ONNX

Designed to be Optimized

Mauro Bennici

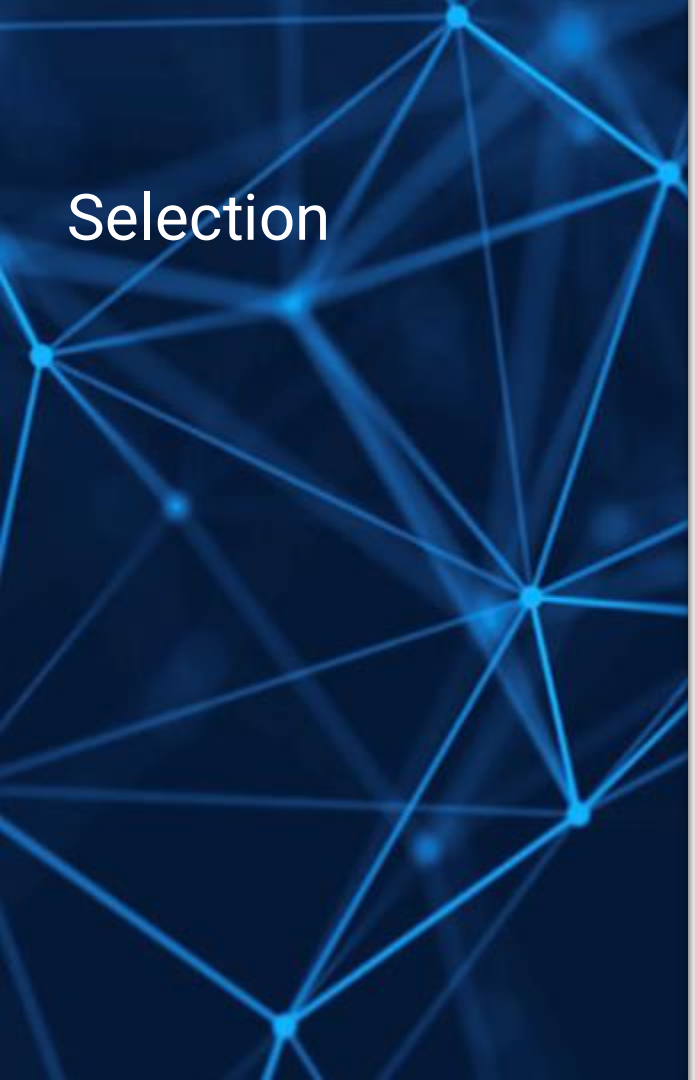
CTO and co-founder - Ghostwriter.AI

About me

- .NET Foundation Member
- DIGITAL SME Alliance - FG on AI
- Techstars Mentor
- Data Scientist
- SCRUM Master (PSM I)
- Torino.NET meetup founder

GhostWriter AI

Twitter: @maurobennici
Linkedin: https://www.linkedin.com/in/maurobennici/

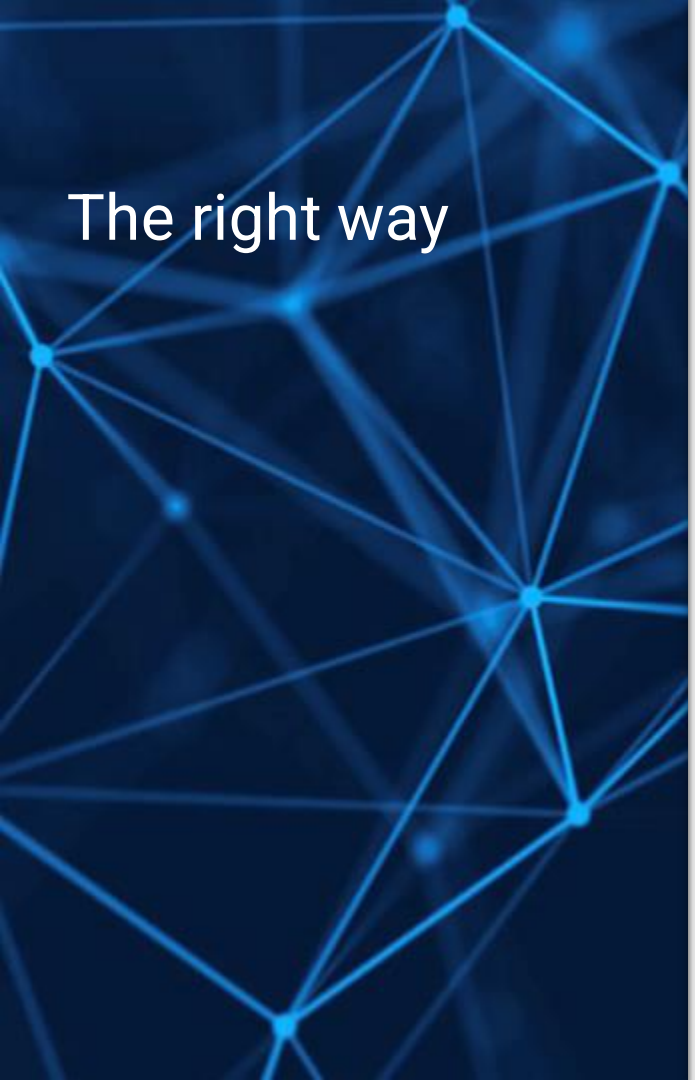


- The training tools
- The inference tools
- The devices
- **The API versions**

ONNX, the old way!

We use what we know best and, at the end, we try to optimize it:

- cloud app / service
- cloud / on-premise server
- offline device
- etc.



A lot of things need to be selected in advance.

Checking them at the end could be too late :(

Know your tool

PyTorch

Adding support for operators

When exporting a model that includes unsupported operators, you'll see an error message like:

```
RuntimeError: ONNX export failed: Couldn't export operator foo
```

Know your tool

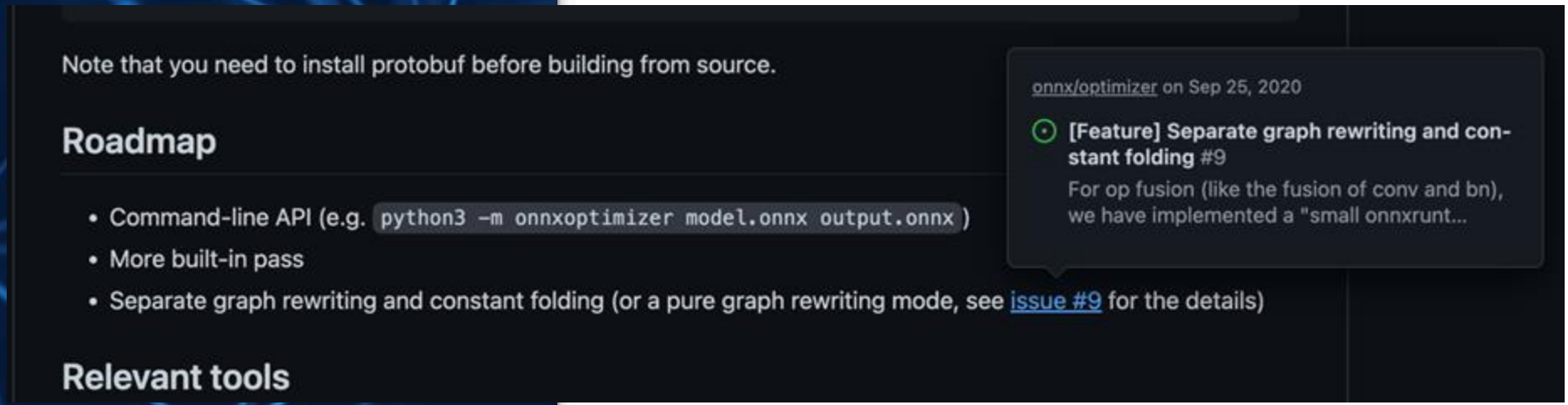
Avoiding Pitfalls

Avoid NumPy and built-in Python types

PyTorch models can be written using NumPy or Python types and functions, but during tracing, any variables of NumPy or Python types (rather than torch.Tensor) are converted to constants, which will produce the wrong result if those values should change depending on the inputs.

Know your tool

ONNX Optimizer



Know your tool

ONNX RUNTIME

ONNX Runtime quantization on CPU can run U8U8, U8S8 and S8S8. S8S8 with QDQ is the default setting and balances performance and accuracy. It should be the first choice. Only in cases that the accuracy drops a lot, you can try U8U8. Note that S8S8 with QOperator will be slow on x86-64 CPUs and should be avoided in general. ONNX Runtime quantization on GPU only supports S8S8.

WHEN AND WHY DO I NEED TO TRY U8U8?

On x86-64 machines with AVX2 and AVX512 extensions, ONNX Runtime uses the VPMADDUBSW instruction for U8S8 for performance. This instruction might suffer from saturation issues: it can happen that the output does not fit into a 16-bit integer and has to be clamped (saturated) to fit. Generally, this is not a big issue for the final result. However, if you do encounter a large accuracy drop, it may be caused by saturation. In this case, you can either try reduce_range or the U8U8 format which doesn't have saturation issues.

There is no such issue on other CPU architectures (x64 with VNNI and ARM).

Know your tool

ONNX Runtime quantization on CPU can run U8U8, U8S8 and S8S8. S8S8 with QDQ is the default setting and balances performance and accuracy. It should be the first choice. Only in cases that the accuracy drops a lot, you can try U8U8. Note that S8S8 with QOperator will be slow on x86-64 CPUs and should be avoided in general. ONNX Runtime quantization on GPU only supports S8S8.

WHEN AND WHY DO I NEED TO TRY U8U8?

On x86-64 machines with AVX2 and AVX512 extensions, ONNX Runtime uses the VPMADDUBSW instruction for U8S8 for performance. This instruction might suffer from saturation issues: it can happen that the output does not fit into a 16-bit integer and has to be clamped (saturated) to fit. Generally, this is not a big issue for the final result. However, if you do encounter a large accuracy drop, it may be caused by saturation. In this case, you can either try reduce_range or the U8U8 format which doesn't have saturation issues.

There is no such issue on other CPU architectures (x64 with VNNI and ARM).

Solution

Check the ONNX doc and the ONNX options of your frameworks, before… Be happy, later ;)

ai.onnx (default)

Operator	Since version
Abs	13, 6, 1
Acos	7
Acosh	9
Add	14, 13, 7, 6, 1
And	7, 1
ArgMax	13, 12, 11, 1
ArgMin	13, 12, 11, 1
Asin	7

Thank you!

ONNX

GhostWriter AI

Twitter: @maurobennici

Linkedin: https://www.linkedin.com/in/maurobennici/