CoreML : Everyone's an Artist

Building a Neural Style Transfer app with CoreML

Alexis Creuzot

What is Neural Style Transfer?

(Or NST)

+

Obtaining a CoreML NST model

Online*

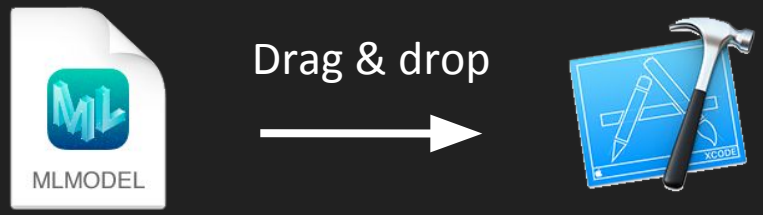
- Github, modelzoo.co…

DIY

- GPU + some ❤️ for Python
- ML framework : Turi Create, Tensorflow, Pytorch…
- Other tools : ONNX, coremltools

*check the licence!

Importing a CoreML model



Xcode provides

- Model Metadata
- Helper class

Metadata

Machine Learning Model

Name	StarryNight
Type	Neural Network
Size	854 KB
Author	Monoqle
Description	starry_night, linear 16 bit
License	Commercial, all rights reserved

Model Class

C StarryNight

Automatically generated Swift model class

Model Evaluation Parameters

Name	Type	Description
Inputs		
inputImage	Image (Color 720 x 720)	Image to stylize
Outputs		
outputImage	Image (Color 720 x 720)	Stylized image

Helper class

StarryNight.swi

```
1  //
2  // StarryNight.sw
3  //
4  // This file was                                   d not be edite
5  //
6
7  import CoreML
8
9
10 /// Model Predict
11 @available(macOS                                   chOS 4.2, *)
12 class StarryNight
13
14     /// Image to                                   ype_32BGRA) im
15     var inputImag
16
17     var featureNa
18         get {
19             retur
20         }
21     }
22
23     func featureV                                   MLFeatureValue
24         if (featu
25             retur                                   utImage)
26         }
27         return nil
```

- C StarryNightInput
  - P inputImage
  - P featureNames
  - M featureValue(for:)
  - M init(inputImage:)
- C StarryNightOutput
  - P provider
  - P outputImage
  - P featureNames
  - M featureValue(for:)
  - M init(outputImage:)
  - M init(features:)
- C StarryNight
  - P model
  - P urlOfModelInThisBundle
  - M init(contentsOf:)
  - M init()
  - M init(configuration:)
  - M init(contentsOf:configuration:)
  - M prediction(input:)
  - M prediction(input:options:)
  - M prediction(inputImage:)
  - M predictions(inputs:options:)

Using a CoreML model

Specific API for CoreML

In our case :

1. Convert input (UIImage) to CoreML format (PixelBuffer)
2. Make prediction
3. Convert output (PixelBuffer) back to target format (UIImage)

Optimizations

- Threading
- Memory offloading to disk

Model encapsulation

- Allow model parameters variability
- Better separation of concern

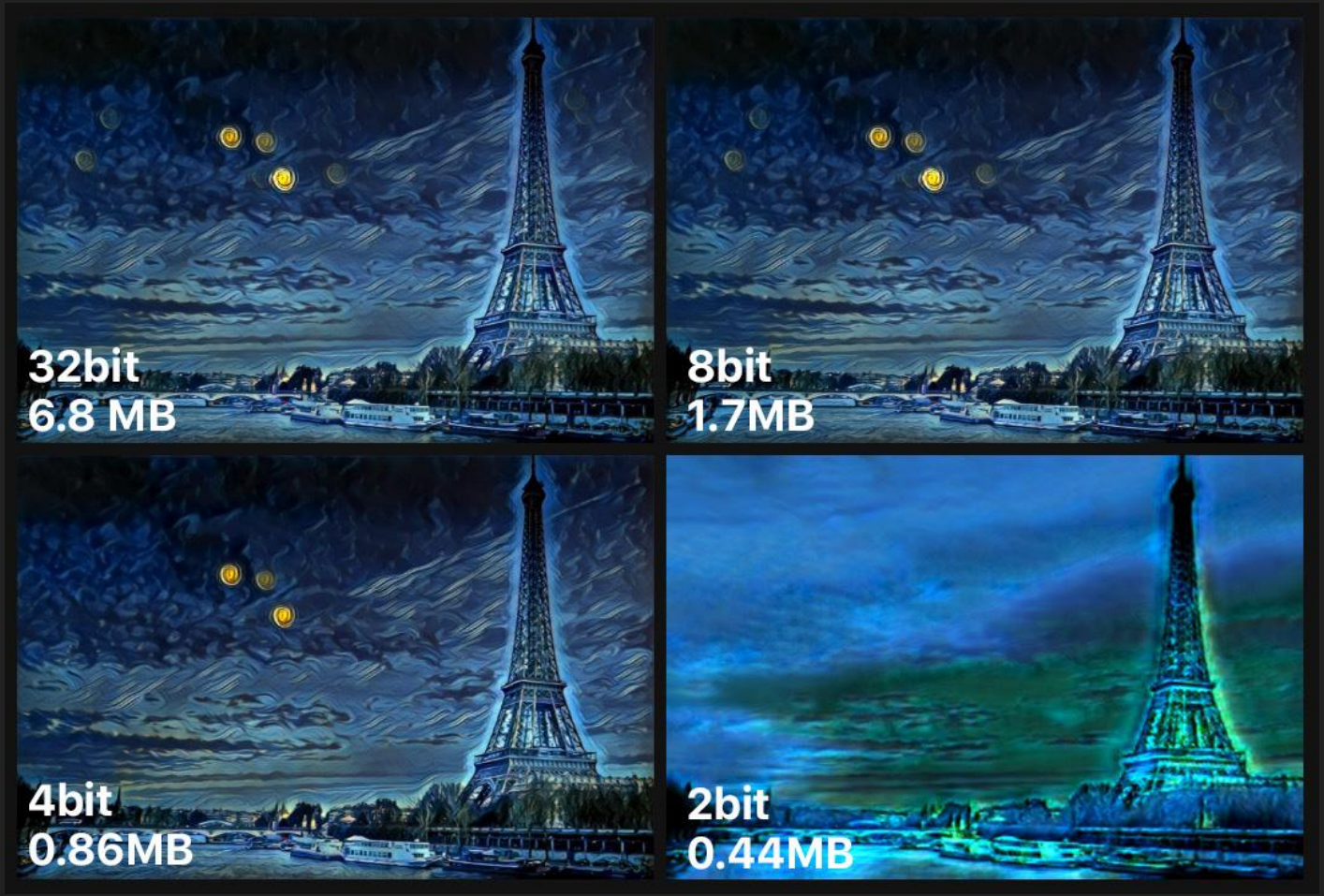
Note: Xcode compiles models as .modelc files on the file system

Livecoding

Demo for a Neural Style Transfer app

One more thing

Quantization


32bit
6.8 MB
8bit
1.7MB
4bit
0.86MB
2bit
0.44MB

Resources

Going further

- NST demo sources (github.com/kirualex/NSTDemo)
- Building a Neural Style Transfer app on iOS (bit.ly/building_nst_ios)
- Using Quantization in iOS 12 (bit.ly/quantization_ios)

Apple

- What's new in CoreML part 1 & 2 (WWDC 18)
- A guide to Turi Create (WWDC 18)

Other

- Jcjohnson (github.com/jcjohnson/fast-neural-style)

Thank you

@alexiscreuzot

@kirualex

looq.monoqle.fr