YOLOV4 TensorFlow Lite

YOLO V4

Yolo é um modelo de rede neural convolucional para detecção de objetos em tempo real. O modelo utiliza o *framework "You Only Look Once"* e é baseado na rede *Darknet53*.

Rede Utilizada

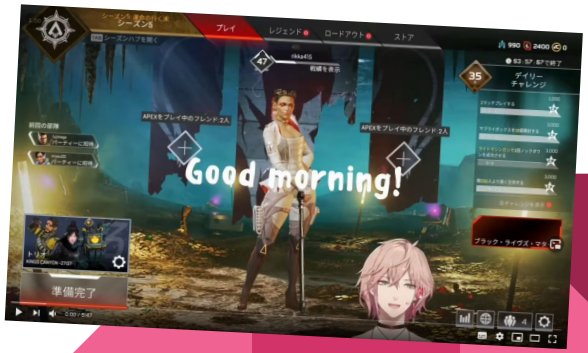
Optamos primeiramente por treinar uma rede para detecção de objetos customizados utilizando a rede *YoloV3* com a rede inicial *darknet53.conv.74*.

Pipeline para o treinamento da rede:

- Extração dos *frames* de imagem;
- Anotação de imagens e composição do *dataset*;
- Treinamento da rede;
- Validação da rede.

Detecção de Modelos de “Youtubers” Virtuais

Em 2020, devido às condições de quarentena, houve um aumento no consumo de entretenimento digital, e com isso, houve o crescimento e surgimento de diversos conteúdos criativos e inusitados na internet. Um deles foi a implementação de personagens animados fictícios, onde a pessoa por trás utiliza esse personagem ao invés de se expor. (Claramente algo que veio do Japão, país que possui diversas peculiaridades esquisitas e criativas).



Resultado (*Dataset* de 1600 *Samples* para 5 “Youtubers” Virtuais)

Rede Pré-treinada para 80 Objetos do *Yolov4*

- O tamanho do *.weights* é de 256MB
- Alguns *Labels*:
    - notebook;
    - pessoa;
    - livro;
    - mouse;
    - celular;
    - garfo;
    - faca.

Otimizações do *TensorFlow Lite*

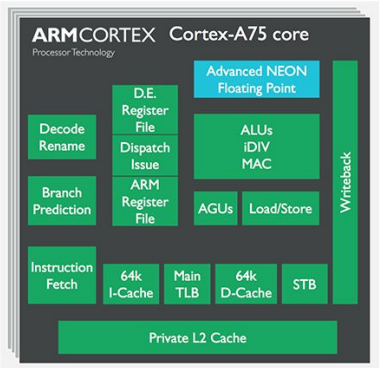
- *Dynamic Quantization*
  - converte os pesos para *int;*
  - executa como *float*.
- *Full integer Quantization*
  - converte parte ou toda rede para *int;*
  - executa como *int*, ou como *float* nas partes que não puderam ser convertidas;
  - necessita de um conjunto de dados representativo.
- *Float16 Quantization*
  - converte os pesos para *float16;*
  - ele é executado como *float32*, a menos que o *hardware* tenha as instruções para *float16*.

Dispositivos Usados

- Motorola Moto G4 Play (2016);
- Samsung Galaxy S9+ (2018).
    - *Advanced NEON Floating Point*.

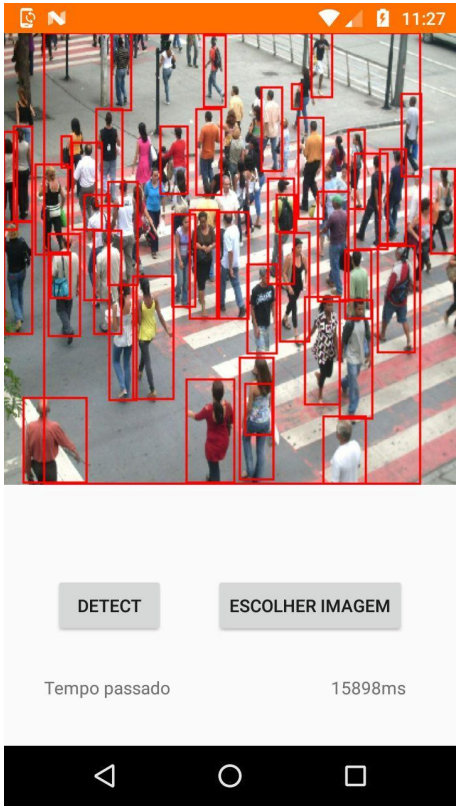
Additional capabilities for NEON/FPU

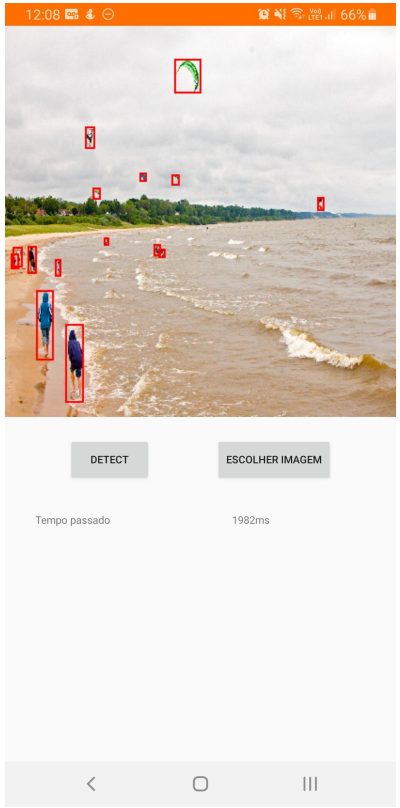
- Dedicated renaming engine for NEON/FPU
- Support for FP16 half-precision processing
  - Double throughput compared to single precision
  - Significant performance uplift for image processing
- Support for Int8 dot product
  - Increased performance on neural network algorithms
- Enhanced floating-point MAC throughput
- Dedicated data store queue



  ARM

Saída da Rede





Métricas dos Experimentos

	Tempo		Tamanho do Modelo *tflite*
Dispositivo	Motorola Moto G4 play	Samsung Galaxy S9+	
Rede com Pesos *int*	15502 ms	1870 ms	124 MB
Rede com *float16*	18503 ms	1990 ms	62 MB

Conclusão

Aprendemos que converter a rede para *tflite* não é tão linear assim, já que nem todas as operações estão implementadas no *tflite*, necessitando a criação dessas operações para poder executar no interpretador. O resultado da rede *yoloV4* foi bom, mas a demora da execução é um problema. Apesar de não termos conseguido aplicar as otimizações que diminuem a latência, podemos ver que o uso da *dynamic quantization* pode diminuir o tamanho da rede de forma considerável.

Grupo 5

Fernando Akio Tutume de Salles Pucci	**(8957197)**
Lucas Nobuyuki Takahashi	**(10295670)**
Vitor Kodhi Teruya	**(10284441)**