An artificial intelligence (AI) or machine learning software program is one that can sense, reason, act and adapt. While AI algorithms have existed for many years, there has recently been a rapid expansion in AI-based capabilities across the enterprise. To deliver promising AI algorithms, innovators must deploy their programs on a sophisticated platform that provides a great end user experience, lower operating costs and ensures a greener planet.

There are two primary tasks in every machine learning program. First it trains and secondly it performs inferencing on the data it has obtained during training. While training is done only once, inferencing is done many times and continuously, which requires a powerful platform to handle.

Ampere’s Armv8 64-bit servers are purpose-built for large-scale public and private cloud environments, making it an extremely good solution for inferencing required by machine learning applications such as real-time object detection. Ampere’s cloud solutions have already proven to deliver significant advantages to developers through its large number of cores, high-speed connectivity, memory throughput and cost-effectiveness. The highly integrated, purpose-built Ampere solution provides high performance at low total cost of ownership (TCO) for private and public clouds.

This paper will describe how the eMAG™ platform can be utilized to demonstrate the YOLO algorithm, a popular object detection system consisting of a single convolutional network that simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. YOLO sees the entire image during training and test time, so it implicitly encodes contextual information about classes as well as their appearance.

In this demonstration, all the training and testing is done on the eMAG platform. This paper will summarize the best known approaches for optimizing the real-time object detection on eMAG.

**Data and Network Architecture**

YOLO takes a completely different approach over competing solutions by not using a traditional classifier. It looks up the image just once and then divides the image into 13 by 13 cells for predicting 5 bounding boxes. Each cell can accommodate 5 bounding boxes, which are the rectangular regions that enclose an object as depicted in figure 2 below.

YOLO uses convolutional neural networks. In this demonstration, a model is implemented as a convolutional neural network and is evaluated on the PASCAL VOC detection dataset. This dataset is used for evaluating algorithms for image classification, image segmentation, and object detection that has 20 classes and 11,530 images overall. This network has 9 convolution layers, enabling the architecture to take input and go through convolution layer followed by max pooling with a 3*3 filter for convolution and 2*2 filter for max pooling. There is no fully connected layer. The convolution layers are pre-trained on ImageNet classification. The final output of this network is the 13 × 13 × 125 tensor of predictions.
Hardware

YOLO Real time object detection was performed on Ampere’s eMAG platform with a Tensorflow 1.0 backend. Below is a summary of all the hardware that was utilized during this demonstration:

**Architecture:** Arm®v8 64-bit server

**CPU op-mode:** 64-bit

**Model name:** Ampere eMAG 8180

**Processor Subsystem**
- 32 Arm v8 64-bit CPU cores up to 3.3 GHz with Turbo
- 32 KB L1 I-cache, 32 KB L1 D-cache per core
- Shared 256 KB L2 cache per 2 cores

**Memory**
- 32 MB globally shared L3 cache
- 8x 72-bit DDR4-2667 channels
- Advanced ECC and DDR4 RAS features
- Up to 16 DIMMs, 1 TB/socket

**System Resources**
- Full interrupt virtualization
- I/O virtualization
- Enterprise server-class RAS – End-to-end data poisoning – Error containment and isolation – Background L3 and DRAM scrubbing

**Connectivity**
- 42 lanes of PCIe Gen 3, with 8 controllers – x16 or two x8/x4 – x16 or two x8/x4 – x8 or two x4 – two x1
- 4 x SATA Gen 3 ports
- 2 x USB 2.0 ports

**Technology & Functionality**
- TSMC 16nm FinFET+
- Arm v8.0-A, SBSA Level 3 – EL3, secure memory and secure boot support
- Advanced power management

**Power**
- TDP: 125 W

---

**Figure 1:** The architecture has 9 convolution layers, each followed by max pooling layer.

**Figure 2:** Model

**Figure 3:** eMAG Block Diagram
Software stack

In order to run the YOLO demonstration, the following dependencies were installed: Tensorflow, Numpy, OpenCV3, and Python3. To get started, the first step was to build the Cython extensions in place by giving the following command: $ python3 setup.py build_ext --inplace.

Training

The convolutional layers were pre-trained on the ImageNet 1000-class competition dataset. For pretraining, the first 9 convolutional layers were used, followed by average-pooling layer.

The next step was to convert the model to perform detection. The very last convolutional layer had a 1*1 kernel, which existed to reduce the data to the shape of 13*13*125. This enabled 125 channels for every grid cell. These 125 numbers contained the data for the bounding boxes and the class predictions. Each grid cell predicted 5 bounding boxes and a bounding box, hence 125 channels.

A bounding box is described by 25 data elements which are width, height, confidence score, and probability distribution over the class. YOLO predicts multiple bounding boxes per grid cell. During training time, it is ideal to only have one bounding box predictor responsible for each object.

One predictor was assigned to be “responsible” for predicting an object based on which prediction had the highest current IOU with the ground truth. This led to specialization between the bounding box predictors. Each predictor got better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall. This was simple process where an input image was resized to 416*416 pixels and went through the convolutional neural network in a single pass. It came out from the other end as 13*13*125 tensor describing the bounding boxes for the grid cells. The final scores for the bounding boxes was computed and eliminated the ones that were less than 30% threshold value.

This training was made easier by running the training model on the eMAG platform because developers were able to deploy custom, deep learning solutions through eMAG’s performant cores, high speed connectivity, memory throughput, and carrier-grade reliability in an existing data center footprint while lowering power and operating costs substantially.

Inference

Just like in training, predicting detections for a test image only requires one network evaluation. On PASCAL VOC, the network predicts 98 bounding boxes per image and class probabilities for each box. YOLO is extremely fast at test time since it only requires a single network evaluation, unlike classifier-based methods. The grid design enforces spatial diversity in the bounding box predictions. Often it is clear which grid cell an object falls in to and the network only predicts one box for each object. However, some large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-maximal suppression can be used to fix these multiple detections. While not critical to performance as it is for R-CNN or DPM, non-maximal suppression adds 2- 3% in mAP. For this demo, the following commands were run on the eMAG platform:

In order to load the tiny yolo weights for training, the following command was given:
python flow --model cfg/tiny-yolo-voc.cfg --load weights/tiny-yolo-voc.weights

For executing the demo on eMAG platform, the following command was given:
python flow --model cfg/tiny-yolo-voc.cfg --load weights/tiny-yolo-voc.weights --demo videofile.avi --saveVideo
Summary

As the demonstration described above, using the Ampere eMAG platform to run the YOLO real-time object detection algorithm does not require any additional software or configuration. The system works out of the box and users are able to directly download prebuilt Python libraries and take advantage of the pretrained models.

Ampere’s eMAG platform continues to develop AI training models to improve accuracy and support evolving workloads and use cases. Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly. YOLO is a state-of-the-art in real-time object detection, and it also generalizes well to new domains, which makes it ideal for applications that rely on fast, robust object detection.

After a neural network is trained, it is deployed to run inference to classify, recognize and process new inputs. Ampere’s eMAG platform is simplified for inference because all the required software and libraries already exist. Developers can take advantage of eMAG’s flexibility to add in-line machine learning capabilities to any customized training models for the real-time inference, good throughputs and low latency improves performance. eMAG delivers better scalability while reducing the platform’s complexity and cutting costs. Arm®v8 64-bit, because of efficient performance, due to the design and code running on eMAG platform, were critical to running the latest object-detection algorithms and traditional approaches such as SIFT, SURF, and ORB.

Ampere was able to deliver this solution by leveraging its highly integrated, purpose-built architecture that has proven to provide high performance at low total cost of ownership (TCO) for private and public clouds.

References

https://modelzoo.co/model/yolo-tensorflow
https://amperecomputing.com/product/
https://www.youtube.com/watch?v=4eIBisqx9_g