Accelerating Genome Workloads Using the 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

In this blog post, we'll show how we accelerated the performance of the Genome Analysis Toolkit (GATK), a popular genome sequencing workload, using the OpenVINO™ integration with TensorFlow. GATK was developed by the Broad Institute, who have worked with us closely on this blog.

By changing **just two lines of code**, we achieved a speed-up of 21 percent on an Intel® Core™ i5 processor; and 16.8 percent on an Intel® Xeon® processor. Further optimizations increased the speed-up to 32 percent and 19.5 percent respectively.

We'll show you how we achieved our results, and how you can replicate them.

	Intel® Core™ i5 processor Speed-Up	Intel® Xeon® processor Speed-Up
OpenVINO™ integration with TensorFlow	21%	16.8%
OpenVINO™ integration with TensorFlow, with optimized inferencing code	32%	19.5%

Table 1: Speed-Ups achieved on GATK using the OpenVINO™ integration with TensorFlow

Introducing OpenVINO integration with TensorFlow

The OpenVINO toolkit was released in 2018. Using the toolkit, developers can optimize the performance of deep learning models on Intel® architecture. Developers use the OpenVINO API to develop their applications, and can import models from TensorFlow, PyTorch, and ONNX.

Now, Intel has released the OpenVINO integration with TensorFlow. Using it, developers can improve the performance of inferencing on Intel architecture, while still using TensorFlow APIs.

No explicit or offline model conversions are required. Optimizations are carried out inline, with only two lines of code required to import and configure OpenVINO. Using the OpenVINO integration requires a minimal amount of additional memory and disk space.

A wide range of TensorFlow models and operators are supported, with more still being added. Accuracy is nearly identical to the original model, as we show later in this blog post.

The OpenVINO integration with TensorFlow partitions TensorFlow graphs into multiple subgraphs. These subgraphs are then dispatched to the OpenVINO runtime for accelerated inferencing or to the TensorFlow runtime if OpenVINO does not support the subgraph. The results are combined at the end to provide the final inferencing results. Figure 1 shows the workflow.

Commented [SM1]: In line with Intel's usual trademark guidance, I'm using the TM symbol on first use in each context. A new context is first heading, first body copy mention, and each figure plus figure captions.

Commented [SM2]: (I'm assuming this will be published on intel.com here)

Commented [PR3R2]: Yes, this will be published on intel.com

Commented [SM4]: To use these stats, Intel Legal will require us to have the full hardware and software specs, details of who did the testing and when, and any security mitigations applied.

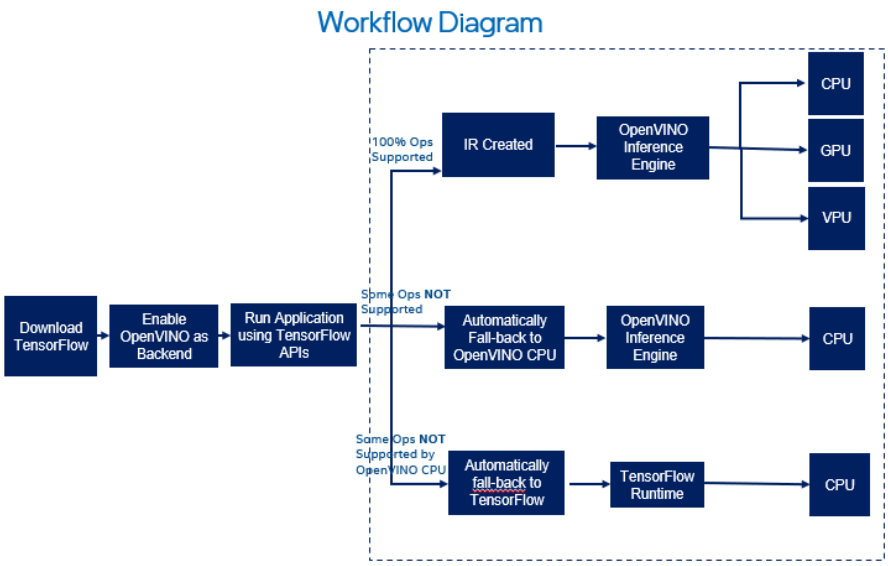


Figure 2: Workflow diagram showcasing OpenVINO™ integration with TensorFlow

The OpenVINO integration with TensorFlow accelerates inference on a range of Intel® processors such as:

- Intel® CPU (e.g., Core, Xeon including the newer SKUs, e.g. TigerLake, IceLake)
  Intel® integrated GPUs
- Intel® Movidius™ Vision Processing Units (VPUs)
- Intel® Vision Accelerator Design with 8 Intel® Movidius™ Myriad™ X VPUs (VAD-M)

The OpenVINO integration with TensorFlow is designed for developers who want to use the OpenVINO toolkit to enhance inferencing performance with minimal code modifications. For maximum performance, efficiency, tooling customization, and hardware control, we recommend adopting the full Intel® Distribution of OpenVINO™ toolkit, using its native OpenVINO APIs and the native OpenVINO runtime.

How to add the OpenVINO integration with TensorFlow

To accelerate TensorFlow performance with the integrated OpenVINO toolkit, add these two lines to your Python code (see Figure 2):

```
import openvino_tensorflow

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

Supported backends include 'CPU', 'GPU', 'MYRIAD', and 'VAD-M'. You can only use one backend at a time in the OpenVINO integration with TensorFlow. The full Intel® Distribution of OpenVINO™ toolkit enables multiple different backends to be used simultaneously.

Commented [SM5]: Where it says "Some Ops NOT Supported", should this say "NOT Supported by GPU/VPU"? According to this figure, they are still supported by the CPU.

Commented [PA6R5]: Fixed picture.

Commented [SM7]: Is this list exhaustive, or are other platforms supported too?

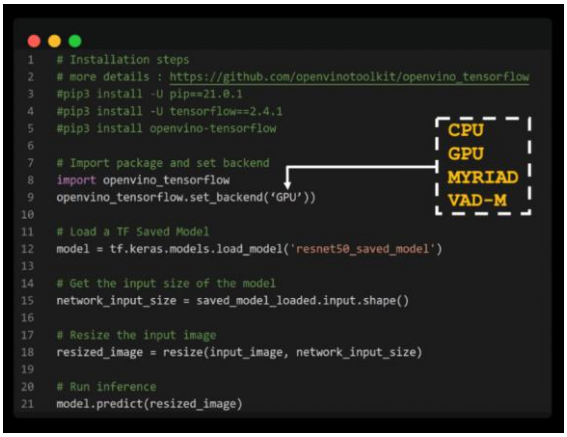
Commented [PA8R7]: Add more clarity

Commented [SM9]: Is this list exhaustive or are other options also available?

Commented [PA10R9]: At the moment its exhaustive. We might be adding other options in the future.

Commented [SM11]: Is this for Intel® Movidius™ Vision Processing Units (VPUs)? If not, can we clarify what this is in relation to the Intel silicon options listed above? Can we also clarify how you would reference Movidius VPUs.

Commented [PA12R11]: Yes this is Movidius.




```
# 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 showing how easy it is to use different hardware with the openvino_tensorflow package.

Accelerating the performance of gene sequencing inference

Terra is a cloud platform for biomedical researchers to access data, run analysis tools, and collaborate. The platform is open source and is co-developed by the Broad Institute of MIT and Harvard, Microsoft, and Verily. The Terra platform includes GATK, the world's most widely used open-source toolkit for variant calling. Variant calling is the process for identifying differences between genome sequences. GATK was developed by the Broad Institute.

CNNScoreVariants is one of the deep learning tools included in GATK. It uses a Convolutional Neural Network (CNN) to annotate gene sequence variations. We'll show you how to accelerate inference performance of CNNScoreVariants without losing accuracy using the new OpenVINO™ integration with TensorFlow.

Installing CNNScoreVariants using the OpenVINO integration with TensorFlow

Figure 3 shows the workflow of CNNScoreVariants when used with the OpenVINO™ integration with TensorFlow.

Commented [SM13]: Is my description here accurate? We don't need to go into too much detail, but this will help readers from outside of the medical research community to benefit from the blog too.

Commented [PR14R13]: Ok. That sounds right

Commented [SM15]: (I've deleted the mention of Intel Optimized TensorFlow from the original draft – I think it's confusing to mention an unrelated but similar sounding product to the one we're talking about. – Let me know if it is essential to include it, and I can add it back in.)

Commented [SM16]: Please check this summary too.

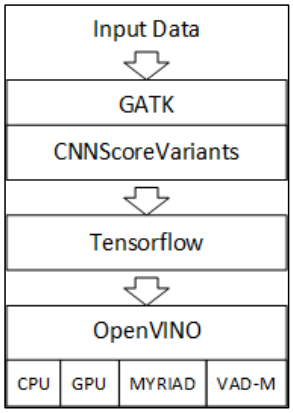


Figure 1: High level overview of the CNNScoreVariants workflow.

You can access this workflow in several ways, which we will describe in this blog. Follow the links below to jump straight to the instructions for your platform of choice:

- The Terra platform includes a docker image you can use with a Jupyter Notebook
- You can run CNNScoreVariants locally using our Docker image
- You can download, build, and install GATK locally on a Linux server

Setting up CNNScoreVariants on Terra Platform

In the Terra platform, the OpenVINO integration with TensorFlow image is available under the Community-Maintained Jupyter images (See Figure 6). Please follow the steps below to use the docker image on the Terra platform.

1. Log in to the Terra platform.
2. Click on the menu icon at the top-left corner of the page (Figure X) and choose "Workspaces" (see Figure 2).
3. Click on the "+" icon to create a new workspace as shown in Figure 3.
4. Chose a workspace name and select the billing project. Then, click on the "CREATE WORKSPACE" button (see Figure 4).
5. Once the workspace opens, click on the "Update cloud information" button on the top-right corner of the page (see Figure 5).
6. Use the drop-down menu "Application configuration" to choose the docker image called "OpenVINO™ integration with TensorFlow (openvino-tensorflow 0.5.0, Python 3.7.10, GATK 4.2.0.0)". Then, click to the "CREATE" button (see Figure 6). Please note that the versions listed might change as we keep updating the images.
7. To start the cloud environment, click to the start button at the top-right corner of the page. Starting the cloud environment might take a few minutes (see Figure 7).
8. Once the cloud environment starts, chose the "NOTEBOOKS" tab (see Figure 8) and create a new notebook (see Figure 9). Choose a name for your notebook and choose Python 3 as language (see Figure 10).
9. Once the notebook is created, click on the notebook that you created and then select "EDIT" (see Figure 11). To test the performance of the OpenVINO integration with TensorFlow, run the sample notebook, GATK-OVTF-Notebook.ipynb.

Commented [SM17]: For consistency with the other sections, can we provide step by step instructions to navigate to this part of the UI and set up the image? This could perhaps include how to run the Docker image and access the Jupyter notebooks, for consistency with the separate Docker section.

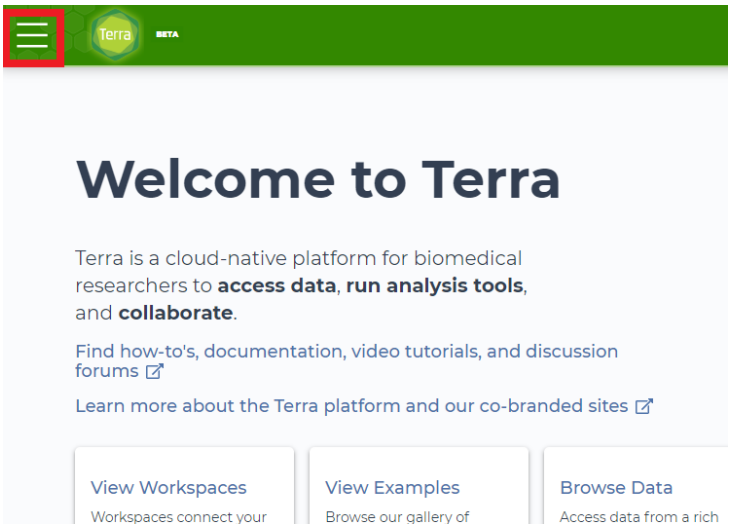


Figure 2: Use the menu at the top-left corner of the page to navigate to the workspaces page.

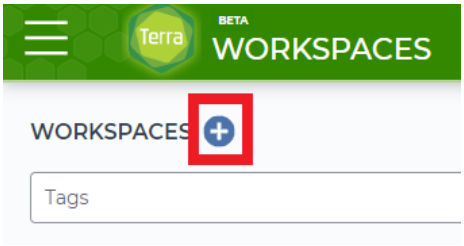


Figure 3: Create a new workspace.

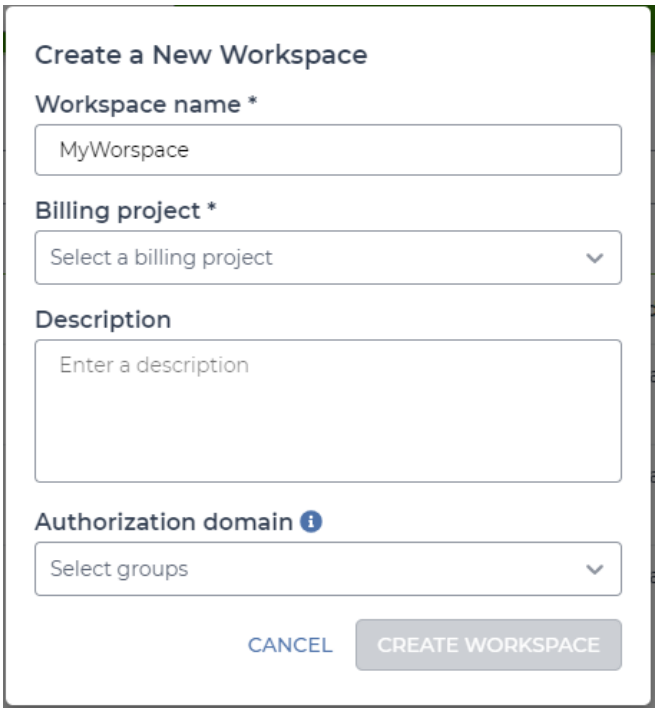


Figure 4: Choose any name and a billing project for the new workspace.

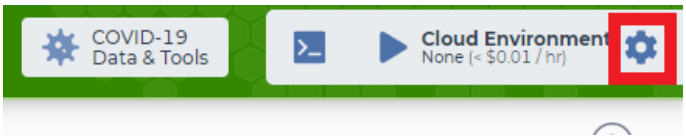


Figure 5: Open the cloud environment options

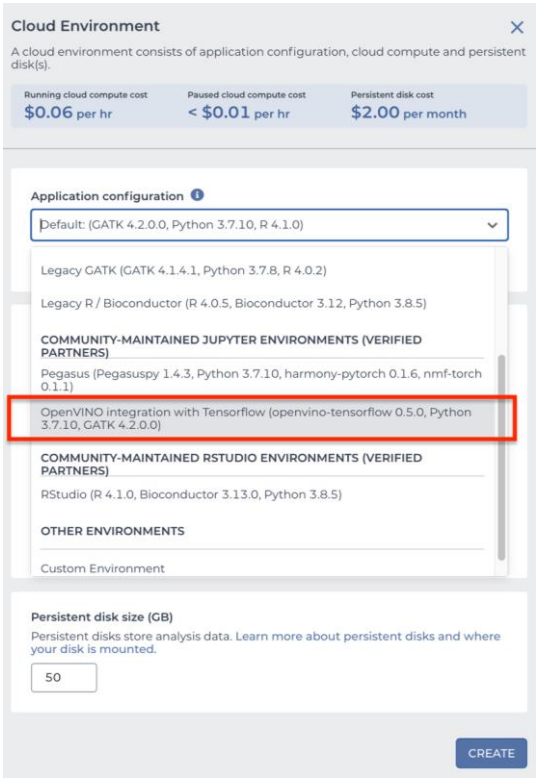


Figure 6: The OpenVINO™ integration with TensorFlow image available in Terra under the Community-Maintained Jupyter Environments.

Figure 7: Start the cloud environment.

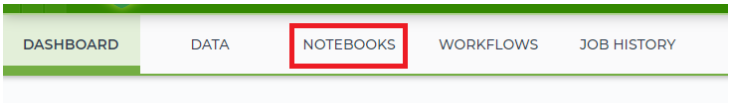


Figure 8: Select Notebook tab.

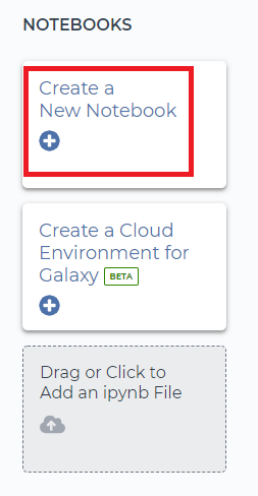


Figure 9: Create a new notebook

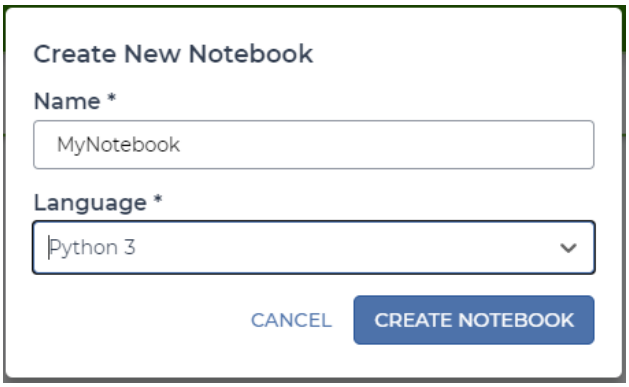


Figure 10: Choose any name and Python 3 as language for the new notebook.

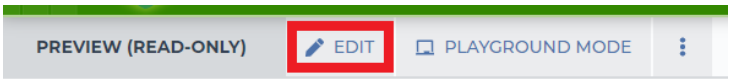


Figure 11*: Choose edit to start experimenting with OpenVINO™ integration with TensorFlow.*

Setting up CNNScoreVariants locally using Docker

Docker provides a quick way to run the optimized CNNScoreVariants with the OpenVINO™ integration with TensorFlow. There are two ways to obtain the Docker image.

Commented [SM18]: Does this include the optimization code changes mentioned later on in this article?

Method 1: Pull the docker image and Run it.

1. Use `us.gcr.io/broad-dsp-gcr-public/terra-jupyter-gatk-ovtf:latest`. See CHANGELOG.md for other versions if needed.

   ```
   docker pull us.gcr.io/broad-dsp-gcr-public/terra-jupyter-gatk-ovtf:latest
   ```

2. Run the docker image.
   ```
   docker run --rm -it -p 8000:8000 us.gcr.io/broad-dsp-gcr-public/terra-jupyter-gatk-ovtf:latest
   ```

3. In a browser, visit http://localhost:8000/notebooks to access the Jupyter UI.

Method 2: Build the docker image locally and Run it

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

2. Change 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.
   ```
   docker run --rm -it -p 8000:8000 terra-jupyter-gatk-ovtf
   ```

5. In a browser, visit http://localhost:8000/notebooks to access the Jupyter UI.

6. To test the performance of the OpenVINO integration with TensorFlow, run the sample notebook - GATK-OVTF-Notebook.ipynb. Find out more about using the Jupyter notebook below.

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 the OpenVINO™ integration with TensorFlow and run GATK on standard TensorFlow, set the OPENVINO_TF_DISABLE environment variable to 1.

```
export OPENVINO_TF_DISABLE =”1”
```

Setting up CNNScoreVariants by installing locally on a Linux Server

In this section, we will show you how to install CNNScoreVariants with OpenVINO™ integration with TensorFlow on a local Linux server.

1. Clone the repository and check out tag 4.2.0.0.

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

Commented [SM19]: Please check this. I've split one step into two here for clarity. Previously the flow was "run docker image", "navigate in browser", "code to run docker image".

Commented [SM20]: Please check this. I've split one step into two here for clarity. Previously the flow was "run docker image", "navigate in browser", "code to run docker image".

Commented [SM21]: Should they download this before going into the Jupyter UI?

Commented [PR22R21]: Yes, they should download it. Or copy the commands into their Jupyter UI.

Commented [SM23]: Jump link to the section below please, to help readers navigate.

Commented [SM24]: Why is this required? Should this step be higher up in the step-by-step guide if it's used for this process? Is this a substitute for Step 4?

Commented [PR25R24]: It is a substitute of Step 4. This allows for root access. I'm moving out of the step-by-step. Making just a note.

Commented [SM26]: Do we need this here? We don't mention this in the previous Terra section, which is also using the Jupyter notebook. Should we move this into the section about the Jupyter notebook below to avoid repetition but ensure it is signposted from wherever the notebook is used?

Commented [PR27R26]: Good Point. Yes, we can move this to the Jupyter Section. This can be used whenever the GATK is used.

Commented [SM28]: Please check. Is this word correct here or should it be omitted?

Commented [PR29R28]: Deleted.

2. Build GATK.
```
./gradlew bundle
```

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

   Option 1: To 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/broadinstitute/hellbender
```

4. Install TensorFlow and OpenVINO integration with TensorFlow
   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
```

5. We need to upgrade CNNScoreVariants to TensorFlow 2.x. Open this file in a code editor:
`src/main/python/org/broadinstitute/hellbender/vqsr_cnn/vqsr_cnn/models.py`

6. Add the line below to import TensorFlow.
```
import tensorflow as tf
```

7. Use the table below to update the lines shown to their new versions.

Old Version	New Version
`import keras.backend as K`	`import tensorflow.compat.v1.keras.backend as K`
`cfg = K.tf.ConfigProto(intra_op_parallelism_threads=intra_ops, inter_op_parallelism_threads=inter_ops)`	`cfg = tf.compat.v1.ConfigProto(intra_op_parallelism_threads=intra_ops, inter_op_parallelism_threads=inter_ops)`
`K.set_session(K.tf.Session(config=cfg))`	`K.set_session(tf.compat.v1.Session(config=cfg))`

Measuring the speed-up from the OpenVINO integration

To measure the speed-up from the OpenVINO integration with TensorFlow, we ran CNNScoreVariants twice. The first time, we ran it using standard TensorFlow. The second time, we ran it using the OpenVINO integration with TensorFlow.

Commented [SM30]: Why is this necessary? Is it because the GATK image we're using has not been updated? Is this a step you've already done in the Docker image and/or Jupyter notebooks?

Commented [PR31R30]: Yes. the Base GATK is not updated.

Commented [PR32R30]: Yes, this is already done in Docker image and so the Jupyter notebook uses TF2.x

Commented [SM33]: Based purely on reading this blog, this suggests TensorFlow was not already being used by CNNScoreVariants. Is that right?

Commented [PA34R33]: No. TF was being used earlier too.

Commented [SM35]: Is this suggested new format combining several steps into a single table okay? It uses less space and may be easier for readers to follow than having several separate steps for individual code changes.

Commented [PR36R35]: This looks great ! Thanks !

Commented [SM37]: Should we add a final step here on how to run the software now it's all been installed, and perhaps a note about using the test data? Would the Jupyter notebook also be used here?

Commented [PR38R37]: Yes, we can use the same jupyter notebook to test it.

Running with standard TensorFlow

If you did not set the PYTHONPATH to the 'gatktools' source directory, you need the rebuild GATK and reinstall the 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, as shown in bold above.

Running with the OpenVINO integration with TensorFlow

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

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

Run CNNScoreVariants with the same parameters to see the performance improvement.

```
gatk CNNScoreVariants \
```

Commented [SM39]: Do these instructions work for all three of the installation options we outlined above?

Commented [PR40R39]: Yes.

Commented [SM41]: The spacing looks different here. I'm guessing your original blog was on US Format Word pages and mine is on UK format A4 with a narrower page width. The final code will be on a web page, where I expect the code will only start a new line at a line break and will scroll horizontally, so I don't think this will be a problem. This comment applies also to later code.

Commented [PR42R41]: OK

Commented [SM43]: (I added the bold)

```
        -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.g
z \

        -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 this:

```
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 is 4456.8, which is **20 percent higher** than the baseline throughput.

Running CNNScoreVariants on Jupyter Notebook

We also provided a sample Jupyter notebook - GATK-OVTF-Notebook.ipynb. The notebook executes the following steps:

1. The notebook uses the command below to run CNNScoreVariants on the 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 \
```

Commented [SM44]: Does this notebook include the code optimizations mentioned later on or does it just include the OpenVINO integration with TensorFlow?

Commented [PR45R44]: The code optimizations are done in GATK.
The notebook has instructions to use OVTF with GATK.
The notebook can run either base GATK or optimized GATK.
The Terra docker image and local build instructions for docker image has optimized GATK

Commented [SM46]: Is there a missing export command here? Earlier in the previous draft paper, the guidance said to disable OV-TF, you use:
`export OPENVINO_TF_DISABLE ="1"`

Commented [PR47R46]: This is a single line command, so it is fine.
If we use export … then it needs to be on a separate line.

Commented [SM48]: Please double-check this is correct. I think this is a line continuation symbol but that 'gatk' starts a new instruction? Should this be a | ?

Commented [PR49R48]: This is correct. We would want to prefix the GATK instruction with the env variable - OPENVINO_TF_DISABLE

```
            -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. The notebook uses the command below to run CNNScoreVariants using the OpenVINO integration with TensorFlow.

```
    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
```

When you run the notebook, you'll see the test results using standard TensorFlow, followed by those using the OpenVINO integration with TensorFlow.

Commented [SM50]: Is this correct?

Commented [PR51R50]: What we meant was, when we finish running the entire notebook. In the notebook there are two "cells". After running both cells, the first jupyter cell output will show standard TF output. After running the second cell, the output of OVTF will be shown

Improving performance further with code optimizations for CNNScoreVariants

We saw a significant improvement with just two additional lines of code, as shown above. Now, we'll share some additional tweaks we made to further improve the CNNScoreVariants workload.

We have shared all these optimizations as a patch in the GitHub repository and in the Dockerfile, so you can replicate these results easily.

Please, apply the changes below before building GATK (see the steps for building GATK in section "Setting up CNNScoreVariants by installing locally on a Linux Server").

Freezing the Keras model

Freezing the Keras model will clear the training ops from the model and prepare it for deployment. This will improve the model coverage of the OpenVINO™ integration and will provide even better performance. Follow the steps below to update the "*src/main/python/org/broadinstitute/hellbender/vqsr_cnn/vqsr_cnn/inference.py*" file.

1. Create global variables for the TensorFlow session and the output tensor by adding the instructions in bold below:

Commented [SM52]: It's not clear to me which file we are editing in these steps. Could we clarify?

Commented [CM53R52]: Thanks for catching this. I added the file name to the description.

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

session = None
out_tensor = None

CIGAR_CODES_TO_COUNT = [
...
```

2. Add the two lines in bold below near the beginning of the function 'score_and_write_batch':

```
...
def score_and_write_batch(model: keras.Model,
  ...
  variant_data = []
  read_batch = []

  global session
  global out_tensor

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

3. **Replace** the line shown in bold 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)
```

Commented [SM54]: If this is the beginning of the function, it might help readers to orientate themselves if we included the function definition line at the start of this code snippet?

Commented [CM55R54]: Function definition line is added but between the function definition and the first line of the function, there are 24 lines including the function parameters and comment lines for function description. I add another "..." to skip those parts.

```
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)
```

```
...
```

Fill the final batch

CNNScoreVariants uses batched input data for inference execution. The batch size was specified by the user by setting the batch parameter. Then, the input data will be divided into small batches of data and each inference iteration will perform on a batch of data until the whole input data is consumed. However, the total size of the input data may not be the multiple of the specified batch size. This may result the last batch of data to have a smaller batch size than the previous batches. This change in the batch size impacts the input shape of the data. It requires the model to be recompiled for the modified input shape which causes an extra latency. To overcome this issue, we can simply fill the batch with fake data to match the batch size of the previous inference iterations. This can prevent the recompilation and result in a better performance. Also, it will not impact the output since we can simply ignore the outputs computed from the fake data.

The changes below are made in the file "src/main/python/org/broadinstitute/hellbender/vqsr_cnn/vqsr_cnn/inference.py".

1. Add the code in bold below before the line '`batch = {}`'

Commented [SM56]: Can we clarify which file we're editing here?

```
...
            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 = {}
        ...
```

Seeing the performance improvement

Now run 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 this:

```
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 31 percent higher than the baseline throughput.

Performance results

We measured the performance using an Intel® Core™ i5 processor on the desktop and an Intel® Xeon® processor on the Terra platform.

We used the "*g94982_chr20_1m_10m_bamout.bam*" dataset. You can find it here:

gs://gatk-tutorials/workshop_2002/2-germline/CNNScoreVariants/bams/g94982_chr20_1m_10m_bamout.bam

We saw significant improvement by simply adopting the OpenVINO integration with TensorFlow, which required just two lines of code. We saw even greater improvements with the code optimizations described above. Table 2 below summarizes the results.

	Intel® Core™ i5 processor (4 cores, 8GB RAM)		**Intel® Xeon® processor (4 Cores, 15GB RAM)**	
Framework used to run CNNScoreVariants workload	**Throughput (Variants/ Min)**	**Speedup**	**Throughput (Variants/ Min)**	**Speedup**
Standard TensorFlow (v2.4.1)	3697.5	-	2037.0	-
OpenVINO™ integration with TensorFlow (v2.4.1, v0.5.0)	4456.8	21%	2380.2	16.8%

Commented [SM57]: We've rounded up from 31.54 here. Above, we rounded down from 20.53 to get 20%. Should we be consistent in how we are rounding? It may be better to round down, to avoid any claim that we are overstating performance.

Commented [PR58R57]: OK

Commented [SM59]: Which one? I think we'll need full specs to be able to publish this blog post. Intel Legal usually requires full specs including hardware, software, date of testing, who did the testing, and what security mitigations were applied.

Commented [SM60]: Will readers understand where this is? This isn't clear to me from the article above.

Commented [PA61R60]: This is a standard URL for GCP buckets. It's well known industry standard.

Commented [SM62]: I've changed "stock TensorFlow" to "standard TensorFlow" for readability throughout. However, near the top of the original draft, it said that GATK Uses the Intel Optimized Version of TensorFlow. Here, we're using GATK but we're testing against "standard TensorFlow". Can we clarify please?

Commented [PA63R62]: This set of test results is versus standard TensorFlow, not intel optimized.

OpenVINO™ integration with TensorFlow (v2.4.1, v0.5.0) with Optimized CNNScoreVariants	4863.9	**32%**	2433.8	**19.5%**

Table 2: Performance for CNNScoreVariants measured on Intel® Core™ i5 processor and Intel® Xeon® processor

The time saving was:

- 1.51 minutes saved over 8.67 mins of execution (~10 mins per hour), on the Intel Core i5 processor
- 2.51 minutes saved over 15.72 mins of execution (~9 mins per hour) on the Intel Xeon processor

Since inferencing with a large dataset can take days, the OpenVINO integration with TensorFlow can save significant inference time and reduce the associated cloud costs for GATK CNN developers.

Accuracy

We compared the output files generated using standard TensorFlow and the OpenVINO integration with TensorFlow. Three samples out of 15870 had a variation in the 3rd significant digit in the CNN_2D feature. However, the final classification is identical.

Here is one of the three samples that showed a variation:

#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT NA12878

Standard 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;ReadPos
RankSum=-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;ReadPos
RankSum=-0.787;SOR=0.998;VQSLOD=22.20;culprit=MQ    GT:AD:DP:GQ:PL  0/1:18,11:29:99:241,0,441
```

Find out more

- OpenVINO integration with TensorFlow at GitHub
- OpenVINO integration with TensorFlow architecture
- Example code snippets
- Installation options

About the Broad Institute

The Broad Institute of MIT and Harvard was founded in 2003 to empower 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. The institute collaborates with more than a hundred private and public institutions in more than 40 countries worldwide. For further information about the Broad Institute, go to www.broadinstitute.org.

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.