Transpile

Transpilation is a crucial process in the deployment of Verifiable Machine Learning models. It involves the transformation of an ONNX model into a Cairo model. These models can generate proofs that can be verified, ensuring the integrity and reliability of the model's predictions.

The transpilation of an ONNX model to a Cairo model is powered by✨Orion✨

The transpilation process begins by reading the model from the specified path. The model is then sent for transpilation. By default, the output of this process is saved in thecairo_model/ folder. However, you can specify a different output path using the--output-path option.

```

Copy

> giza transpile awesome_model.onnx --output-path my_awesome_model [giza][2024-02-07 16:31:20.844] No model id provided, checking if model exists ✅ [giza][2024-02-07 16:31:20.845] Model name is: awesome_model [giza][2024-02-07 16:31:21.599] Model Created with id -> 1! ✅ [giza][2024-02-07 16:31:22.436] Version Created with id -> 1! ✅ [giza][2024-02-07 16:31:22.437] Sending model for transpilation ✅ [giza][2024-02-07 16:32:13.511] Transpilation is fully compatible. Version compiled and Sierra is saved at Giza ✅ [giza][2024-02-07 16:32:13.516] Transpilation recieved! ✅ [giza][2024-02-07 16:32:14.349] Transpilation saved at: my_awesome_model

```

The result of the transpilation process is saved at the provided path, in this case,my_awesome_model/ .

```

Copy

> tree my_awesome_model/ my_awesome_model ├── inference | ├── Scarb.toml | └── src | └── lib.cairo └── initializers ├── node_1 | ├── Scarb.toml | └── src | └── lib.cairo

```

When we transpile a model we have two possibilities: a fully compatible model and a partially compatible one.

A model is fully compatible when all the operators that the model uses are supported by the Transpiler and Orion, if this happens the model is compiled after transpilation and we save the .sierra file on behalf of the user to use later for deployment (endpoint docs). This will be shown in the output of the transpile command:

```

Copy [giza][2024-02-07 16:32:13.511] Transpilation is fully compatible. Version compiled and Sierra is saved at Giza ✅

```

If a model is partially supported, we will create a warning in the output stating that not all the operators are supported right now. If it is partially supported the Cairo code can still be modified for later compilation and endpoint.

```

Copy [WARN][2024-02-07 16:42:31.209] Transpilation is partially supported. Some operators are not yet supported in the Transpiler/Orion [WARN][2024-02-07 16:42:31.211] Please check the compatibility list in Orion: https://cli.gizatech.xyz/frameworks/cairo/transpile#supported-operators

```

Supported Operators

Operator Implemented Abs ✅ Acos ✅ Acosh ✅ Add ✅ And ✅ Div ✅ Mul ✅ Sub ✅ Argmax ✅ Argmin ✅ Asin ✅ Asinh ✅ Atan ✅ Relu ✅ Constant ✅ MatMul ✅ Gemm ✅

How do we transpile a model?

There are three main methods for transpiling a model:

Method 1: Using the giza transpile command

```

Copy giza transpile awesome_model.onnx --output-path my_awesome_model

```

This is the simplest method and is recommended for most users. When you run this command, Giza handles everything for you:

- It first checks if a model with the specified name already exists. If not, it creates a new model and then transpiles it.
- The output of this process is saved in thecairo_model/
- folder by default, but you can specify a different output path using the--output-path

- option.
-

This is the strategy that we followed in the example before.

Method 2: Manually creating a model and then transpiling it

This method gives you more control over the process.

1. First, you create a model manually using thegiza models create
2. command.
3. After the model is created, you can transpile it using thegiza transpile --model-id ...
4.

This method is useful when you want to specify particular options or parameters during the model creation and transpilation process.

```

Copy

    giza models create --name awesome_model --description"A Model for testing different models" [giza][2023-09-13 14:04:59.532] Creating model ✅ { "id": 2, "name": "awesome_model", "description": "A Model for testing different models" }

```

```

Copy

    giza transpile --model-id 2 awesome_model.onnx --output-path new_awesome_model [giza][2023-09-13 14:08:38.022] Model found with id -> 2! ✅ [giza][2024-02-07 14:08:38.432] Version Created with id -> 1! ✅ [giza][2023-09-13 14:08:38.712] Sending model for transpilation ✅ [giza][2023-09-13 14:08:49.879] Transpilation recieved! ✅ [giza][2023-09-13 14:08:49.885] Transpilation saved at: new_awesome_model

```

Method 3: Using a previous model

If you have a previously created model, you can transpile it by indicating the model-id in thegiza transpile --model-id ... orgiza versions transpile --model-id command.

- This method is useful when you want to create a new version of an existing model.
- The output of the transpilation process is saved in the same location as the original model.
-

```

Copy

# Using the previous model (id:2) we can transpile a new model, which will create version 2 of the model.

giza transpile --model-id 29 awesome_model.onnx --output-path new_awesome_model [giza][2023-09-13 14:11:30.015] Model found with id -> 2! ✅ [giza][2024-02-07 14:11:30.225] Version Created with id -> 2! ✅ [giza][2023-09-13 14:11:30.541] Sending model for transpilation ✅ [giza][2023-09-13 14:11:41.601] Transpilation recieved! ✅ [giza][2023-09-13 14:11:41.609] Transpilation saved at: new_awesome_model

```

Transpilation Results

When a version is transpiled the version can be in the following statuses:

- FAILED
- : the transpilation of the model failed
- COMPLETED: the version transpilation is completed and the version is FULLY compatible. This means that the version operators all are supported during transpilation and Orion, and the model has been compiled to sierra and saved in the platform. The model can be directly deployed without the need to provide a sierra file. This model is "frozen" so it will not allow for code or model updates and if any changes are done to the model a new version should be created.
- PARTIALLY_SUPPORTED
- : not all the operators are supported in the transpilation but they might be supported in Orion, so a partially working code will be returned, allowing for modifications of the code to update the version into a fully compatible one. Once this version is updated we will compile the version and if it is successful the new code and the sierra will be uploaded to Giza, the status will be updated toCOMPLETED
- and the version will be frozen not allowing any more modifications.
-

We try to support all the available operators on Orion but there might be a little lag between Orion's implementation and transpilation availability

How to update a transpilation

If your model version is in aPARTIALLY_SUPPORTED status, you can work towards achieving aCOMPLETED status by updating the transpilation. The update process involves modifying the unsupported operators and compiling the model. Here's how to update a transpilation, from creation to fully supported:

1. Transpile the model withgiza transpile
2. Modify your cairo model to address the unsupported operators.
3. Execute the giza version update command. This command needsscarb
4. to be installed (docs
5.), compilation will be attempted and if successful code and sierra file will be updated in Giza.
6.

Example: Updating a Version

Say you have anawesome_model.onnx that isPARTIALLY_SUPPORTED :

```

Copy ❯giza transpile awesome_model.onnx [giza][2024-02-12 13:19:55.957] No model id provided, checking if model exists ✅ [giza][2024-02-12 13:19:55.958] Model name is: awesome_model [giza][2024-02-12 13:19:56.207] Model Created with id -> 1! ✅ [giza][2024-02-12 13:19:56.710] Version Created with id -> 1! ✅ [giza][2024-02-12 13:19:56.711] Sending model for transpilation ✅ [WARN][2024-02-12 13:20:07.207] Transpilation is partially supported. Some operators are not yet supported in the Transpiler/Orion [WARN][2 13:20:07.209] Please check the compatibility list in Orion: https://cli.gizatech.xyz/frameworks/cairo/transpile#supported-operators [giza] [2024-02-12 13:20:07.773] Downloading model ✅ [giza][2024-02-12 13:20:07.783] model saved at: cairo_model

```

This version has some operators that are not available in the transpilation, but they might be supported in Orion. When a model is not fully compatible, in theinference/lib.cairo a comment will be shown:

```

Copy let node_8 = // Operator LogSoftmax is not yet supported by the Giza transpiler. If Orion supports it, consider manual implementation.;

```

Let's say thatLogSoftMax is the unsupported operator, if we check the Orion Documentation, we can see that its supported . Now we could add the necessary code to add our operator (including imports):

```

Copy let node_8 = NNTrait::logsoftmax(node_7_output_0, 1);

```

LogSoftMax serves as an example and does not mean that it is not currently supported After the manual implementation, we can trigger the update with theupdate command:

```

Copy ❯ giza versions update --model-id 1 --version-id 1 --model-path cairo_model [giza][2024-02-12 13:35:28.993] Checking version ✅ scarb 2.4.3 (5dbab1f31 2024-01-04) cairo: 2.4.3 (https://crates.io/crates/cairo-lang-compiler/2.4.3) sierra: 1.4.0

[giza][2024-02-12 13:35:29.138] Scarb is installed, proceeding with the build. Compiling inference v0.1.0 (/Users/gizabrain/cairo_model/inference/Scarb.toml) error: Unexpected argument type. Expected: "@orion::operators::tensor::core::Tensor:: <?13>", found: "orion::operators::tensor::core::Tensor::". --> /Users/gizabrain/cairo_model/inference/src/lib.cairo:15:34 let node_8 = NNTrait::logsoftmax(node_7_output, 1); ^*****^

error: could not compile inference due to previous error [ERROR][2024-02-12 13:35:34.847] Compilation failed [ERROR][2024-02-12 13:35:34.848] Error building the scarb model [ERROR][2024-02-12 13:35:34.848] Version could not be updated [ERROR][2024-02-12 13:35:34.849] Check scarb documentation https://docs.swmansion.com/scarb/

```

Here's what is going on:

- We want to update the first version of the first model with our new code, the code is at--model-path cairo_model
- The CLI checks ifscarb
- is available in the system
- scarb build
- is attempted
- We still have some errors that we have to fix
-

In this case, we purposely forgot to add the@ to showcase a common scenario:

```
```

Copy let node_8 = NNTrait::logsoftmax(@node_7_output, 1);

```

Once everything is fixed we can attempt the update again:

```

Copy ❯ giza versions update --model-id 1 --version-id 1 --model-path cairo_model [giza][2024-02-12 13:43:25.913] Checking version ✅ scarb 2.4.3 (5dbab1f31 2024-01-04) cairo: 2.4.3 (https://crates.io/crates/cairo-lang-compiler/2.4.3) sierra: 1.4.0

[giza][2024-02-12 13:43:26.064] Scarb is installed, proceeding with the build. Compiling inference v0.1.0 (/Users/gizabrain/cairo_model/inference/Scarb.toml) Finished release target(s) in 6 seconds [giza][2024-02-12 13:43:32.326] Compilation successful [giza][2024-02-12 13:43:33.708] Sierra updated ✅ [giza][2024-02-12 13:43:34.962] Version updated ✅ { "version": 1, "size": 8858, "status": "COMPLETED", "message": null, "description": "Initial version", "created_date": "2024-02-12T12:19:56.324501", "last_update": "2024-02-12T12:43:34.906667" }

```

The version has been updated successfully! Now we have a fully compatible model that generated a sierra and can be easily deployed! Now the version will be frozen so it won't allow for any more updates.

When we refer to a version of a model, we refer to the code/artifact of a specific model at a specific point in time. The model is frozen for tracking purposes.

What is happening with the models and versions?

In Giza, a model is essentially a container for versions. Each version represents a transpilation of a machine learning model at a specific point in time. This allows you to keep track of different versions of your model as it evolves and improves over time.

To check the current models and versions that have been created, you can use the following steps:

1. Use thegiza models list
2. command to list all the models that have been created.
3. For each model, you can use thegiza versions list --model-id ...
4. command to list all the versions of that model.
5.

Remember, each version represents a specific transpilation of the model. So, if you have made changes to your machine learning model and transpiled it again, it will create a new version.

This system of models and versions allows you to manage and keep track of the evolution of your machine learning models over time.

For example, let's say you have created a model calledawesome_model and transpiled it twice. This will create two versions of the model, version 1 and version 2. You can check the status of these versions using thegiza versions list --model-id ... command.

```

Copy giza versions list --model-id 29 [giza][2023-09-13 14:17:08.006] Listing versions for the model ✅ [{ "version": 1, "size": 52735, "status": "COMPLETED", "message": "Transpilation Successful!", "description": "Initial version", "created_date": "2023-09-13T12:08:38.177605", "last_update": "2023-09-13T12:08:43.986137" }, { "version": 2, "size": 52735, "status": "COMPLETED", "message": "Transpilation Successful!", "description": "Initial version", "created_date": "2023-09-13T12:11:30.165440", "last_update": "2023-09-13T12:11:31.625834" }]

```

Previous Cairo Next Deploy

Last updated17 days ago
```