Building Screenshot OCR with Vision and CoreML

Victor Pavlychko

About this Workshop

During the workshop we will:

- Understand how text recognition is done
- Learn how to to build a text recognizer app using iOS frameworks

And write some code:

- CoreImage kernel to adjust contrast
- Use Vision to find text in the image
- Use CoreML to classify characters

bit.ly/2JXs4Ch

Pokémon GO: A Brief Intro

Pokémon Stats

- What players want to know:
  - Attack, defense, stamina
  - Exact level

Pokémon Stats

- What players want to know:
  - Attack, defense, stamina
  - Exact level

- What players know:

Pokémon Stats

- What players want to know:
  - Attack, defense, stamina
  - Exact level

- What players know:
  - CP = f(level, attack, defense, stamina)

Pokémon Stats

- What players want to know:
  - Attack, defense, stamina
  - Exact level

- What players know:
  - CP = f(level, attack, defense, stamina)
  - HP = g(level, stamina)

Pokémon Stats

- What players want to know:
  - Attack, defense, stamina
  - Exact level

- What players know:
  - CP = f(level, attack, defense, stamina)
  - HP = g(level, stamina)
  - Approximate level

IV Checker Apps

- Checking Pokémon stats
  - Take a screenshot
  - Switch to an IV Checker app
  - Use OCR to read values from the image
  - Calculate values player needs

IV Checker Apps

- Checking Pokémon stats
  - Take a screenshot
  - Switch to an IV Checker app
  - Use OCR to read values from the image
  - Calculate values player needs

- Can this be done faster?

IV Checker Apps

- Checking Pokémon stats
  - Take a screenshot
  - Switch to an IV Checker app
  - Use OCR to read values from the image
  - Calculate values player needs

- Can this be done faster?
  - Let's build a keyboard!

IV Checker Apps

- Checking Pokémon stats
  - Take a screenshot
  - Switch to an IV Checker app
  - Use OCR to read values from the image
  - Calculate values player needs

- Can this be done faster?
  - Let's build a keyboard!

Tesseract OCR: An Easy Win?

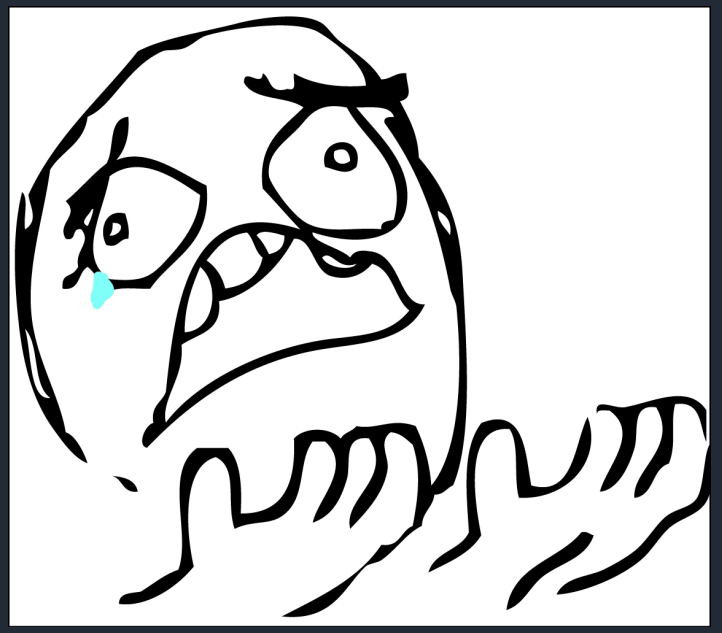
- Ready to use OCR engine maintained by Google
- Supports multiple languages
- Works on many platforms
- Has iOS wrappers available on GitHub
- What could go wrong?

Keyboard Memory Limits

Keyboard Memory Limits


```
Thread 3: EXC_RESOURCE RESOURCE_TYPE_MEMORY (limit=66 MB, unused=0x0)
```

Keyboard extensions is a constrained environment with 66 MB memory limit.




CHALLENGE ACCEPTED

Building Custom OCR

Image Processing Pipeline

Original

Image Processing Pipeline

Original

Remove Color

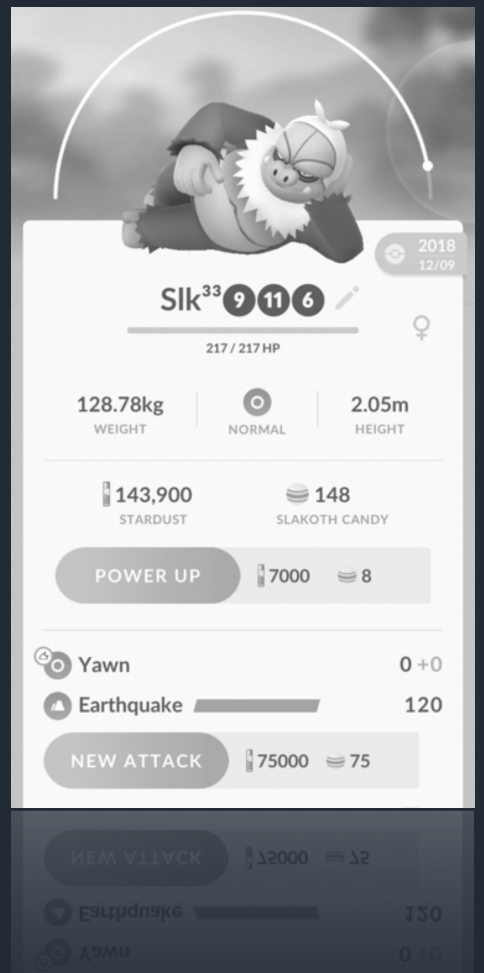
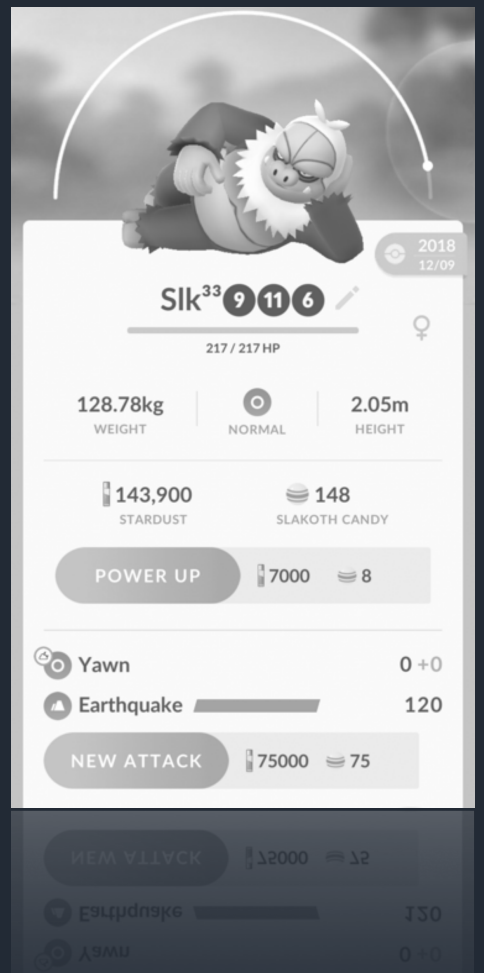


Image Processing Pipeline

Original

Remove Color



Add Contrast

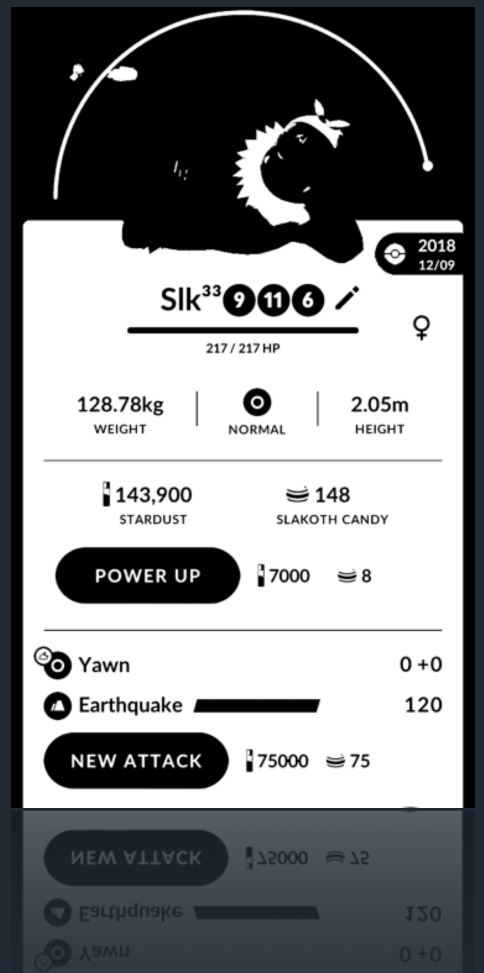
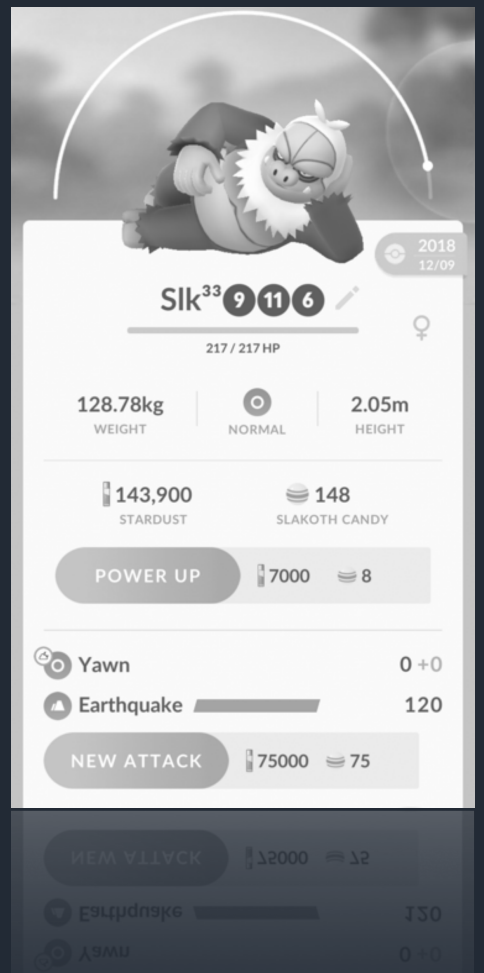


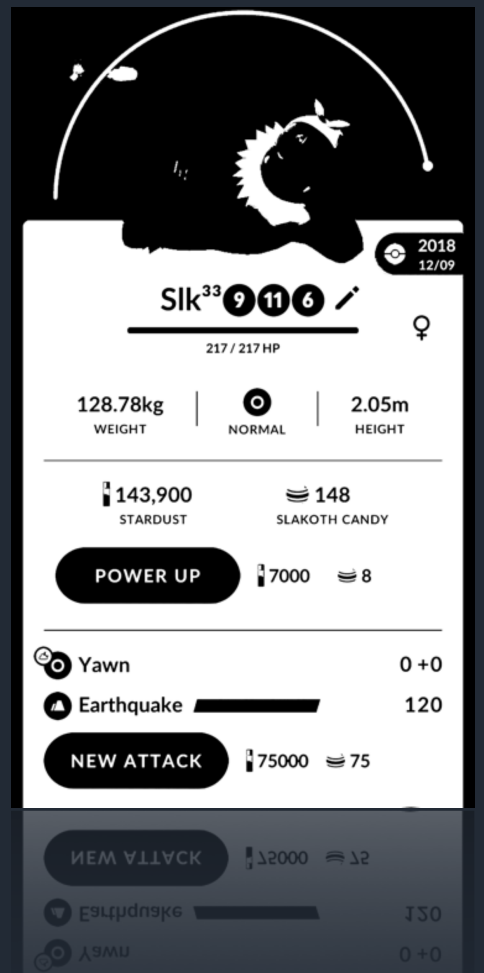
Image Processing Pipeline

Original

Remove Color

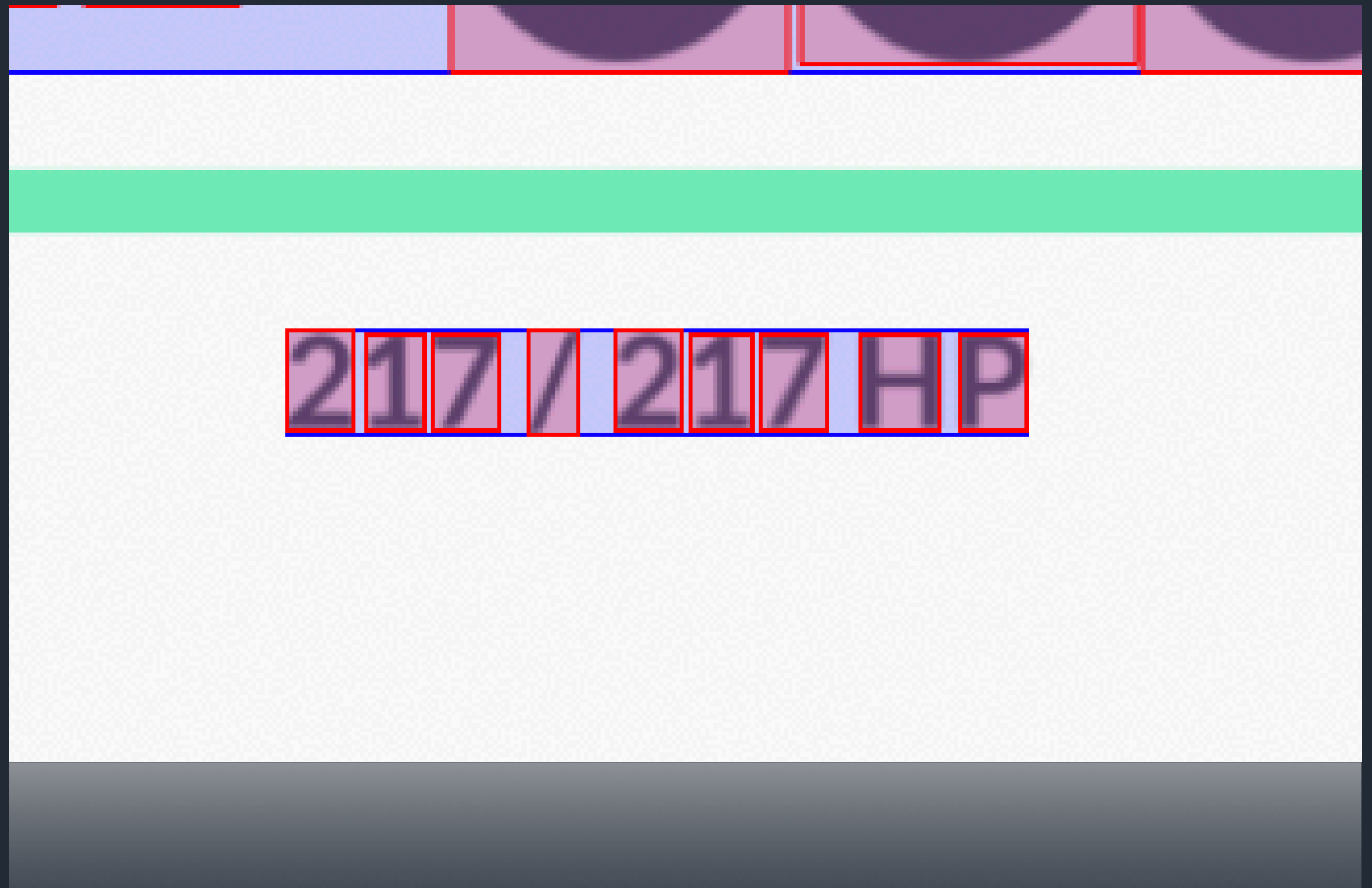


Add Contrast



Find Text

Text Boxes



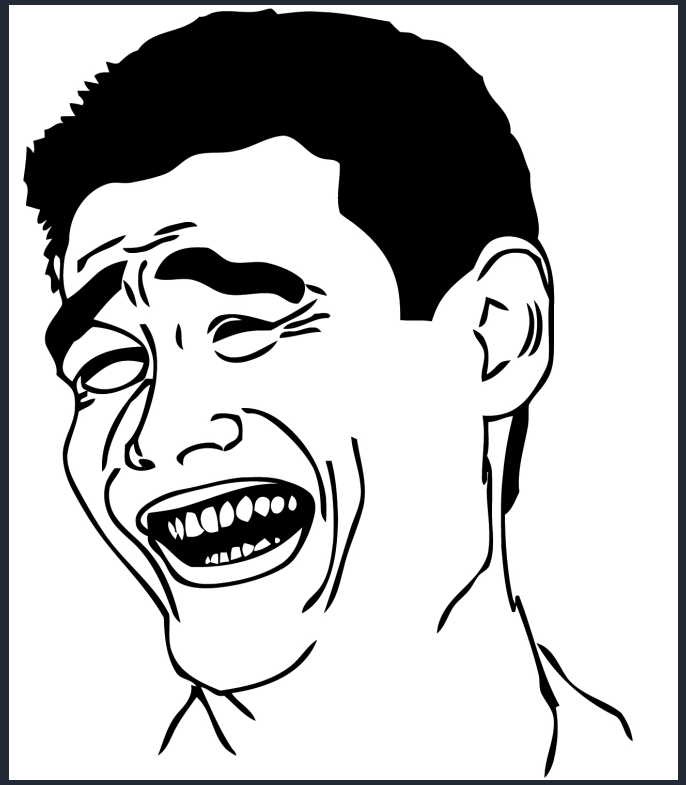
Building a Classifier: CreateML

Let’s Give it a Try!

- Apple: it’s best to use images that are at least 299x299 pixels
- The resulting classifier seems to accept 299x299 images only

Let’s Give it a Try!

- Apple: it’s best to use images that are at least 299x299 pixels
- The resulting classifier seems to accept 299x299 images only
- Good luck classifying letters with that…



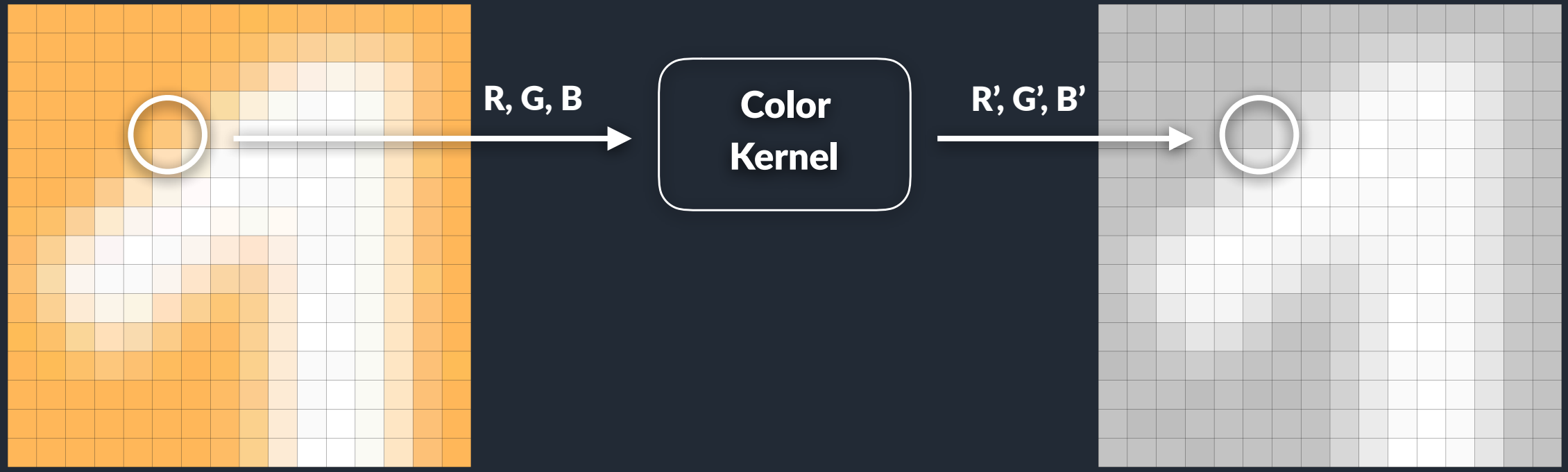
Preprocessing

Grayscale Filter

$$L = R \times 0.299 + G \times 0.587 + B \times 0.114$$

Grayscale Filter

$$L = R \times 0.299 + G \times 0.587 + B \times 0.114$$



Grayscale Kernel


```
kernel vec4 grayscale_kernel(__sample s) {
    float v = dot(s.rgb, vec3(0.299, 0.587, 0.114));
    return vec4(v, v, v, 1);
}
```

Grayscale Kernel


```
kernel vec4 grayscale_kernel(__sample s) {
    float v = dot(s.rgb, vec3(0.299, 0.587, 0.114));
    return vec4(v, v, v, 1);
}
```

0.965
0.796
0.576

Grayscale Kernel


```
kernel vec4 grayscale_kernel(__sample s) {
    float v = dot(s.rgb, vec3(0.299, 0.587, 0.114));
    return vec4(v, v, v, 1);
}
```

0.965	0.299
0.796	0.587
0.576	0.114

Grayscale Kernel


```
kernel vec4 grayscale_kernel(__sample s) {
    float v = dot(s.rgb, vec3(0.299, 0.587, 0.114));
    return vec4(v, v, v, 1);
}
```

0.965	×	0.299	=	0.289			
0.796	×	0.587	=	0.467	+	=	0.822
0.576	×	0.114	=	0.066			

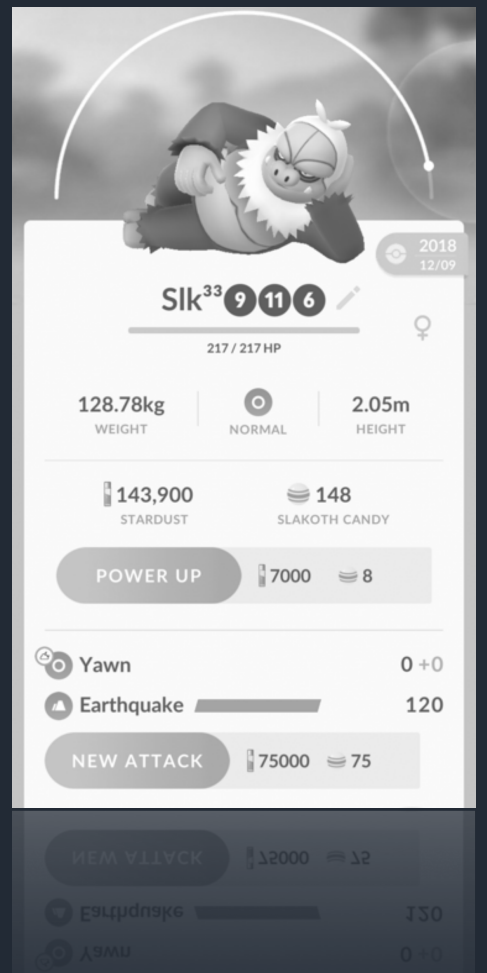
Grayscale Kernel


```
kernel vec4 grayscale_kernel(__sample s) {
    float v = dot(s.rgb, vec3(0.299, 0.587, 0.114));
    return vec4(v, v, v, 1);
}
```

0.965	×	0.299	=	0.289			
0.796	×	0.587	=	0.467	+	=	0.822
0.576	×	0.114	=	0.066			

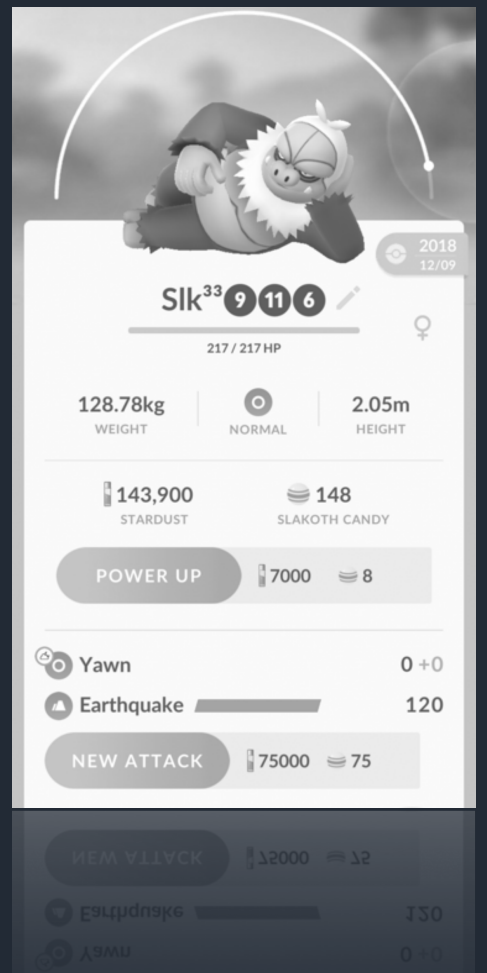
What is Contrast?

Grayscale

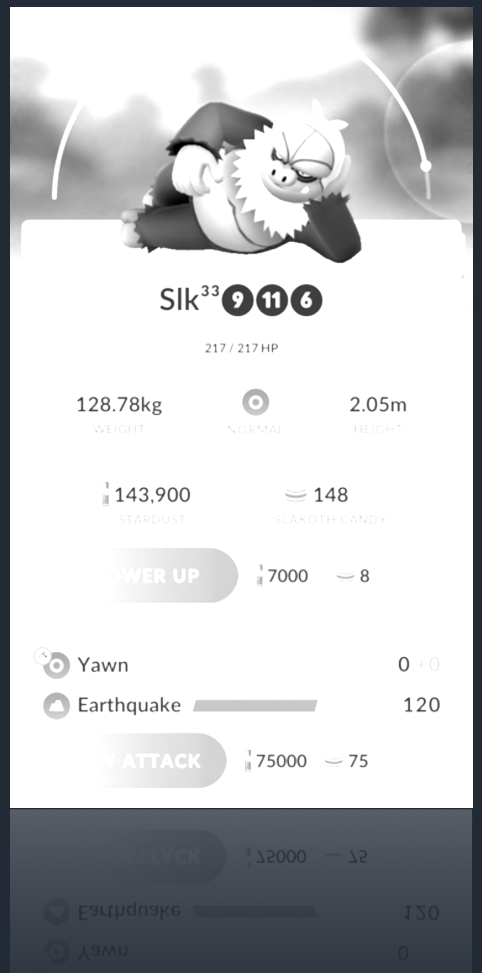


What is Contrast?

Grayscale

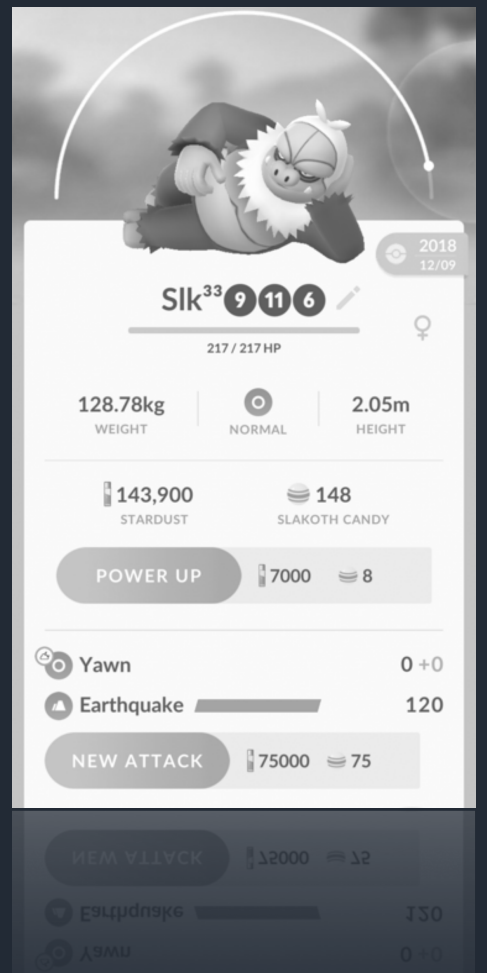


Contrast

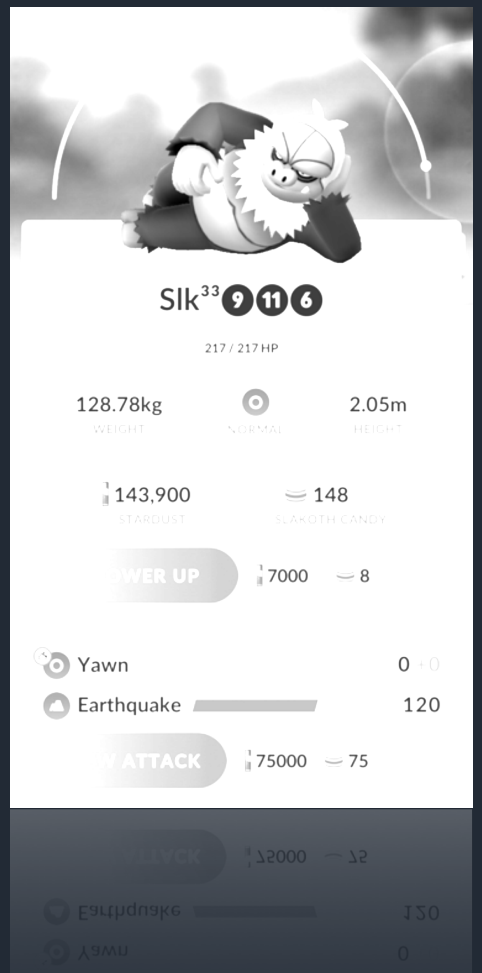


What is Contrast?

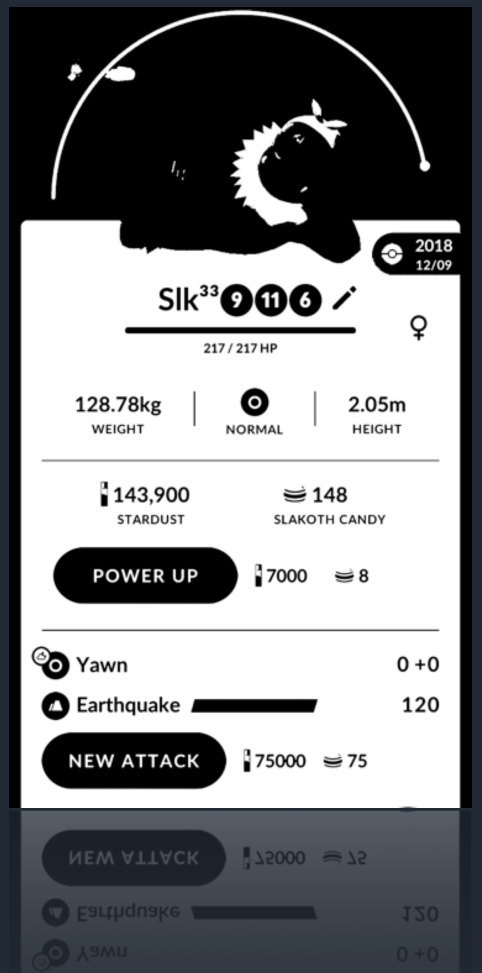
Grayscale



Contrast



Local Contrast

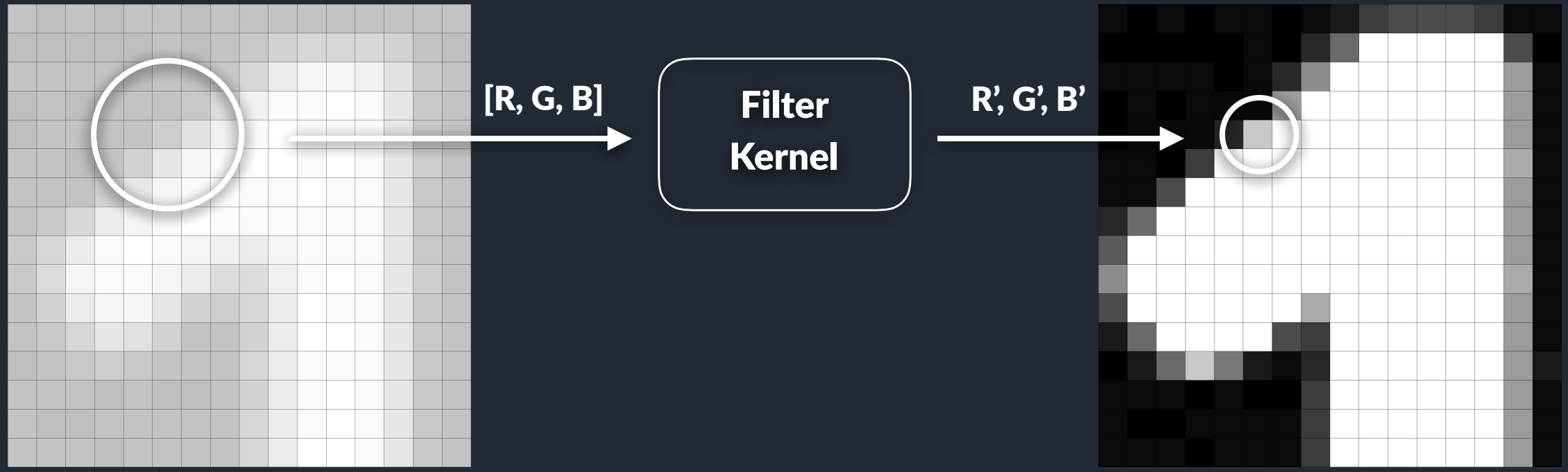


Local Contrast Filter

$$L' = L' = (L - L_{min}) / (L_{max} - L_{min})$$

Local Contrast Filter

$$L' = L' = (L - L_{min}) / (L_{max} - L_{min})$$



Why Use CoreImage Kernels?

Why Use CoreImage Kernels?

iPhone XS resolution is 1125 x 2436 giving us 2.7 million pixels

Why Use CoreImage Kernels?

iPhone XS resolution is 1125 x 2436 giving us 2.7 million pixels

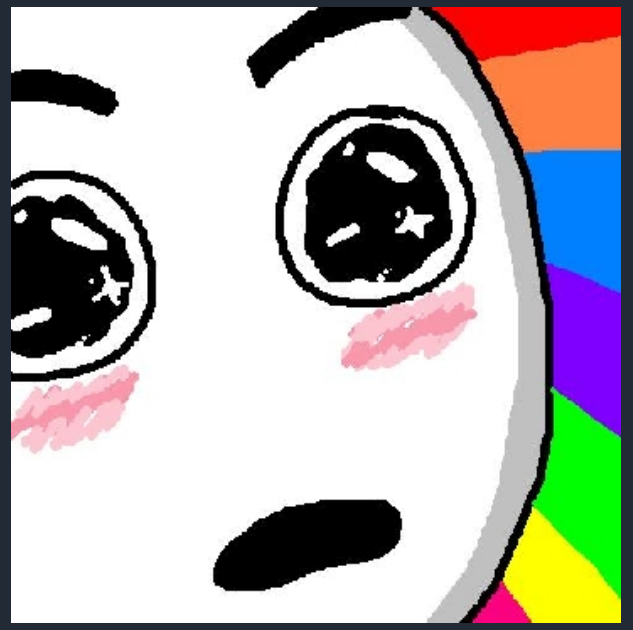
Contrast filter with 10 pixel radius reads 400 pixels per each pixel

Why Use CoreImage Kernels?

iPhone XS resolution is **1125 x 2436** giving us **2.7 million pixels**

Contrast filter with 10 pixel radius reads **400 pixels** per each pixel

Which total to **1 billion iteration** loop



Code Time!

Detecting Text with Vision

Working with Text Detection API

Working with Text Detection API


```
let request = VNDetectTextRectanglesRequest { request, error in
    if let result = request.results?.first as? VNTextObservation {
        print(result.boundingBox)
    }
}
```

Working with Text Detection API


```
let request = VNDetectTextRectanglesRequest { request, error in
    if let result = request.results?.first as? VNTextObservation {
        print(result.boundingBox)
    }
}

request.reportCharacterBoxes = true
request.revision = VNDetectTextRectanglesRequestRevision1
```

Working with Text Detection API


```
let request = VNDetectTextRectanglesRequest { request, error in
    if let result = request.results?.first as? VNTextObservation {
        print(result.boundingBox)
    }
}

request.reportCharacterBoxes = true
request.revision = VNDetectTextRectanglesRequestRevision1


let handler = VNImageRequestHandler(cvPixelBuffer: pixelBuffer, options: [:])
try handler.perform([request])
```

OK, Show me the Text!

OK, Show me the Text!

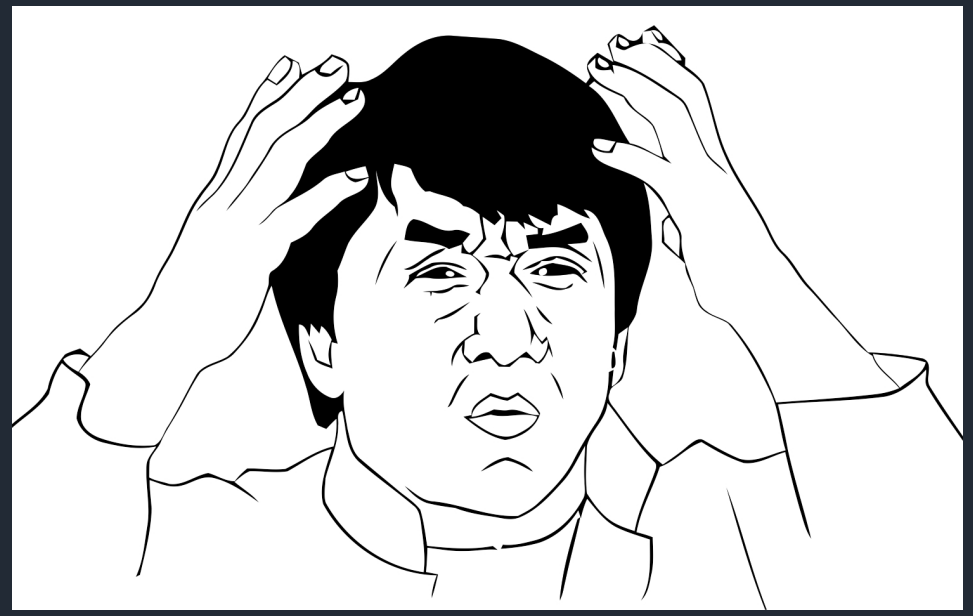


Image Coordinate Systems

Image Coordinate Systems

Top-left Coordinates

0, 0

1125, 2436

Image Coordinate Systems

Top-left Coordinates

0, 0

1125, 2436

Vision Coordinates

1, 1

0, 0

Dealing with Vision Coordinates

Dealing with Vision Coordinates


```
extension VNRectangleObservation {
    var flippedBoundingBox: CGRect {
        return CGRect(
            x: boundingBox.minX, y: 1 - boundingBox.minY - boundingBox.height,
            width: boundingBox.width, height: boundingBox.height
        )
    }
}
```

Dealing with Vision Coordinates

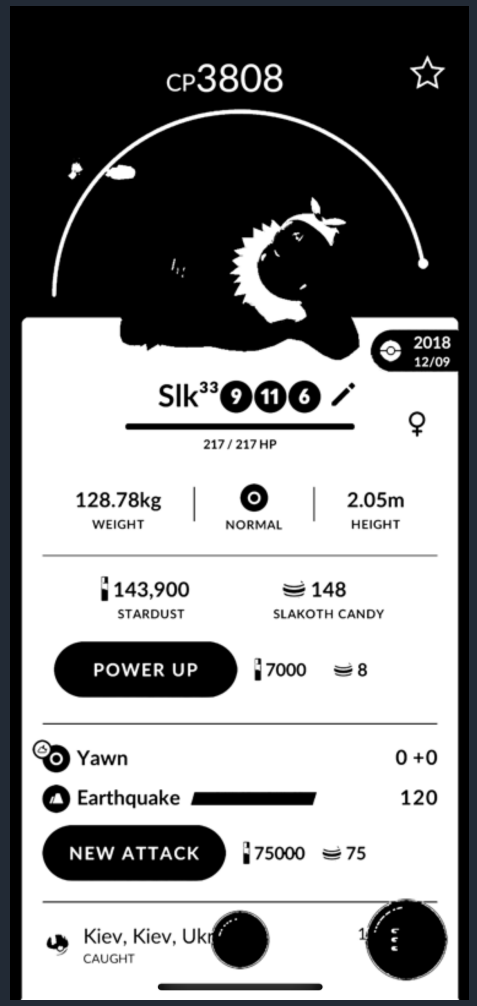

```
extension VNRectangleObservation {
    var flippedBoundingBox: CGRect {
        return CGRect(
            x: boundingBox.minX, y: 1 - boundingBox.minY - boundingBox.height,
            width: boundingBox.width, height: boundingBox.height
        )
    }
}

extension CGRect {
    func denormalized(for size: CGSize) -> CGRect {
        return VNImageRectForNormalizedRect(self, Int(size.width), Int(size.height))
    }
}
```

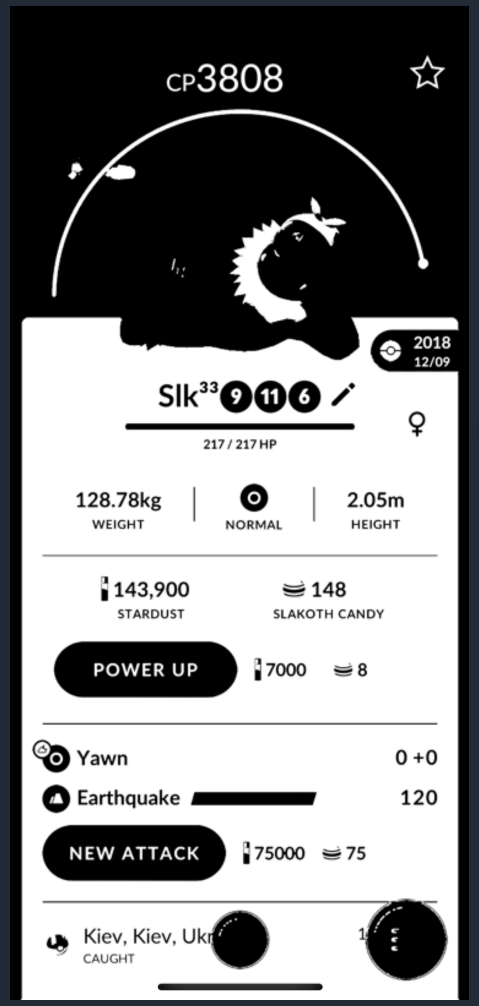
Code Time!

Finding Interesting Text

Analyzing Screenshot Layout

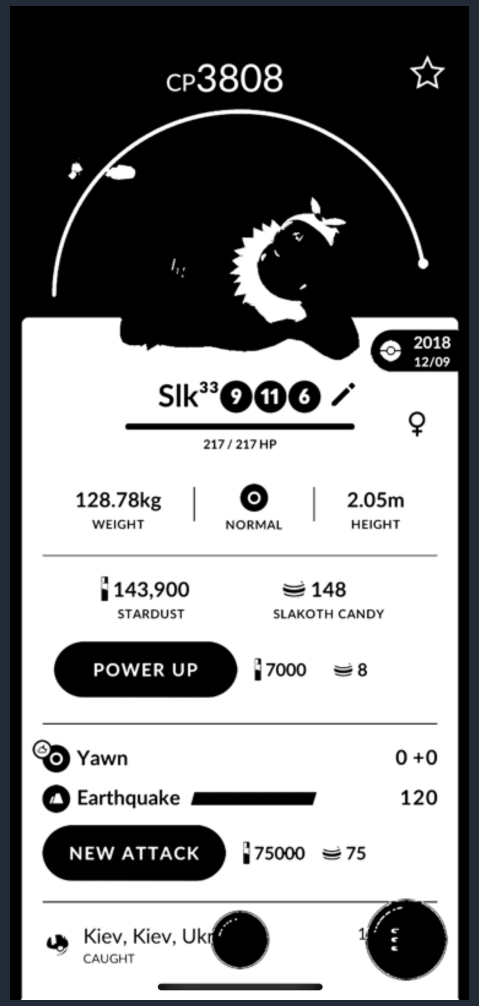


Analyzing Screenshot Layout



Health: blackWidth > w / 3 && blackWidth < w * 2 / 3

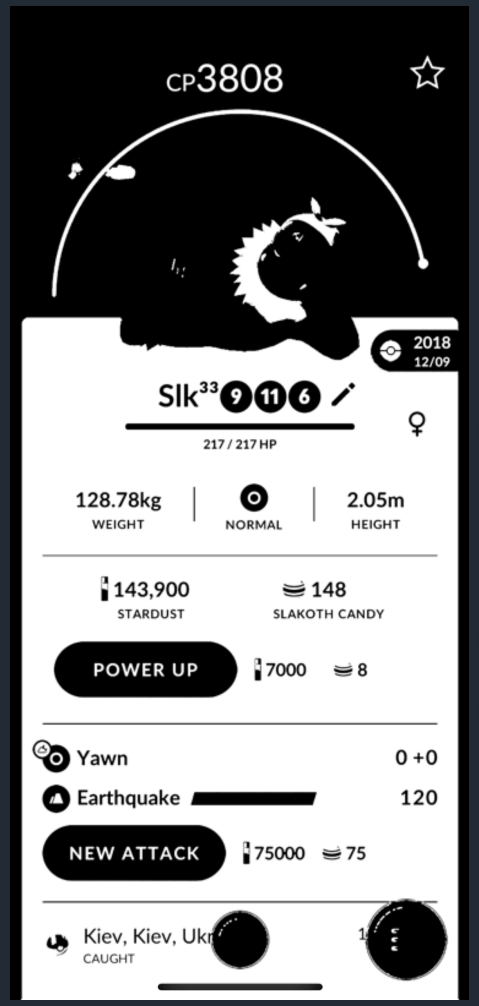
Analyzing Screenshot Layout



Health: blackWidth > w / 3 && blackWidth < w * 2 / 3

White panel: whiteWidth > w / 2

Analyzing Screenshot Layout

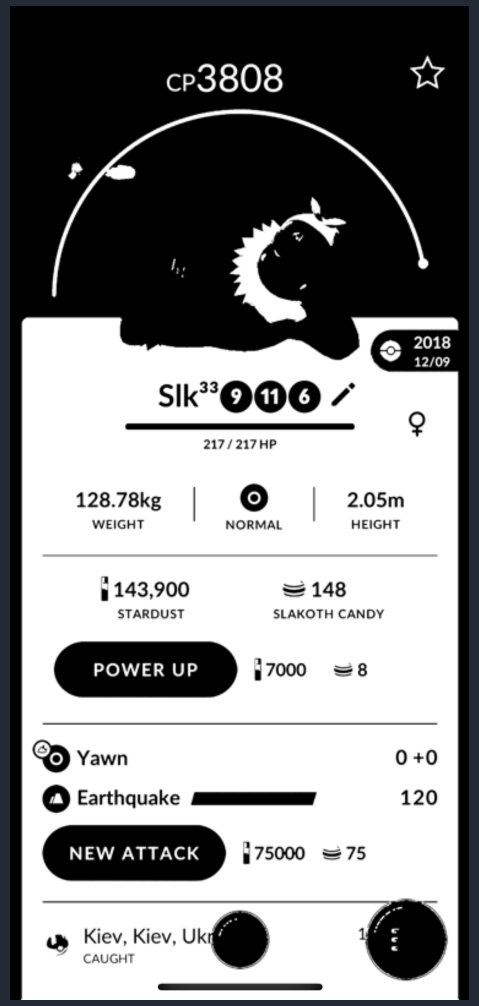


Health: blackWidth > w / 3 && blackWidth < w * 2 / 3

White panel: whiteWidth > w / 2

Separator: blackWidth > w * 2 / 3

Analyzing Screenshot Layout



Health: blackWidth > w / 3 && blackWidth < w * 2 / 3

White panel: whiteWidth > w / 2

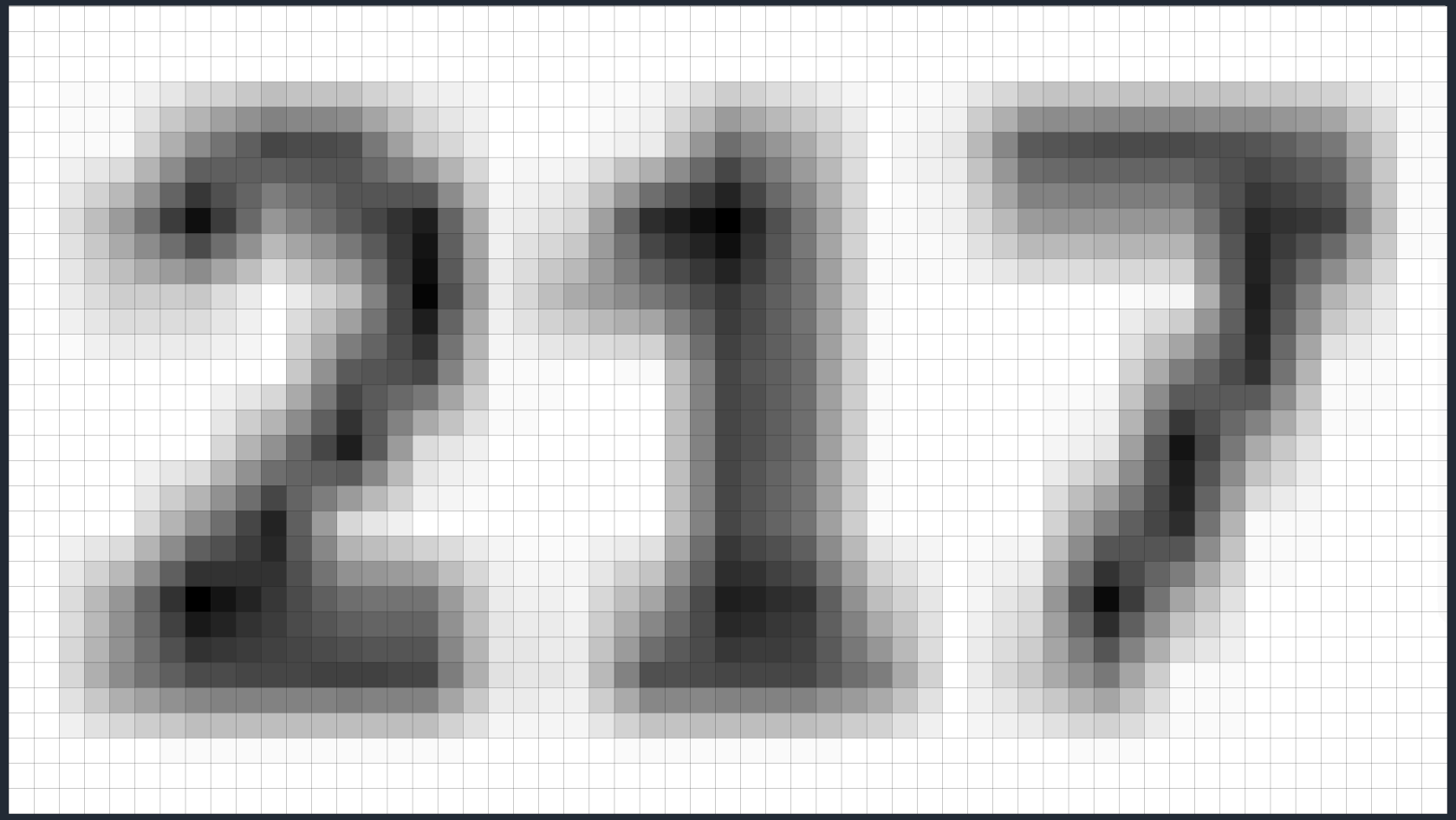
Separator: blackWidth > w * 2 / 3

Unknown: everything else

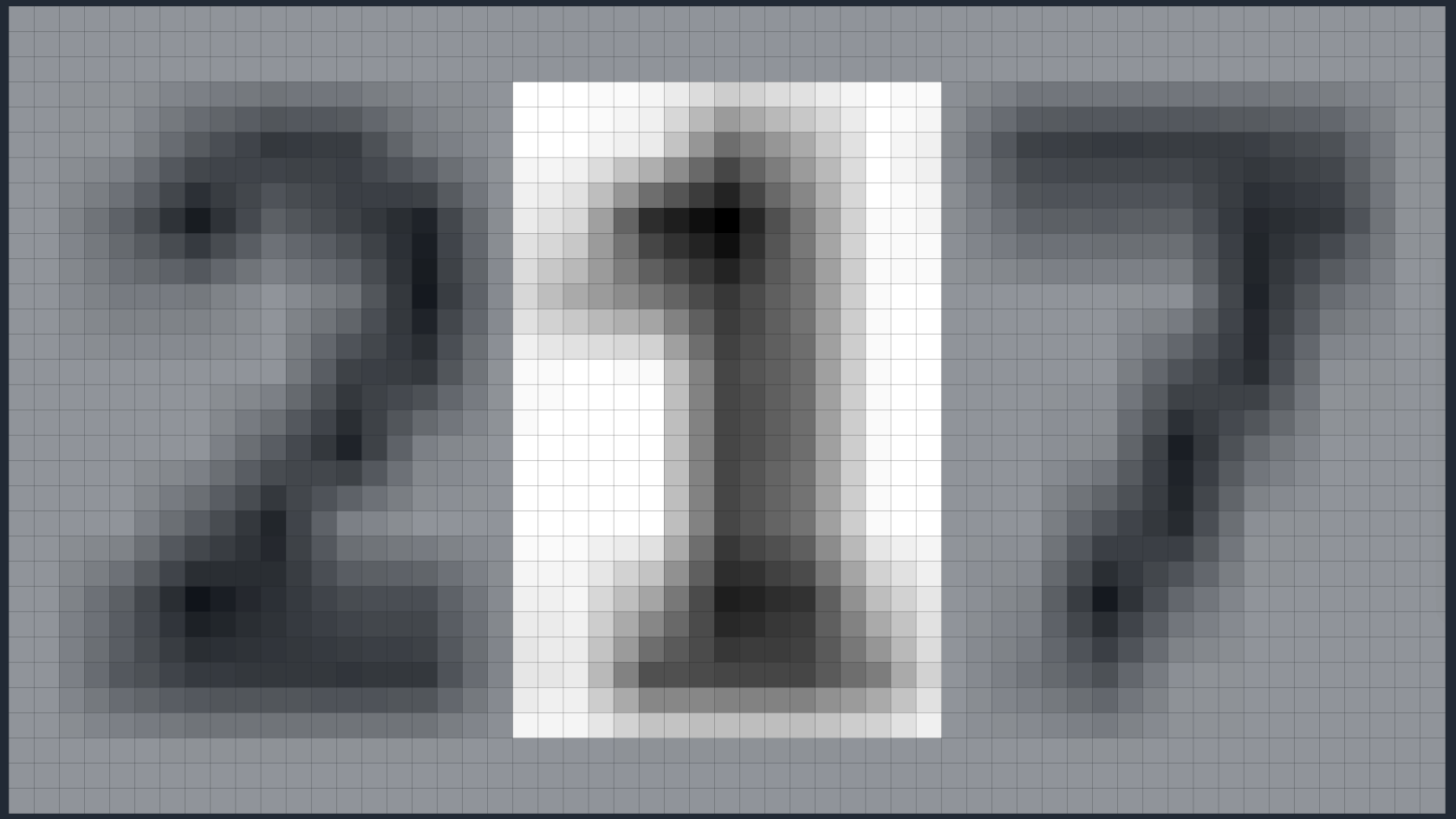
Code Time!

Classifying Images with CoreML

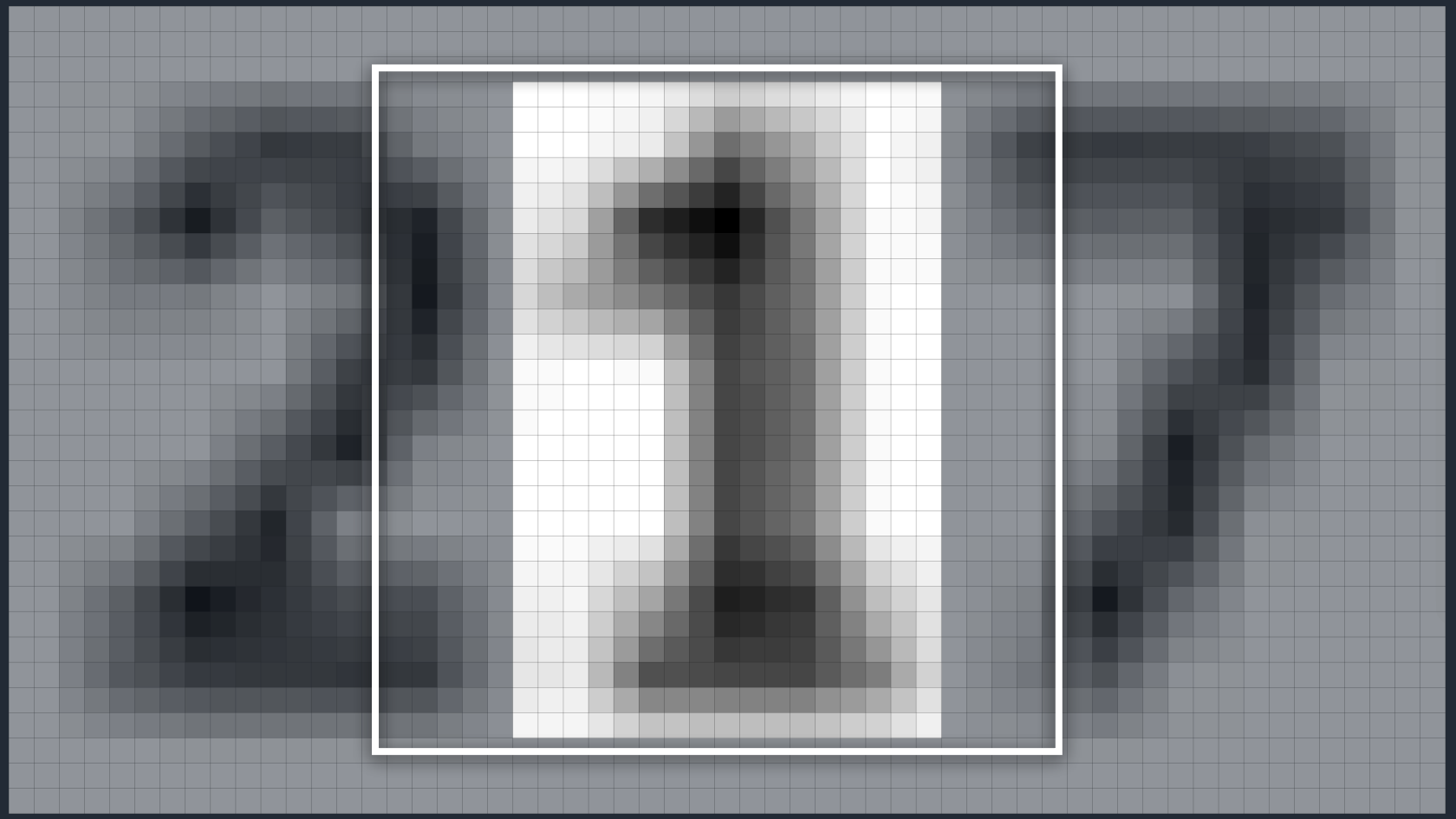
Preparing an Image for Classification



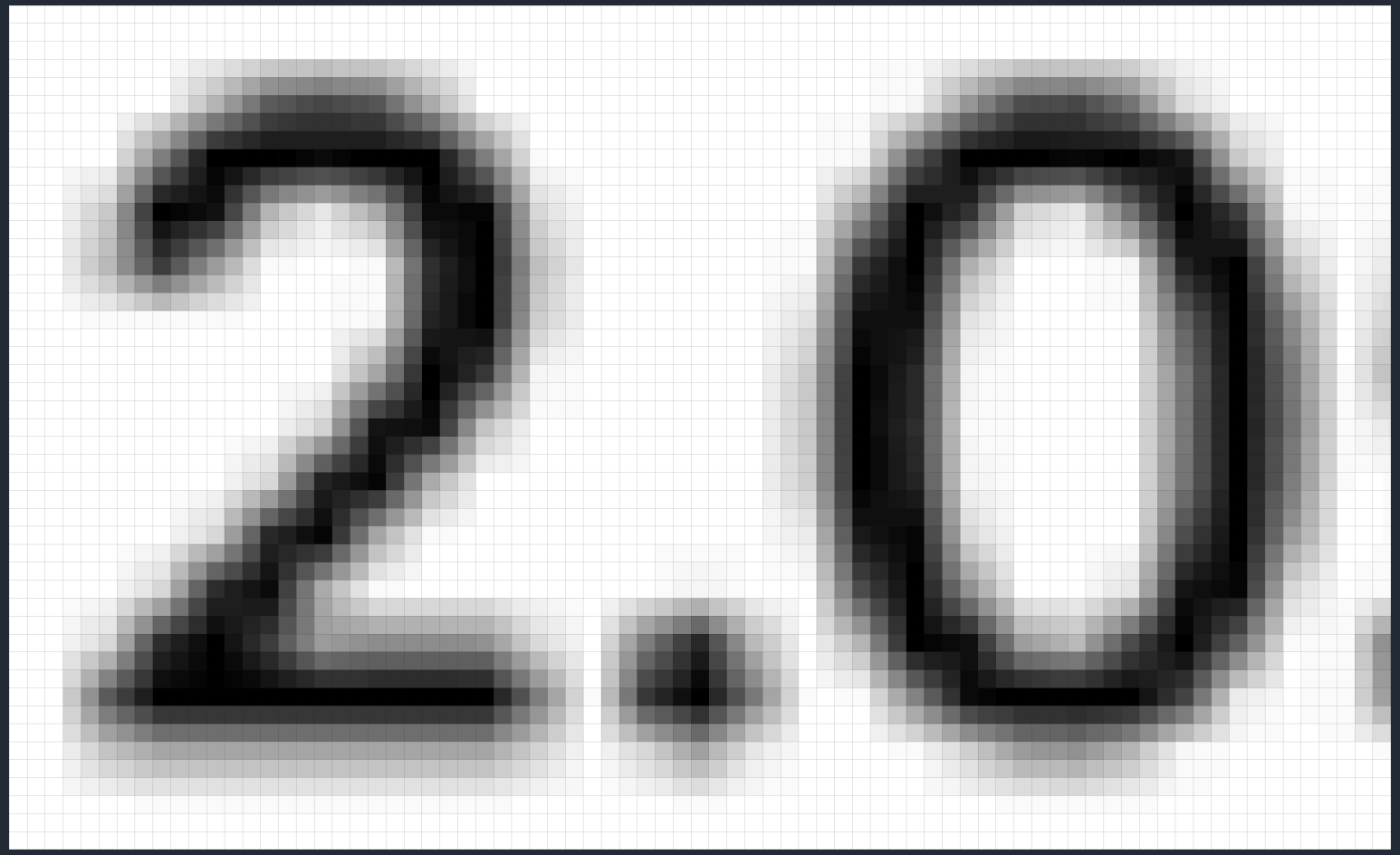
Preparing an Image for Classification



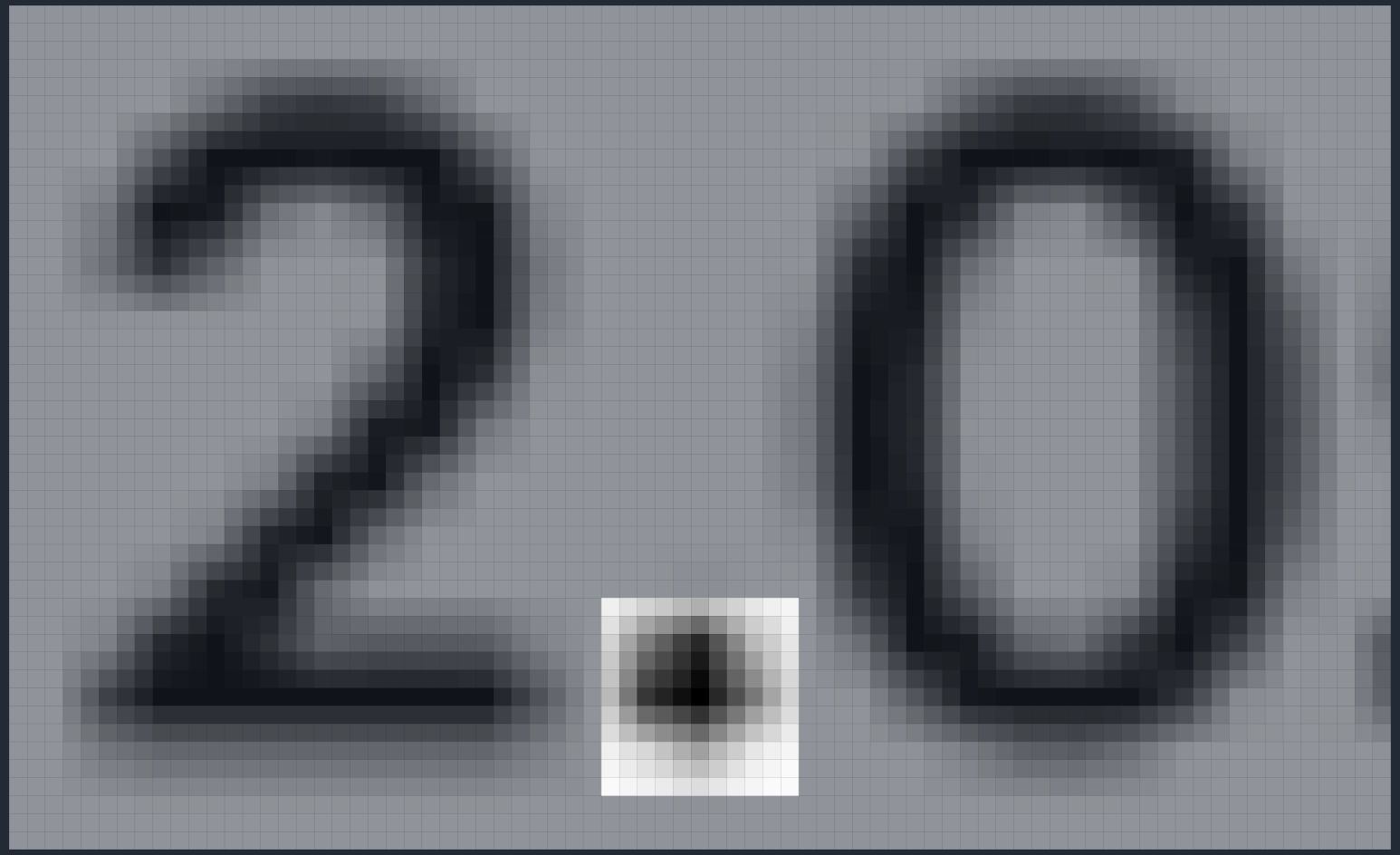
Preparing an Image for Classification



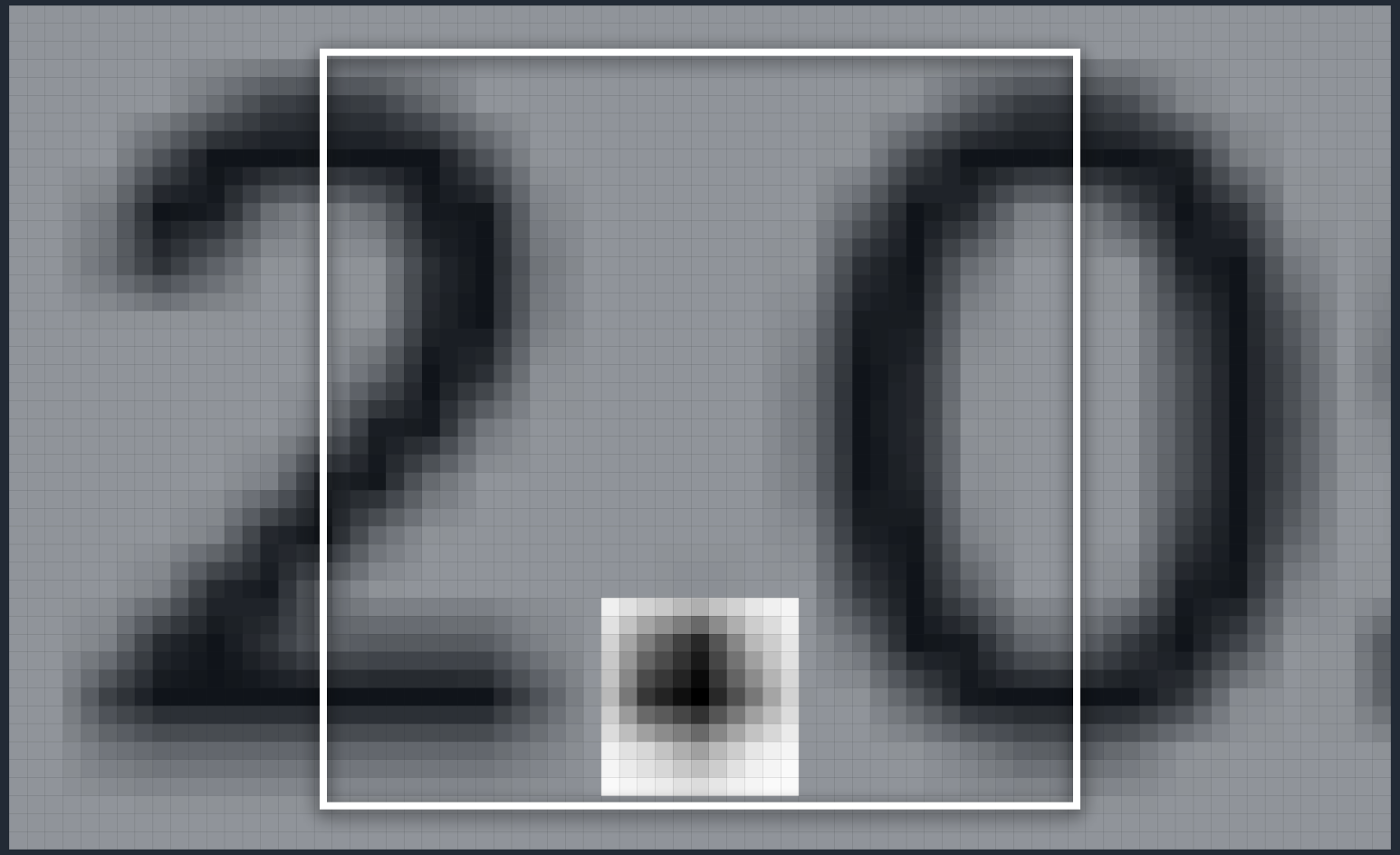
Preparing an Image for Classification



Preparing an Image for Classification

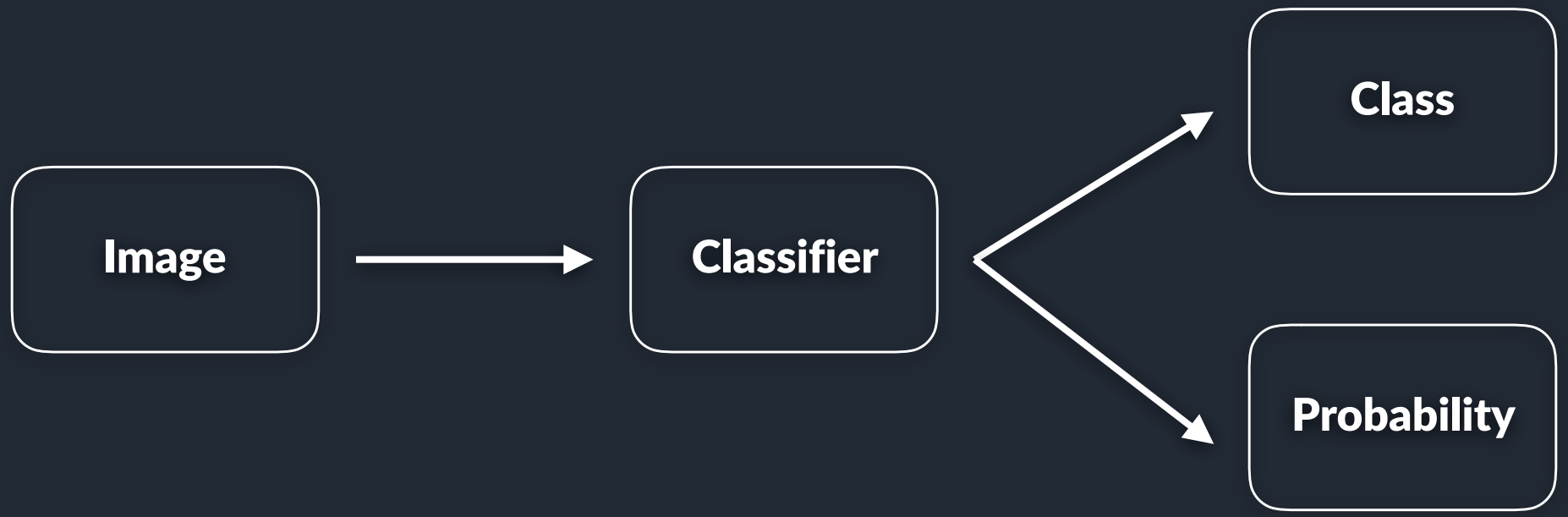


Preparing an Image for Classification



Classifying an Image with CoreML

Classifying an Image with CoreML



Using CoreML Classifier Model

Using CoreML Classifier Model


```
private func classify(_ pixelBuffer: CVPixelBuffer) throws -> String? {
```

Using CoreML Classifier Model


```
private func classify(_ pixelBuffer: CVPixelBuffer) throws -> String? {
    let prediction = try hpClassifier.prediction(image: pixelBuffer)
```

Using CoreML Classifier Model


```
private func classify(_ pixelBuffer: CVPixelBuffer) throws -> String? {
    let prediction = try hpClassifier.prediction(image: pixelBuffer)

    let probability = prediction.output[classLabel] ?? 0
    guard probability > threshold else {
        return nil
    }
```

Using CoreML Classifier Model


```
private func classify(_ pixelBuffer: CVPixelBuffer) throws -> String? {
    let prediction = try hpClassifier.prediction(image: pixelBuffer)

    let probability = prediction.output[classLabel] ?? 0
    guard probability > threshold else {
        return nil
    }

    let classLabel = prediction.classLabel
    return classLabel
}
```

Using CoreML Classifier Model


```
private func classify(_ pixelBuffer: CVPixelBuffer) throws -> String? {
    let prediction = try hpClassifier.prediction(image: pixelBuffer)

    let probability = prediction.output[classLabel] ?? 0
    guard probability > threshold else {
        return nil
    }

    let classLabel = prediction.classLabel
    return classLabel
}
```

Code Time!

Questions?

Thank You!

Victor Pavlychko

Facebook: victor_pavlychko
Twitter: @victorpavlychko
GitHub: victor-pavlychko

bit.ly/2JXs4Ch