Broad Institute Accelerates Genome Workloads Leveraging OpenVINO™ Integration with TensorFlow

Authors: Mustafa Cavus, Ravi Panchumarthy, Anthony Reina, Yamini Nimmagadda, Prashant Shah, Sesh Seshagiri, Arindam Paul
Guest Authors from Broad Institute: Samuel Friedman, Geraldine Van der Auwera

1 OVERVIEW 2

2 OPENVINO™ INTEGRATION WITH TENSORFLOW 2

3 BENEFITS FOR GATK WITH OPENVINO™ INTEGRATION WITH TENSORFLOW 3

4 ACCELERATING CNNSCOREVARIANTS LEVERAGING OPENVINO™ INTEGRATION WITH TENSORFLOW 4

4.1 Running CNNScoreVariants on Terra Platform 4

4.2 Running CNNScoreVariants locally using Docker 5

4.3 **Running CNNScoreVariants by installing locally in a Linux Server 6**

4.4 Running CNNScoreVariants with Stock TensorFlow 7

4.5 Upgrading CNNScoreVariants to Enable OpenVINO™ integration with TensorFlow 8

4.6 Additional CNNScoreVariants Workload-Specific Optimizations for Even Better Performance 8

4.7 Running CNNScoreVariants with OpenVINO™ integration with TensorFlow on Jupyter Notebook 11

5 PERFORMANCE RESULTS 11

5.1 Accuracy 12

6 ABOUT BROAD INSTITUTE 13

7 ABOUT INTEL 13

1 Overview

In the fields of life sciences and biomedical sciences, the genomics discipline is currently exhibiting a massive growth in data collection and data generation, leading the path to advance genomic analytics. Genome Analysis Toolkit (GATK) developed by the Broad Institute is the world's most widely used open-source toolkit for variant calling. Terra is a secure, scalable, open-source platform for biomedical researchers to access data, run analysis tools and collaborate. The cloud-based platform is co-developed by the Broad Institute of MIT and Harvard, Microsoft, and Verily. Terra platform includes GATK tools and pipelines for the research community to run their analytics.

GATK supports using Intel Optimized Tensorflow for running deep learning workloads. CNNScoreVariants is one of the deep learning tools included in GATK which apply a Convolutional Neural Net to filter annotated variants. In this blog, we showcase how to further accelerate inference performance of CNNScoreVariants without losing accuracy using OpenVINO™ by simply adding two lines of code.

2 OpenVINO™ Integration with TensorFlow

OpenVINO™ integration with TensorFlow delivers OpenVINO™ inline optimizations and runtime needed for an enhanced level of TensorFlow compatibility (See Figure 1). It is designed for developers who want to get started with OpenVINO™ in their inferencing applications to enhance inferencing performance with minimal code modifications. OpenVINO™ integration with TensorFlow provides accelerated TensorFlow performance by smartly partitioning TensorFlow graphs into multiple subgraphs which are then dispatched to either the TensorFlow runtime or the OpenVINO™ runtime for optimal accelerated inferencing. The results are finally assembled to provide the final inference results. It accelerates inference across many AI models on a variety of Intel® silicon such as:

- Intel® CPUs
- Intel® integrated GPUs
- Intel® Movidius™ Vision Processing Units - referred as VPU.
- Intel® Vision Accelerator Design with 8 Intel Movidius™ MyriadX VPUs - referred as VAD-M or HDDL

Developers can greatly accelerate the inferencing of their TensorFlow models by adding the following two lines of code to their Python code or Jupyter notebooks.

```
import openvino_tensorflow
openvino_tensorflow.set_backend('<backend_name>')
```

Supported backends include 'CPU', 'GPU', 'MYRIAD', and 'VAD-M' as shown in Figure 2. Please visit the Github repo for more details. The architecture is described here. Example code snippets are here. Installation options are described here.

Note: For maximum performance, efficiency, tooling customization, and hardware control, we recommend going beyond this component to adopt native OpenVINO™ APIs and its runtime.

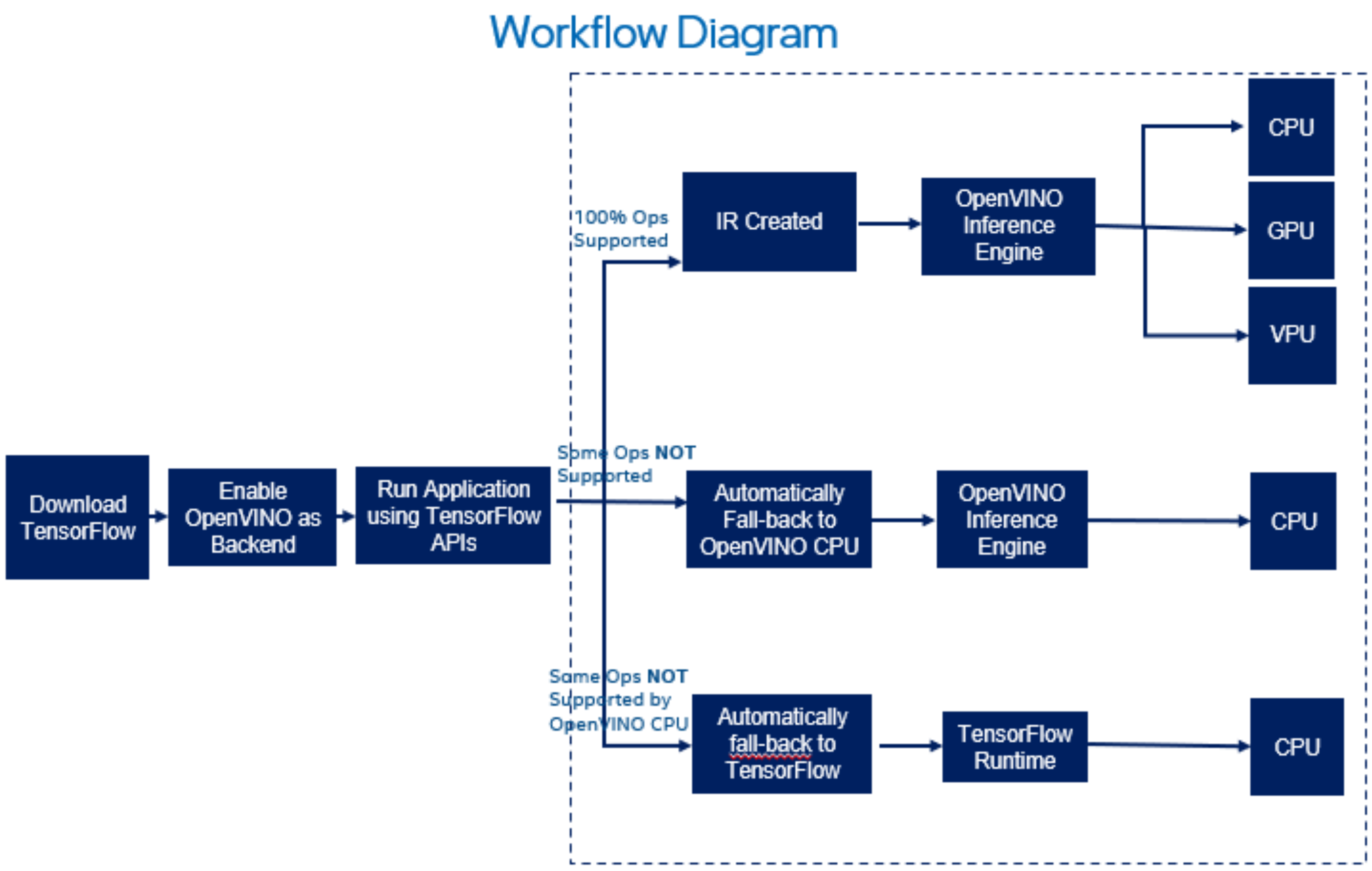
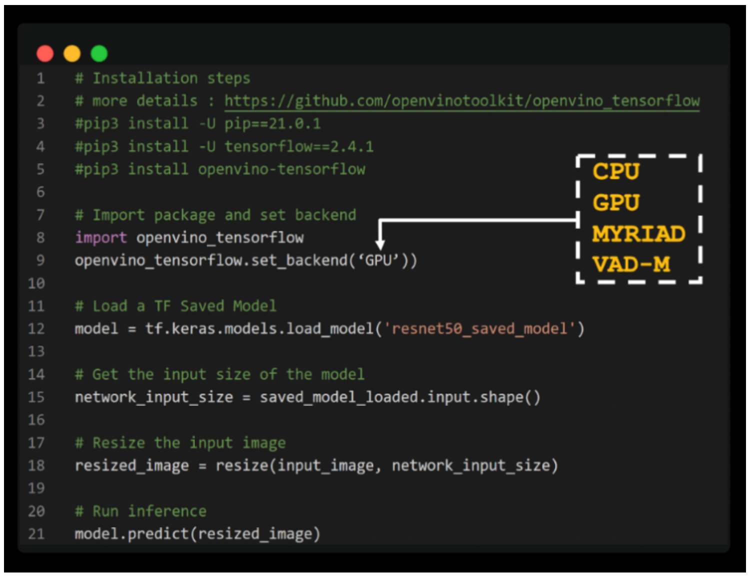

Workflow Diagram
Download TensorFlow
Enable OpenVINO as Backend
Run Application using TensorFlow APIs
100% Ops Supported
IR Created
OpenVINO Inference Engine
CPU
GPU
VPU
Some Ops NOT Supported
Automatically Fall-back to OpenVINO CPU
OpenVINO Inference Engine
CPU
Some Ops NOT Supported by OpenVINO CPU
Automatically fall-back to TensorFlow
TensorFlow Runtime
CPU

Figure 1: Workflow diagram showcasing OpenVINO™ integration with TensorFlow



```
# Installation steps
# more details : https://github.com/openvinotoolkit/openvino_tensorflow
#pip3 install -U pip==21.0.1
#pip3 install -U tensorflow==2.4.1
#pip3 install openvino-tensorflow

# Import package and set backend
import openvino_tensorflow
openvino_tensorflow.set_backend('GPU'))

# Load a TF Saved Model
model = tf.keras.models.load_model('resnet50_saved_model')

# Get the input size of the model
network_input_size = saved_model_loaded.input.shape()

# Resize the input image
resized_image = resize(input_image, network_input_size)

# Run inference
model.predict(resized_image)
```

Figure 2: Code Snippet highlighting the ease of using different hardware with openvino_tensorflow package.

3 Benefits for GATK with OpenVINO™ integration with TensorFlow

There are several advantages of using OpenVINO™ integration with TensorFlow.

- **Near zero switching cost:** TensorFlow developers can continue to use TensorFlow APIs for inferencing. There is no need to refactor code for acceleration.

- **Easy:** TensorFlow developers can accelerate their TensorFlow models seamlessly – with just two lines of code. No explicit or off-line model conversions are required.

- **Wide model coverage:** OpenVINO™ integration with TensorFlow is architected to support a wide range of TensorFlow models and operators. Check here for our ever increasing list of supported models.

- **Performance acceleration**: OpenVINO™ integration with TensorFlow greatly accelerates TensorFlow models. See section below for Performance Acceleration Results.

- **Accuracy:** OpenVINO™ integration with TensorFlow preserves model accuracy nearly identical to the original model.

- **Lightweight**: Minimal incremental memory and disk footprint are required.

- **Support for broad range of Intel powered devices:** CPUs (Xeon, Core), iGPUs, VPUs (Myriad-X), Vision Accelerator Design (VAD-M) with Movidius MyriadX etc.

4 Accelerating CNNScoreVariants leveraging OpenVINO™ integration with TensorFlow

Figure 3 depicts a high-level overview of the entire workflow of the CNNScoreVariants with OpenVINO™ integration with TensorFlow. For ease of use, we created a docker image that is compatible with notebook service in Terra platform to showcase the accelerated performance of CNNScoreVariants leveraging OpenVINO™ integration with TensorFlow. In the following sections, we describe instructions on how to use accelerated CNNScoreVariants in (1) Terra platform, (2) Locally using Docker (3) installing locally and later describe in detail the changes made to achieve this performance acceleration.

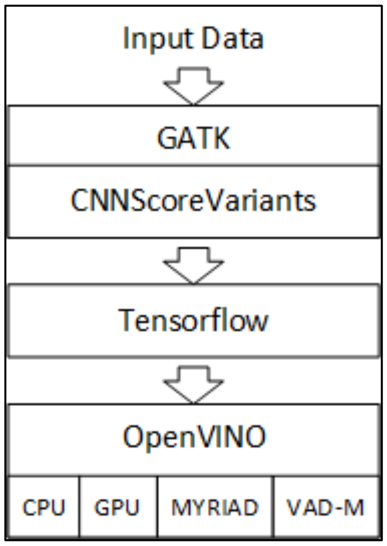
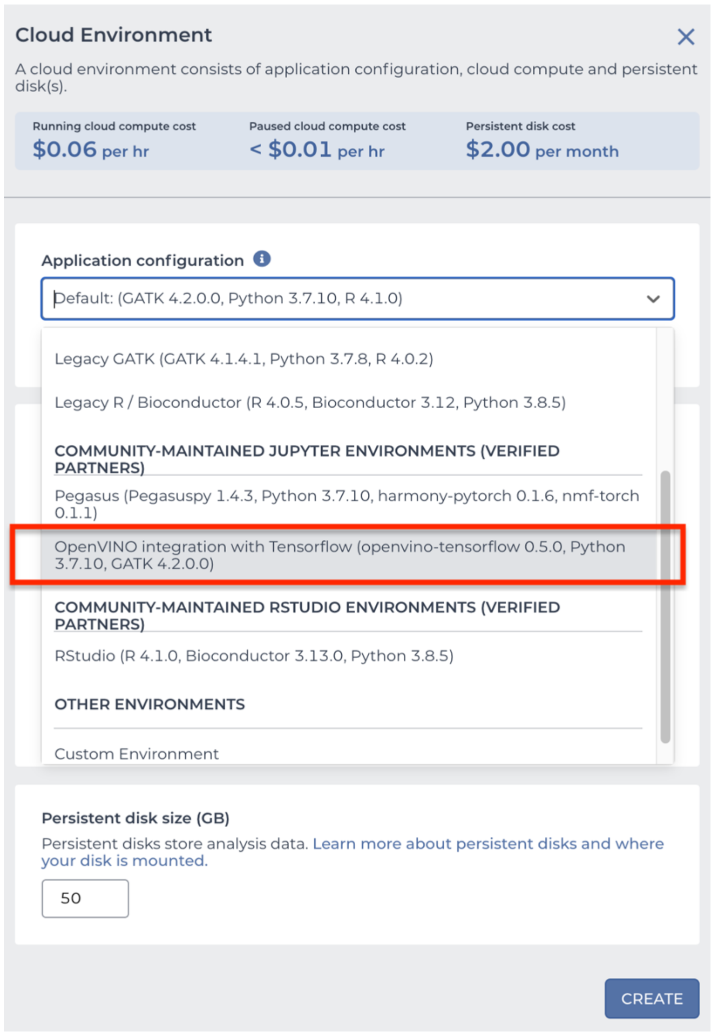


Figure 3: High level overview of the CNNScoreVariants workflow.

4.1 Running CNNScoreVariants on Terra Platform

In the Terra platform, the OpenVINO integration with Tensorflow image is available under the Community-Maintained Jupyter images (See Figure 4). The docker image that is compatible with notebook service in Terra called Leonardo is available at:
`us.gcr.io/broad-dsp-gcr-public/terra-jupyter-gatk-ovtf:0.1.0`

A sample notebook ***GATK-OVTF-Notebook.ipynb*** which showcases the performance benefits obtained by using OpenVINO™ integration with TensorFlow is available HERE.


Cloud Environment

A cloud environment consists of application configuration, cloud compute and persistent disk(s).

Running cloud compute cost	Paused cloud compute cost	Persistent disk cost
$0.06 per hr	< $0.01 per hr	$2.00 per month

Application configuration

Default: (GATK 4.2.0.0, Python 3.7.10, R 4.1.0)

Legacy GATK (GATK 4.1.4.1, Python 3.7.8, R 4.0.2)

Legacy R / Bioconductor (R 4.0.5, Bioconductor 3.12, Python 3.8.5)

COMMUNITY-MAINTAINED JUPYTER ENVIRONMENTS (VERIFIED PARTNERS)

Pegasus (Pegasuspy 1.4.3, Python 3.7.10, harmony-pytorch 0.1.6, nmf-torch 0.1.1)

OpenVINO integration with Tensorflow (openvino-tensorflow 0.5.0, Python 3.7.10, GATK 4.2.0.0)

COMMUNITY-MAINTAINED RSTUDIO ENVIRONMENTS (VERIFIED PARTNERS)

RStudio (R 4.1.0, Bioconductor 3.13.0, Python 3.8.5)

OTHER ENVIRONMENTS

Custom Environment

Persistent disk size (GB)

Persistent disks store analysis data. Learn more about persistent disks and where your disk is mounted.

50

Figure 4: OpenVINO integration with Tensorflow image available in TERRA under the Community-Maintained Jupyter Environments.

4.2 Running CNNScoreVariants locally using Docker

A simple way to quickly run the optimized CNNScoreVariants running the CNNScoreVariants with OpenVINO™ integration with TensorFlow is using Docker. Follow the steps below to build and run the docker image locally:

1. Clone the repository.
   ```
   git clone https://github.com/DataBiosphere/terra-docker
   ```

2. Go into the “terra-jupyter-gatk-ovtf” directory.
   ```
   cd terra-docker/terra-jupyter-gatk-ovtf
   ```

3. Build the docker image

```
    docker build . -t terra-jupyter-gatk-ovtf
```

4. Run the docker image, then navigate a browser to http://localhost:8000/notebooks to **access the Jupyter UI.**
   ```
   docker run --rm -it -p 8000:8000 terra-jupyter-gatk-ovtf
   ```

5. To test the performance, run a sample notebook - GATK-OVTF-Notebook.ipynb, which is available HERE.

6. **NOTE:** You can gain root access and open a bash terminal as follows:
   ```
     docker run --rm -it -u root -p 8000:8000 --entrypoint /bin/bash terra-
   jupyter-gatk-ovtf
   ```

Note: To disable OpenVINO™ integration with TensorFlow and run GATK on stock TensorFlow, set OPENVINO_TF_DISABLE environment variable to 1.

```
    export OPENVINO_TF_DISABLE ="1"
```

4.3 Running CNNScoreVariants by installing locally in a Linux Server

In this section, we will describe in detail how to install CNNScoreVariants with OpenVINO™ integration with TensorFlow in a Linux server locally.

Step 1: Download and Build GATK

First clone GATK and checkout to tag 4.2.0.0.

```
    git clone https://github.com/broadinstitute/gatk.git
    git checkout 4.2.0.0
```

Use the command below to build GATK

```
    ./gradlew bundle
```

After building GATK, an executable named as “gatk” should be created in the project folder. Additionally, you need to either install or set the PYTHONPATH to the “gatktools” python package.

Option 1: Install gatktools using pip:

```
    pip install build/gatkPythonPackageArchive.zip
```

Option 2: Set the python path to the gatktools source:

```
    export
    PYTHONPATH=$PYTHONPATH:<gatk_directory_path>/src/main/python/org/broadinstit
    ute/hellbender
```

Step 2: Install TensorFlow and OpenVINO^{TM} integration with TensorFlow

Install TensorFlow 2.5.0 and Intel OpenVINO^{TM} integration with TensorFlow using the commands below.
Note: Verify the supported version of Tensorflow at openvino-tensorflow PyPi project page

```
    pip install --force-reinstall tensorflow==2.5.0
    pip install openvino_tensorflow
```

Step 3: Upgrading CNNScoreVariants to TF 2.x

We need to upgrade CNNScoreVariants to support TF 2.x first. A few lines of code should be updated in the file below.

```
src/main/python/org/broadinstitute/hellbender/vqsr_cnn/vqsr_cnn/models.py
```

Step 3.1: Update the below line...

```
import keras.backend as K
```

to

```
import tensorflow.compat.v1.keras.backend as K
```

Step 3.2: Add the line below to import TensorFlow.

```
import tensorflow as tf
```

Step 3.3: Update the below line...

```
cfg = K.tf.ConfigProto(intra_op_parallelism_threads=intra_ops,
     inter_op_parallelism_threads=inter_ops)
```

to

```
cfg = tf.compat.v1.ConfigProto(intra_op_parallelism_threads=intra_ops,
     inter_op_parallelism_threads=inter_ops)
```

Step 3.4: Update the below line...

```
K.set_session(K.tf.Session(config=cfg))
```

to

```
K.set_session(tf.compat.v1.Session(config=cfg))
```

4.4 Running CNNScoreVariants with Stock TensorFlow

In this section we will describe how to run CNNScoreVariants with stock TensorFlow 2.4.1. If you did not set the PYTHONPATH to the “gatktools” source directory, you need the rebuild GATK and reinstall gatktools python package before running this command.

```
gatk CNNScoreVariants \
        -I gs://gatk-tutorials/workshop_2002/2-
germline/CNNScoreVariants/bams/g94982_chr20_1m_10m_bamout.bam \
        -V gs://gatk-tutorials/workshop_2002/2-
germline/CNNScoreVariants/vcfs/g94982_b37_chr20_1m_15871.vcf.gz \
        -R gs://gcp-public-data--broad-
references/hg19/v0/Homo_sapiens_assembly19.fasta \
        -O data/my_2d_cnn_scored.vcf \
        --tensor-type read_tensor \
        --transfer-batch-size 256 \
        --inference-batch-size 256
```

The last few lines of the output should look like this:

```
...
03:15:56.984 INFO  ProgressMeter -           20:8840293              8.6
31742           3697.5
03:15:56.984 INFO  ProgressMeter - Traversal complete. Processed 31742 total
variants in 8.6 minutes.
03:15:56.984 INFO  CNNScoreVariants - Done scoring variants with CNN.
03:15:57.073 INFO  CNNScoreVariants - Shutting down engine
[May 18, 2021 at 3:15:57 a.m. PDT]
org.broadinstitute.hellbender.tools.walkers.vqsr.CNNScoreVariants done. Elapsed
time: 8.67 minutes.
Runtime.totalMemory()=364904448
```

The overall throughput of this execution is 3697.5 variants per minute.

4.5 Upgrading CNNScoreVariants to Enable OpenVINO^{TM} integration with TensorFlow

As mentioned before, to enable OpenVINO^{TM} integration with TensorFlow, we need to simply add two lines of code to import OpenVINO^{TM} integration and set the backend in *inference.py*.

```
      import openvino_tensorflow
      openvino_tensorflow.set_backend('CPU')
```

Run CNNScoreVariants with same parameters to see the performance improvement.

```
      gatk CNNScoreVariants \
              -I gs://gatk-tutorials/workshop_2002/2-
      germline/CNNScoreVariants/bams/g94982_chr20_1m_10m_bamout.bam \
              -V gs://gatk-tutorials/workshop_2002/2-
      germline/CNNScoreVariants/vcfs/g94982_b37_chr20_1m_15871.vcf.gz \
              -R gs://gcp-public-data--broad-
      references/hg19/v0/Homo_sapiens_assembly19.fasta \
              -O data/my_2d_cnn_scored.vcf \
              --tensor-type read_tensor \
              --transfer-batch-size 256 \
              --inference-batch-size 256
```

The last few lines of the output should like like below.

```
03:25:06.333 INFO  ProgressMeter -           20:8840293              7.1
31742           4456.8
03:25:06.333 INFO  ProgressMeter - Traversal complete. Processed 31742 total
variants in 7.1 minutes.
03:25:06.333 INFO  CNNScoreVariants - Done scoring variants with CNN.
03:25:06.429 INFO  CNNScoreVariants - Shutting down engine
[May 18, 2021 at 3:25:06 a.m. PDT]
org.broadinstitute.hellbender.tools.walkers.vqsr.CNNScoreVariants done. Elapsed
time: 7.60 minutes.
Runtime.totalMemory()=315621376
```

The overall throughput of this execution it 4456.8 which is 20% higher than the baseline throughput.

4.6 Additional CNNScoreVariants Workload-Specific Optimizations for Even Better Performance

Although we observed significant improvement with just 2 additional lines of code as shown in the previous section, in this section we describe additional tweaks to further improve CNNScoreVariants workload. Freezing the Keras model will clear the training ops from the model and prepare it for deployment. This will improve the model coverage of OpenVINO^{TM} integration and will provide even better performance.

Note: We shared all these optimizations as patch in the Github repository and in the Dockerfile, so you can replicate these results easily. See Section 4.1 on using the docker image in Terra platform and Section 4.2 to run locally using Docker.

Step 1: Create global variables for TF session, and the output tensor:

```
...
VARIANT_TYPE_FIFO_INDEX = 6
VARIANT_FIFO_FIELDS = 7

session = None
out_tensor = None

CIGAR_CODES_TO_COUNT = [
...
```

Step 2: And add the two lines at the beginning of the function, “score_and_write_batch”:

```
...
variant_data = []
read_batch = []

global session
global out_tensor

for _ in range(batch_size):
...
```

Step 3: Then, replace the line below.

```
...
elif tensor_type in defines.TENSOR_MAPS_2D:
     predictions = model.predict(
                [np.array(read_batch), np.array(annotation_batch)],
                batch_size=python_batch_size)
else:
     raise ValueError('Unknown tensor mapping.  Check architecture file.',
                    tensor_type)
...
```

With the code below

```
...
elif tensor_type in defines.TENSOR_MAPS_2D:
     if session is None:
          full_model = tf.function(lambda x: model(x))
          full_model = full_model.get_concrete_function(
               (tf.TensorSpec(model.inputs[0].shape,
               model.inputs[0].dtype, name="read_tensor"),
               tf.TensorSpec(model.inputs[1].shape,
               model.inputs[1].dtype, name="best_practices")))
```

```
                frozen_func = convert_variables_to_constants_v2(full_model)
                frozen_func.graph.as_graph_def()
                session = tf.compat.v1.Session(graph=frozen_func.graph)
                prob_tensor = frozen_func.graph.get_tensor_by_name(
                           "model_1/softmax_predictions/Softmax:0")
            batch = {}
            batch["read_tensor:0"] = np.array(read_batch)
            batch["best_practices:0"] = np.array(annotation_batch)
            predictions = session.run(prob_tensor, feed_dict=batch)
        else:
            raise ValueError('Unknown tensor mapping.  Check architecture file.',
                           tensor_type)
        ...
```

Although the batch size while running CNNScoreVariants is fixed for every iteration, last iteration may not have enough data to fill the full batch so this may cause the actual inference batch size to be reduced and lead to the model to be recompiled for OpenVINO™. Creating fake data to fill the rest of the batch will prevent the model to be recompiled while still keeping the computation and the output as expected.

Step 4: Please add the code below before the line "`batch = {}`"

```
...
            read_batch.append(tensor)

        for i in range(python_batch_size-batch_size):
            tensor = np.empty(read_batch[0].shape)
            read_batch.append(tensor)
            tensor = np.empty(annotation_batch[0].shape)
            annotation_batch.append(tensor)

        batch = {}
        ...
```

Step 5: Now run the CNNScoreVariants with the same parameters again to see the further improvement in the performance:

```
        gatk CNNScoreVariants \
                -I gs://gatk-tutorials/workshop_2002/2-
        germline/CNNScoreVariants/bams/g94982_chr20_1m_10m_bamout.bam \
                -V gs://gatk-tutorials/workshop_2002/2-
        germline/CNNScoreVariants/vcfs/g94982_b37_chr20_1m_15871.vcf.gz \
                -R gs://gcp-public-data--broad-
        references/hg19/v0/Homo_sapiens_assembly19.fasta \
                -O data/my_2d_cnn_scored.vcf \
                --tensor-type read_tensor \
                --transfer-batch-size 256 \
                --inference-batch-size 256
```

The last few lines of the output should look like below.

```
03:36:44.913 INFO  ProgressMeter -          20:8840293              6.5
31742           4863.9
03:36:44.913 INFO  ProgressMeter - Traversal complete. Processed 31742 total
variants in 6.5 minutes.
03:36:44.913 INFO  CNNScoreVariants - Done scoring variants with CNN.
03:36:44.969 INFO  CNNScoreVariants - Shutting down engine
```

```
[May 18, 2021 at 3:36:44 a.m. PDT]
org.broadinstitute.hellbender.tools.walkers.vqsr.CNNScoreVariants done. Elapsed
time: 7.16 minutes.
Runtime.totalMemory()=434110464
```

The throughput of this execution is **4863.9 variants per minute which is 32% higher** than the baseline throughput.

4.7 Running CNNScoreVariants with OpenVINO™ integration with TensorFlow on Jupyter Notebook

We also provided a sample notebook - GATK-OVTF-Notebook.ipynb, which is available HERE. The notebook executes the following steps:

1. Use the command below to run CNNScoreVariants on native TensorFlow runtime.

```
      OPENVINO_TF_DISABLE="1" \
      gatk CNNScoreVariants \
              -I gs://gatk-tutorials/workshop_2002/2-
germline/CNNScoreVariants/bams/g94982_chr20_1m_10m_bamout.bam \
              -V gs://gatk-tutorials/workshop_2002/2-
germline/CNNScoreVariants/vcfs/g94982_b37_chr20_1m_15871.vcf.gz \
              -R gs://gcp-public-data--broad-
references/hg19/v0/Homo_sapiens_assembly19.fasta \
              -O my_2d_cnn_scored.vcf \
              --tensor-type read_tensor \
              --transfer-batch-size 256 \
              --inference-batch-size 256
```

2. Use the command below to run CNNScoreVariants on with the OpenVINO™ integration with TensorFlow runtime.

```
      gatk CNNScoreVariants \
              -I gs://gatk-tutorials/workshop_2002/2-
germline/CNNScoreVariants/bams/g94982_chr20_1m_10m_bamout.bam \
              -V gs://gatk-tutorials/workshop_2002/2-
germline/CNNScoreVariants/vcfs/g94982_b37_chr20_1m_15871.vcf.gz \
              -R gs://gcp-public-data--broad-
references/hg19/v0/Homo_sapiens_assembly19.fasta \
              -O my_2d_cnn_scored.vcf \
              --tensor-type read_tensor \
              --transfer-batch-size 256 \
              --inference-batch-size 256
```

5 Performance Results

We did performance benchmarking on Intel® Core™ i5 processor and on Terra platform with Intel® Xeon® processor using the "*g94982_chr20_1m_10m_bamout.bam*" dataset. On local computer with Intel® Core™ i5, we observed a 32% improved performance using OpenVINO™ integration with TensorFlow compared to stock TensorFlow (See Table 1). On the Terra workspace with Intel® Xeon® processor, we observed 19.5% improved performance using OpenVINO™ integration with TensorFlow (See Table 2). Please note that with simple additional two lines of additional code, we observed significant performance improvement. We observed further performance improvement with specific optimizations in the CNNScoreVariants application as described previously.

The location of dataset used:
`gs://gatk-tutorials/workshop_2002/2-germline/CNNScoreVariants/bams/g94982_chr20_1m_10m_bamout.bam`

Intel i5 performance (4-Core, 8GB RAM) – CNNScoreVariants

- 32% bump in **performance**
- 1.51 mins saved over 8.67 mins of execution (~10 mins per hour)

Table 1: Performance metrics on local computer using Intel i5 processor

Framework used to run CNNScoreVariants workload	Throughput (Variants/Min)	Speedup
Stock TF (v2.4.1)	3697.5	-
OV-TF (v2.4.1, v0.5.0)	4456.8	21%
OV-TF (v2.4.1, v0.5.0) with Optimized CNNScoreVariants	4863.9	**32%**

Terra Workspace performance (4-Core Xeon, 15GB RAM) – CNNScoreVariants

- 19.5% bump in **performance**
- 2.51 mins saved over 15.72 mins of execution (~9 mins per hour)

Table 2: Performance metrics on Terra platform using Intel Xeon processor.

Framework used to run CNNScoreVariants workload	Throughput (Variants/Min)	Speedup
Stock TF (v2.4.1)	2037.0	-
OV-TF (v2.4.1, v0.5.0)	2380.2	16.8%
OV-TF (v2.4.1, v0.5.0) with Optimized CNNScoreVariants	2433.8	**19.5%**

As demonstrated in the table above, OpenVINO™ integration with TensorFlow is able to provide significant acceleration for the GATK CNN workload. Since inferencing with the large dataset can take days, OpenVINO™ integration with TensorFlow is able to save significant inference time and reduce the associated cloud costs for GATK CNN developers.

5.1 Accuracy

We compared the output files generated using stock TensorFlow and OpenVINO^{TM} integration with TensorFlow. 3 samples out of 15870 had a variation in the 3rd significant digit in the CNN_2D feature. However, the final classification is identical. Below are the differences highlighted.

```
#CHROM POS    ID     REF    ALT    QUAL   FILTER  INFO    FORMAT NA12878
-----SAMPLE 1----------
Stock Tensorflow
20  1834907 rs149222   G   A   212.77  PASS
AC=1;AF=0.500;AN=2;BaseQRankSum=0.960;CNN_2D=3.466;ClippingRankSum=0.000;DB;DP=30;ExcessHet=3.0103;FS=1.447;MLEAC=1;MLEAF=
0.500;MQ=60.00;MQRankSum=0.000;POSITIVE_TRAIN_SITE;QD=7.34;ReadPosRankSum=-0.787;SOR=0.998;VQSLOD=22.20;culprit=MQ
GT:AD:DP:GQ:PL  0/1:18,11:29:99:241,0,441

OpenVINO™ integration with TensorFlow
20  1834907 rs149222   G   A   212.77  PASS
AC=1;AF=0.500;AN=2;BaseQRankSum=0.960;CNN_2D=3.467;ClippingRankSum=0.000;DB;DP=30;ExcessHet=3.0103;FS=1.447;MLEAC=1;MLEAF=
0.500;MQ=60.00;MQRankSum=0.000;POSITIVE_TRAIN_SITE;QD=7.34;ReadPosRankSum=-0.787;SOR=0.998;VQSLOD=22.20;culprit=MQ
GT:AD:DP:GQ:PL  0/1:18,11:29:99:241,0,441

-----SAMPLE 2----------
Stock Tensorflow
```

```
20  2623191 rs2422818   C   T   491.77  PASS
AC=1;AF=0.500;AN=2;BaseQRankSum=3.426;CNN_2D=8.268;ClippingRankSum=0.000;DB;DP=34;ExcessHet=3.0103;FS=5.231;MLEAC=1;MLEAF=
0.500;MQ=60.00;MQRankSum=0.000;POSITIVE_TRAIN_SITE;QD=14.46;ReadPosRankSum=0.191;SOR=1.141;VQSLOD=24.17;culprit=MQ
GT:AD:DP:GQ:PL  0/1:15,19:34:99:520,0,290
```

OpenVINO™ integration with TensorFlow

```
20  2623191 rs2422818   C   T   491.77  PASS
AC=1;AF=0.500;AN=2;BaseQRankSum=3.426;CNN_2D=8.267;ClippingRankSum=0.000;DB;DP=34;ExcessHet=3.0103;FS=5.231;MLEAC=1;MLEAF=
0.500;MQ=60.00;MQRankSum=0.000;POSITIVE_TRAIN_SITE;QD=14.46;ReadPosRankSum=0.191;SOR=1.141;VQSLOD=24.17;culprit=MQ
GT:AD:DP:GQ:PL  0/1:15,19:34:99:520,0,290
```

-----SAMPLE 3----------

Stock Tensorflow

```
20  5873144 rs1287079   C   T   522.77  PASS
AC=1;AF=0.500;AN=2;BaseQRankSum=0.796;CNN_2D=9.162;ClippingRankSum=0.000;DB;DP=40;ExcessHet=3.0103;FS=4.677;MLEAC=1;MLEAF=
0.500;MQ=60.00;MQRankSum=0.000;POSITIVE_TRAIN_SITE;QD=13.07;ReadPosRankSum=-0.666;SOR=0.877;VQSLOD=23.84;culprit=MQ
GT:AD:DP:GQ:PL  0/1:18,22:40:99:551,0,412
```

OpenVINO™ integration with TensorFlow

```
20  5873144 rs1287079   C   T   522.77  PASS
AC=1;AF=0.500;AN=2;BaseQRankSum=0.796;CNN_2D=9.161;ClippingRankSum=0.000;DB;DP=40;ExcessHet=3.0103;FS=4.677;MLEAC=1;MLEAF=
0.500;MQ=60.00;MQRankSum=0.000;POSITIVE_TRAIN_SITE;QD=13.07;ReadPosRankSum=-0.666;SOR=0.877;VQSLOD=23.84;culprit=MQ
GT:AD:DP:GQ:PL  0/1:18,22:40:99:551,0,412
```

----

6 About Broad Institute

The Broad Institute of MIT and Harvard was founded in 2003 to empower this generation of creative scientists to transform medicine with new genome-based knowledge. The Broad Institute seeks to describe the molecular components of life and their connections; discover the molecular basis of major human diseases; develop effective new approaches to diagnostics and therapeutics; and disseminate discoveries, tools, methods and data openly to the entire scientific community.

Founded by MIT, Harvard and its affiliated hospitals, and the visionary Los Angeles philanthropists Eli and Edythe L. Broad, the Broad Institute includes faculty, professional staff and students from throughout the MIT and Harvard biomedical research communities and beyond, with collaborations spanning over a hundred private and public institutions in more than 40 countries worldwide. For further information about the Broad Institute, go to www.broadinstitute.org.

7 About Intel

Founded in 1968, Intel's technology has been at the heart of computing breakthroughs. We are an industry leader, creating world-changing technology that enables global progress and enriches lives. We stand at the brink of several technology inflections—artificial intelligence (AI), 5G network transformation, and the rise of the intelligent edge—that together will shape the future of technology. Silicon and software drive these inflections, and Intel is at the heart of it all.

Notices & Disclaimers
Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex .
Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.
Your costs and results may vary.
Intel technologies may require enabled hardware, software or service activation.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.
© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.