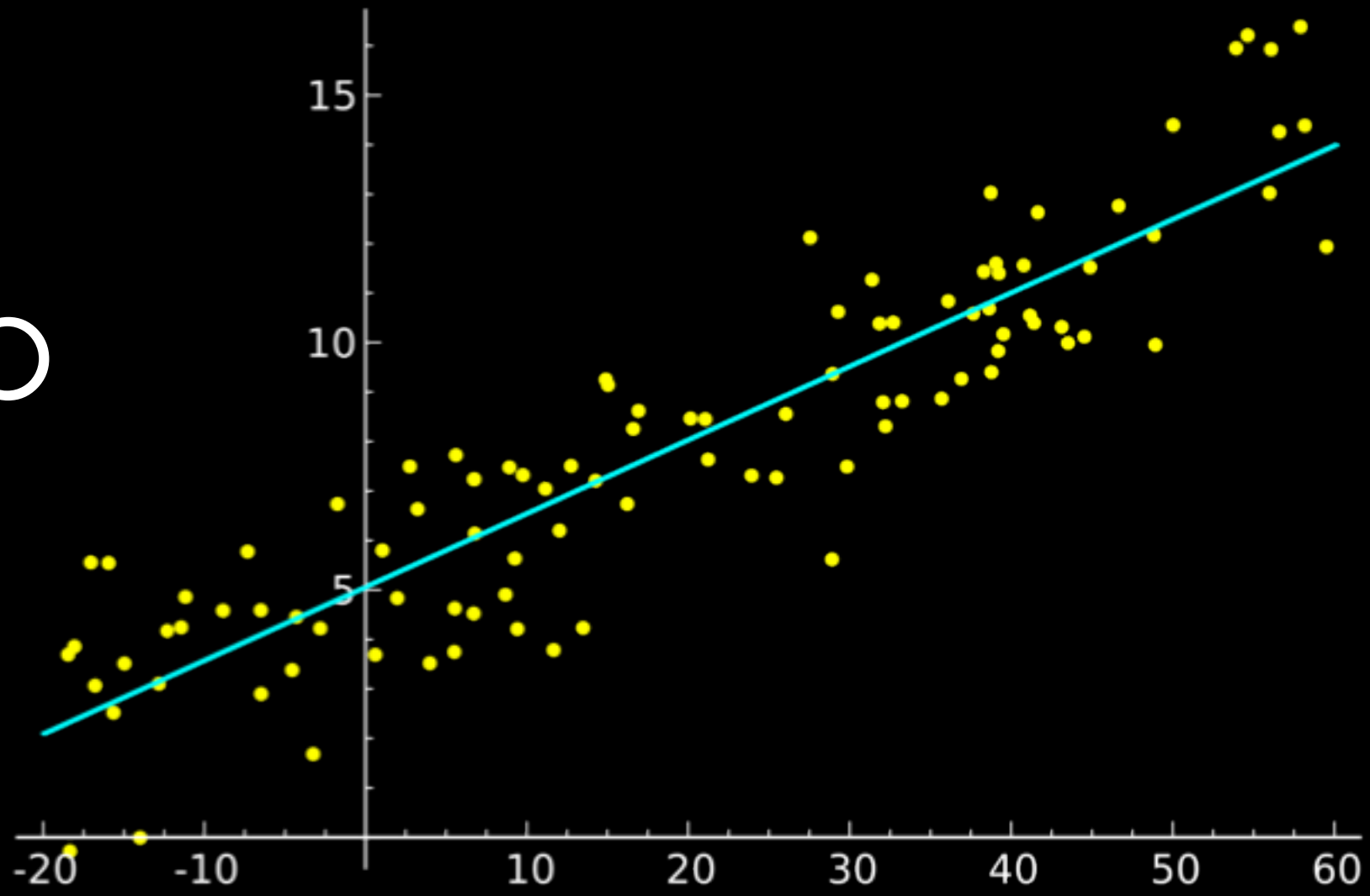
Stochastic Programming

Artificial Intelligence

Statistics

Regression

$$y = mx + b$$



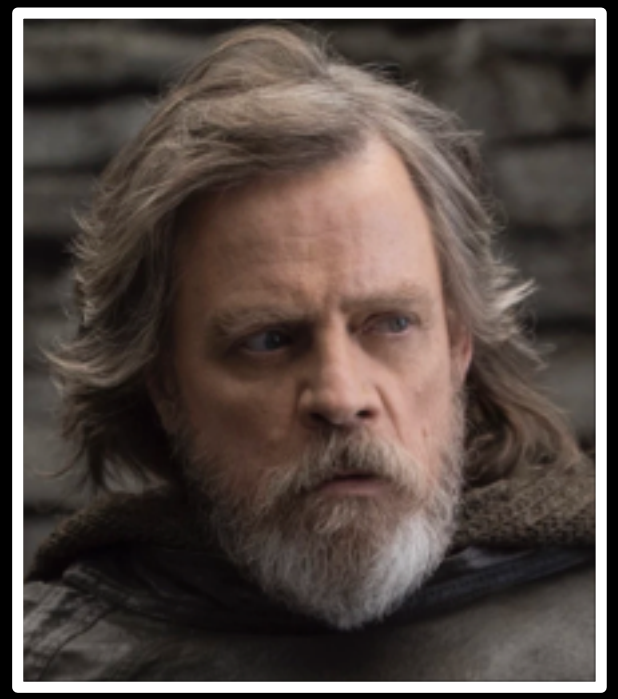
Differentiation

High-Dimensional Numbers

High-Dimensional Vectors

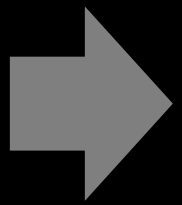
High-Dimensional **Arrays**

0.3200	0.5421	0.5110	0.7389	0.4918	0.9184
0.8660	0.3020	0.4890	0.6987	0.3051	0.4029
0.0722	0.6375	0.7953	0.1546	0.2867	0.9654
0.3608	0.6891	0.8220	0.0024	0.2030	0.1865
0.2222	0.6622	0.5402	0.7650	0.1721	0.3517
0.4795	0.8597	0.4049	0.7070	0.0784	0.6631
0.3131	0.7966	0.4261	0.7823	0.7319	0.0807
0.2659	0.7687	0.7874	0.8179	0.0292	0.4352
0.4201	0.2081	0.2831	0.3404	0.4661	0.6139
0.0548	0.6564	0.0637	0.5374	0.1792	0.4786
0.3202	0.4883	0.9120	0.7424	0.8177	0.8120
0.8694	0.7283	0.8922	0.2756	0.9849	0.4665
0.2098	0.9957	0.5271	0.7638	0.4820	0.9617
0.2385	0.8299	0.3855	0.9500	0.3676	0.2845
0.1923	0.7167	0.6323	0.9345	0.2605	0.0331
0.5322	0.1652	0.4930	0.2563	0.5717	0.1103
0.7212	0.4764	0.9196	0.6196	0.4345	0.6993
0.8939	0.5753	0.7510	0.1955	0.7078	0.9869
0.9766	0.7216	0.0188	0.8569	0.8033	0.7709
0.3885	0.6761	0.4612	0.8000	0.8435	0.8812



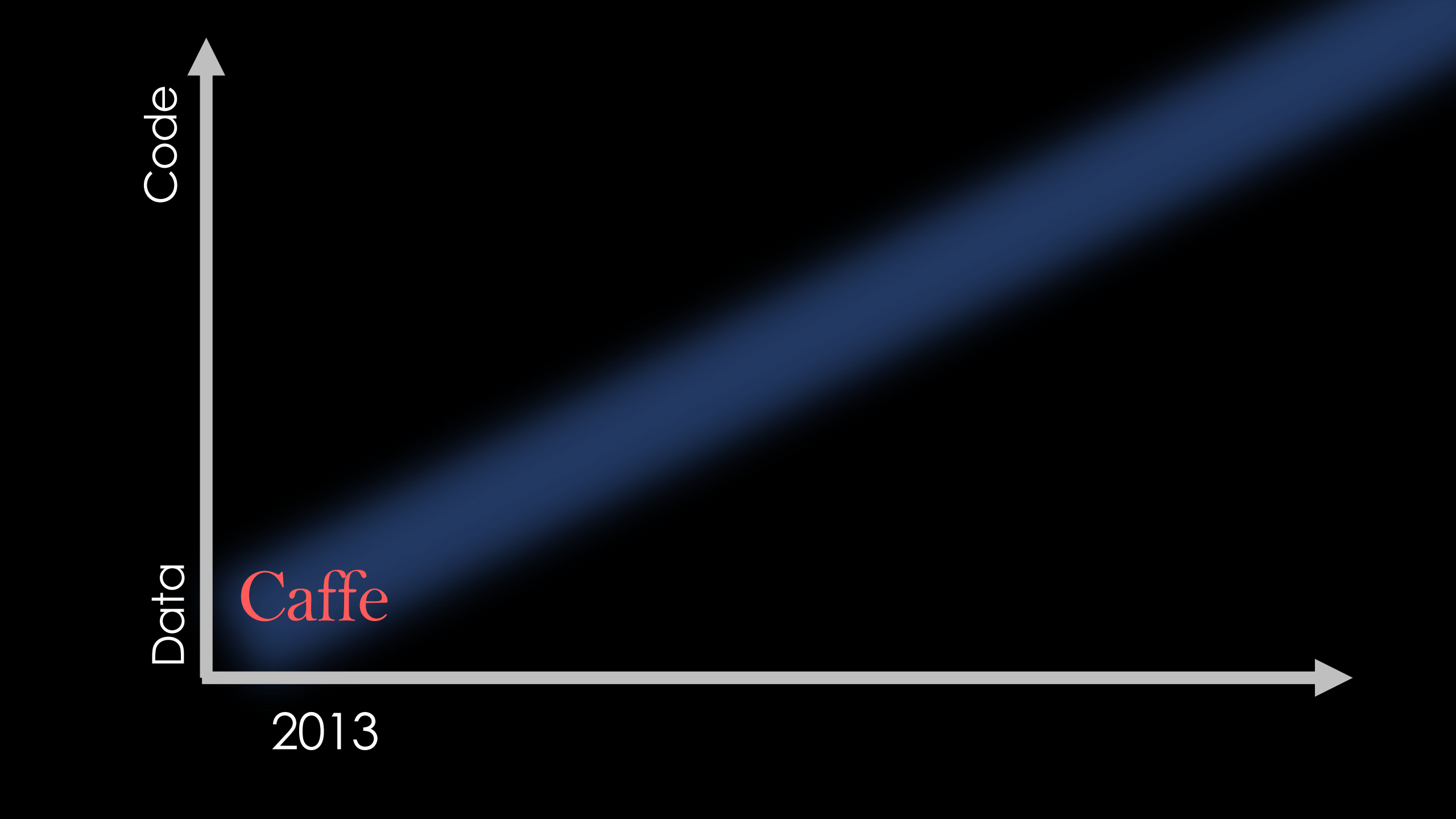
		0.3200	0.5421	0.5110	0.7389	0.4918	0.9184	
		0.8660	0.3020	0.4890	0.6987	0.3051	0.4029	
		0.0722	0.6375	0.7953	0.1546	0.2867	0.9654	
		0.3608	0.6891	0.8220	0.0024	0.2030	0.1865	
		0.6227	0.2532	0.9418	0.8587	0.0778	0.5965	
	0.2222	0.6622	0.5402	0.7650	0.1721	0.3517	0.3071	
	0.4795	0.8597	0.4049	0.7070	0.0784	0.6631	0.1203	
	0.3131	0.7966	0.4261	0.7823	0.7319	0.0807	0.7669	
	0.2659	0.7687	0.7874	0.8179	0.0292	0.4352	0.8521	
	0.4201	0.2091	0.2931	0.3404	0.4661	0.6139	0.7875	
	0.3099	0.1022	0.0893	0.3600	0.8400	0.5494	0.9331	
0.0548	0.6564	0.0637	0.5374	0.1792	0.4786	0.0082		
0.3202	0.4883	0.9120	0.7424	0.8177	0.8120	0.8012		
0.8694	0.7283	0.8922	0.2756	0.9849	0.4665	0.5517		
0.2098	0.9957	0.5271	0.7638	0.4820	0.9617	0.2544		
0.2385	0.8299	0.3855	0.9500	0.3676	0.2845	0.5787		
0.1923	0.7167	0.6323	0.9345	0.2605	0.0331			
0.5322	0.1652	0.4930	0.2563	0.5717	0.1103			
0.7212	0.4764	0.9196	0.6196	0.4345	0.6993			
0.8939	0.5753	0.7510	0.1955	0.7078	0.9869			
0.9766	0.7216	0.0188	0.8569	0.8033	0.7709			
0.3885	0.6761	0.4612	0.8000	0.8435	0.8812			

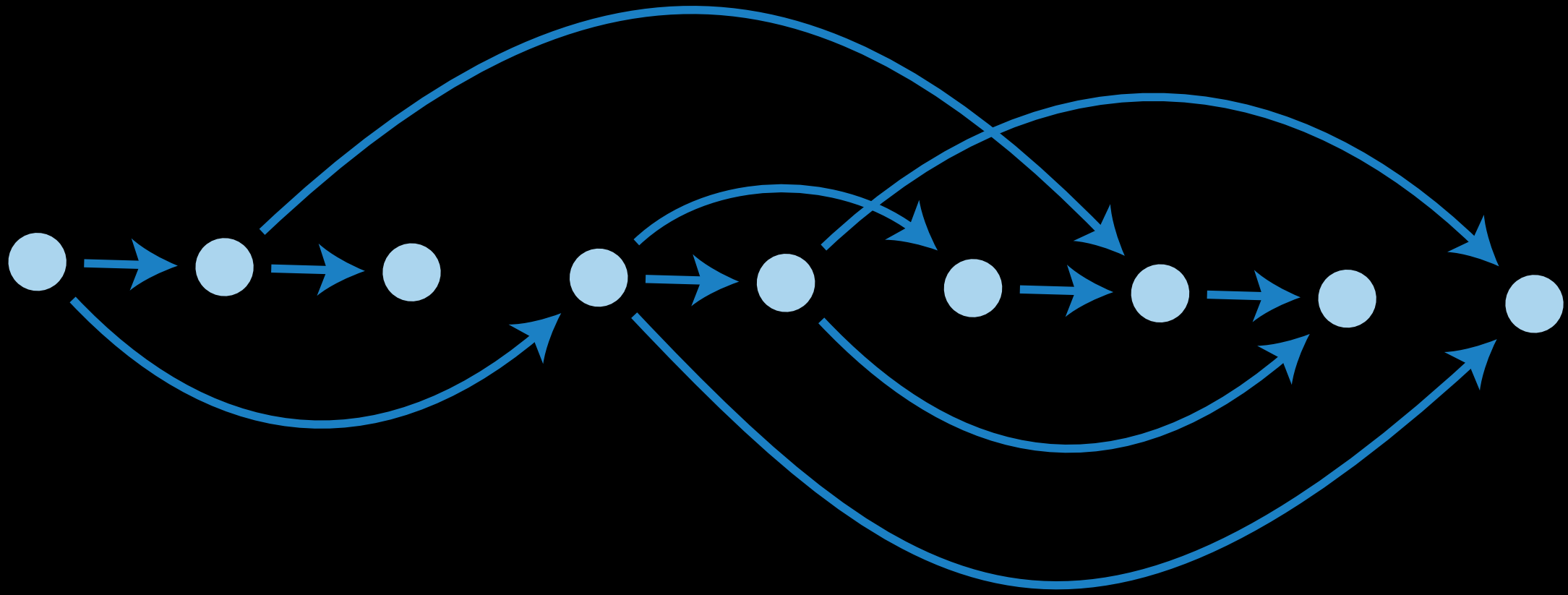
0.1489
0.0026
0.0128
0.0731
0.1212
0.0933
0.0110
0.0011
0.0432
0.0002
0.0299
0.0054
0.0255
0.0035
0.1043

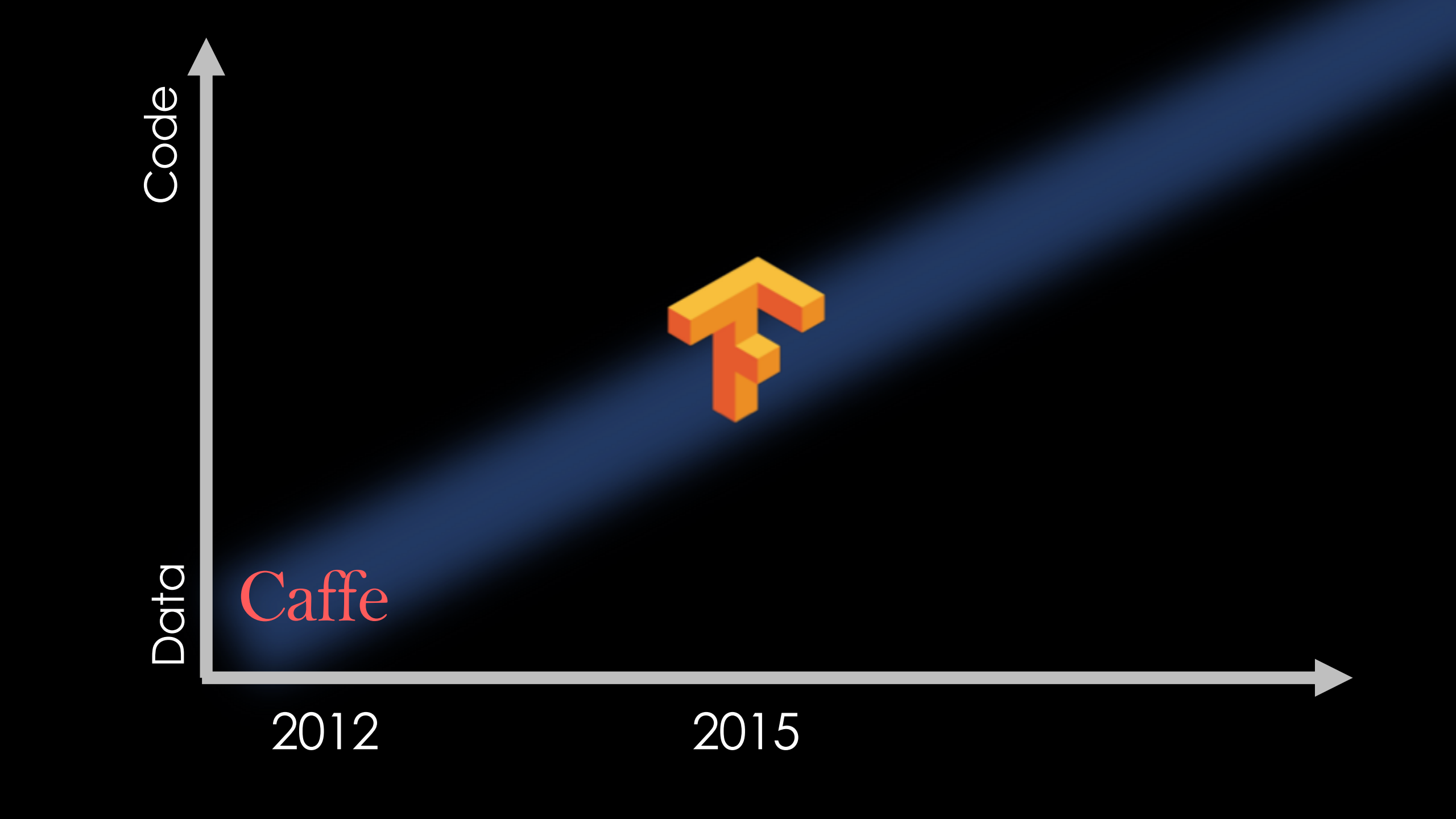


63% Luke Skywalker
24% Jedi Knight
0.1% Chewbacca

Conv2D
MaxPool
ReLU
Sigmoid
TanH
LSTM
GRU


Code
Data
Caffe
2013

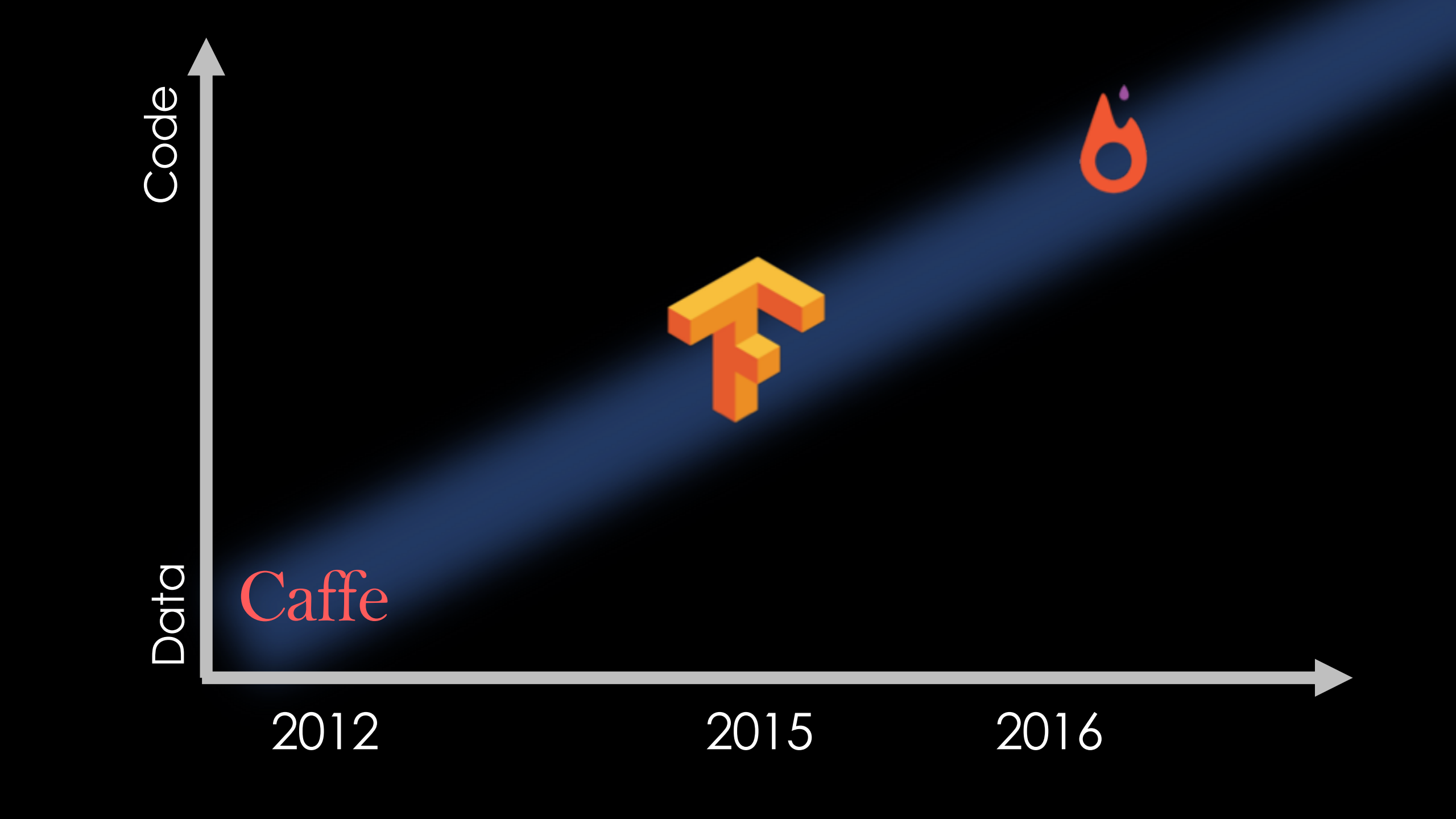



Code
Data
Caffe
2012
2015

Model as Data Structure

Model as Code

Automatic Differentiation


Code
Data
Caffe
2012
2015
2016

$$y = f(x)$$

0.1489
0.0026
0.0128
0.0731
0.1212
0.0933
0.0110
0.0011
0.0432
0.0002
0.0299
0.0054
0.0255
0.0035
0.1043

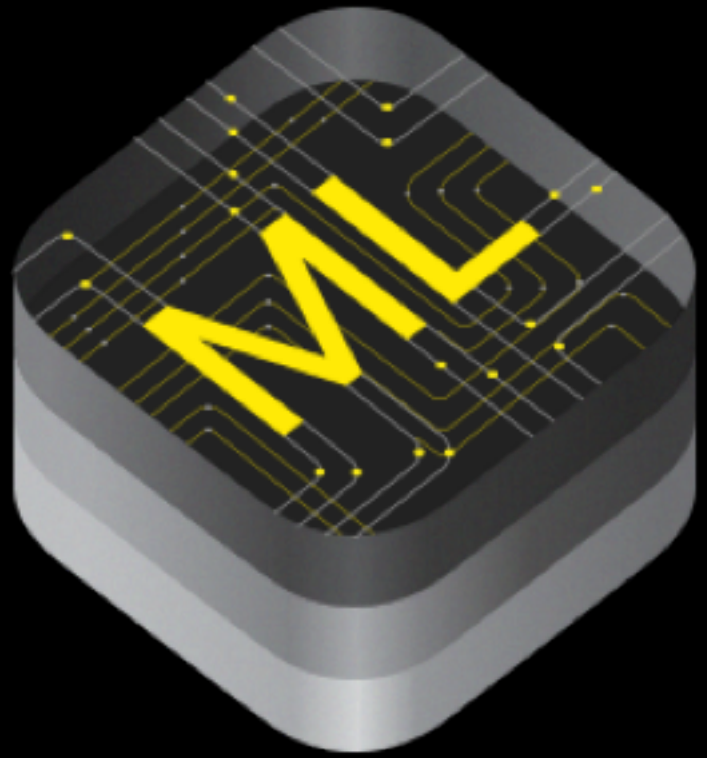
$$= f\left(\begin{matrix}
0.0548 & 0.6564 & 0.0637 & 0.5374 & 0.1792 & 0.4786 \\
0.3202 & 0.4883 & 0.9120 & 0.7424 & 0.8177 & 0.8120 \\
0.8694 & 0.7283 & 0.8922 & 0.2756 & 0.9849 & 0.4665 \\
0.2098 & 0.9957 & 0.5271 & 0.7638 & 0.4820 & 0.9617 \\
0.2385 & 0.8299 & 0.3855 & 0.9500 & 0.3676 & 0.2845 \\
0.1923 & 0.7167 & 0.6323 & 0.9345 & 0.2605 & 0.0331 \\
0.5322 & 0.1652 & 0.4930 & 0.2563 & 0.5717 & 0.1103 \\
0.7212 & 0.4764 & 0.9196 & 0.6196 & 0.4345 & 0.6993 \\
0.8939 & 0.5753 & 0.7510 & 0.1955 & 0.7078 & 0.9869 \\
0.9766 & 0.7216 & 0.0188 & 0.8569 & 0.8033 & 0.7709 \\
0.3885 & 0.6761 & 0.4612 & 0.8000 & 0.8435 & 0.8812
\end{matrix}\right)$$

$$\left(\begin{matrix}
0.3200 & 0.5421 & 0.5110 & 0.7389 & 0.4918 & 0.9184 \\
0.8660 & 0.3020 & 0.4890 & 0.6987 & 0.3051 & 0.4029 \\
0.0722 & 0.6375 & 0.7953 & 0.1546 & 0.2867 & 0.9654 \\
0.2222 & 0.6622 & 0.5402 & 0.7650 & 0.1721 & 0.3517 \\
0.4795 & 0.8597 & 0.4049 & 0.7070 & 0.0784 & 0.6631 \\
0.3131 & 0.7966 & 0.4261 & 0.7823 & 0.7319 & 0.0807 \\
0.4352 & 0.6139 & 0.5494 & 0.0082 & 0.8012 & 0.5517 \\
0.2544 & 0.5787
\end{matrix}\right)$$

$$\left(\begin{matrix}
0.1865 \\
0.5965 \\
0.3071 \\
0.1203 \\
0.7669 \\
0.8521 \\
0.7875 \\
0.9331
\end{matrix}\right)$$

55% Millennium Falcon
23% Hunk of Junk
0.1% Chewbacca

= f()


ML

File Specification

API

Conversion Tools

ProtoBuf

apple.github.io/coremltools/coremlspecification

Feature Engineering

Input Conversion

Execution
Metal

Output Conversion

75% Darth Vader
24% Sith Lord
10% Samurai

VisionKit

MLModel

Natural Language

GameplayKit

VNTrackObjectRequest

VNDetectFaceRectanglesRequest

VNDetectBarcodesRequest

VNDetectTextRectanglesRequest

```
func captureOutput(_ output: AVCaptureOutput …) {
  let cvp = CMSampleBufferGetImageBuffer(sampleBuffer)
  let irh = VNImageRequestHandler(cvPixelBuffer: cvp, options: [:])
  irh.perform([r])
}
```

VNCoreMLRequest

VNClassificationObservation

```
let modelURL = URL(string: thisURL)
let m = try? MLModel(contentsOf: modelURL)
let vnmodel = try? VNCoreMLModel(for: m)
let r = VNCoreMLRequest(model: vnmodel) { request, error in
    guard error == nil, let results = request.results else { return }
    var out:[[String:Any]] = []
    for result in results {
        if let co = result as? VNClassificationObservation {
            let confidence = String(Int(co.confidence * 100))
            print("Label: " + co.identifier + " " + confidence + "%")
        }
    }
}
```

MLModel.compile(url:)

MLModel(url:)

DictionaryFeatureProvider

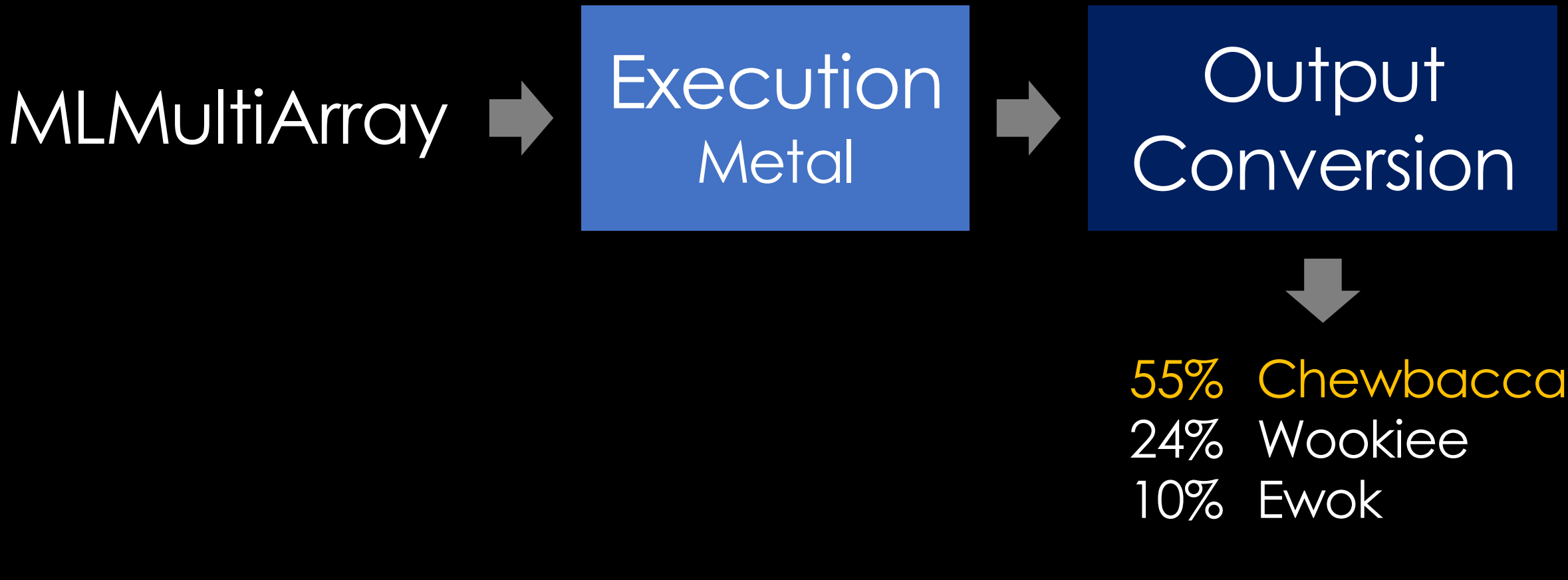
```
let c = MLModel.compile(url: u)
let m = MLModel(url: c)
let d:[String:Any] = ["input": image]
let dfpin = MLDictionaryFeatureProvider(dictionary: d)
let fp = m.predict(dfp)
let dfpout = fp.featureValue(for: "labels")
let dic = dfpout.dictionaryValue
let topResult = dic.first()
```

MLMultiArray

Input Conversion

Execution
Metal

MLMultiArray

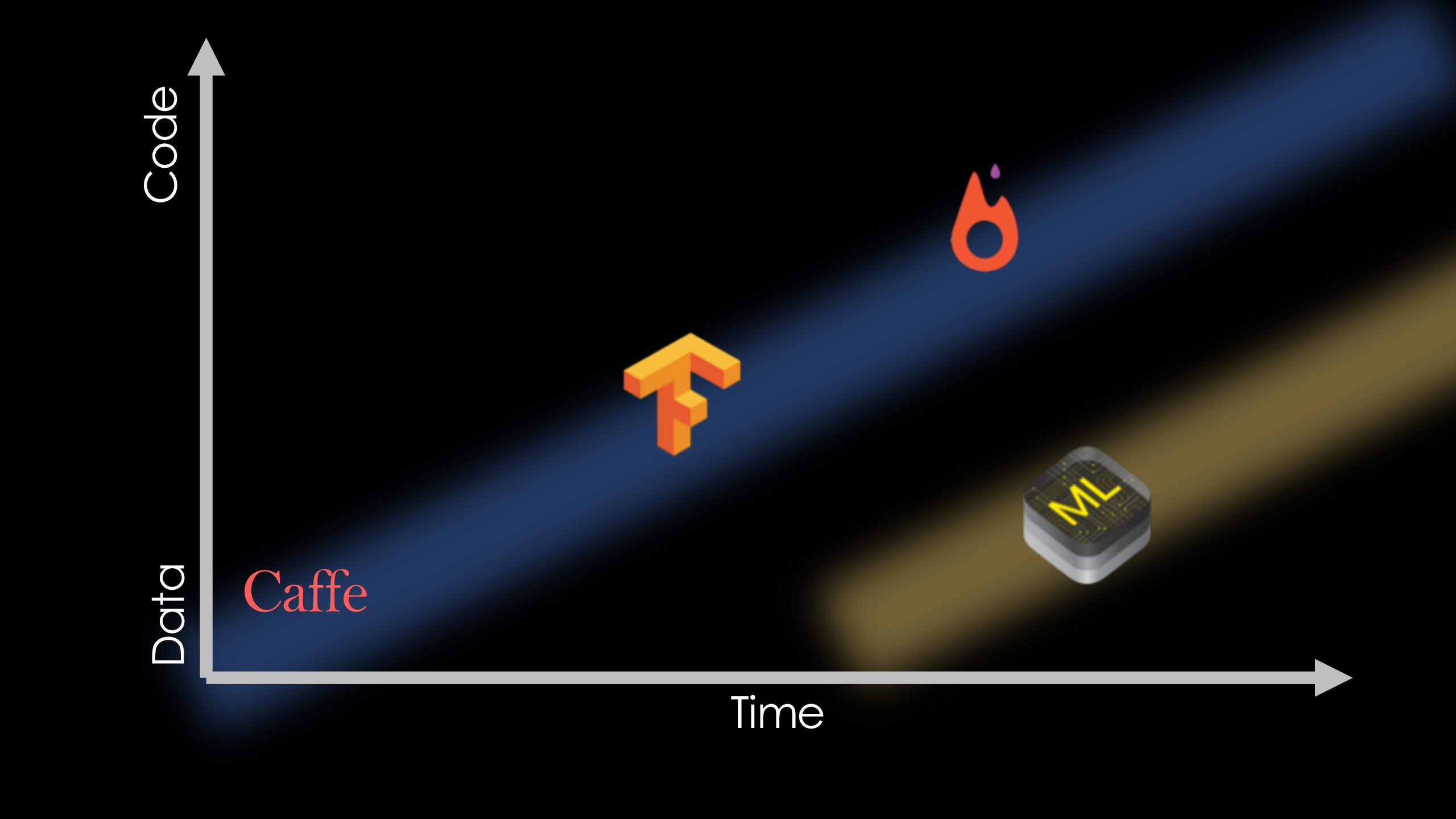

MLMultiArray
Execution
Metal
Output
Conversion
55% Chewbacca
24% Wookiee
10% Ewok

```
let m = MLModel(url: c)
let d:[String:Any] = ["input": image]
let dfpin = MLDictionaryFeatureProvider(dictionary: _d)
let fp = m.predict(dfp)
let c2 = MLModel.compile(url: u2)
let m2 = MLModel(url: c2)
let fp2 = m2.predict(fp)
let dfpout = fp2.featureValue(for: "labels")
let dic = dfpout.dictionaryValue
let topResult = dic.first()
```

Bottlenecking

Composition

Packaging


Code
Data
Caffe
ML
Time


ML

Quantization

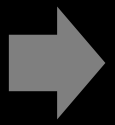
NLModel(url:)

predictedLabel(for:)

predictedLabels(forTokens:)

“I find your lack of faith disturbing”

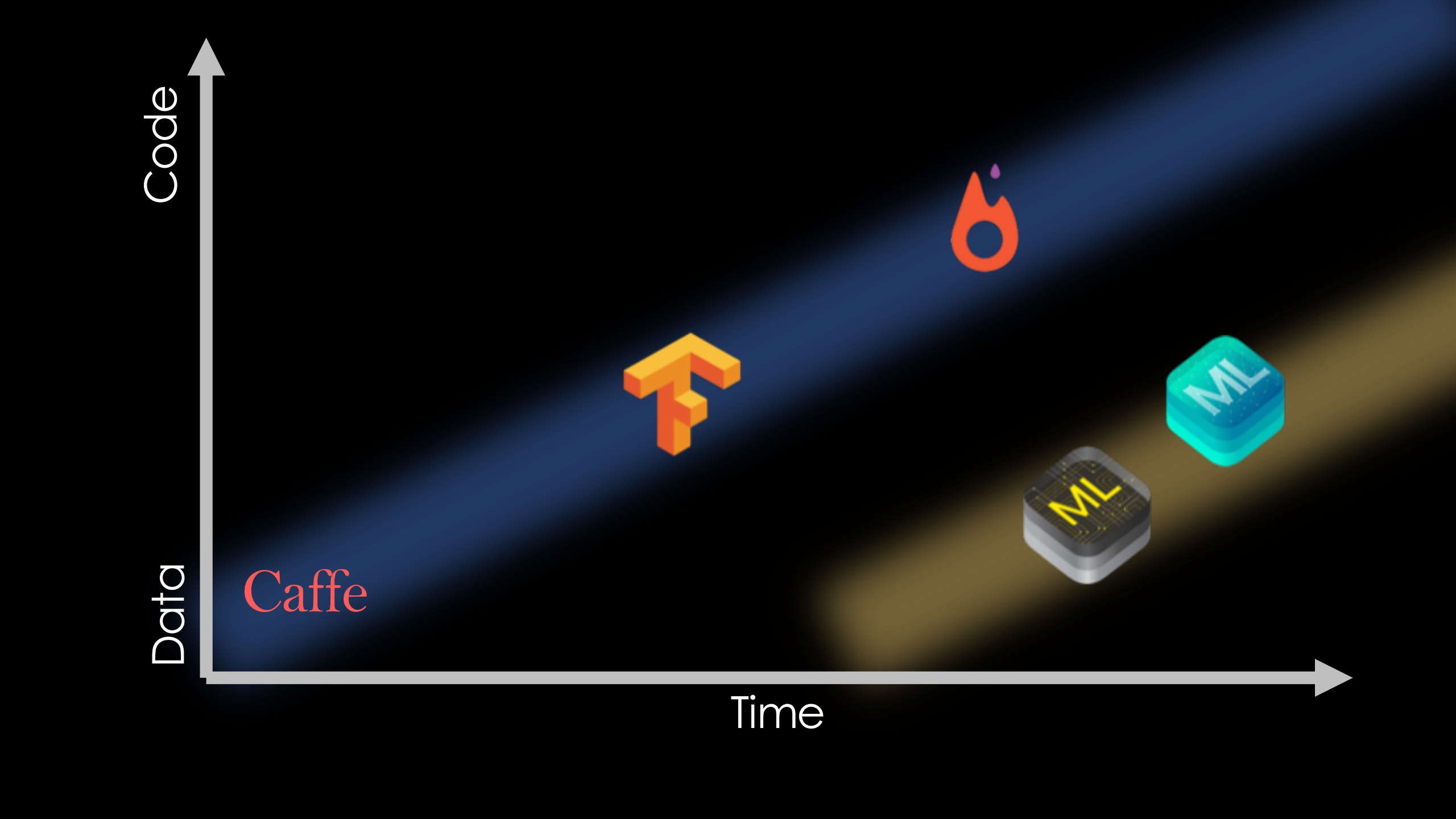
Input Conversion



Execution
Metal

Output Conversion

“Intimidation”

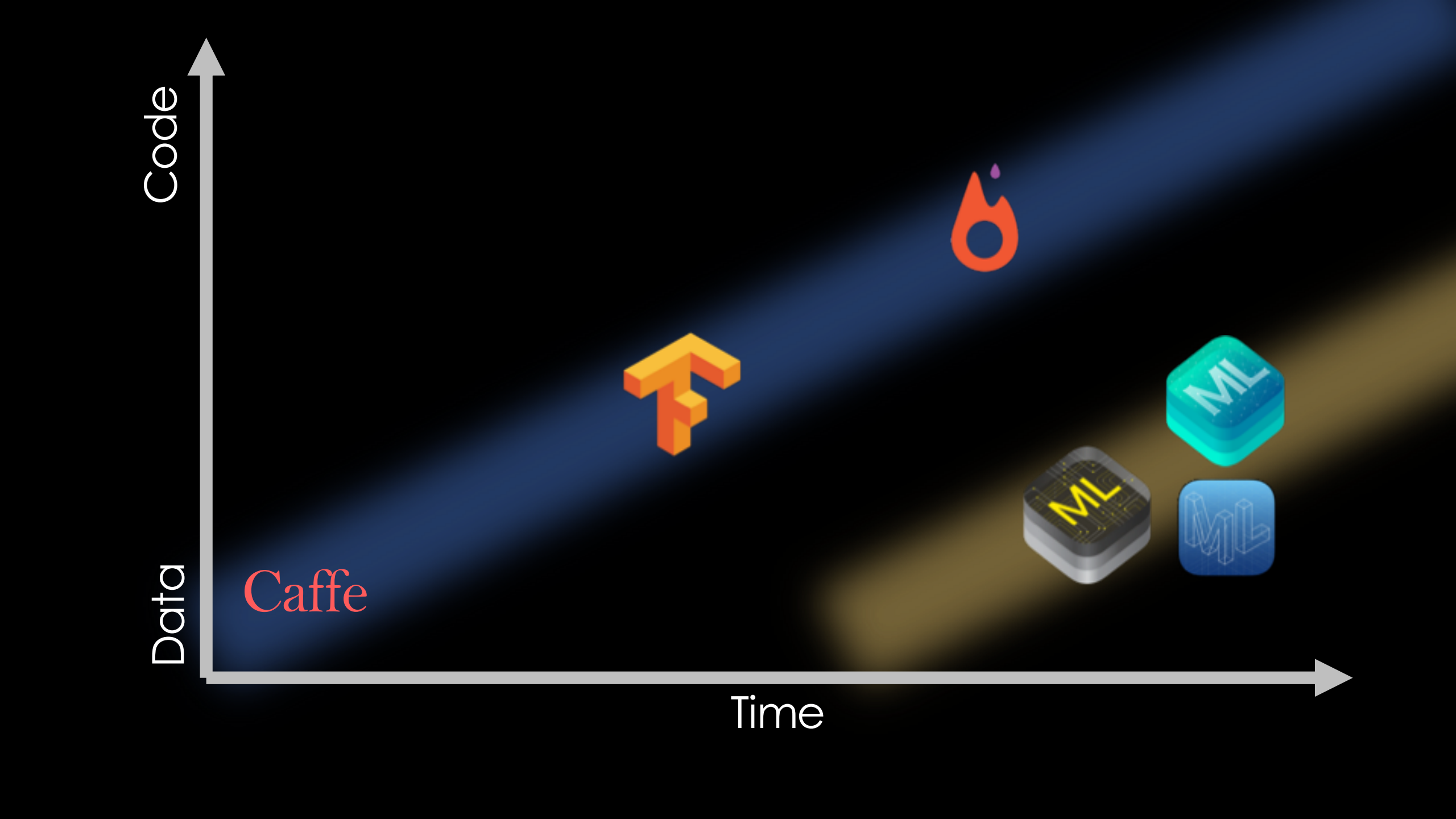

Code
Data
Time
Caffe
ML
ML

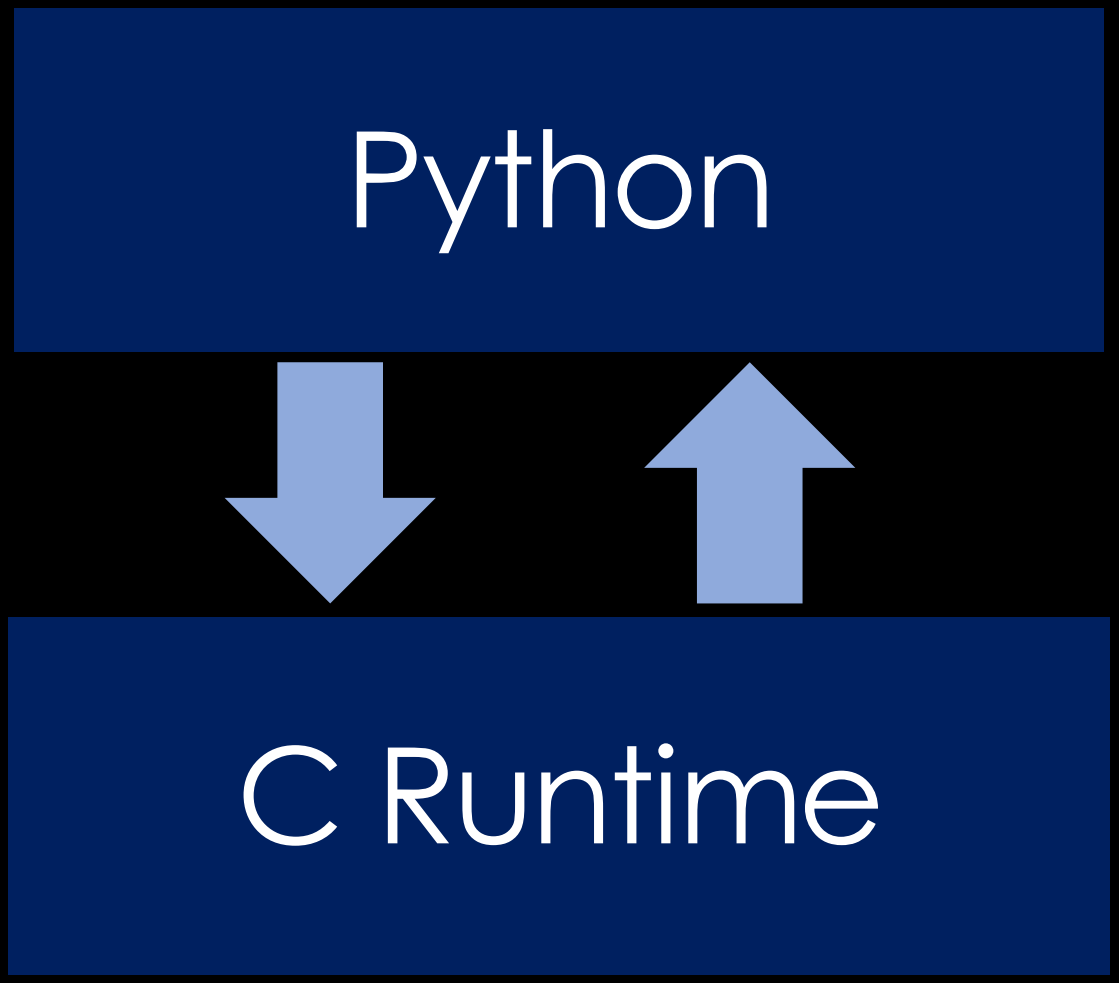
MLImageClassifier

MLTextClassifier

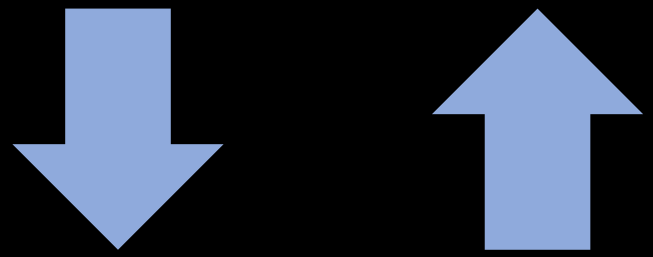
MLWordTagger

CreateMLUI


Code
Data
Time
Caffe
ML
ML


Python
C Runtime

Python



C Runtime

Compiler
& Runtime

```
import Python

let np = Python.import("numpy")
let gzip = Python.import("gzip")
```

Automatic Differentiation

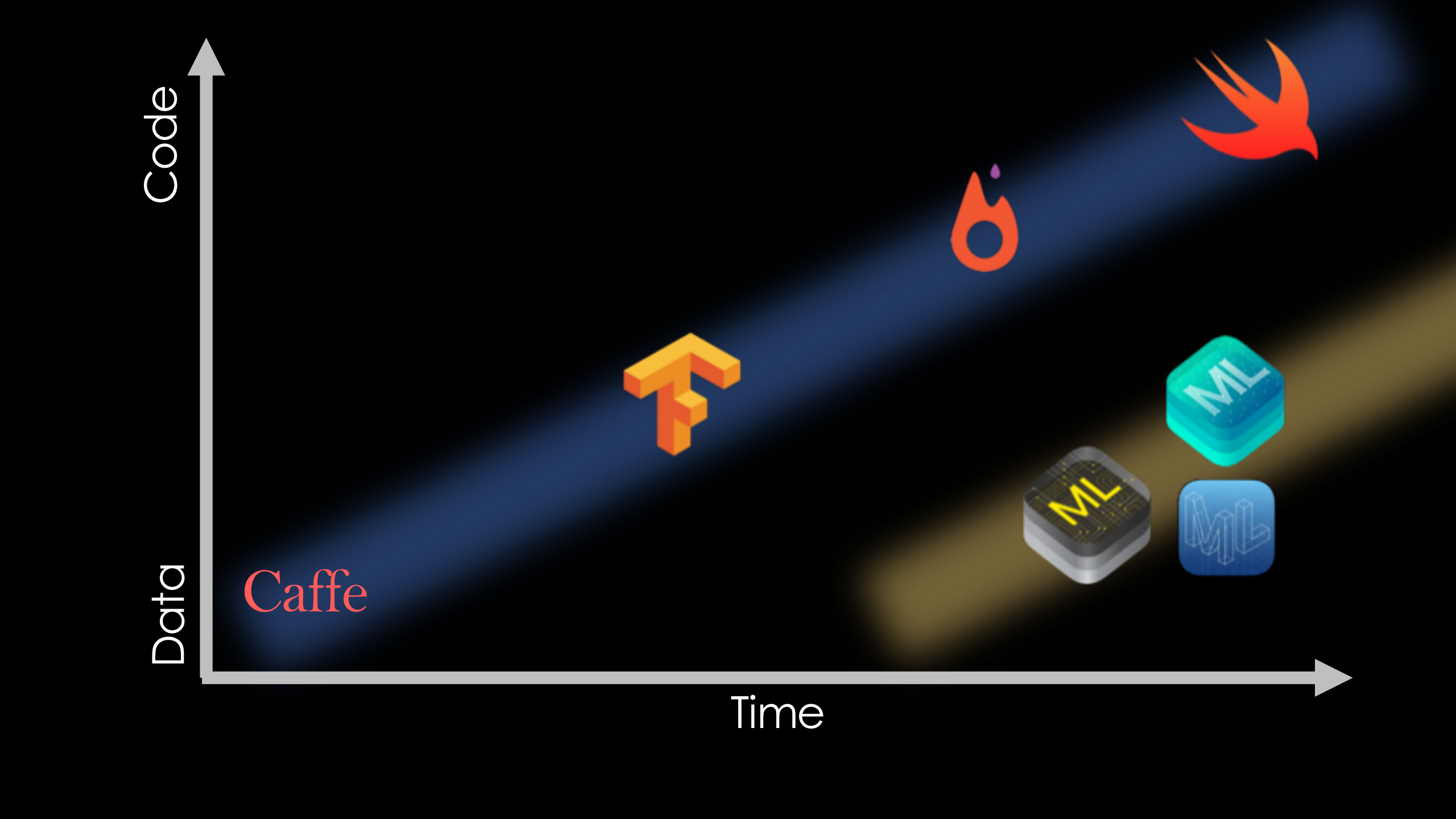
@differentiable

```
@differentiable(reverse, adjoint: dTanh)
func tanh(_ x: Float) -> Float {
  // ... some super low-level assembly…}
func dTanh(x: Float, y: Float, seed: Float) -> Float {
  return (1.0 - (y * y)) * seed
}
// Get the gradient function of tanh.
let dtanh_dx = #gradient(of: tanh)
dtanh_dx(2)
```

```
// Get the gradient function of foo with respect to
the first parameter.
let dfoo_dx = #gradient(of: foo, withRespectTo: .0)
dfoo_dx(3, 4)
```

Eager Execution

Composition


Code
Data
Time
Caffe
ML
ML
ML




Thank You

github.com/rhdeck/swiftfest-2018

Cartoon from *Darth Vader and Son* by Jeffrey Brown