Geekbench ML 0.5
Inference Workloads

Introduction

This document outlines the workloads included in the Geekbench ML Inference Benchmark suite.

Inference Benchmark scores are used to evaluate and optimize CPU, GPU, NPU, and DSP performance using workloads that include Computer Vision and Natural Language Processing tasks.

Platform Support

Platform	Minimum Version	Comment
Android	Android 9	
iOS	iOS 14	
Linux	Ubuntu 18.04 LTS	Coming in v0.6.0
macOS	macOS 10.15	Coming in v0.6.0
Windows	Windows 10	Coming in v0.6.0

Machine Learning Framework Support

Platform	API	Comment
Android	TensorFlow Lite	
iOS	TensorFlow LIte	Core ML Coming in v0.6.0
Linux	TensorFlow Lite	Coming in v0.6.0
macOS	TensorFlow Lite	Coming in v0.6.0
Windows	TensorFlow Lite	Coming in v0.6.0

Compilers

Platform	Compiler	Comment
Android	Clang 11.0	Provided by NDK r22b
iOS	Xcode 12.5	
Linux	Clang 9.0	Coming in v0.6.0
macOS	Xcode 12.5	Coming in v0.6.0
Windows	Clang 9.0	Coming in v0.6.0

Runtime

Geekbench ML runs Inference workloads in the order listed here as the Inference Benchmark. Each workload is run for 5 iterations by default.

Scores

Geekbench ML workload scores are calibrated using results from a baseline system (a Lenovo ThinkStation P340 with a Core i7-10700 processor). Scores are normalized against these results, where a score of 1,500 indicates equality between the two. Higher scores are better, with double the score indicating double the performance.

Geekbench ML provides one overall score for the Inference Benchmark. The overall score is the geometric mean of the scores of the individual Inference workloads.

Accuracy

Geekbench ML uses the predictions computed by the F32 model running on an Intel i7 CPU as the ground truth for accuracy calculations. The output from each input is compared against the ground truth using each task's evaluation metric. The metrics used to evaluate the models are standard and well established for their corresponding tasks.

Task	Evaluation Metric
Image Classification	Top-1 Accuracy
Image Segmentation	Pixel Accuracy
Object Detection	Mean IOU
Face Detection	Mean IOU
Pose Estimation	Object Keypoint Similarity
Machine Translation	Bilingual Evaluation Understudy
Text Classification	Top-1 Accuracy

Computer Vision Workloads

Computer Vision is a field of artificial intelligence that develops techniques for training computers to process and understand digital images. With the development of deep neural networks, we have obtained higher accuracy and better performance in increasingly challenging Computer Vision tasks.

Geekbench ML includes five Computer Vision workloads that span a wide range of inference tasks relevant to mobile devices and applications:

- **Image Classification** identifies the category class to which the object belongs.
- **Image Segmentation** identifies different objects, along with corresponding boundaries.
- **Object Detection** identifies the objects with their spatial positions.
- **Face Detection** identifies the human faces with their spatial positions.
- **Pose Estimation** identifies the position of human joints.

Image Classification

Image Classification is a task where a label is predicted for a digital image. For example, an image of a dog would be classified as “dog”. The model takes a fixed-size image as input and returns a vector of confidence scores for the trained image labels. The label with the highest confidence score is used as the label for the image.

Image Classification uses MobileNetV1 as its network. MobileNetV1 uses a depth-wise separable convolution to build a lightweight network reducing model size and complexity.MobileNetV1 is more common than VGG and ResNet in mobile and embedded vision applications because of its compact nature.

Network	Input Resolution	Data Type
MobileNetV1	224 * 224 * 3	float32
MobileNetV1	224 * 224 * 3	float16
MobileNetV1	224 * 224 * 3	int8

Image Segmentation

Image Segmentation is a task where all of the pixels of a digital image are separated into different categories. Unlike Image Classification, which classifies the entire image, Image Segmentation classifies each pixel of the image. The model takes a fixed-size image as input and returns a vector of confidence scores for each pixel of the image. The label with the highest score is used as the label for the pixel. The overall image is returned as a multi-colour mosaic where each colour represents an object type.

Image Segmentation uses DeepLabV3+ as its network. DeepLabV3+ includes DeepLabV3's Atrous Spatial Pyramid Pooling (ASPP) to capture the contextual information at multiple scales, but also adds an effective decoder module to refine the results. We use MobileNetV2 as the backbone for feature extraction to reduce the model's overall size and complexity.

Image Segmentation Example Input

Image Segmentation Example Output

Network	Backbone	Input Resolution	Data Type
DeepLabV3+	MobileNetV2	384 * 384 * 3	float32
DeepLabV3+	MobileNetV2	384 * 384 * 3	float16
DeepLabV3+	MobileNetV2	384 * 384 * 3	int8

Pose Estimation

Pose Estimation is the task of estimating the pose of a person in a digital image by estimating the location of key joints in the image. The model takes a fixed-size image as input and returns the relative location of individual body parts with confidence scores. The parts include 13 human body parts, five facial keypoints, and one background location.

Pose Estimation uses OpenPoseV2 as its network. OpenPoseV2 was chosen because it increases the network depth and uses three consecutive 3*3 kernels instead of one 7*7 convolutional kernel. These structural changes reduce the number of operations, which improves the accuracy and speed of the model. We use VGG19 as the backbone for feature extraction to test hardware performance on more complex models.

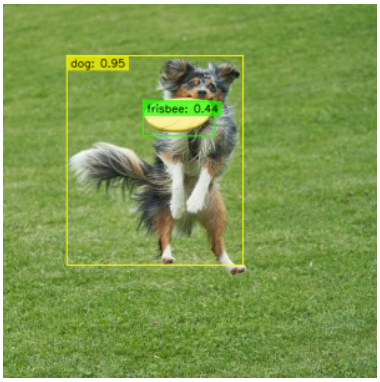
Pose Estimation Output Image

Network	Backbone	Input Resolution	Data Type
OpenPoseV2	VGG19	368 * 368 * 3	float32
OpenPoseV2	VGG19	368 * 368 * 3	float16
OpenPoseV2	VGG19	368 * 368 * 3	int8

Object Detection

Object Detection is the task of identifying which objects are present in a digital image along with where these objects appear in the image. Unlike Image Classification which only returns one prediction for an image, Object Detection detects the positions and classifications for all of the objects in the input image.

Object Detection uses SSD as its network. SSD, a Single Shot MultiBox Detector, is simpler than methods that require object proposals because it completely eliminates the proposal generation and subsequent pixel or feature resampling stages. It also encapsulates all computation in a single network. It makes SSD easy to train and setup. In order to reduce the model's size and complexity, we use MobileNetV1 as the backbone for feature extraction.



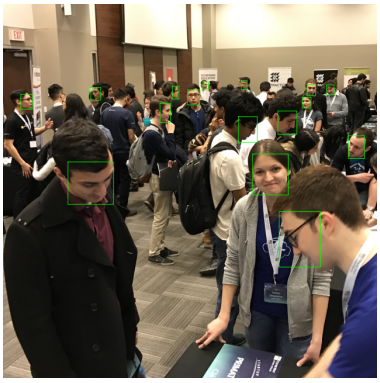
Object Detection Example Output

Network	Backbone	Input Resolution	Data Type
SSD	MobileNetV1	300 * 300 * 3	float32
SSD	MobileNetV1	300 * 300 * 3	float16
SSD	MobileNetV1	300 * 300 * 3	int8

Face Detection

Face Detection is the task of detecting faces from a digital image. The model takes an image as input and returns the confidence score and the coordinates for each detected face. A threshold can be set to exclude faces with confidence scores that fall below the threshold. We used a robust single-stage face detector as our model.

Face Detection uses Retinaface as its network. Retinaface takes advantage of extra and self-supervised multitasking to perform pixel-wise face localization on various scales of faces. In order to reduce the model's size and complexity, we used MobileNetV2 as the backbone for feature extractor.



Face Detection Example Output

Network	Backbone	Input Resolution	Data Type
RetinaFace	MobileNetV2	640 * 640 * 3	float32
RetinaFace	MobileNetV2	640 * 640 * 3	float16
RetinaFace	MobileNetV2	640 * 640 * 3	int8

Natural Language Processing Workloads

Natural Language Processing is a branch of artificial intelligence in linguistics. Its goal is to help the interaction between computers and human languages. It does this by allowing computers to understand humans' natural languages. Natural Language Processing is a complex problem. A comprehensive understanding of human language requires both low-level (words) and high-level (abstract) concepts.

With the development of deep neural networks, many Natural Language Processing tasks can now achieve high performance on mobile devices.

Geekbench ML includes two Natural Language Processing workloads:

- **Text Classification** performs sentiment analysis on open-ended text.
- **Machine Translation** translates text from one language to another.

Text Classification

Text Classification is the task of assigning open-ended text to a set of pre-defined categories.The model takes tokenized text and returns a vector of confidence scores for each category.The Text Classification workload takes movie reviews as input and determines whether the review is a positive or negative review.

Text Classification uses Compressed BERT (BERT-Tiny) as its network. BERT-Tiny was chosen because it retains the high accuracy found in larger versions of the model, but also provides simple, small, and effective model.

Network	Input Size (Words)	Data Type
BERT-Tiny	128	float32
BERT-Tiny	128	float16

Machine Translation

Machine Translation is the task of translating text from one language to another. Our Machine Translation workload uses neural networks as the critical part of the end-to-end translation pipeline. It takes English sentences as input and produces French sentences as output.

Machine Translation uses Sequence to Sequence Learning with Neural Networks (Seq2Seq) as its network. Seq2Seq was chosen because of its performance when converted to TFLite. Gated Recurrent Unit (GRU) was implemented to ensure high accuracy. In the future, we plan to use Transformer and Attention to maintain accuracy while enabling support for GPU and NNAPI delegates.

Network	Input Size (Words)	Data Type
Seq2Seq GRU	17	float32
Seq2Seq GRU	17	float16