Geekbench AI Workloads

Introduction

This document outlines the workloads included in the Geekbench AI Benchmark suite.

AI 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
Android	Android 12
iOS	iOS 17
Linux	Ubuntu 22.04 LTS
macOS	macOS 14
Windows	Windows 10

AI Framework Support

Platform	API
Android	TensorFlow Lite
iOS	CoreML
Linux	TensorFlow Lite, ONNX, OpenVINO
macOS	CoreML
Windows	ONNX, OpenVINO

Compilers

Platform	Compiler
Android	Clang 16
iOS	Xcode 15
Linux	Clang 16
macOS	Xcode 15
Windows	Clang 16

Runtime

Geekbench AI runs AI workloads in the order listed here as the AI Benchmark. Each workload is run for at least 5 iterations and at least one second by default.

Accuracy

Geekbench AI uses the predictions computed by the Full Precision 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	F1 Score
Face Detection	F1 Score
Pose Estimation	Object Keypoint Similarity
Depth Estimation	Root Mean Square Error (RMSE)
Super Image Resolution	Structural Similarity Index Measure (SSIM)
Style Transfer	Structural Similarity Index Measure (SSIM)
Machine Translation	BiLingual Evaluation Understudy
Text Classification	Top-1 Accuracy

Scores

Geekbench AI workload scores are calibrated using results from a baseline system (a Lenovo ThinkStation P340 with a Core i7-10700 processor). Workload scores are normalized against these results, where a score of 1,500 indicates equality between the two. Workload scores are also adjusted using a factor based on the accuracy.

Geekbench AI provides three overall scores for the AI Benchmark — a Single Precision score, a Half Precision score, and a Quantized score. Each score is the geometric mean of the corresponding workload scores. For example, the Quantized score is the geometric mean of all the Quantized workload scores.

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 AI includes the following Computer Vision workloads that span a wide range of inference tasks relevant to applications used by laptops and mobile devices.

- **Image Classification** identifies the category class to which an object belongs.
- **Image Segmentation** identifies different objects, along with corresponding boundaries.
- **Object Detection** identifies objects along with their spatial locations.
- **Face Detection** identifies human faces along with their spatial positions.
- **Pose Estimation** identifies the position of human joints.
- **Depth Estimation** creates a pixel-depth map for an image.
- **Image Super Resolution** converts a low-resolution image into a high-resolution image.
- **Style Transfer** composes an image based on the style of another.

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.

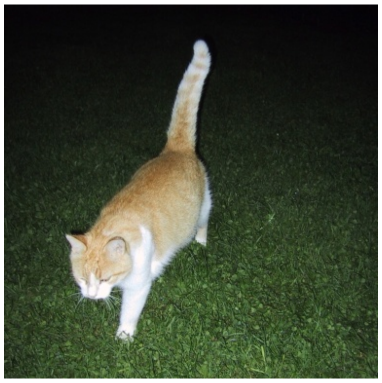


Image Segmentation Example Input

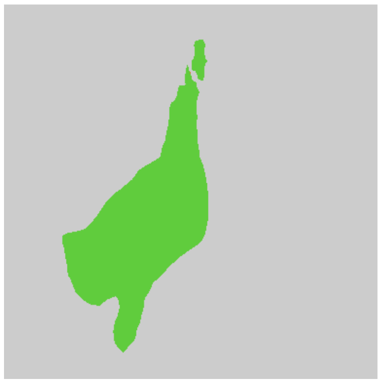


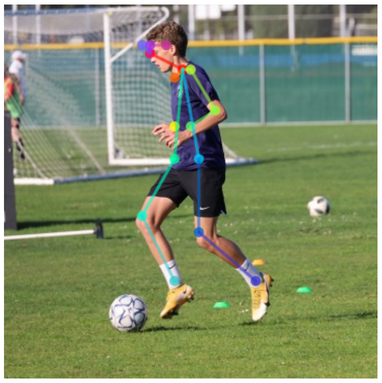
Image Segmentation Example Output

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.



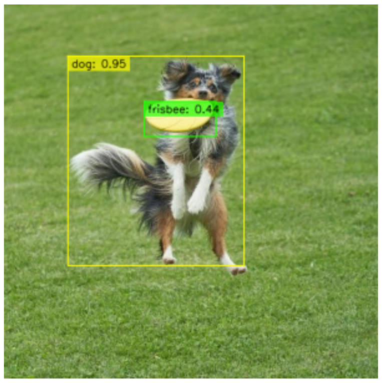
Pose Estimation Output Image

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. It draws a bounding box around each object and assigns a class label and a confidence score. The confidence score reflects the confidence that the object is correctly classified.

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 set up. 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 extraction.

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

Depth Estimation

Depth Estimation is the task of creating a depth map by measuring the distance of each pixel in an image relative to the camera's focal point. The model takes an RGB image and creates a depth map that conveys depth using grayscale values. Black areas are farther away from the camera. White areas are closer.

The model uses ConvNets, a deep convolutional network, and follows the approach proposed by Ke Xian, et al. (2018) in Monocular Relative Depth Perception with Web Stereo Data Supervision. This approach uses EfficientNet-lite3 as a feature extraction backbone, which is a fast and lightweight convolutional neural network architecture and scaling method. It results in highly accurate estimates of depth.

Depth Estimation Example Input

Depth Estimation Example Output

Network	Backbone	Input Resolution	Data Type
ConvNets	EfficientNet-lite3	256 * 256 * 3	float32
ConvNets	EfficientNet-lite3	256 * 256 * 3	float16
ConvNets	EfficientNet-lite3	256 * 256 * 3	int8

Image Super Resolution

Single Image Super Resolution (SISR) is the task of converting a low-resolution (LR) image into a high-resolution (HR) image with enhanced details by a factor of 4x upscaling. A popular application of SISR is to prepare LR images for further processing by computer vision tasks.

Image Super Resolution uses Residual Feature Distillation Network (RFDN) as its network. RFDN contains multiple Feature Distillation Connections (FDC) and Shallow Residual Blocks (SRB). The FDC provides the same function as a channel splitting operation, allowing RFDN to learn more discriminate feature representation. The SRB enables RFDN to do residual learning. With those two parts, RFDN becomes a lightweight and accurate model.

Network	Input Resolution	Data Type
RFDN	48 * 48 * 3	float32
RFDN	48 * 48 * 3	float16
RFDN	48 * 48 * 3	int8

Image Super Resolution Example Input

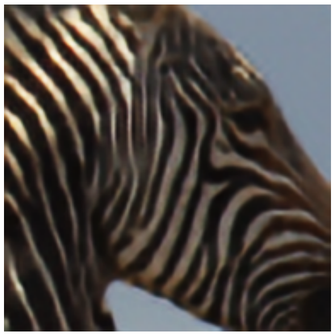
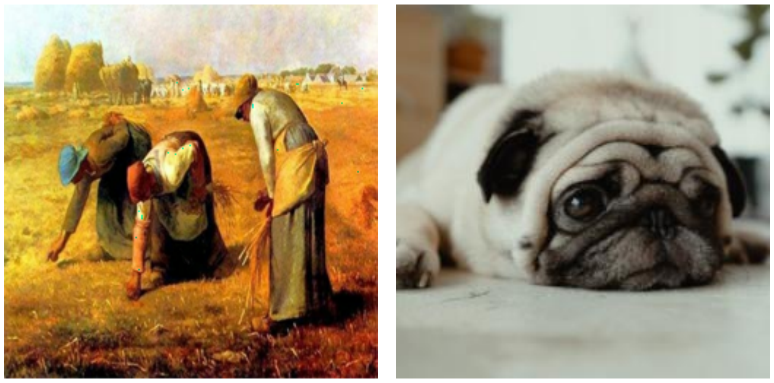


Image Super Resolution Example Output

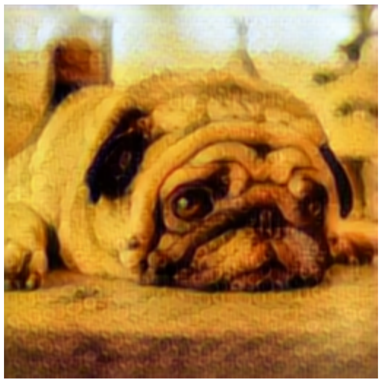
Style Transfer

Style Transfer is the task that creates an image that resembles another image's style. It does this by taking two images—an original content image and a style reference image—and blending them together to create an image that looks like the original content image has been painted using the style of the reference image.

The model uses the Fast Real-Time Style Transfer approach described by Johnson, et al. (2016) in Perceptual Losses for Real-Time Style Transfer and Super-Resolution. This approach uses perceptual loss functions to train the Image Transform Net and is more effective in reconstructing fine details than methods trained with per-pixel loss.



Style Transfer Example Input



Style Transfer Output Image

Network	Input Resolution	Data Type
Image Transform Net	256 * 256 * 3	float32
Image Transform Net	256 * 256 * 3	float16
Image Transform Net	256 * 256 * 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 concrete components (words) and high-level abstract concepts (idioms or phrasing).

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

Geekbench AI 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 A simple, small, and effective model.

Network	Input Size (Words)	Data Type
BERT-Tiny	128	float32
BERT-Tiny	128	float16
BERT-Tiny	128	int8

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 Transformer as its network. The Transformer architecture was introduced in the paper Attention is All You Need by Vaswani et al. in 2017. It has become one of the most influential and widely used architectures in NLP tasks. Transformers use a self-attention module to capture dependencies between words in a sequence. This enables the model to handle long-range dependencies more effectively. The attention module allows the model to weigh the importance of different words and capture both local and global relationships.

Network	Input Size (Words)	Data Type
Transformer	49	float32
Transformer	49	float16
Transformer	49	int8