Google Discussions



# EXAM PROFESSIONAL MACHINE LEARNING ENGINEER TOPIC 1 QUESTION 145 DISCUSSI...

Actual exam question from Google's Professional Machine Learning Engineer

Question #: 145

Topic #: 1

[All Professional Machine Learning Engineer Questions]

You have trained a DNN regressor with TensorFlow to predict housing prices using a set of predictive features. Your default precision is tf.float64, and you use a standard TensorFlow estimator:

```
estimator = tf.estimator.DNNRegressor(
     feature_columns=[YOUR_LIST_OF_FEATURES],
     hidden_units=[1024, 512, 256],
     dropout=None)
```

Your model performs well, but just before deploying it to production, you discover that your current serving latency is 10ms @ 90 percentile and you currently serve on CPUs. Your production requirements expect a model latency of 8ms @ 90 percentile. You're willing to accept a small decrease in performance in order to reach the latency requirement.

Therefore your plan is to improve latency while evaluating how much the model's prediction decreases. What should you first try to quickly lower the serving latency?

A. Switch from CPU to GPU serving.

B. Apply quantization to your SavedModel by reducing the floating point precision to tf.float16.

C. Increase the dropout rate to 0.8 and retrain your model.

D. Increase the dropout rate to 0.8 in _PREDICT mode by adjusting the TensorFlow Serving parameters.

Show Suggested Answer

by imamapri at *Feb. 4, 2023, 5:02 a.m.*

## Comments

Type your comment...

Submit

**baimus** 1 month, 1 week ago

Selected Answer: B

I know the answer is B becaue the question is telegraphing it so much: "You can lower quality a bit" (waggles eyebrows) - that obviously means quantizing (the other changes are silly). But in reality A would be much more normal thing to do. It's unusual to even attempt serving an NN on CPU these days.

upvoted 1 times

**fitri001** 6 months ago

Selected Answer: B

Reduced model size: Quantization reduces the model size by using lower precision data types like tf.float16 instead of the default tf.float64. This smaller size leads to faster loading and processing during inference.
Minimal performance impact: Quantization often introduces a small decrease in model accuracy, but it's a good initial step to explore due to the potential latency gains with minimal performance trade-offs.

upvoted 4 times

**gscharly** 6 months, 1 week ago

Selected Answer: B

I went with B.

upvoted 1 times

**Carlose2108** 7 months, 3 weeks ago

Selected Answer: B

I went with B.

upvoted 1 times

**Tayoso** 10 months ago

Selected Answer: B

Switching from CPU to GPU serving could also improve latency, but it may not be considered a "quick" solution compared to model quantization because it involves additional hardware requirements and potentially more complex deployment changes. Additionally, not all models see a latency improvement on GPUs, especially if the model is not large enough to utilize the GPU effectively or if the infrastructure does not support GPU optimizations.
Therefore, the first thing to try would be quantization, which can be done relatively quickly and directly within the TensorFlow framework. After applying quantization, you should evaluate the model to ensure that the decrease in precision does not lead to an unacceptable drop in prediction accuracy.

upvoted 4 times

**Mickey321** 11 months, 1 week ago

Selected Answer: A

Very confusing A or B but leaning to A

upvoted 1 times

**Mickey321** 11 months, 1 week ago

Changed to B

upvoted 2 times

**andresvelasco** 1 year, 1 month ago

Selected Answer: B

B based on the consideration: "Therefore your plan is to improve latency while evaluating how much the model's prediction decreases"

upvoted 2 times

**Voyager2** 1 year, 4 months ago

Selected Answer: B

To me is
B. Apply quantization to your SavedModel by reducing the floating point precision to tf.float16.
Obviously that swithing to GPU improve latency BUT.... it says "Therefore your plan is to improve latency while evaluating

how much the model's prediction decreases. If you want evaluate how much decrease is because you are going to make changes that affect the prediction

upvoted 4 times

julliet 1 year, 4 months ago

according to the documentation we have to convert to TensorFlowLite before applying quantization or use an API https://www.tensorflow.org/model_optimization/guide/quantization/training
doesn't look to be first option. Second maybe?

upvoted 2 times

Voyager2 1 year, 4 months ago

Selected Answer: A

Going with A:
My reason to discard B: from https://www.tensorflow.org/lite/performance/post_training_quantization#float16_quantization
The advantages of float16 quantization are as follows:

It reduces model size by up to half (since all weights become half of their original size).
It causes minimal loss in accuracy.
It supports some delegates (e.g. the GPU delegate) which can operate directly on float16 data, resulting in faster execution than float32 computations.
The disadvantages of float16 quantization are as follows:

It does not reduce latency as much as a quantization to fixed point math.
By default, a float16 quantized model will "dequantize" the weights values to float32 when run on the CPU. (Note that the GPU delegate will not perform this dequantization, since it can operate on float16 data.)

upvoted 1 times

aryaavinash 1 year, 5 months ago

Going with B because quantization can reduce the model size and inference latency by using lower-precision arithmetic operations, while maintaining acceptable accuracy. The other options are either not feasible or not effective for lowering the serving latency. Switching from CPU to GPU serving may not be possible or cost-effective, increasing the dropout rate may degrade the model performance significantly, and dropout is not applied in _PREDICT mode by default.

upvoted 4 times

M25 1 year, 5 months ago

Selected Answer: A

For tf.float16 [Option B], we would have to be on TFLite:
https://discuss.tensorflow.org/t/convert-tensorflow-saved-model-from-float32-to-float16/12130 and resp.
https://www.tensorflow.org/lite/performance/post_training_quantization#float16_quantization (plus "By default, a float16 quantized model will "dequantize" the weights values to float32 when run on the CPU. (Note that the GPU delegate will not perform this dequantization, since it can operate on float16 data.)"

upvoted 2 times

M25 1 year, 5 months ago

But even before that, tf.estimator.DNNRegressor is deprecated, "Use tf.keras instead":
https://www.tensorflow.org/api_docs/python/tf/estimator/DNNRegressor.
When used with Keras (a high-level NN library that runs on top of TF), for training though, "It is not recommended to set this to float16 for training, as this will likely cause numeric stability issues. Instead, mixed precision, which is using a mix of float16 and float32, can be used": https://www.tensorflow.org/api_docs/python/tf/keras/backend/set_floatx.

upvoted 1 times

M25 1 year, 5 months ago

But then, "On CPUs, mixed precision will run significantly slower, however.":
https://www.tensorflow.org/guide/mixed_precision#supported_hardware.
And, "The policy will run on other GPUs and CPUs but may not improve performance.":
https://www.tensorflow.org/guide/mixed_precision#setting_the_dtype_policy.

upvoted 1 times

M25 1 year, 5 months ago

"This can take around 500ms to process a single Tweet (of at most 128 tokens) on a CPU-based machine. The processing time can be greatly reduced to 20ms by running the model on a GPU instance (…). An option to dynamically quantize a TensorFlow model wasn't available, so we updated the script to convert the TensorFlow models into TFLite and created the options to apply int8 or fp16 quantization.":
https://blog.twitter.com/engineering/en_us/topics/insights/2021/speeding-up-transformer-cpu-inference-in-google-cloud

upvoted 1 times

[Removed] 1 year, 6 months ago

Selected Answer: A

A and B both work well here, but I prefer A since B would imply some minor tradeoff between latency and model accuracy, which isn't the case for A. So I would consider quantization after switching to GPU serving. Can anyone explain why B might be better than A?

be better than A?

upvoted 1 times

**frangm23** 1 year, 6 months ago

Since you're allowing a small decrease in accuracy, you should choose B as it is more cost effective than A.

upvoted 3 times

**[Removed]** 1 year, 5 months ago

There's no mention that we're cost sensitive in this scenario. Why should we assume that reducing cost is more important than model accuracy?

upvoted 2 times

**tavva_prudhvi** 1 year, 2 months ago

Yes you're right. But, Quantization reduces the floating point precision, which can result in a smaller model size and lower memory footprint. This can lead to faster serving times and improved latency. In comparison, switching to GPU serving doesn't necessarily reduce the model size or memory footprint.Also, it provides a more direct way to balance the trade-off between model performance and latency, as it directly impacts the model's precision. Switching to GPU serving may improve latency but doesn't directly address the trade-off between performance and latency. While Switching to GPU serving may require changes to your existing serving infrastructure, which can be time-consuming and may not be compatible with your current setup. Quantization, on the other hand, is a model optimization technique that can be applied directly to the model without requiring changes to the serving infrastructure.

upvoted 2 times

**TNT87** 1 year, 7 months ago

Selected Answer: A

A makes sense too

upvoted 4 times

**TNT87** 1 year, 7 months ago

But answer is B , i dnt know how i clicked A

upvoted 2 times

**TNT87** 1 year, 7 months ago

GPU serving can significantly speed up the serving of models due to the parallel processing power of GPUs. By switching from CPU to GPU serving, you can quickly lower the serving latency without making changes to the model architecture or precision. Once you have switched to GPU serving, you can evaluate the impact on the model's prediction quality and consider further optimization techniques if necessary. Therefore, the correct option is A.

upvoted 2 times

**TNT87** 1 year, 8 months ago

Answer B
Applying quantization to your SavedModel by reducing the floating point precision can help reduce the serving latency by decreasing the amount of memory and computation required to make a prediction. TensorFlow provides tools such as the tf.quantization module that can be used to quantize models and reduce their precision, which can significantly reduce serving latency without a significant decrease in model performance

upvoted 1 times

**imamapri** 1 year, 8 months ago

Selected Answer: B

Vote B. https://www.tensorflow.org/lite/performance/post_training_float16_quant

upvoted 4 times

## Social Media

Facebook , Twitter
YouTube , Reddit
Pinterest

We are the biggest and most updated IT certification exam material website.

Using our own resources, we strive to strengthen the IT professionals community for free.