#WWDC19

Text Recognition in Vision Framework

Frank Doepke, Vision Team
Cédric Bray, Vision Team

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNDetectTextRectanglesRequest { (request, error) in
    guard let observations = request.results as? [VNTextObservation] else { return }

}

request.reportCharacterBoxes = true
try? requestHandler.perform([request])
```

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNDetectTextRectanglesRequest { (request, error) in
    guard let observations = request.results as? [VNTextObservation] else { return }









}

request.reportCharacterBoxes = true
try? requestHandler.perform([request])
```

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNDetectTextRectanglesRequest { (request, error) in
    guard let observations = request.results as? [VNTextObservation] else { return }

    for currentObservation in observations
    {
        // iterate over the character boxes in the observation
        for currentCharacterBox in currentObservation.characterBoxes!
        {



        }



    }
}

request.reportCharacterBoxes = true
try? requestHandler.perform([request])
```

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNDetectTextRectanglesRequest { (request, error) in
    guard let observations = request.results as? [VNTextObservation] else { return }

    for currentObservation in observations
    {
        // iterate over the character boxes in the observation
        for currentCharacterBox in currentObservation.characterBoxes!
        {
            // TODO: Train CoreML model to do character recognition


        }



    }
}

request.reportCharacterBoxes = true
try? requestHandler.perform([request])
```

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNDetectTextRectanglesRequest { (request, error) in
    guard let observations = request.results as? [VNTextObservation] else { return }

    for currentObservation in observations
    {
        // iterate over the character boxes in the observation
        for currentCharacterBox in currentObservation.characterBoxes!
        {
            // TODO: Train CoreML model to do character recognition
            // TODO: run model on character box

        }



    }
}

request.reportCharacterBoxes = true
try? requestHandler.perform([request])
```

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNDetectTextRectanglesRequest { (request, error) in
    guard let observations = request.results as? [VNTextObservation] else { return }

    for currentObservation in observations
    {
        // iterate over the character boxes in the observation
        for currentCharacterBox in currentObservation.characterBoxes!
        {
            // TODO: Train CoreML model to do character recognition
            // TODO: run model on character box
            // TODO: threshold against possible garbage results
        }



    }
}

request.reportCharacterBoxes = true
try? requestHandler.perform([request])
```

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNDetectTextRectanglesRequest { (request, error) in
    guard let observations = request.results as? [VNTextObservation] else { return }

    for currentObservation in observations
    {
        // iterate over the character boxes in the observation
        for currentCharacterBox in currentObservation.characterBoxes!
        {
            // TODO: Train CoreML model to do character recognition
            // TODO: run model on character box
            // TODO: threshold against possible garbage results
        }
        // TODO: concatenate characters into string


    }
}

request.reportCharacterBoxes = true
try? requestHandler.perform([request])
```

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNDetectTextRectanglesRequest { (request, error) in
    guard let observations = request.results as? [VNTextObservation] else { return }

    for currentObservation in observations
    {
        // iterate over the character boxes in the observation
        for currentCharacterBox in currentObservation.characterBoxes!
        {
            // TODO: Train CoreML model to do character recognition
            // TODO: run model on character box
            // TODO: threshold against possible garbage results
        }
        // TODO: concatenate characters into string
        // TODO: Fix recognized words based on dictionary
        //       and other probability heuristics for character pairs
    }
}

request.reportCharacterBoxes = true
try? requestHandler.perform([request])
```

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNRecognizeTextRequest { (request, error) in
    guard let observations = request.results as? [VNRecognizedTextObservation] else { return }

    for currentObservation in observations
    {
        let topCandidate = currentObservation.topCandidates(1)
        if let recognizedText = topCandidate.first {
                print(recognizedText.string)
        }
    }
}
request.recognitionLevel = .accurate

try? requestHandler.perform([request])
```

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNRecognizeTextRequest { (request, error) in
    guard let observations = request.results as? [VNRecognizedTextObservation] else { return }

    for currentObservation in observations
    {
        let topCandidate = currentObservation.topCandidates(1)
        if let recognizedText = topCandidate.first {
                print(recognizedText.string)
        }
    }
}
request.recognitionLevel = .accurate

try? requestHandler.perform([request])
```

```
let requestHandler = VNImageRequestHandler(url: imageURL, options: [:])

let request = VNRecognizeTextRequest { (request, error) in
    guard let observations = request.results as? [VNRecognizedTextObservation] else { return }

    for currentObservation in observations
    {
        let topCandidate = currentObservation.topCandidates(1)
        if let recognizedText = topCandidate.first {
                print(recognizedText.string)
        }
    }
}
request.recognitionLevel = .accurate

try? requestHandler.perform([request])
```

The Health Benefits of Reading

Reduce stress levels (by 68 percent!)

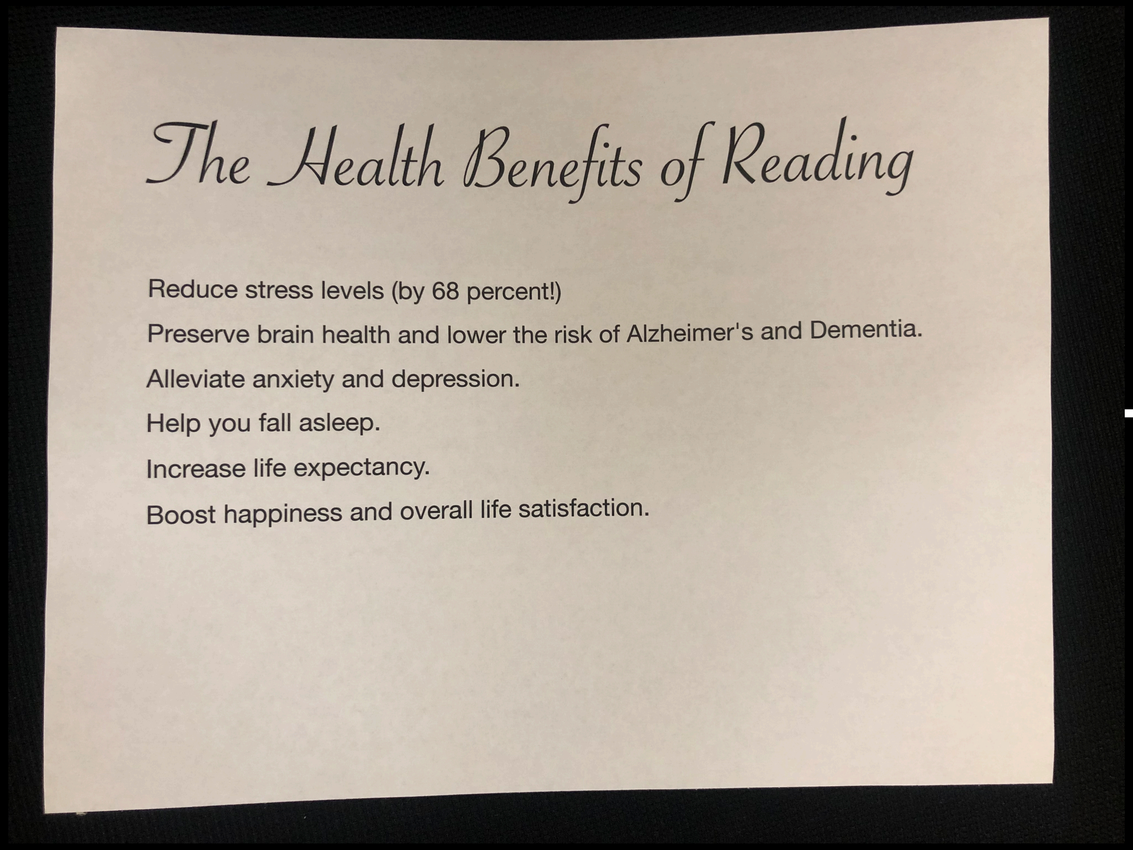
Preserve brain health and lower the risk of Alzheimer's and Dementia.

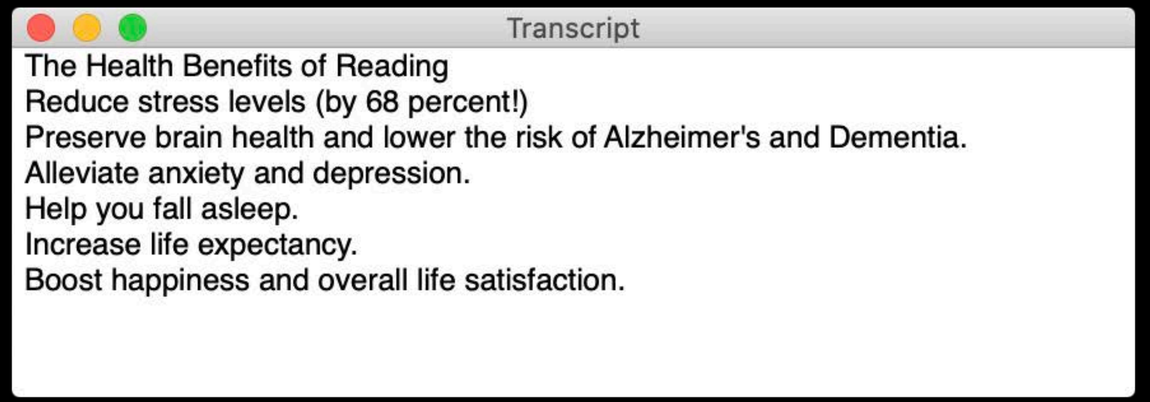
Alleviate anxiety and depression.

Help you fall asleep.

Increase life expectancy.

Boost happiness and overall life satisfaction.


The Health Benefits of Reading
Reduce stress levels (by 68 percent!)
Preserve brain health and lower the risk of Alzheimer's and Dementia.
Alleviate anxiety and depression.
Help you fall asleep.
Increase life expectancy.
Boost happiness and overall life satisfaction.


Transcript
The Health Benefits of Reading
Reduce stress levels (by 68 percent!)
Preserve brain health and lower the risk of Alzheimer's and Dementia.
Alleviate anxiety and depression.
Help you fall asleep.
Increase life expectancy.
Boost happiness and overall life satisfaction.

Our Journey Today

How Text Recognition works

Example applications

Best practices

Our Journey Today

How Text Recognition works

Example applications

Best practices

Our Journey Today

How Text Recognition works

Example applications

Best practices

Our Journey Today

How Text Recognition works

Example applications

Best practices

What It Is and How It Works

Two Paths to Choose From

Two Paths to Choose From

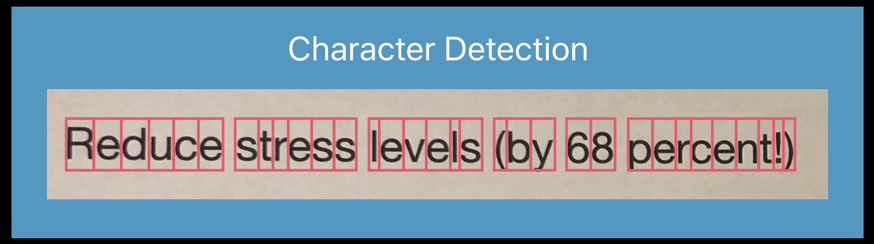
Fast

Accurate

Two Paths to Choose From

Fast

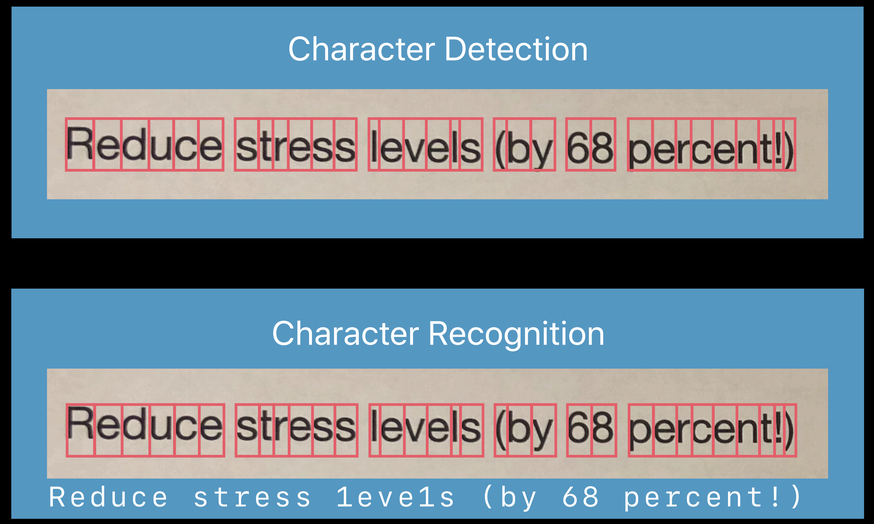
Accurate



Two Paths to Choose From

Fast

Accurate



Two Paths to Choose From

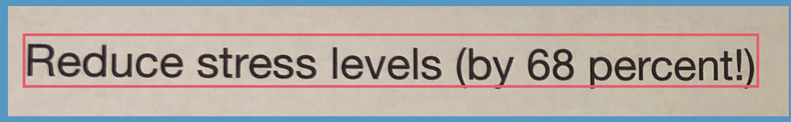
Fast

Character Detection

Character Recognition

Accurate

NN based Text Detection



Two Paths to Choose From

Fast

Character Detection

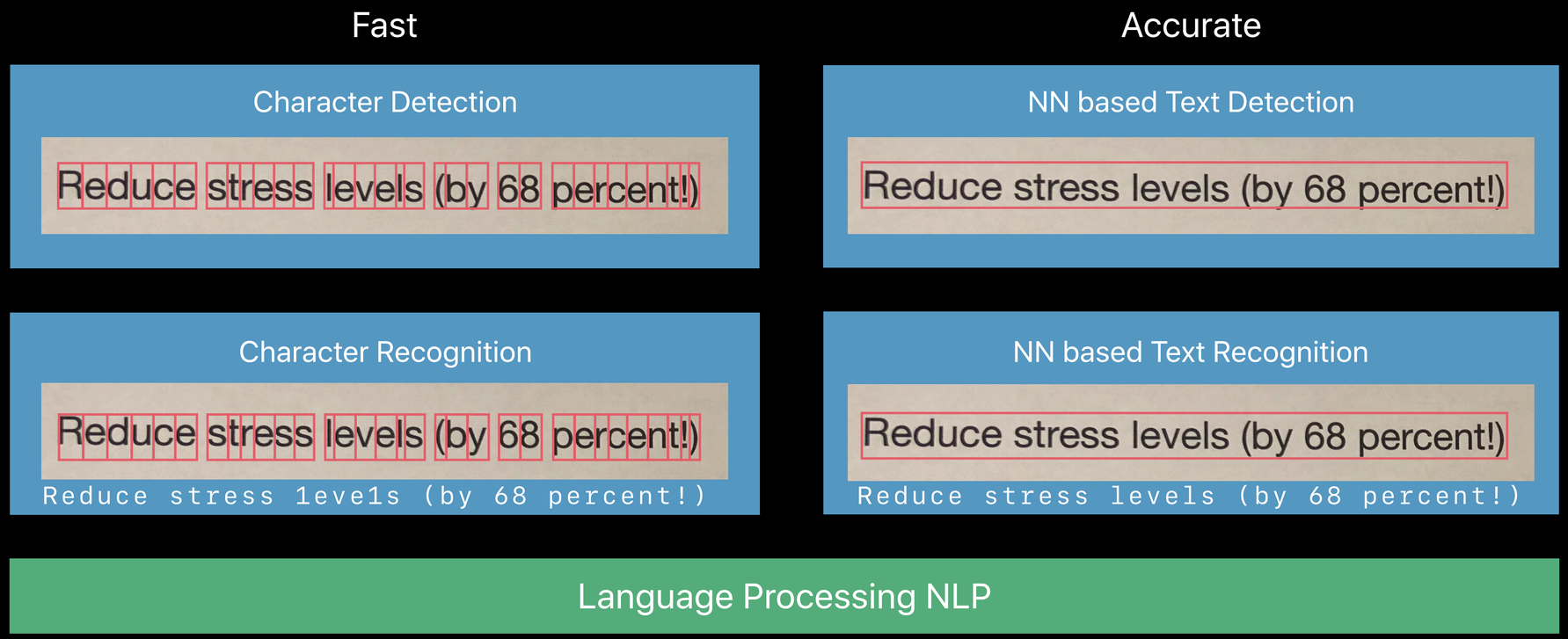
Character Recognition

Accurate

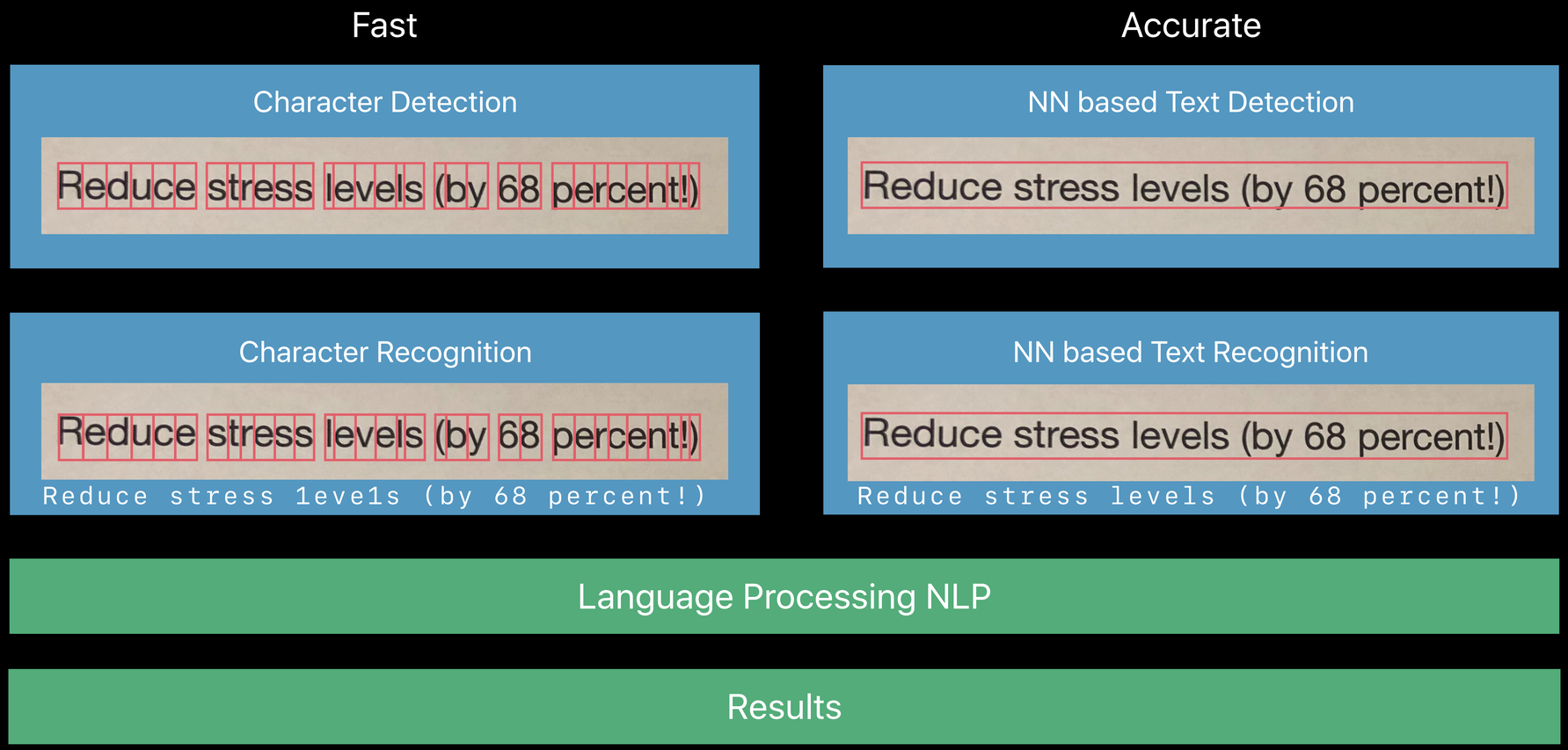
NN based Text Detection

NN based Text Recognition

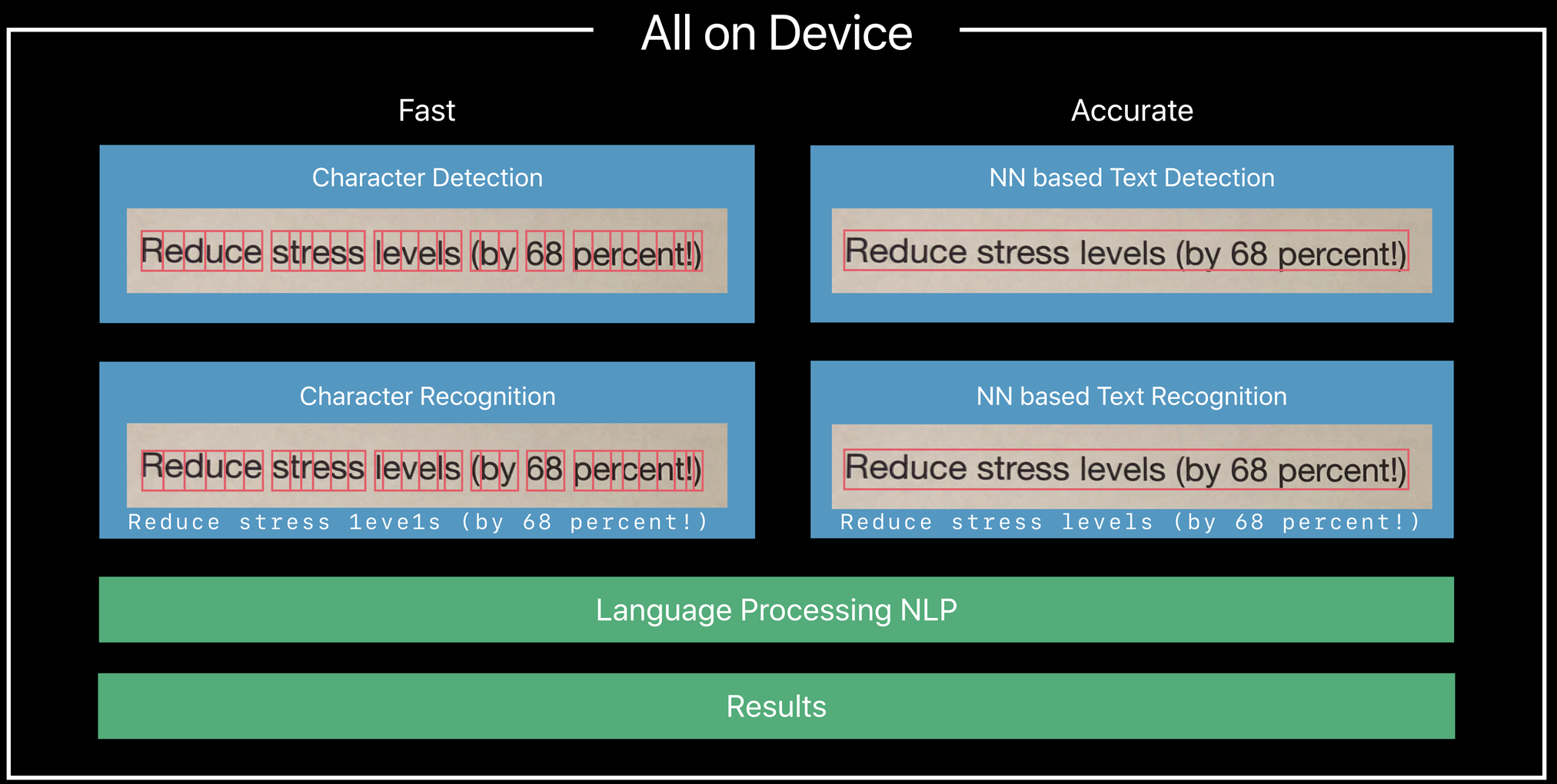
Two Paths to Choose From



Two Paths to Choose From

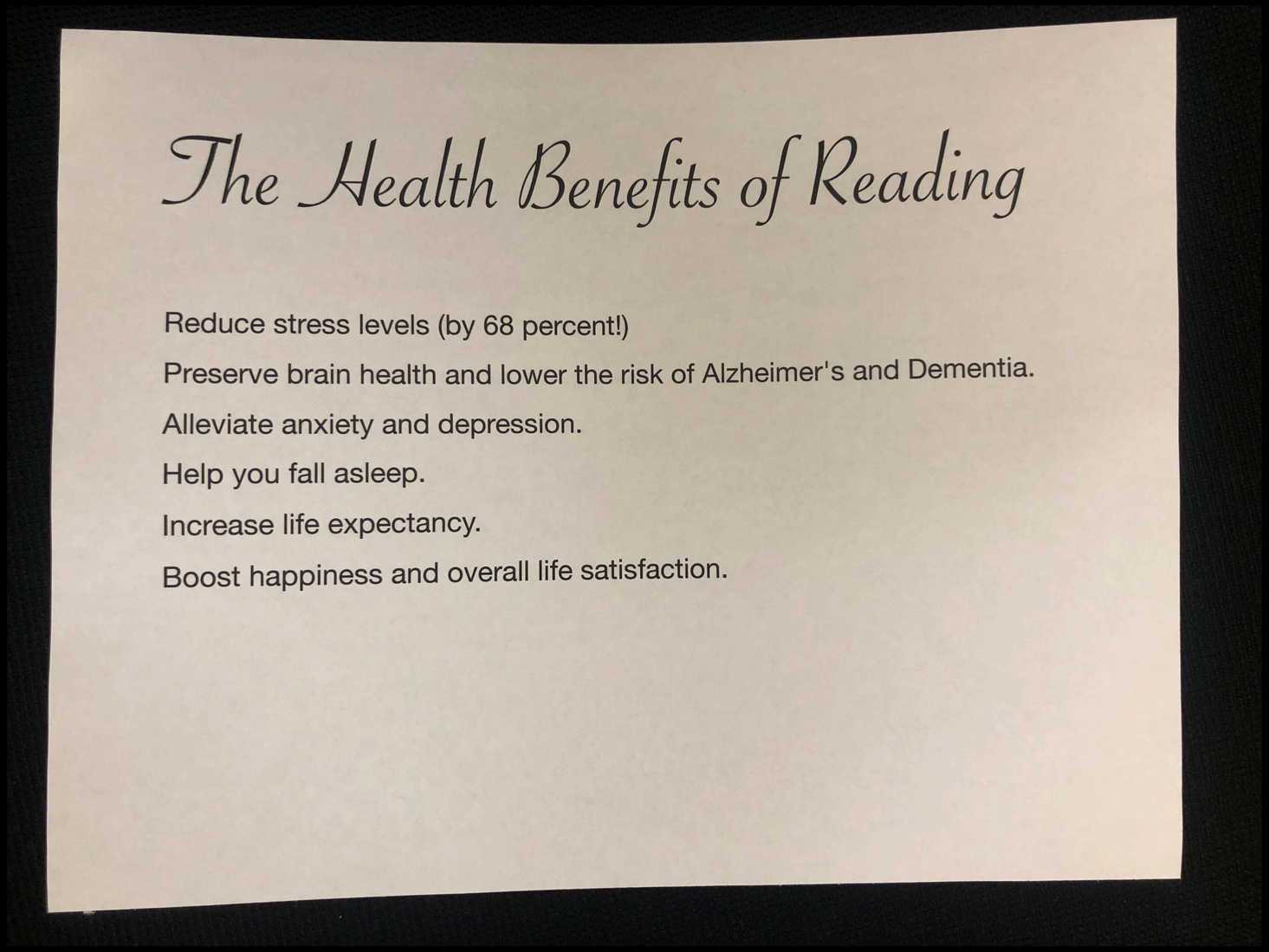


Two Paths to Choose From



Fast Versus Accurate

Fast Versus Accurate



Fast Versus Accurate

Fast Versus Accurate

Fast Versus Accurate

🕐 0.25 sec

🕐 2.0 sec

Fast Versus Accurate

	Fast	Accurate

Fast Versus Accurate

	Fast	Accurate
Processing time	Optimized for real-time	Asynchronous processing

Fast Versus Accurate

	Fast	Accurate
Processing time	Optimized for real-time	Asynchronous processing
Memory footprint	Smallest	Larger

Fast Versus Accurate

	Fast	Accurate
Processing time	Optimized for real-time	Asynchronous processing
Memory footprint	Smallest	Larger
Support for rotated text	Limited	Broad

Fast Versus Accurate

	Fast	Accurate
Processing time	Optimized for real-time	Asynchronous processing
Memory footprint	Smallest	Larger
Support for rotated text	Limited	Broad
Support for variety of fonts	Limited	Diverse font styles

Fast Versus Accurate

	Fast	Accurate
Processing time	Optimized for real-time	Asynchronous processing
Memory footprint	Smallest	Larger
Support for rotated text	Limited	Broad
Support for variety of fonts	Limited	Diverse font styles
Accuracy for natural language	Good	Best

Use Cases Drive How to Configure the Request

Use Cases Drive How to Configure the Request

What is my input?

What are my processing constraints?

What am I going to do with the results?

Use Cases Drive How to Configure the Request

Camera capture

Use Cases Drive How to Configure the Request

Camera capture

- Live capture at high frame rate — go fast

Use Cases Drive How to Configure the Request

Camera capture

- Live capture at high frame rate — go fast
- Opportunistic capture — go accurate

Use Cases Drive How to Configure the Request

Camera capture

- Live capture at high frame rate — go fast
- Opportunistic capture — go accurate
- Text size drives the resolution

Use Cases Drive How to Configure the Request

Post processing

Use Cases Drive How to Configure the Request

Post processing

- Favor accuracy over speed

Reading Codes Versus Reading Natural Language

Language processing

Reading Codes Versus Reading Natural Language

Language processing

- Corrects typical recognition errors

Reading Codes Versus Reading Natural Language

Language processing

- Corrects typical recognition errors
- It can get in the way for codes

Reading Codes Versus Reading Natural Language

Language processing

- Corrects typical recognition errors
- It can get in the way for codes
- Increases the processing time and memory budget

Performing Text Recognition

```
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])
```

```
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// Create request
let myTextRecognitionRequest = VNRecognizeTextRequest()
```

```
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// Create request
let myTextRecognitionRequest = VNRecognizeTextRequest()

// Specify a completion handler
myTextRecognitionRequest.completionHandler = myCompletionHandler
```

```
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// Create request
let myTextRecognitionRequest = VNRecognizeTextRequest()

// Specify a completion handler
myTextRecognitionRequest.completionHandler = myCompletionHandler

// Select the recognition level
myTextRecognitionRequest.recognitionLevel = VNRequestTextRecognitionLevel.fast
```

```
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// Create request
let myTextRecognitionRequest = VNRecognizeTextRequest()

// Specify a completion handler
myTextRecognitionRequest.completionHandler = myCompletionHandler

// Select the recognition level
myTextRecognitionRequest.recognitionLevel = VNRequestTextRecognitionLevel.accurate
```

```
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// Create request
let myTextRecognitionRequest = VNRecognizeTextRequest()

// Specify a completion handler
myTextRecognitionRequest.completionHandler = myCompletionHandler

// Select the recognition level
myTextRecognitionRequest.recognitionLevel = VNRequestTextRecognitionLevel.accurate
```

```
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// Create request
let myTextRecognitionRequest = VNRecognizeTextRequest()

// Specify a completion handler
myTextRecognitionRequest.completionHandler = myCompletionHandler

// Select the recognition level
myTextRecognitionRequest.recognitionLevel = VNRequestTextRecognitionLevel.accurate

// Set the revision
myTextRecognitionRequest.revision = VNRecognizeTextRequestRevision1
```

```
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// Create request
let myTextRecognitionRequest = VNRecognizeTextRequest()

// Specify a completion handler
myTextRecognitionRequest.completionHandler = myCompletionHandler

// Select the recognition level
myTextRecognitionRequest.recognitionLevel = VNRequestTextRecognitionLevel.accurate

// Set the revision
myTextRecognitionRequest.revision = VNRecognizeTextRequestRevision1

// Control language correction
myTextRecognitionRequest.usesLanguageCorrection = false
```

```
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// Create request
let myTextRecognitionRequest = VNRecognizeTextRequest()

// Specify a completion handler
myTextRecognitionRequest.completionHandler = myCompletionHandler

// Select the recognition level
myTextRecognitionRequest.recognitionLevel = VNRequestTextRecognitionLevel.accurate

// Set the revision
myTextRecognitionRequest.revision = VNRecognizeTextRequestRevision1

// Control language correction
myTextRecognitionRequest.usesLanguageCorrection = true
```

```swift
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// Create request
let myTextRecognitionRequest = VNRecognizeTextRequest()

// Specify a completion handler
myTextRecognitionRequest.completionHandler = myCompletionHandler

// Select the recognition level
myTextRecognitionRequest.recognitionLevel = VNRequestTextRecognitionLevel.accurate

// Set the revision
myTextRecognitionRequest.revision = VNRecognizeTextRequestRevision1

// Control language correction
myTextRecognitionRequest.usesLanguageCorrection = true
```

```swift
// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// Create request
let myTextRecognitionRequest = VNRecognizeTextRequest()

// Specify a completion handler
myTextRecognitionRequest.completionHandler = myCompletionHandler

// Select the recognition level
myTextRecognitionRequest.recognitionLevel = VNRequestTextRecognitionLevel.accurate

// Set the revision
myTextRecognitionRequest.revision = VNRecognizeTextRequestRevision1

// Control language correction
myTextRecognitionRequest.usesLanguageCorrection = true

// Send the request to the request handler
try myRequestHandler.perform([myTextRecognitionRequest])
```

```swift
// The results are in the request upon return
guard let results = request.results as? [VNRecognizedTextObservation] else {
      return
     }

// Iterate over the line results
for visionResult in results {
     let maximumCandidates = 1
     guard let candidate = visionResult.topCandidates(maximumCandidates).first else {
         continue
      }
      print(candidate.string)

      // Get the bounding box
      let boundingBox = visionResult.boundingBox

      // Get the bounding box for a specific word
      if let range = candidate.string.range(of: "stress"),
         let boxObservation = try? candidate.boundingBox(for: range) {
          // Draw the box in the view
          let foundTextBox = boxObservation.boundingBox
      }
}
```

```swift
// The results are in the request upon return
guard let results = request.results as? [VNRecognizedTextObservation] else {
      return
     }

// Iterate over the line results
for visionResult in results {
     let maximumCandidates = 1
     guard let candidate = visionResult.topCandidates(maximumCandidates).first else {
         continue
      }
      print(candidate.string)

      // Get the bounding box
      let boundingBox = visionResult.boundingBox

      // Get the bounding box for a specific word
      if let range = candidate.string.range(of: "stress"),
         let boxObservation = try? candidate.boundingBox(for: range) {
          // Draw the box in the view
          let foundTextBox = boxObservation.boundingBox
      }
}
```

```swift
// The results are in the request upon return
guard let results = request.results as? [VNRecognizedTextObservation] else {
      return
     }

// Iterate over the line results
for visionResult in results {
     let maximumCandidates = 1
     guard let candidate = visionResult.topCandidates(maximumCandidates).first else {
         continue
      }
      print(candidate.string)

      // Get the bounding box
      let boundingBox = visionResult.boundingBox

      // Get the bounding box for a specific word
      if let range = candidate.string.range(of: "stress"),
         let boxObservation = try? candidate.boundingBox(for: range) {
          // Draw the box in the view
          let foundTextBox = boxObservation.boundingBox
      }
}
```

```swift
// The results are in the request upon return
guard let results = request.results as? [VNRecognizedTextObservation] else {
      return
     }

// Iterate over the line results
for visionResult in results {
      let maximumCandidates = 1
      guard let candidate = visionResult.topCandidates(maximumCandidates).first else {
          continue
       }
       print(candidate.string)

       // Get the bounding box
       let boundingBox = visionResult.boundingBox

       // Get the bounding box for a specific word
       if let range = candidate.string.range(of: "stress"),
          let boxObservation = try? candidate.boundingBox(for: range) {
           // Draw the box in the view
           let foundTextBox = boxObservation.boundingBox
       }
}
```

Reduce stress levels (by 68 percent!)

```swift
// The results are in the request upon return
guard let results = request.results as? [VNRecognizedTextObservation] else {
      return
     }

// Iterate over the line results
for visionResult in results {
     let maximumCandidates = 1
     guard let candidate = visionResult.topCandidates(maximumCandidates).first else {
         continue
      }
      print(candidate.string)

      // Get the bounding box
      let boundingBox = visionResult.boundingBox

      // Get the bounding box for a specific word
      if let range = candidate.string.range(of: "stress"),
         let boxObservation = try? candidate.boundingBox(for: range) {
          // Draw the box in the view
          let foundTextBox = boxObservation.boundingBox
      }
}
```

Reduce stress levels (by 68 percent!)

```swift
// The results are in the request upon return
guard let results = request.results as? [VNRecognizedTextObservation] else {
      return
     }

// Iterate over the line results
for visionResult in results {
     let maximumCandidates = 1
     guard let candidate = visionResult.topCandidates(maximumCandidates).first else {
         continue
      }
      print(candidate.string)

      // Get the bounding box
      let boundingBox = visionResult.boundingBox

      // Get the bounding box for a specific word
      if let range = candidate.string.range(of: "stress"),
         let boxObservation = try? candidate.boundingBox(for: range) {
          // Draw the box in the view
          let foundTextBox = boxObservation.boundingBox
      }
}
```

Real-Time Text Recognition

When to Use .fast

Use case

When to Use .fast

Use case

- Read codes/serial numbers just like a barcode reader

When to Use .fast

Use case

- Read codes/serial numbers just like a barcode reader
- Constrained camera usage

When to Use .fast

Use case

- Read codes/serial numbers just like a barcode reader
- Constrained camera usage
- Interactivity is key

When to Use .fast

Use case

- Read codes/serial numbers just like a barcode reader
- Constrained camera usage
- Interactivity is key

```
request = VNRecognizeTextRequest(completionHandler: recognizeTextHandler)
request.recognitionLevel = .fast
```

Demo

Phone Number Reader

Demo Recap

Fast path to maintain frame rate

Guide the user on how to use the camera

Use ROI to target the recognition area

Disable language correction

Use our domain knowledge of phone numbers

Build evidence over time to reduce the noise

Document Camera

The perfect companion

Notes Document Camera Available for Everyone

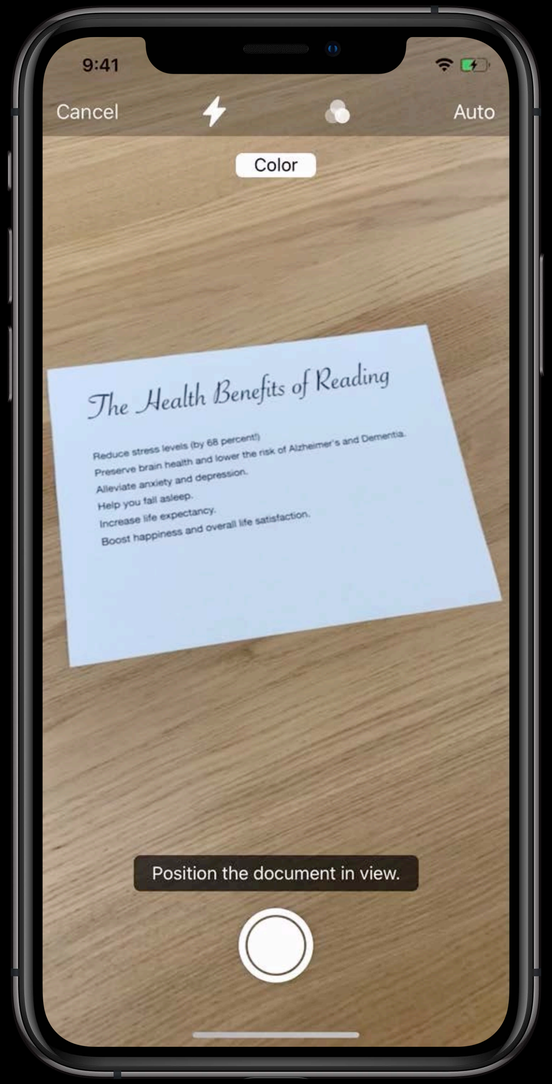
Notes Document Camera Available for Everyone

Used in Notes, Mail, Files, and Messages

Notes Document Camera Available for Everyone

Used in Notes, Mail, Files, and Messages

Perfect for reading documents



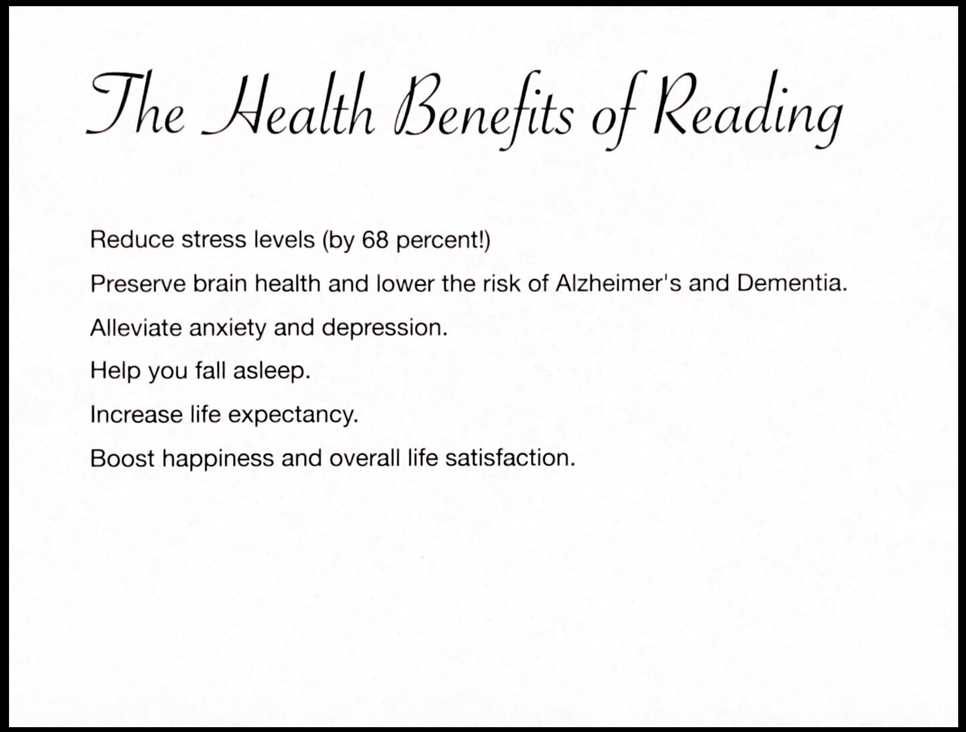
Notes Document Camera Available for Everyone

Used in Notes, Mail, Files, and Messages

Perfect for reading documents

Provides cleaned up “scans“ of your document

- Perspective corrected
- Evenly lit



```
import Vision
import VisionKit

let documentCameraViewController = VNDocumentCameraViewController()
documentCameraViewController.delegate = self
present(documentCameraViewController, animated: true)

…
extension ViewController: VNDocumentCameraViewControllerDelegate {

    public func documentCameraViewController(_ controller: VNDocumentCameraViewController,
                                             didFinishWith scan: VNDocumentCameraScan) {

        textRecognitionWorkQueue.async {
            for pageIndex in 0 ..< scan.pageCount {
                let image = scan.imageOfPage(at: pageIndex)
                if let cgImage = image.cgImage {
                    let requestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:])
                    do {
                        try requestHandler.perform(self.requests)
                    } catch {
                        print(error)
                    }
                }
            }
    }
}
```

```swift
import Vision
import VisionKit

let documentCameraViewController = VNDocumentCameraViewController()
documentCameraViewController.delegate = self
present(documentCameraViewController, animated: true)
```

…

```swift
extension ViewController: VNDocumentCameraViewControllerDelegate {

    public func documentCameraViewController(_ controller: VNDocumentCameraViewController,
                                             didFinishWith scan: VNDocumentCameraScan) {

        textRecognitionWorkQueue.async {
            for pageIndex in 0 ..< scan.pageCount {
                let image = scan.imageOfPage(at: pageIndex)
                if let cgImage = image.cgImage {
                    let requestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:])
                    do {
                        try requestHandler.perform(self.requests)
                    } catch {
                        print(error)
                    }
                }
            }
    }
}
```

```
import Vision
import VisionKit

let documentCameraViewController = VNDocumentCameraViewController()
documentCameraViewController.delegate = self
present(documentCameraViewController, animated: true)

…

extension ViewController: VNDocumentCameraViewControllerDelegate {

    public func documentCameraViewController(_ controller: VNDocumentCameraViewController,
                                             didFinishWith scan: VNDocumentCameraScan) {

        textRecognitionWorkQueue.async {
            for pageIndex in 0 ..< scan.pageCount {
                let image = scan.imageOfPage(at: pageIndex)
                if let cgImage = image.cgImage {
                    let requestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:])
                    do {
                        try requestHandler.perform(self.requests)
                    } catch {
                        print(error)
                    }
                }
            }
    }
}
```

```
import Vision
import VisionKit

let documentCameraViewController = VNDocumentCameraViewController()
documentCameraViewController.delegate = self
present(documentCameraViewController, animated: true)

…
extension ViewController: VNDocumentCameraViewControllerDelegate {

    public func documentCameraViewController(_ controller: VNDocumentCameraViewController,
                                             didFinishWith scan: VNDocumentCameraScan) {

        textRecognitionWorkQueue.async {
            for pageIndex in 0 ..< scan.pageCount {
                let image = scan.imageOfPage(at: pageIndex)
                if let cgImage = image.cgImage {
                    let requestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:])
                    do {
                        try requestHandler.perform(self.requests)
                    } catch {
                        print(error)
                    }
                }
            }
    }
}
```

```swift
import Vision
import VisionKit

let documentCameraViewController = VNDocumentCameraViewController()
documentCameraViewController.delegate = self
present(documentCameraViewController, animated: true)

…
extension ViewController: VNDocumentCameraViewControllerDelegate {

    public func documentCameraViewController(_ controller: VNDocumentCameraViewController,
                                             didFinishWith scan: VNDocumentCameraScan) {

        textRecognitionWorkQueue.async {
            for pageIndex in 0 ..< scan.pageCount {
                let image = scan.imageOfPage(at: pageIndex)
                if let cgImage = image.cgImage {
                    let requestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:])
                    do {
                        try requestHandler.perform(self.requests)
                    } catch {
                        print(error)
                    }
                }
            }
    }
}
```

Best Practices For Text Recognition

Cédric Bray, Vision Team

Language knowledge

Performance

Processing results

Leverage Language Knowledge

Language-based correction

- Specify the language
- Improve transcription with on-device language models

Leverage Language Knowledge

Custom lexicon

- Use custom vocabulary for domain-specific text
- Increase recognition accuracy of confusable or illegible text

```
// Check supported languages for current revision

let languages = VNRecognizeTextRequest.supportedRecognitionLanguages(
    for: VNRequestTextRecognitionLevel.accurate,
    revision: VNRecognizeTextRequestRevision1)
```

```
// Check supported languages for current revision

let languages = VNRecognizeTextRequest.supportedRecognitionLanguages(
    for: VNRequestTextRecognitionLevel.accurate,
    revision: VNRecognizeTextRequestRevision1)

// Enable language-based correction
myTextRecognitionRequest.usesLanguageCorrection = true
```

```
// Check supported languages for current revision

let languages = VNRecognizeTextRequest.supportedRecognitionLanguages(
    for: VNRequestTextRecognitionLevel.accurate,
    revision: VNRecognizeTextRequestRevision1)

// Enable language-based correction
myTextRecognitionRequest.usesLanguageCorrection = true

// Specify custom lexicon words
myTextRecognitionRequest.customWords = [ "1337", "h4x0r", "sp34k"]
```

Tune for Better Performance

Minimum text height

- Use when it is acceptable to ignore text below a certain size
- Improves execution time and memory usage by downscaling the image

```
myTextRecognitionRequest.minimumTextHeight = 0.5 // Text bigger than 50% of the image height
```

The Case of Background Tasks

CPU versus GPU/Neural Engine

- Configure for CPU only
- Leaves faster resources for higher-priority tasks

```
myTextRecognitionRequest.usesCPUOnly = true // Run on CPU only
```

Manage Progress

Progress updates

```
myTextRecognitionRequest.progressHandler = myProgressHandler
```

Cancellation

```
myTextRecognitionRequest.cancel()
```

Demo

My First Image Reader

Demo Recap

Pick recognition level that fits desired user interaction

Configure language support for your targeted use case

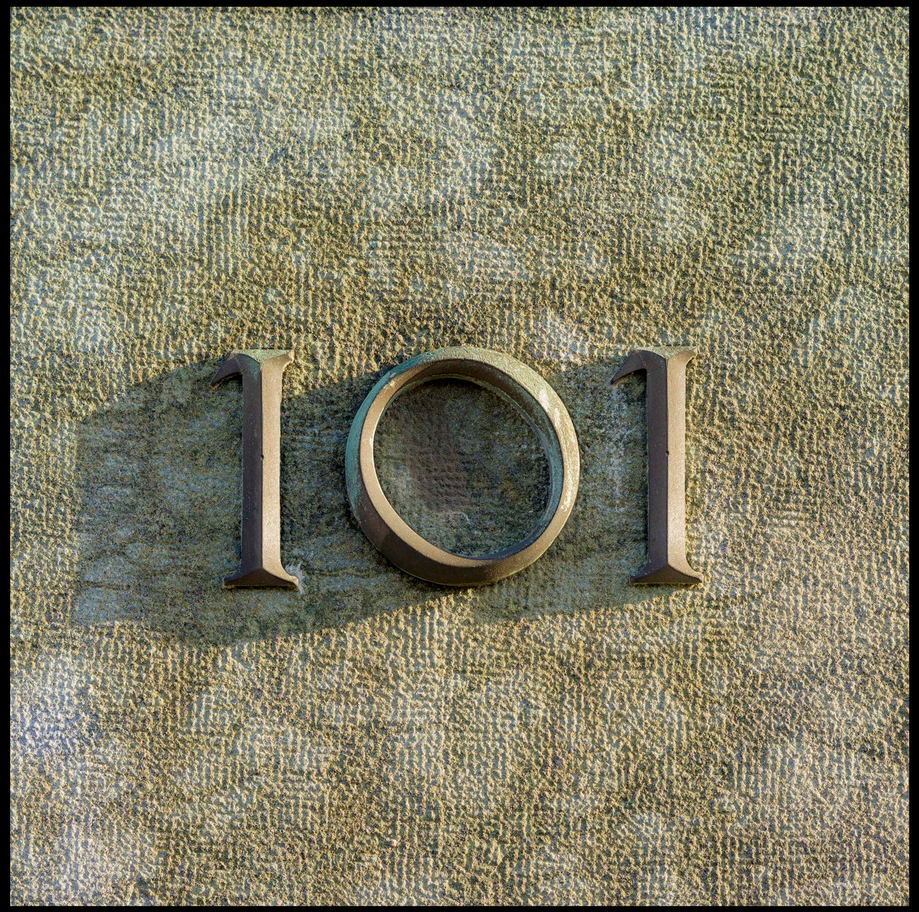
Leverage progress updates

Processing Results

Expect Ambiguity in the Input

The case of house numbers

- lol
- lOl
- 101



Dive into the Candidate List

VNRecognizedTextObservation

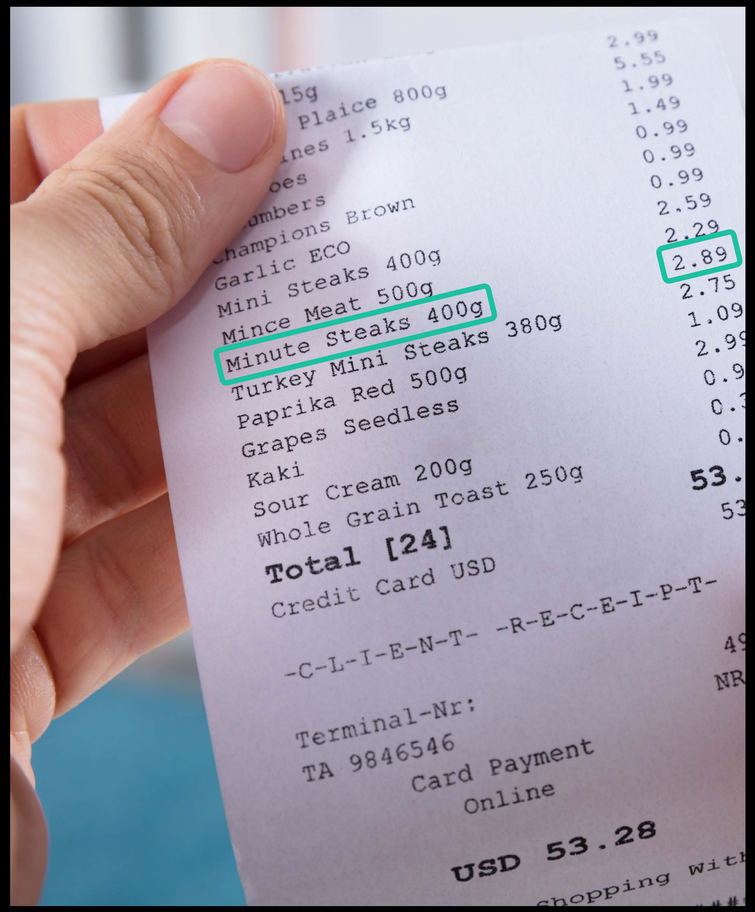
- Process transcription candidates

```
let maxCount = 3
let candidates = currentObservation.topCandidates(maxCandidateCount: maxCount)
```

Use Geometry to Map Results

Compare spatial information

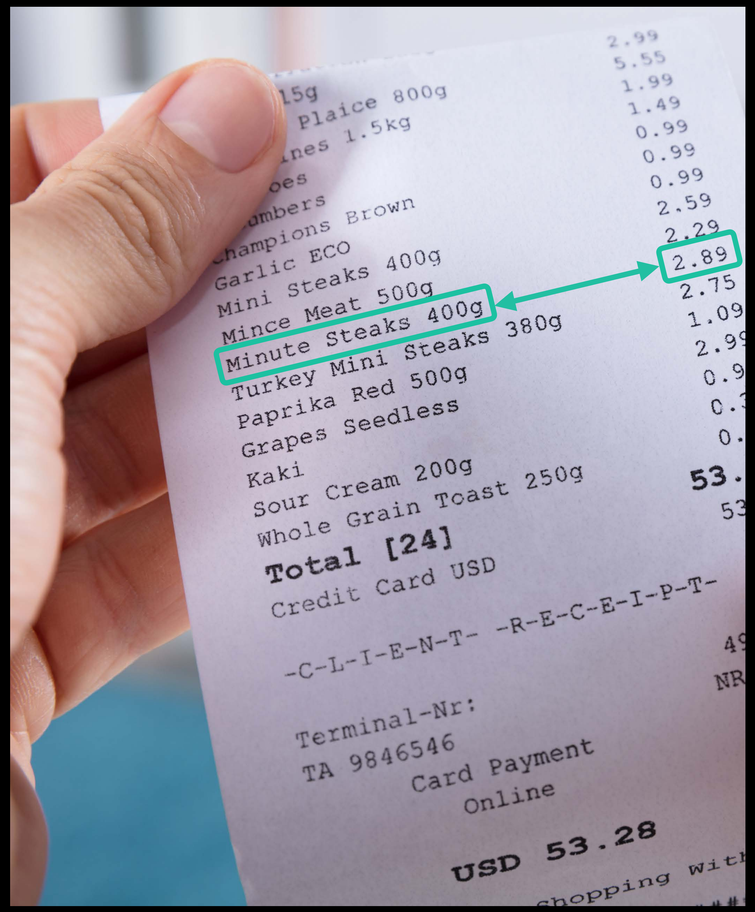
- Position and scale
- Rotation



Use Geometry to Map Results

Compare spatial information

- Position and scale
- Rotation



Use Parsers to Label Results

Data Detectors

- NSDataDetector for types of interest
- Addresses, URLs, dates, and phone numbers

Use Parsers to Label Results

Data Detectors

- NSDataDetector for types of interest
- Addresses, URLs, dates, and phone numbers

Use Parsers to Label Results

Domain-specific filters

- Your own lexicon
- Regular expressions

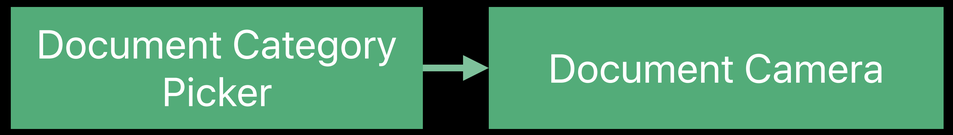
A Business Companion App

Business Companion App

Business Companion App

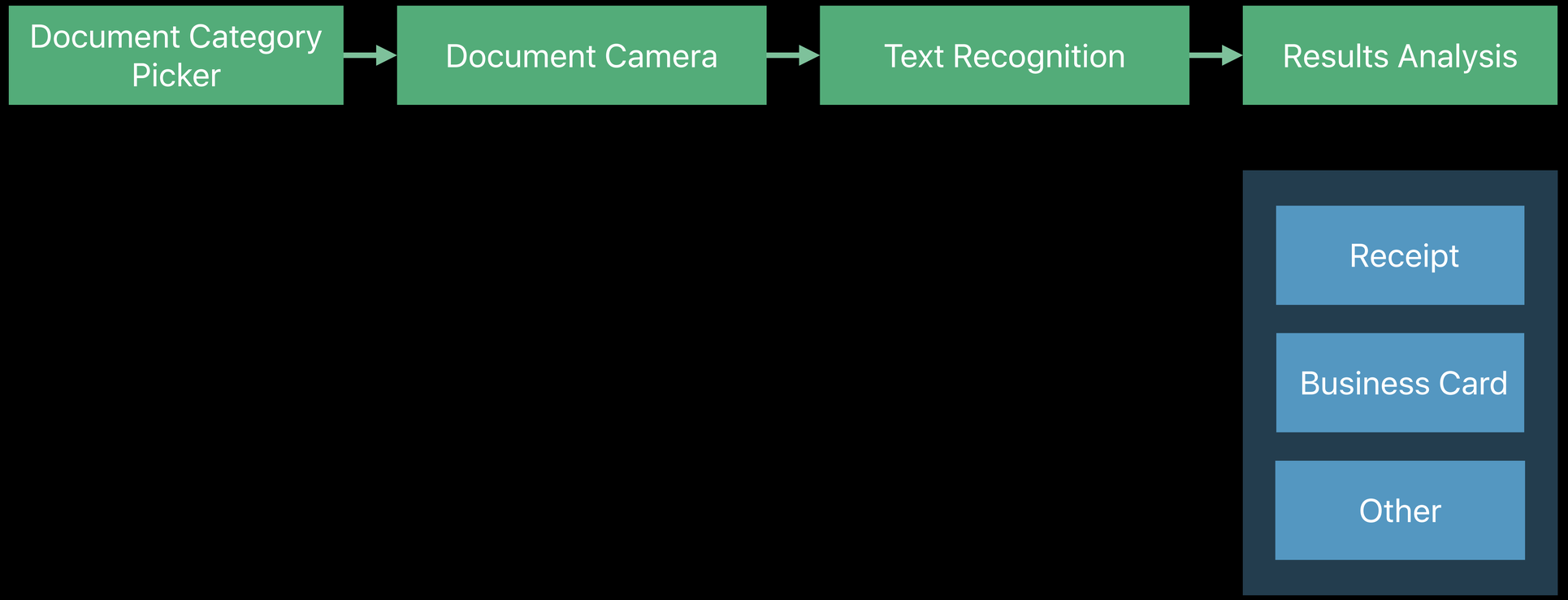
Document Category Picker

Business Companion App

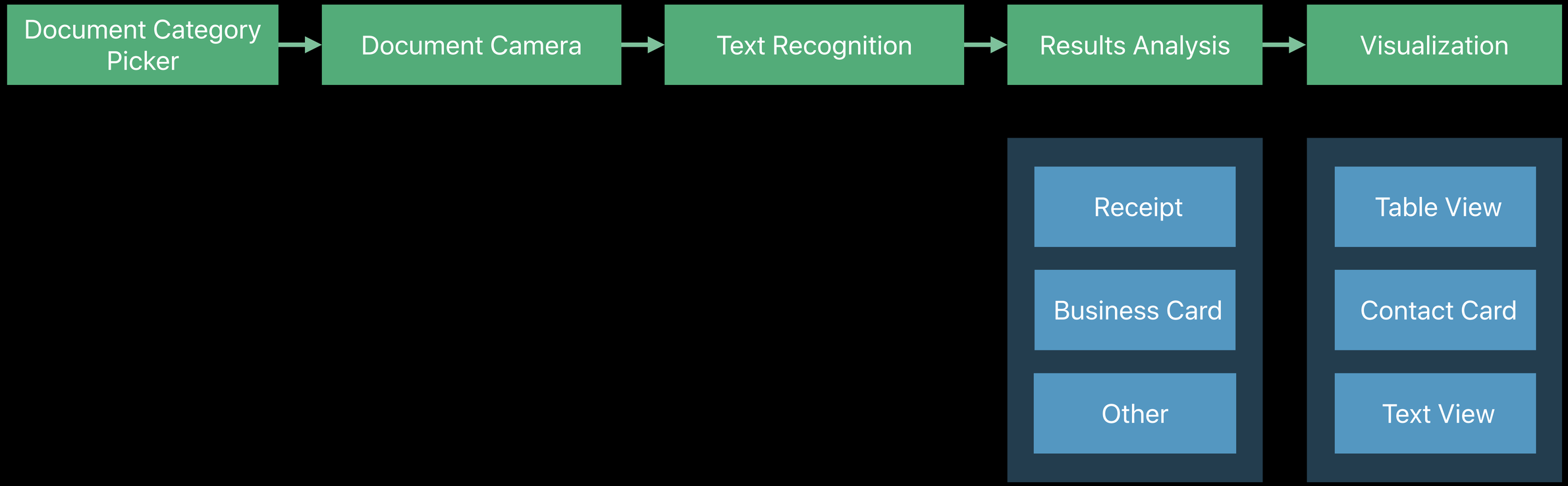


Business Companion App

Business Companion App



Business Companion App



Demo

Business Companion

Demo Recap

Use geometric information to map results

Use Data Detectors, existing language API, or regular expressions

Use your domain knowledge

More Information

developer.apple.com/wwdc19/234

Machine Learning Lab

Friday, 2:00


WWDC19