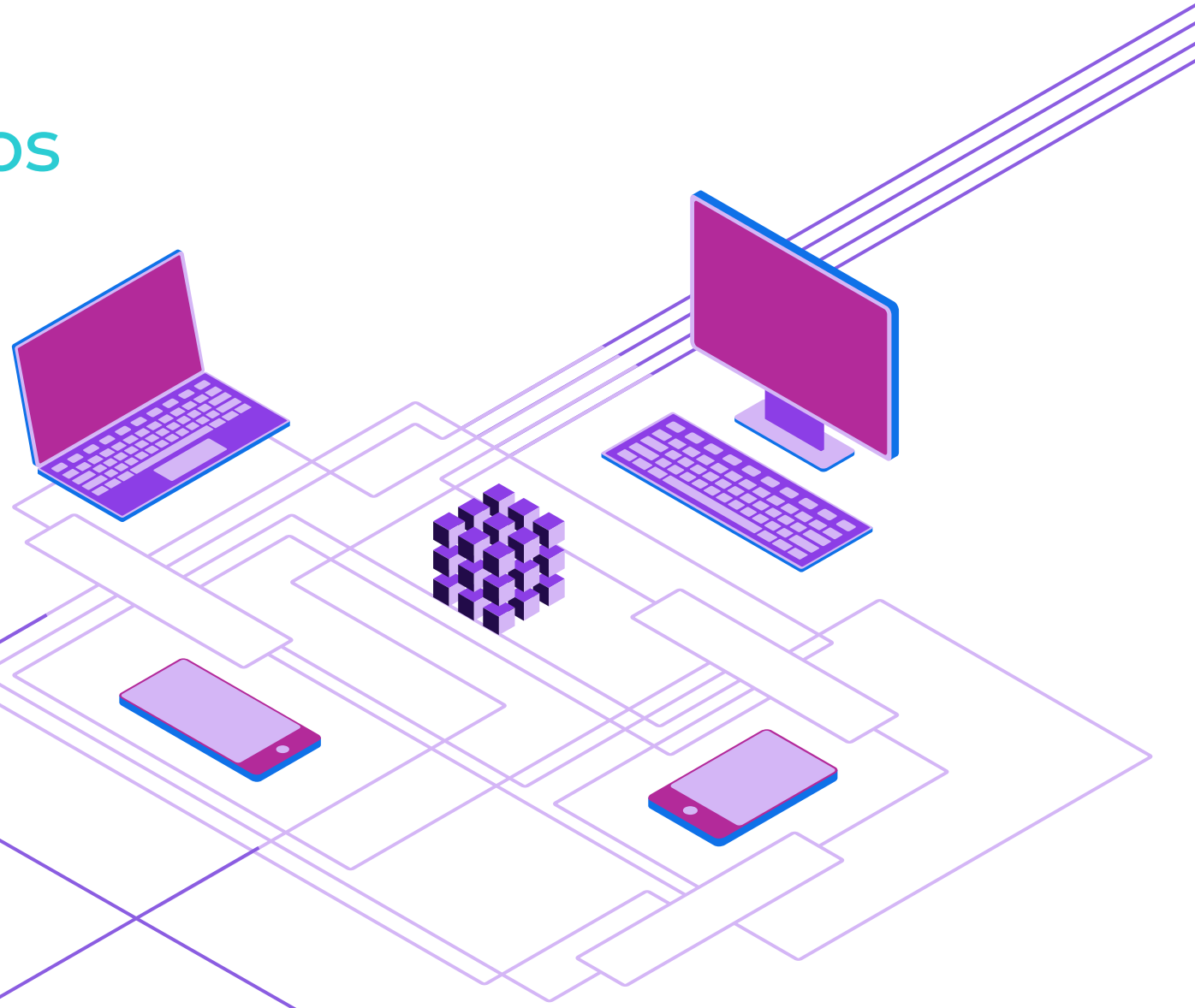
plain concepts

Optimización de modelos de Deep Learning

Mejora la eficiencia y el rendimiento de tus predicciones



plain concepts

Alexander González

AI SOFTWARE ENGINEER

Geek del cloud y IA. Especialmente de todo lo que tenga que ver con tratamiento de imágenes/videos. Tinerfeño de pura cepa. Siempre abierto al conocimiento, a aprender, observar, y conseguir lo mejor de mí mismo y compartirlo con los demás. ¡Interesado todo el tiempo en escuchar a maestros, desarrolladores, geeks o personas que simplemente son interesantes y creativas!

@alexndrglez

plain concepts

Agenda

¿Por que surge todo esto de la "cuantización" o aceleración de los procesos de inferencia?

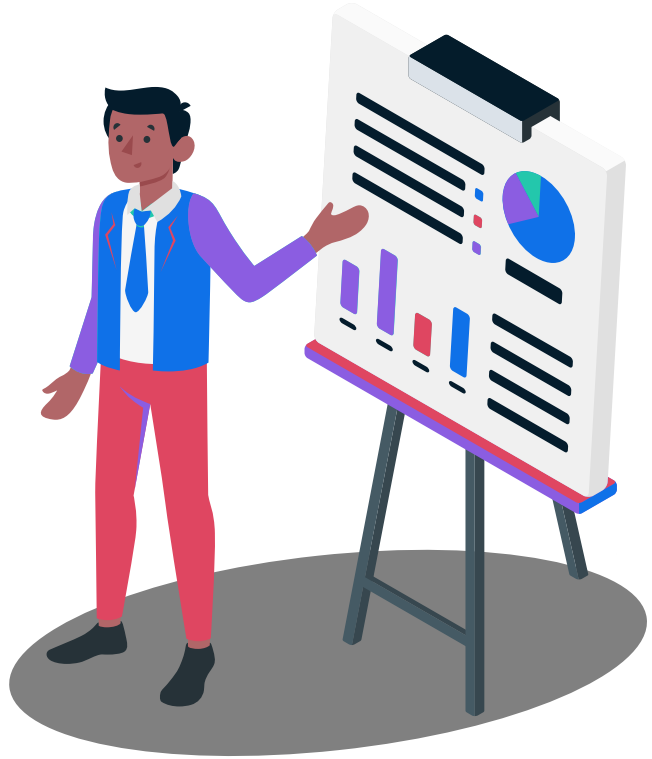
¿De que me sirve cuantizar o optimizar mis modelos?

¿Que beneficios obtengo al optimizar?

Tipos de optimizaciones actuales

¿Que librerías existen actualmente para realizar estas optimizaciones?

Pongámoslo en practica

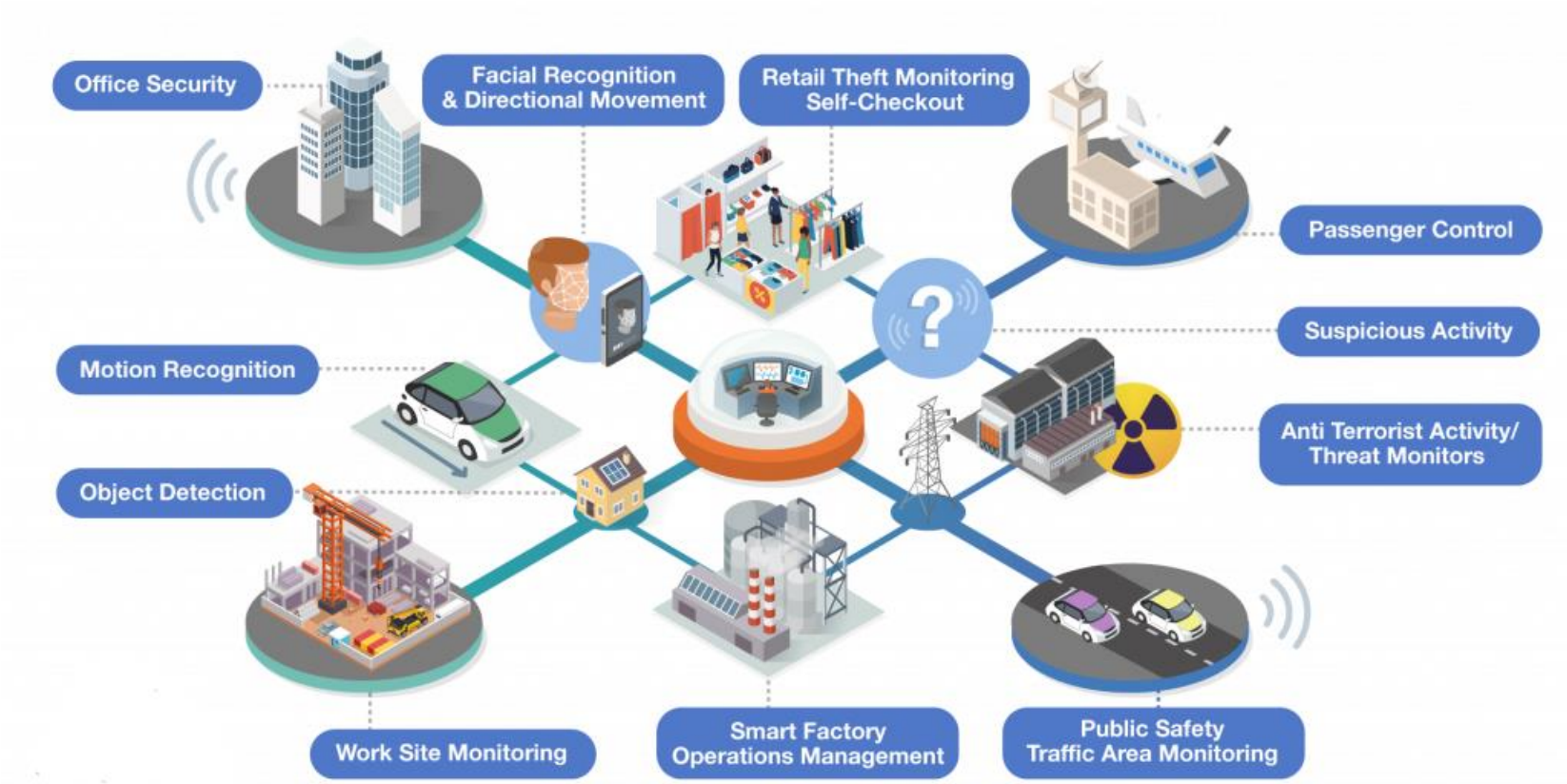


plain concepts

¿Por que surge todo esto de la "cuantización" o aceleración de los procesos de inferencia?

plain concepts

IoT



plain concepts

Velocidad y eficiencia

plain concepts

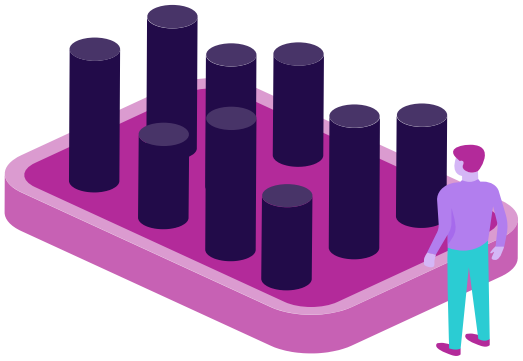
¿De que sirve optimizar mis modelos?

¿Que beneficios obtengo al optimizar?

- Tamaño de almacenamiento más pequeño
- Tamaño de descarga más pequeño
- Menos uso de memoria
- Reducción de latencia
- Compatibilidad con aceleradores

Tipos de optimizaciones

Pruning o recorte



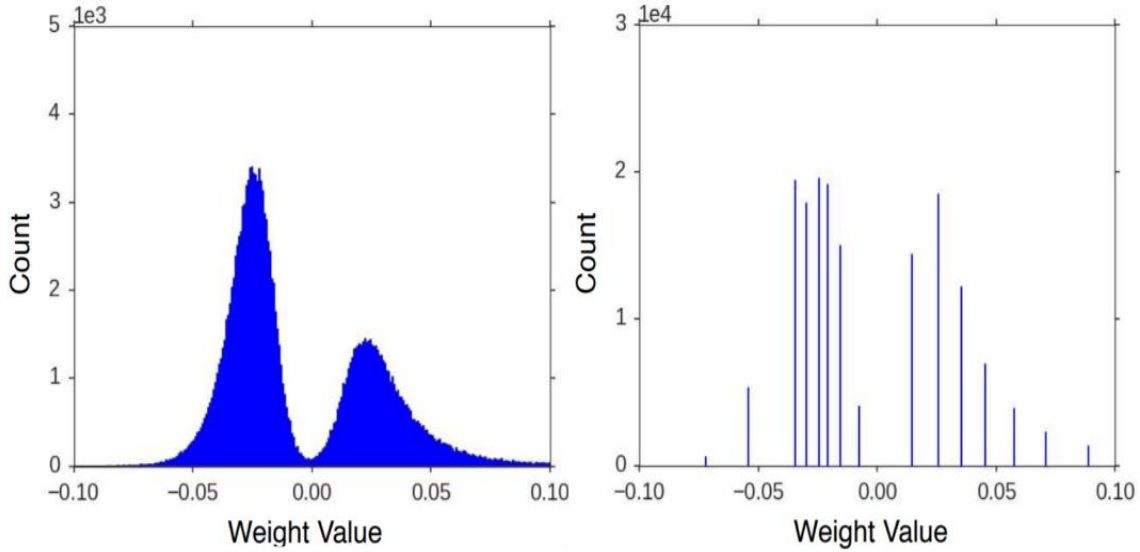
Cuantización

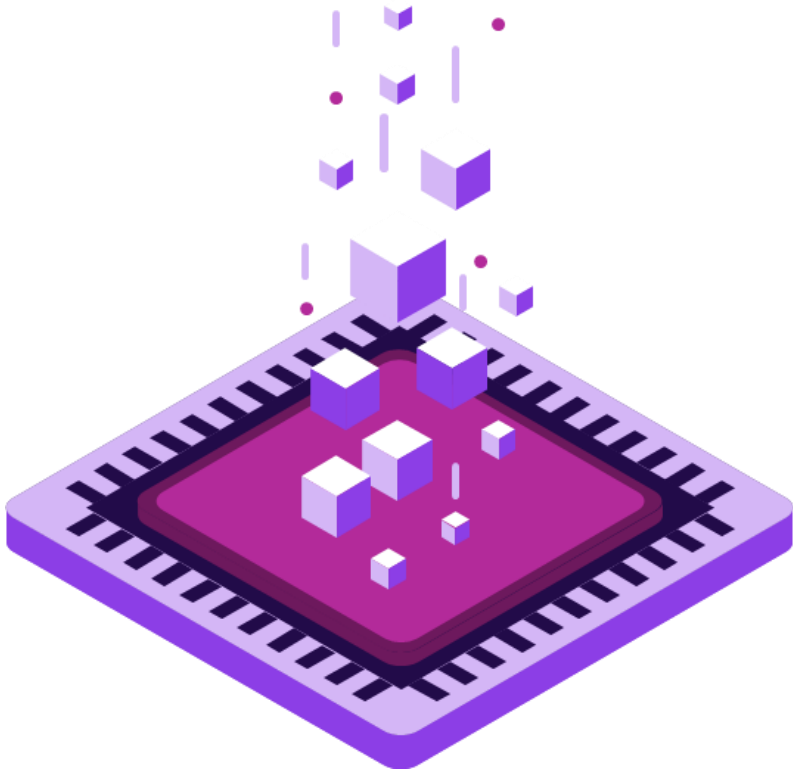
plain concepts

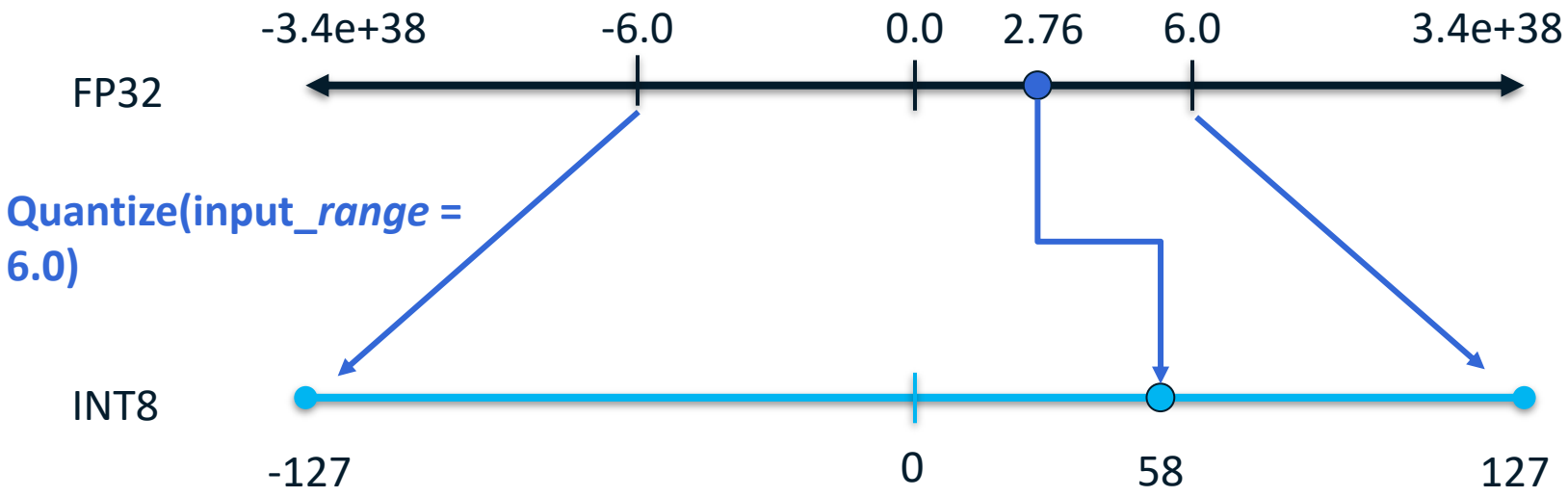
Técnica de “Pruning”: Recorte de pesos

```
pruning_params = {
      'pruning_schedule': sparsity.PolynomialDecay(initial_sparsity=0.50,
                                                   final_sparsity=0.90,
                                                   begin_step=2000,
                                                   end_step=end_step,
                                                   frequency=100)
}

pruned_model = tf.keras.Sequential([
    sparsity.prune_low_magnitude(
        l.Conv2D(32, 5, padding='same', activation='relu'),
        input_shape=input_shape,
        **pruning_params),
    l.MaxPooling2D((2, 2), (2, 2), padding='same'),
    l.BatchNormalization(),
    sparsity.prune_low_magnitude(
        l.Conv2D(64, 5, padding='same', activation='relu'), **pruning_params),
    l.MaxPooling2D((2, 2), (2, 2), padding='same'),
    l.Flatten(),
    sparsity.prune_low_magnitude(l.Dense(1024, activation='relu'),
                                 **pruning_params),
    l.Dropout(0.4),
    sparsity.prune_low_magnitude(l.Dense(num_classes, activation='softmax'),
                                 **pruning_params)
])

pruned_model.summary()
```

¿Que es la cuantización?







Tipos de cuantización

- Post-training
- Quantization-aware training

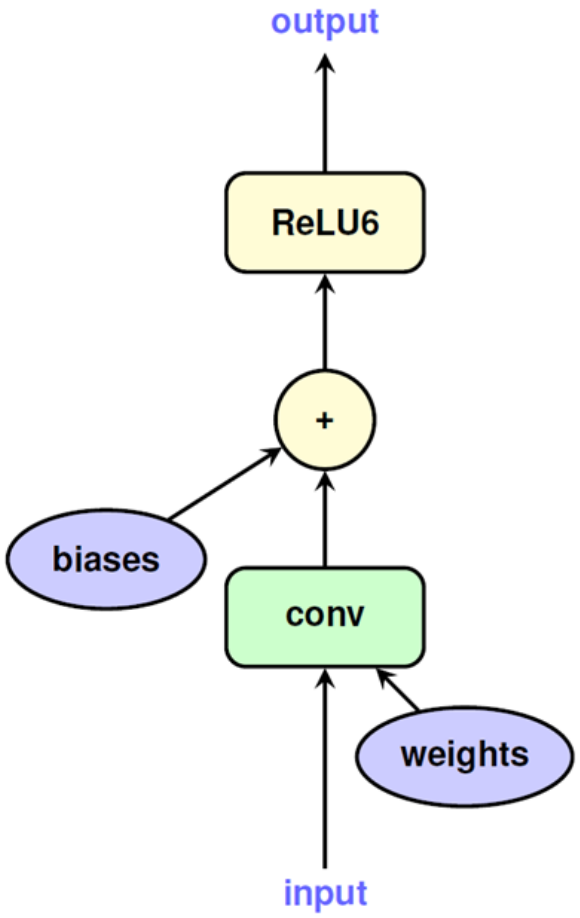
plain concepts

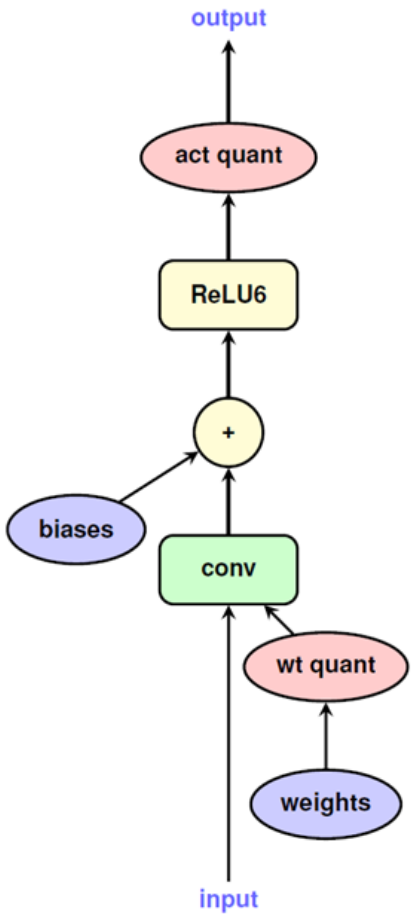
Post-training quantization

Tecnica	Datos	Reduccion de tamaño	Precision	Hardware soportado
Post-training quantization	No	Mas del 50%	Perdida de precision insignificante	CPU, GPU
Post-training dynamic range quantization	No	Mas del 50%	Perdida de precision	CPU
Post-training integer quantization	Muestra representativa no etiquetada	Mas del 75%	Perdida pequeña de precision	CPU, EdgeTPU

plain concepts

Quantization Aware Training





Model	Top-1 Accuracy: Floating point	Top-1 Accuracy: Fixed point: 8 bit weights and activations
Mobilenet-v1-128-0.25	0.415	0.399
Mobilenet-v1-128-0.5	0.563	0.549
Mobilenet-v1-128-0.75	0.621	0.598
Mobilenet-v1-128-1	0.652	0.64
Mobilenet-v1-160-0.25	0.455	0.435
Mobilenet-v1-160-0.5	0.591	0.577
Mobilenet-v1-160-0.75	0.653	0.639
Mobilenet-v1-160-1	0.68	0.673
Mobilenet-v1-192-0.25	0.477	0.458
Mobilenet-v1-192-0.5	0.617	0.604

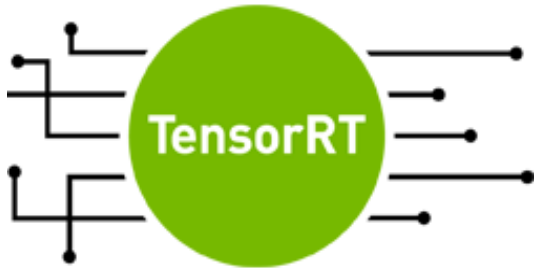
plain concepts

Benchmarks

Model	Top-1 Accuracy (Original)	Top-1 Accuracy (Post Training Quantized)	Top-1 Accuracy (Quantization Aware Training)	Latency (Original) (ms)	Latency (Post Training Quantized) (ms)	Latency (Quantization Aware Training) (ms)	Size (Original) (MB)	Size (Optimized) (MB)
Mobilenet-v1-1-224	0.709	0.657	0.70	124	112	64	16.9	4.3
Mobilenet-v2-1-224	0.719	0.637	0.709	89	98	54	14	3.6
Inception_v3	0.78	0.772	0.775	1130	845	543	95.7	23.9
Resnet_v2_101	0.770	0.768	N/A	3973	2868	N/A	178.3	44.9

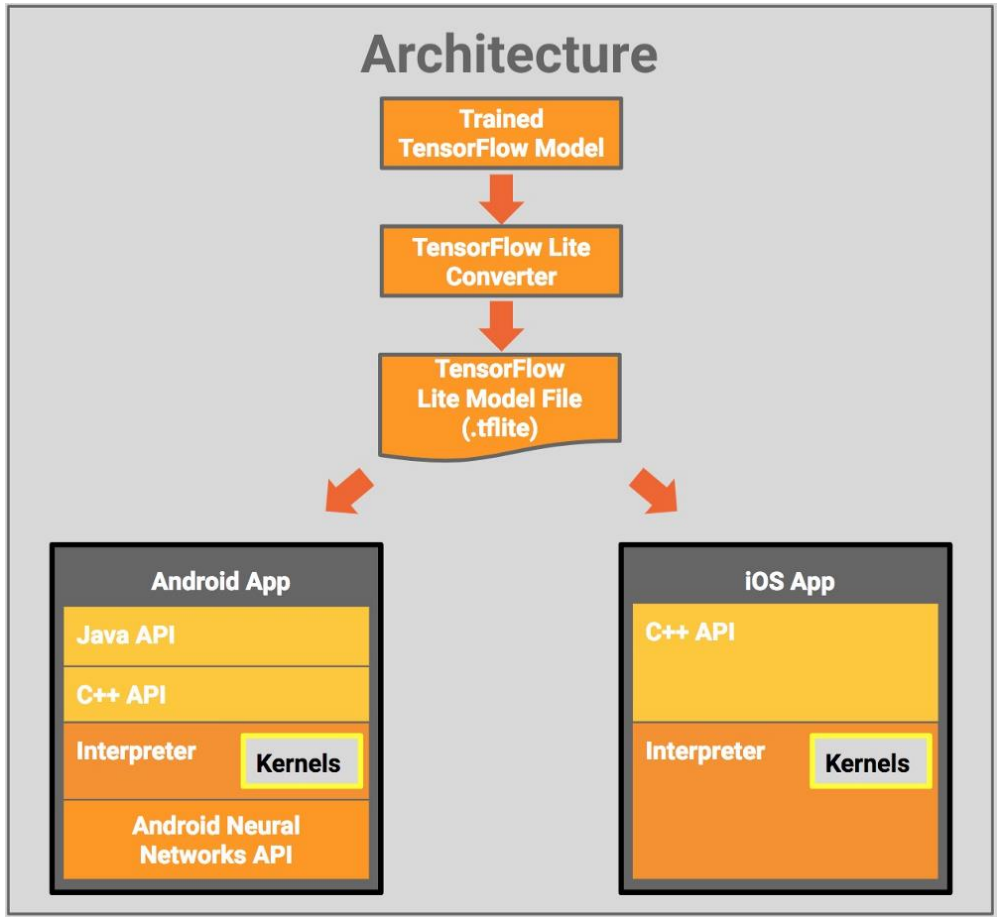
plain concepts

¿Que librerías existen actualmente para realizar estas optimizaciones?



plain concepts

TensorFlow Lite



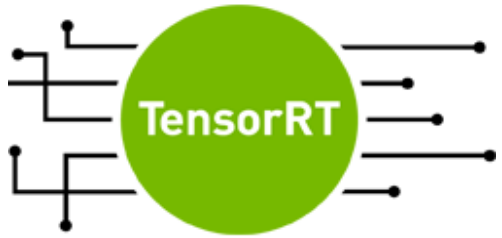
Weight & Activation Precision Calibration
Maximizes throughput by quantizing models to INT8 while preserving accuracy

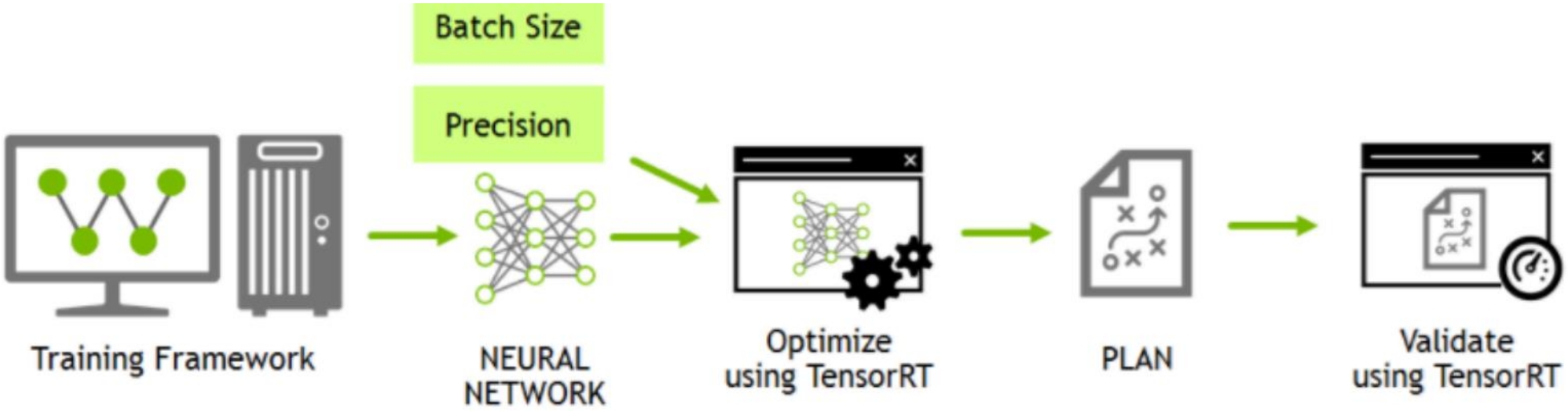
Layer & Tensor Fusion
Optimizes use of GPU memory and bandwidth by fusing nodes in a kernel

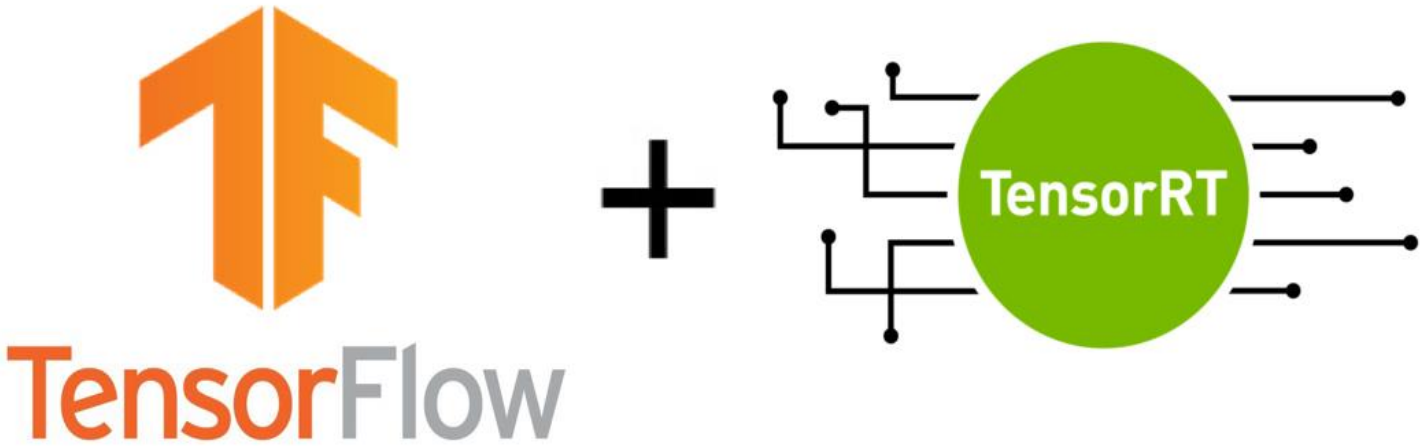
Kernel Auto-Tuning
Selects best data layers and algorithms based on target GPU platform

Dynamic Tensor Memory
Minimizes memory footprint and re-uses memory for tensors efficiently

Multi-Stream Execution
Scalable design to process multiple input streams in parallel







TF-TRT = TF + TRT

Aceleradores de inferencia

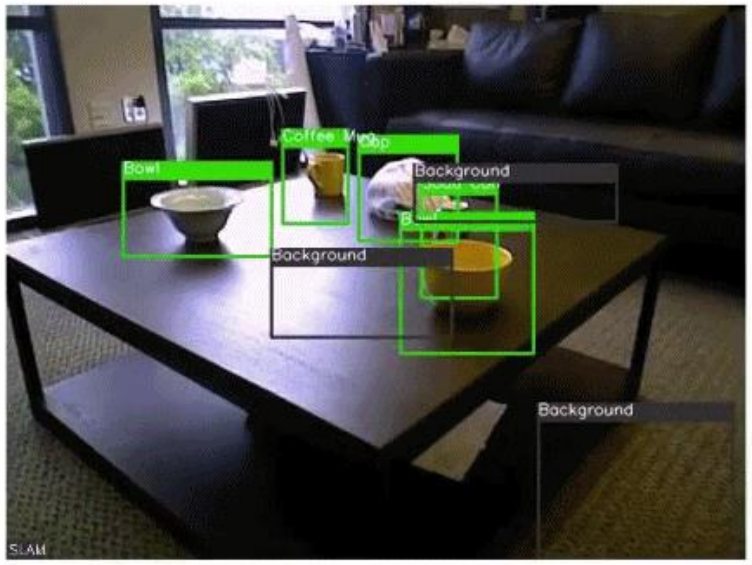
NVIDIA Jetson Nano

Google Coral Edge TPU

Raspberry Pi 3B +
Intel Neural Compute Stick 2

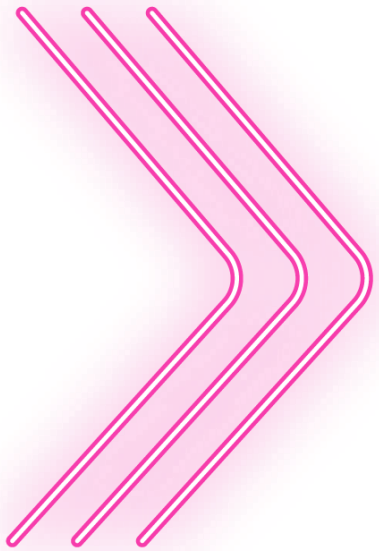
TensorFlow Lite

USB Accelerator



NVIDIA Jetson Nano

Siguientes pasos



Fundamentos de optimización de modelos

Estado del arte frameworks de optimización

Ejemplos de ejecución

Aceleradores disponibles

Despliegue en dispositivos

Contenido del Virtual Coffee!

Repositorio: https://github.com/alexandergg/Accelerate-Deep-Learning-Inferences

Articulo: https://azurebrains.com/2020/04/17/fundamentos-para-mejorar-el-rendimiento-y-la-eficiencia-de-tus-modelos-de-deep-learning/

plain concepts

Thank you

@alexndrglez

@plainconcepts

www.plainconcepts.com