



life.augmented

# From Heterogenous Development Frameworks to Unified Benchmarking in the Cloud

Marco Lattuada, PhD

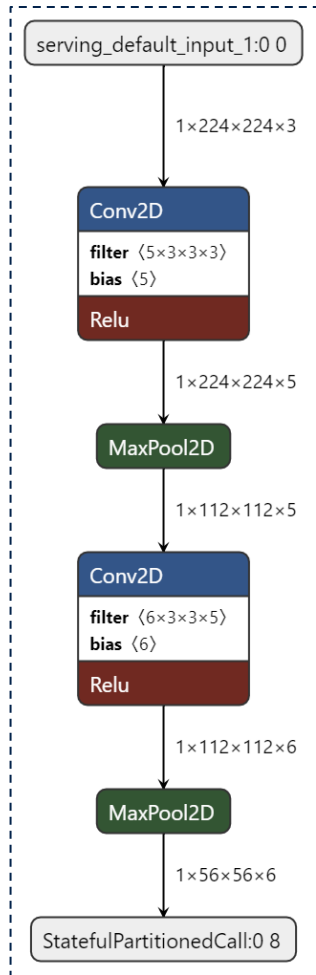
STMicroelectronics

# Introduction

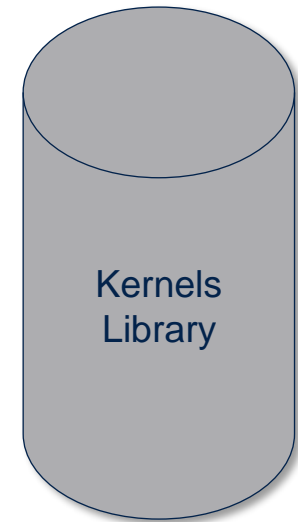
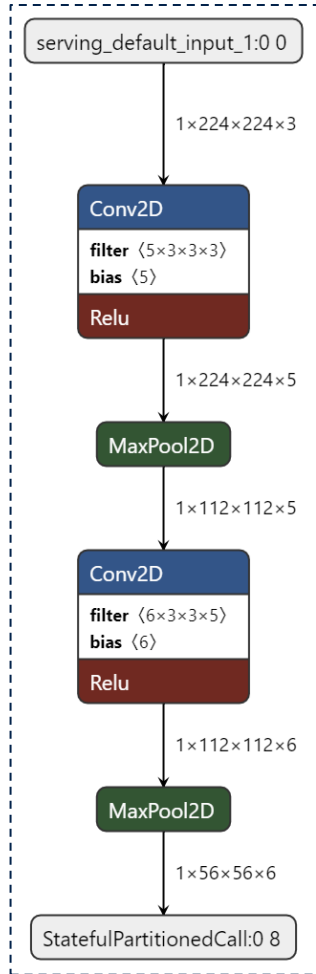
- Deployment of **Machine Learning** models on **Embedded Systems** is becoming **more and more relevant**
- Machine learning models can be very different in terms of algorithms, size, formats, etc.
- An automatic deployment flow must tackle this **heterogeneity** to reduce as much as possible the restrictions to the end users
- This presentation will show:
  - What characterize heterogeneity of Machine Learning models
  - How heterogeneity can be addressed to **maximize the support** and to **maximize sharing and reuse**



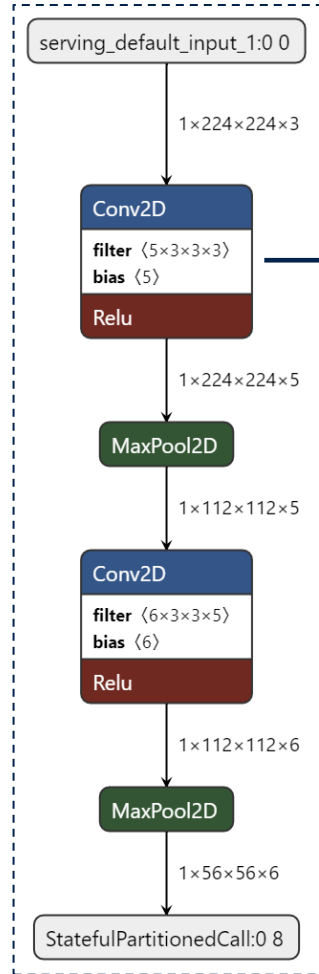
# From Machine Learning to Implementation



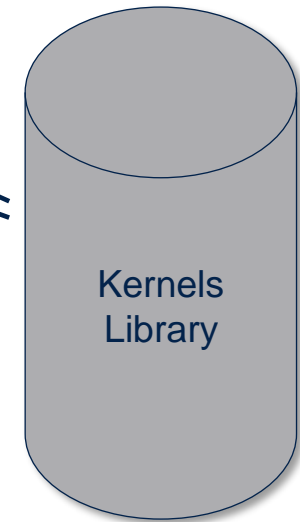
# From Machine Learning to Implementation



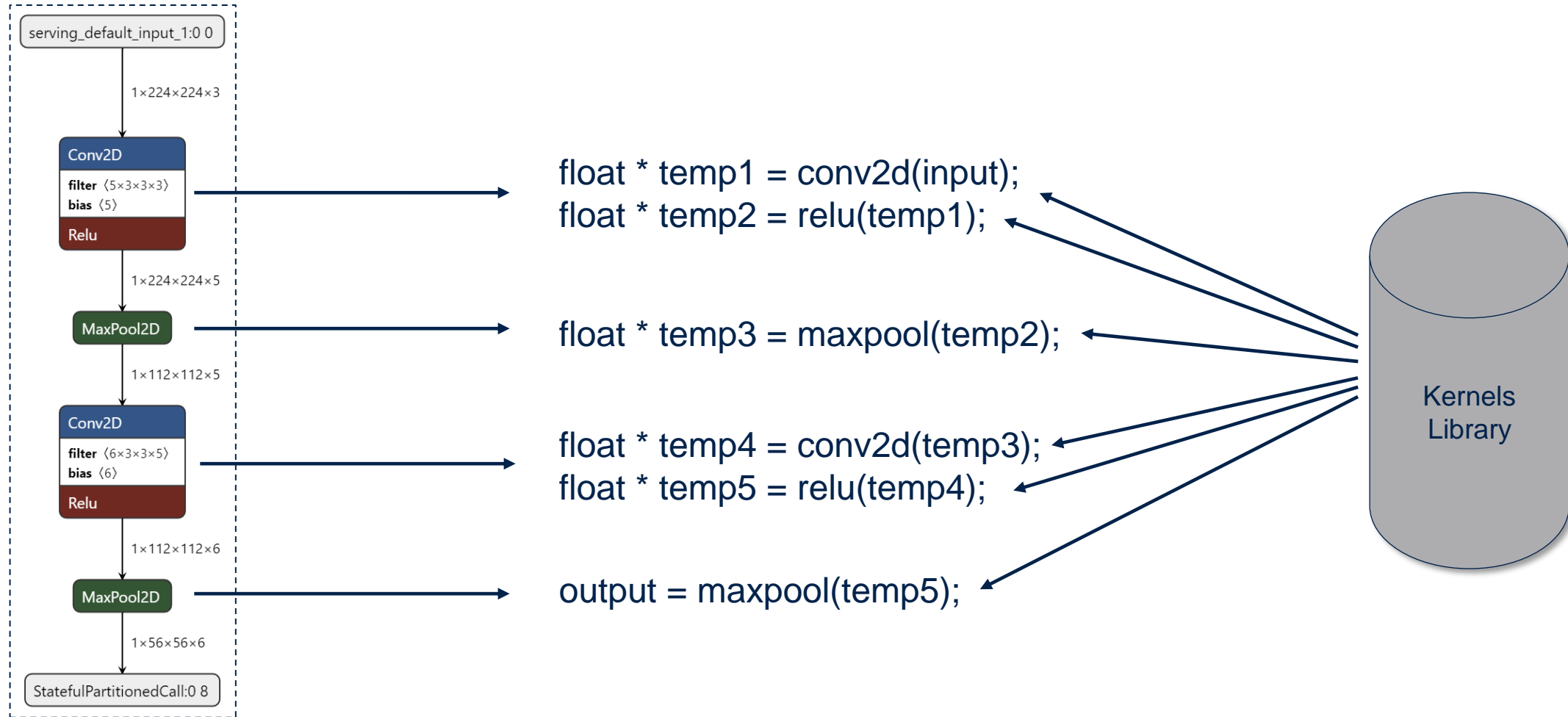
# From Machine Learning to Implementation



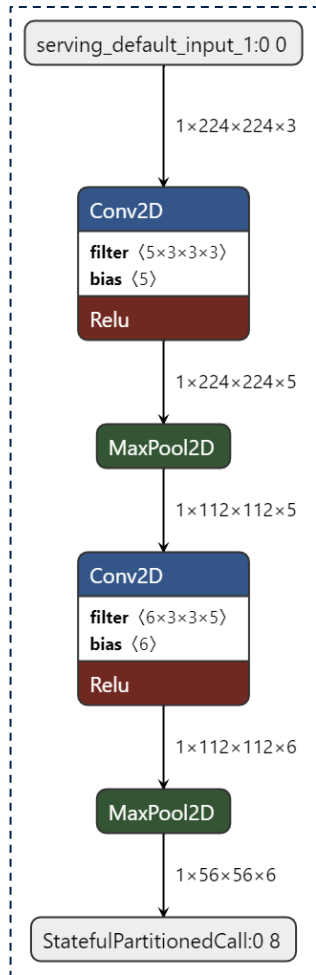
```
float * temp1 = conv2d(input);  
float * temp2 = relu(temp1);
```



# From Machine Learning to Implementation



# From Machine Learning to Implementation



```
void network(float * input, float * output) {
```

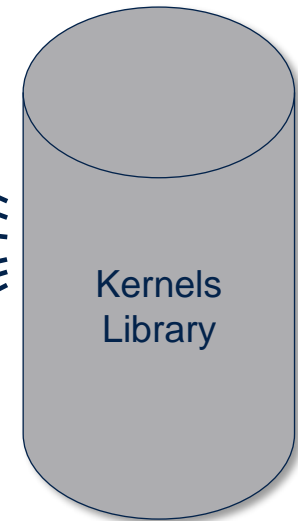
```
float * temp1 = conv2d(input);  
float * temp2 = relu(temp1);
```

```
float * temp3 = maxpool(temp2);
```

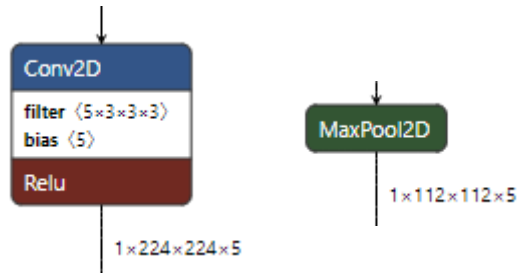
```
float * temp4 = conv2d(temp3);  
float * temp5 = relu(temp4);
```

```
output = maxpool(temp5);
```

```
}
```



# Heterogeneity: Operators

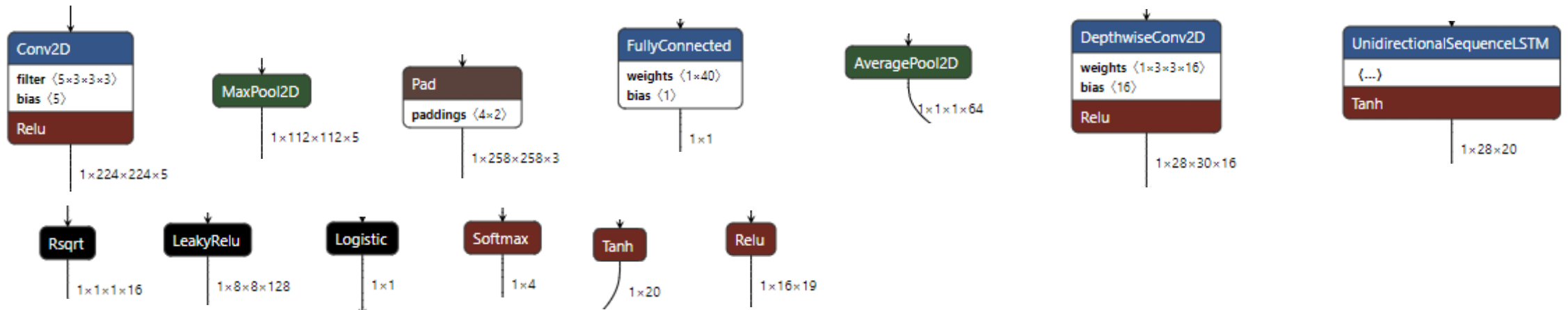




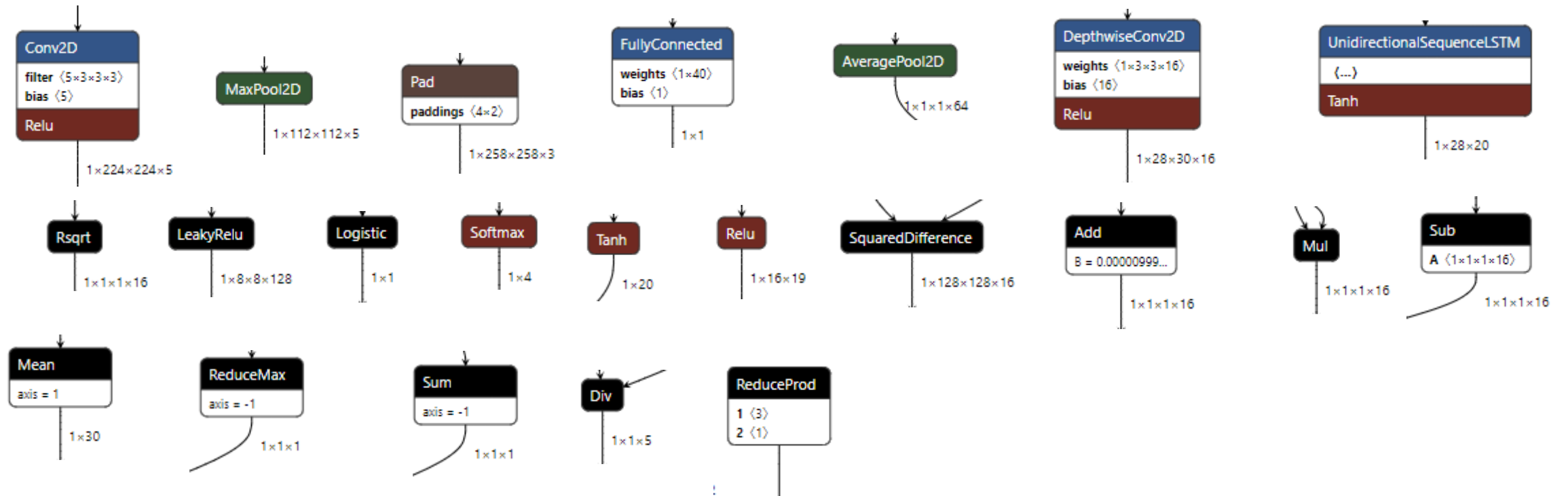
# Heterogeneity: Operators



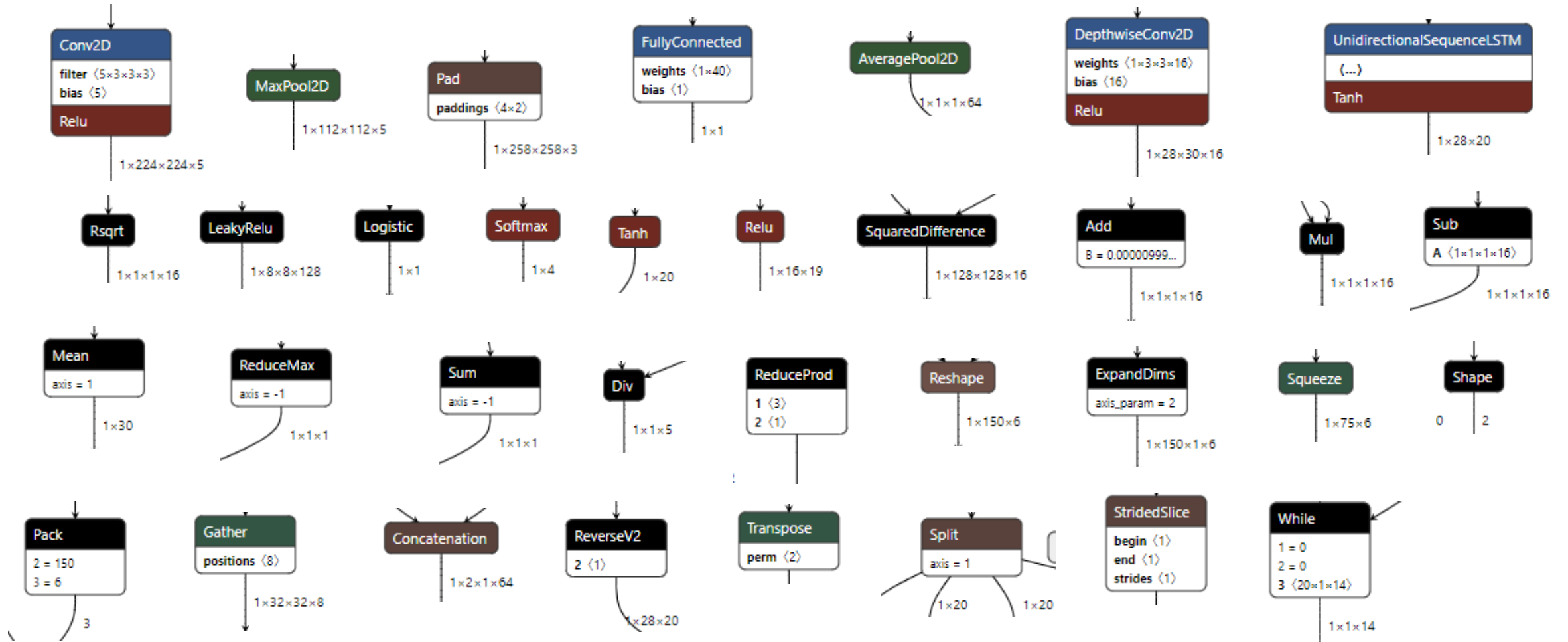
# Heterogeneity: Operators



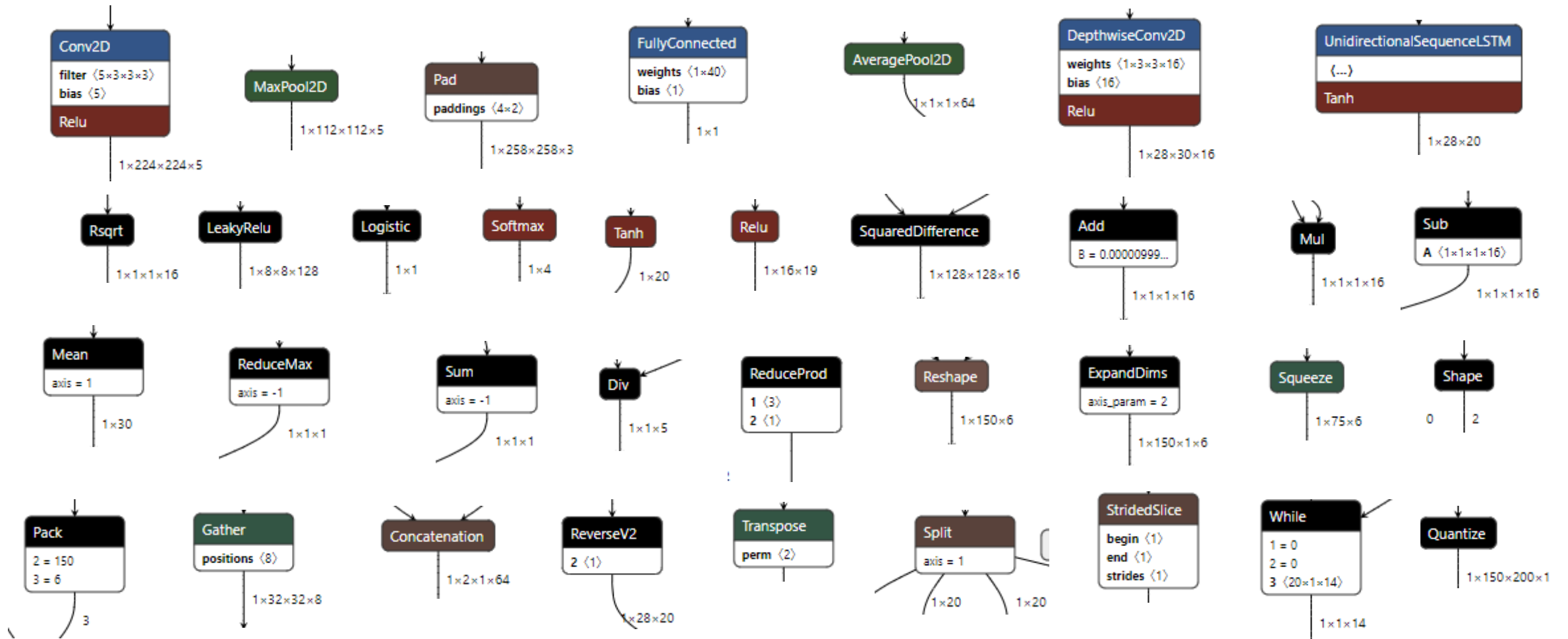
# Heterogeneity: Operators



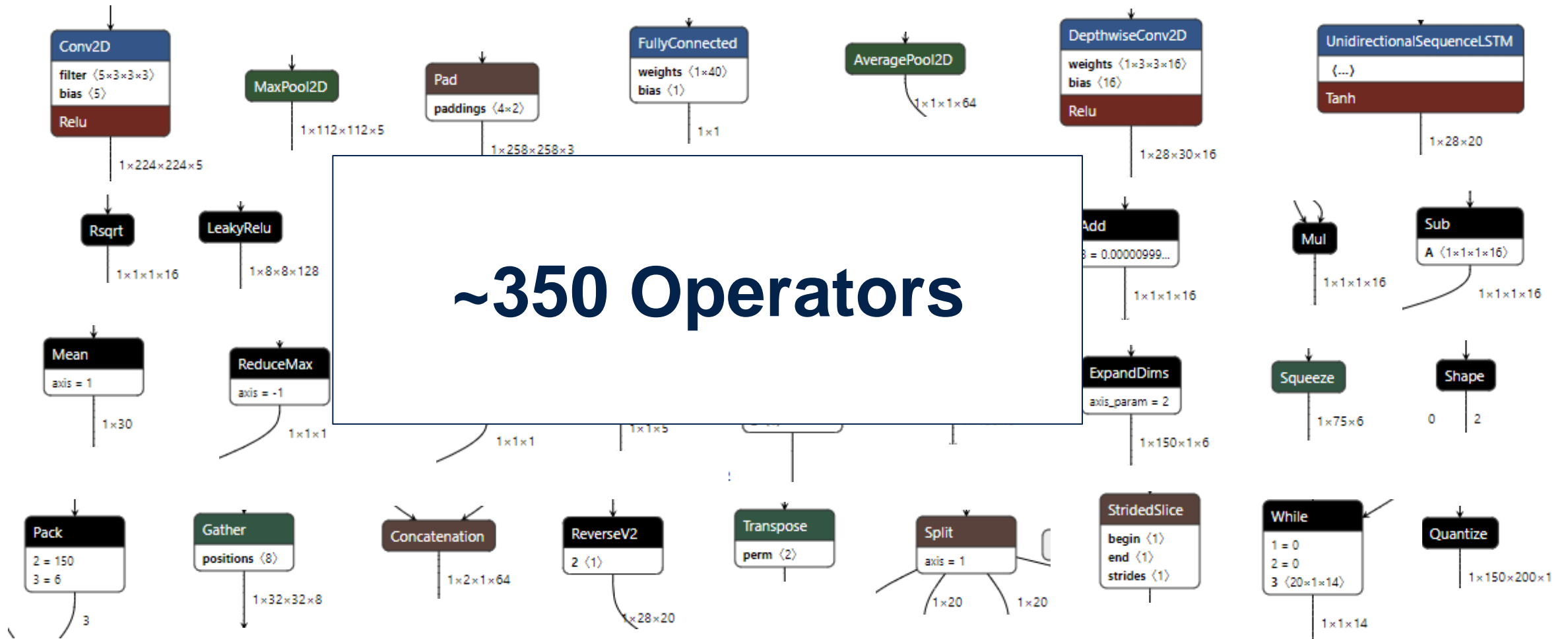
# Heterogeneity: Operators



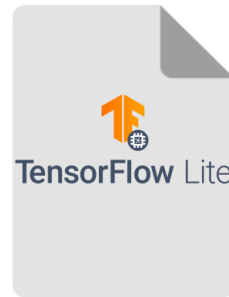
# Heterogeneity: Operators



# Heterogeneity: Operators

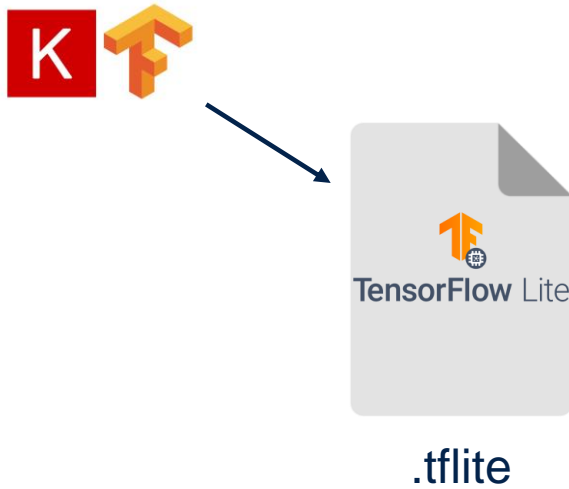


# Heterogeneity: DL Formats



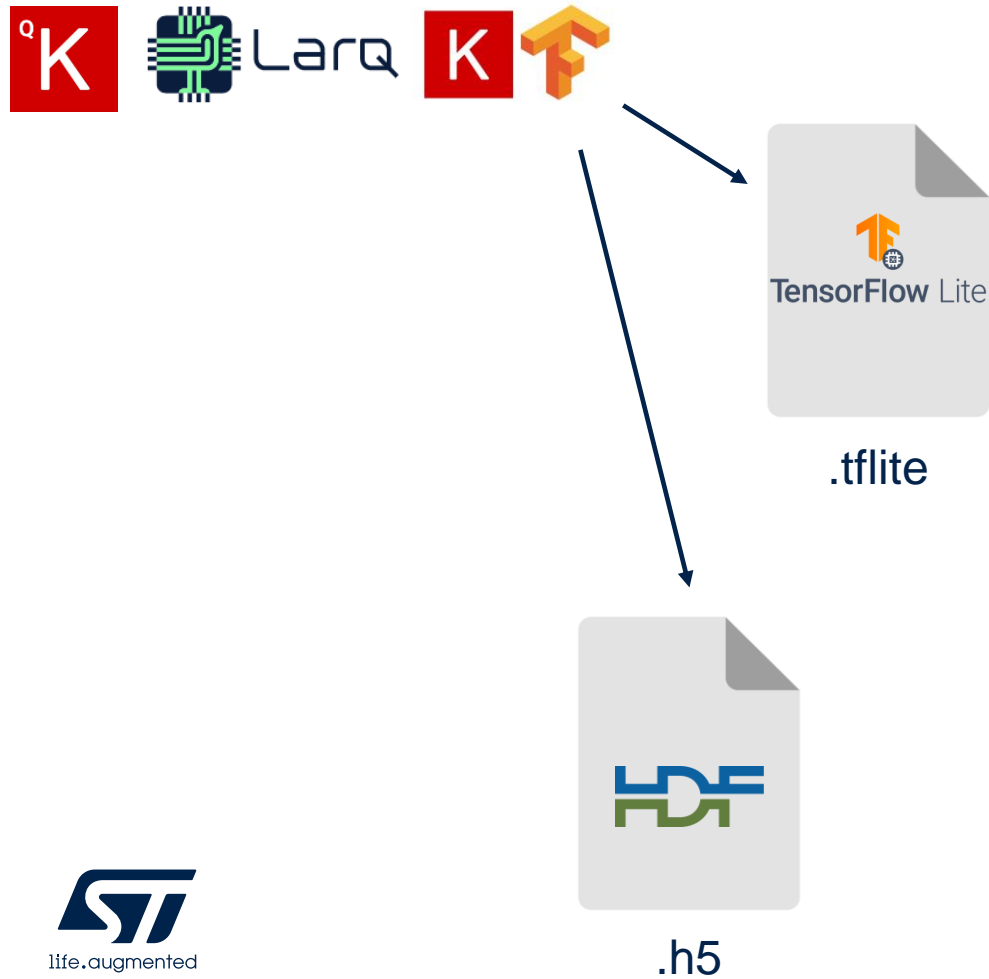
.tflite

# Heterogeneity: DL Formats

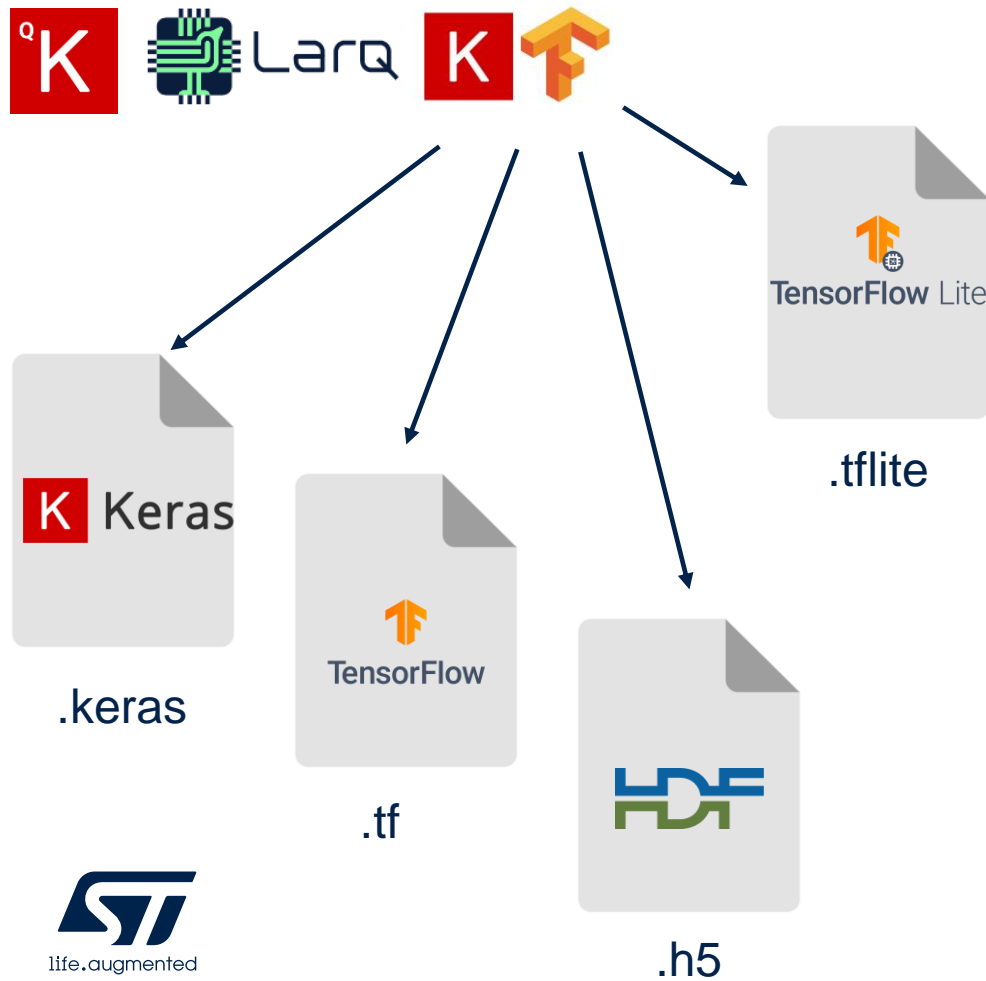




# Heterogeneity: DL Formats



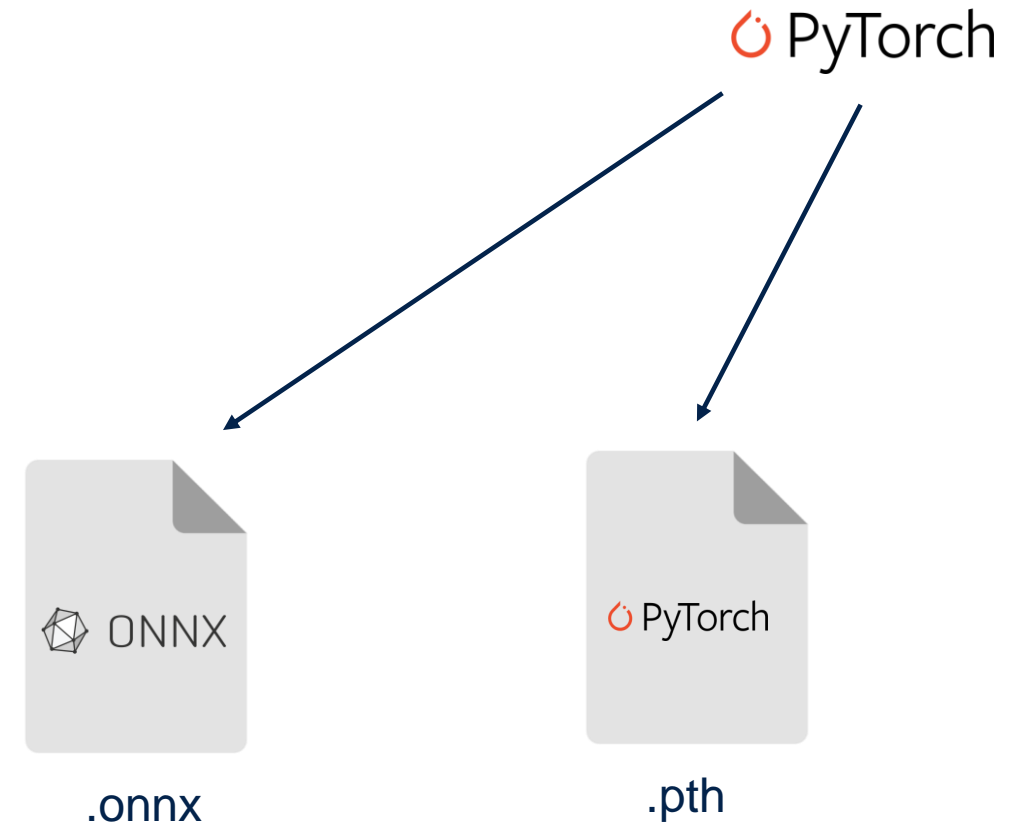
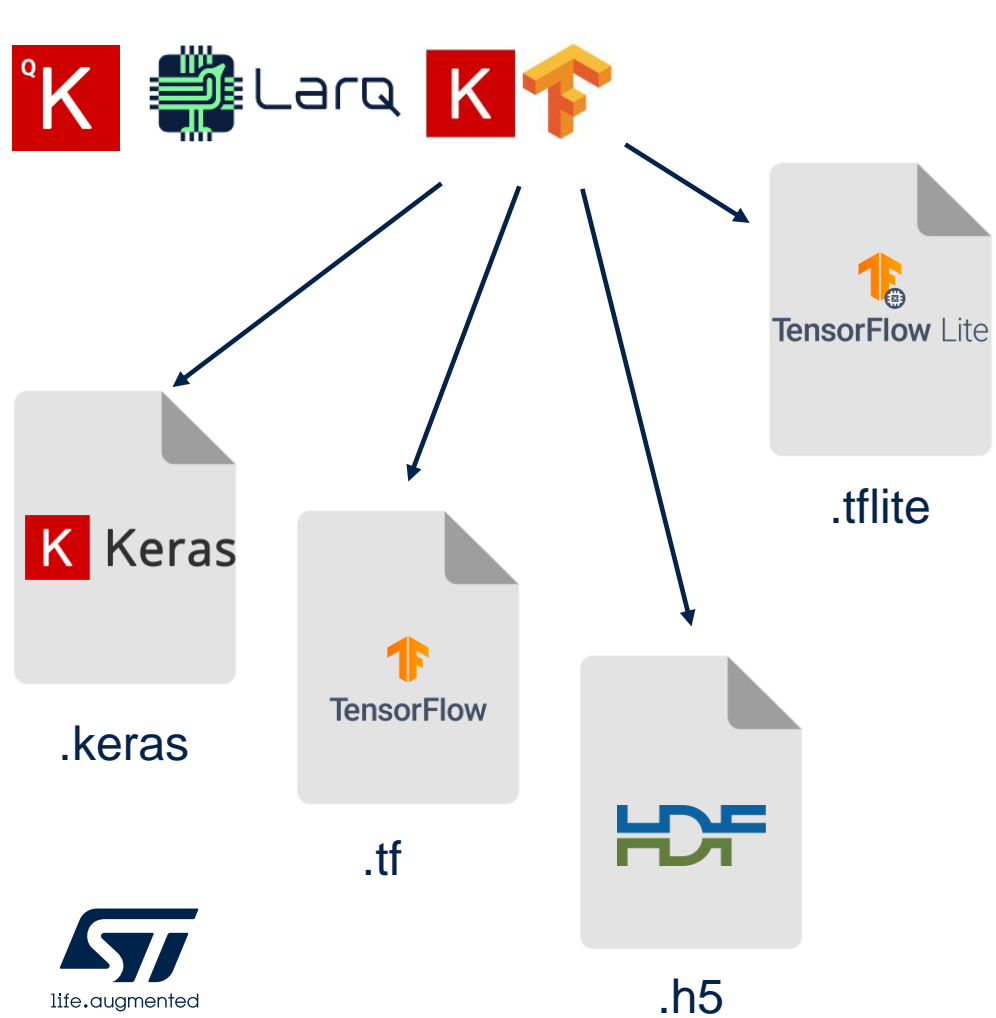
# Heterogeneity: DL Formats



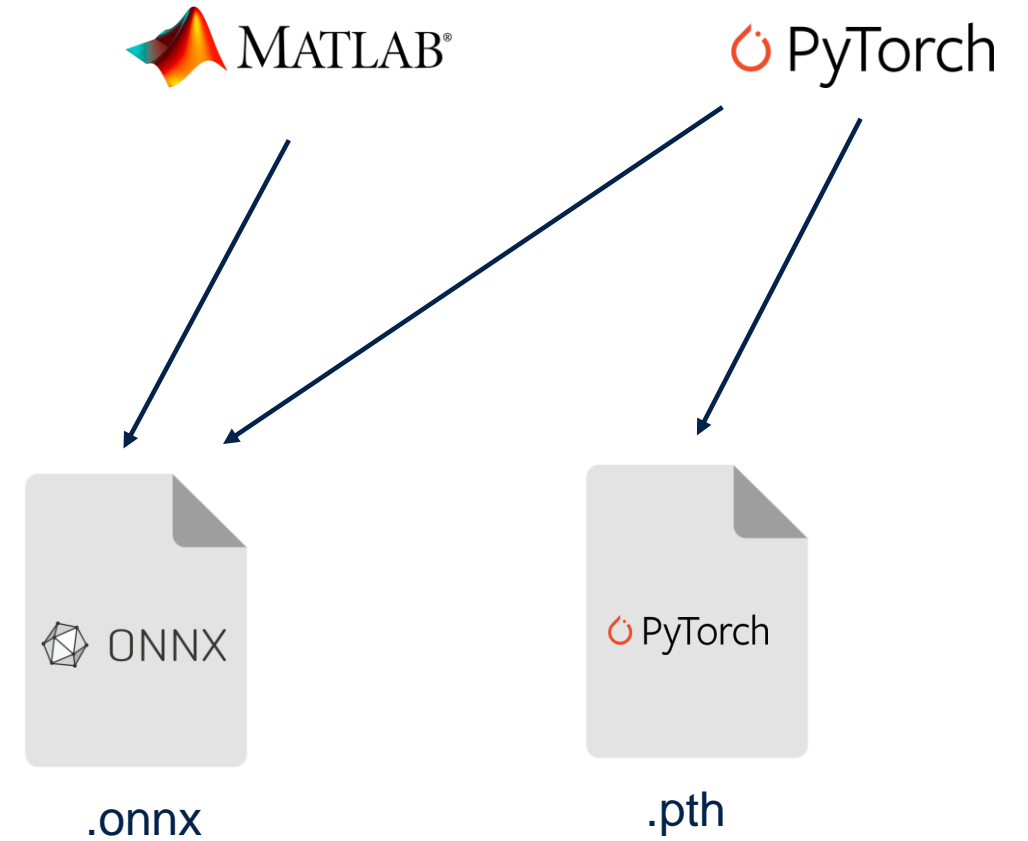
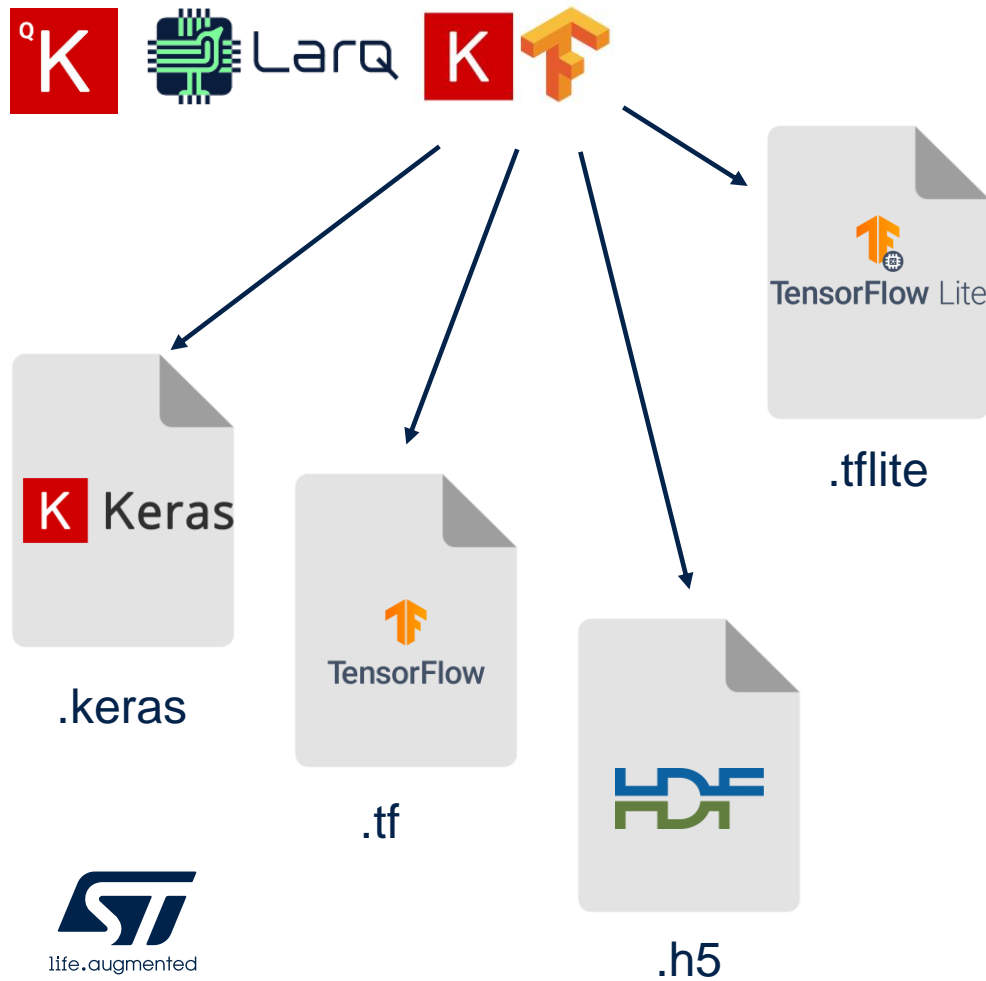
# Heterogeneity: DL Formats



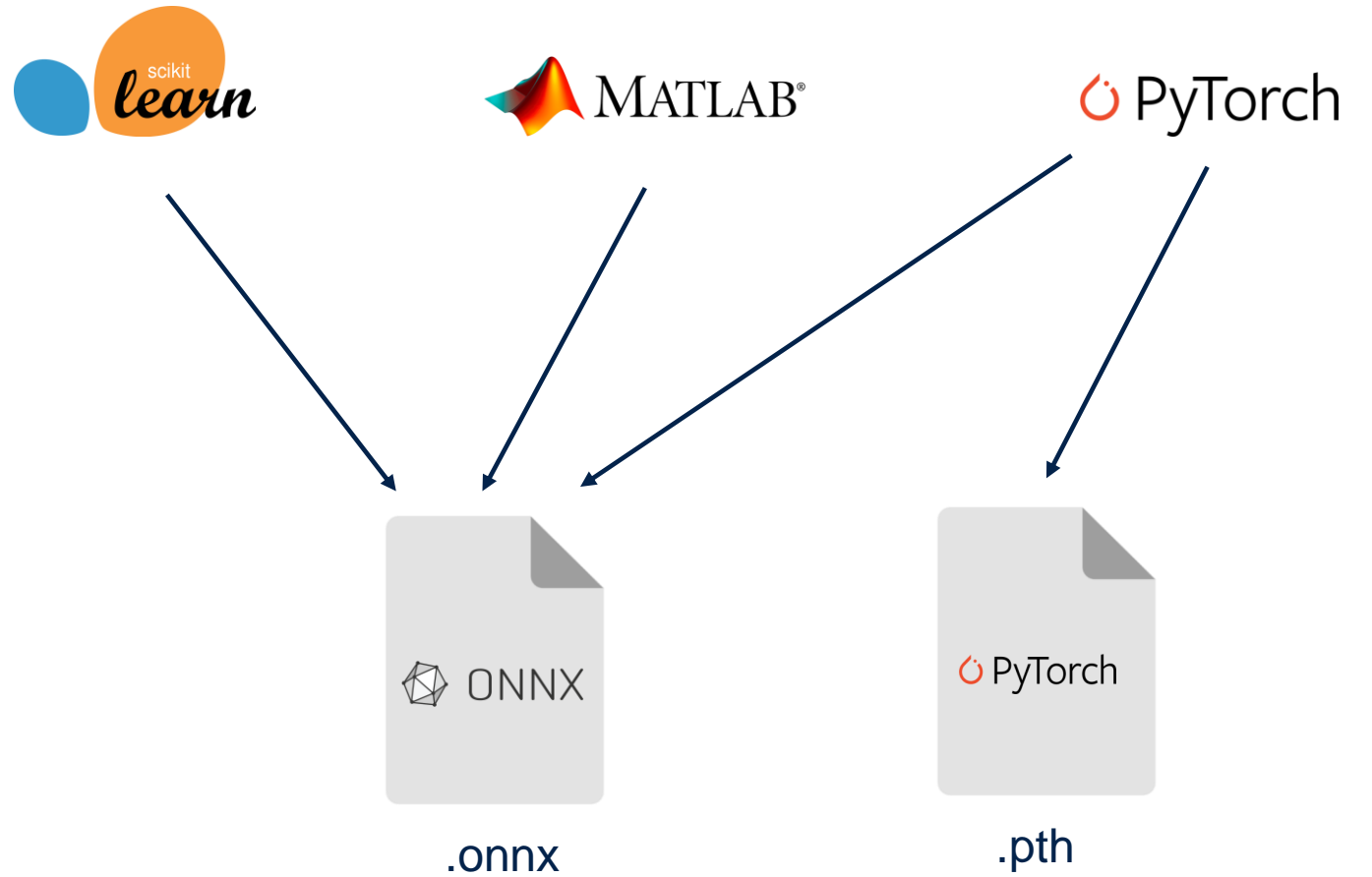
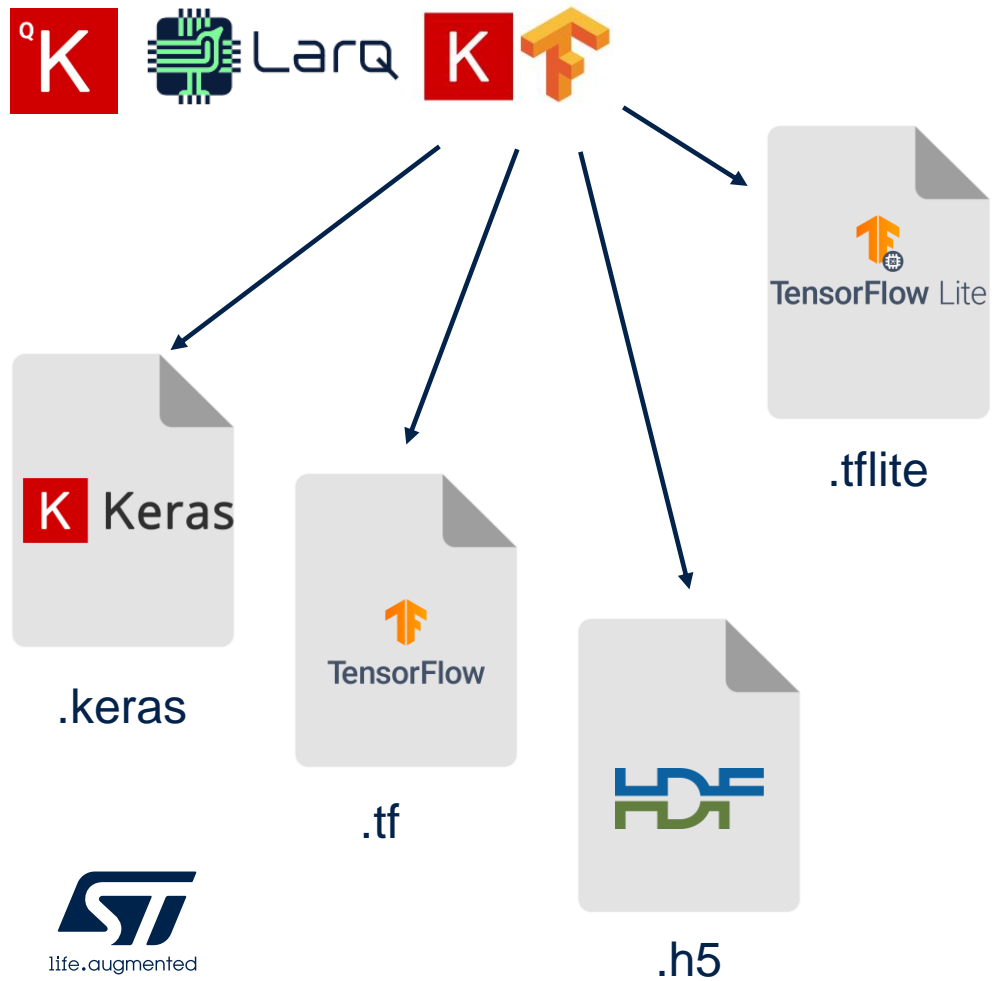
# Heterogeneity: DL Formats



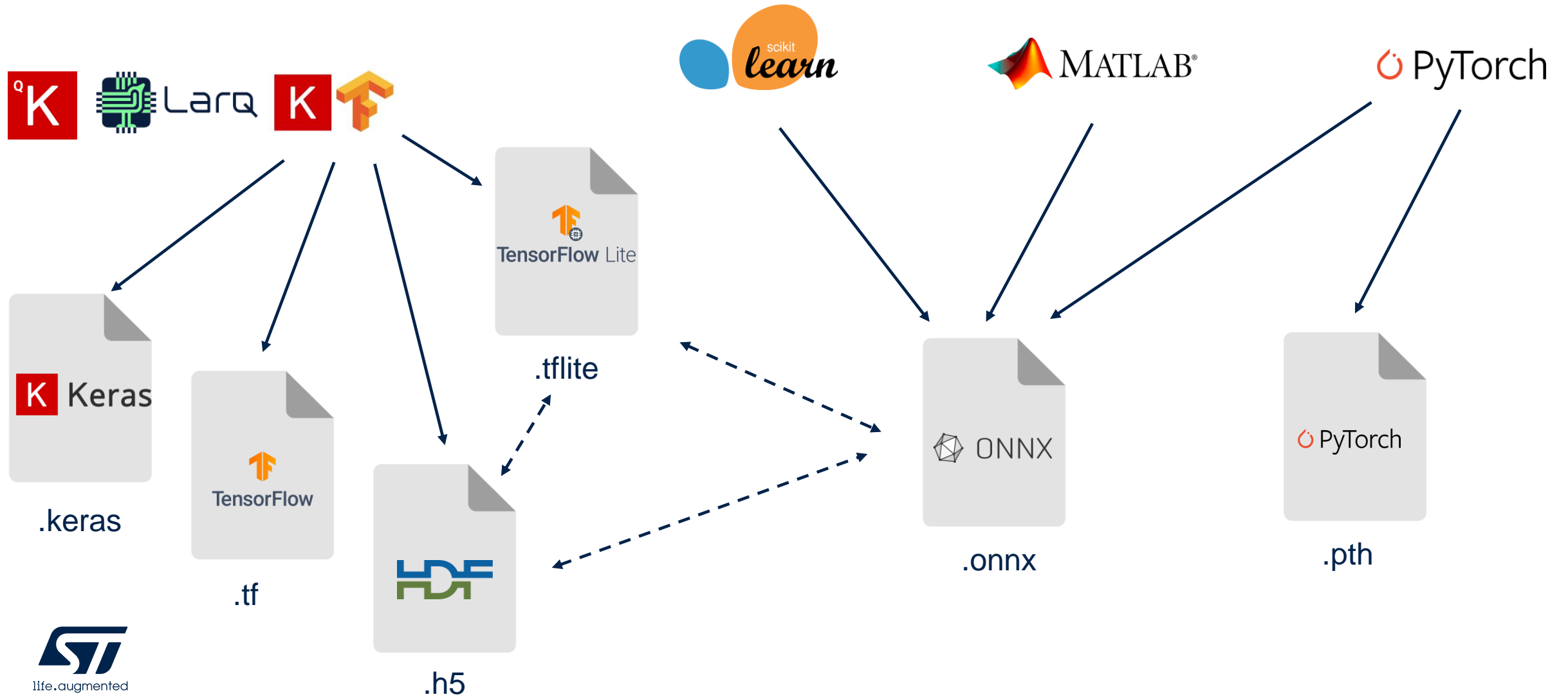
# Heterogeneity: DL Formats



# Heterogeneity: DL Formats



# Heterogeneity: DL Formats



# Heterogeneity: DL Differences

- Availability of operators: not all operators are available in all frameworks
- Attributes: different attributes, different meaning, different default values
- Data layout, i.e., channel first vs. channel last
- Quantization, i.e., available precisions to represent data



# Heterogeneity: DL Differences

- Availability of operators: not all operators are available in all frameworks
- Attributes: different attributes, different meaning, different default values
- Data layout, i.e., channel first vs. channel last
- Quantization, i.e., available precisions to represent data

TFLite  
MEAN

ONNX  
Mean

# Heterogeneity: DL Differences

- Availability of operators: not all operators are available in all frameworks
- Attributes: different attributes, different meaning, different default values
- Data layout, i.e., channel first vs. channel last
- Quantization, i.e., available precisions to represent data



# Heterogeneity: DL Differences

- Availability of operators: not all operators are available in all frameworks
- Attributes: different attributes, different meaning, different default values
- Data layout, i.e., channel first vs. channel last
- Quantization, i.e., available precisions to represent data

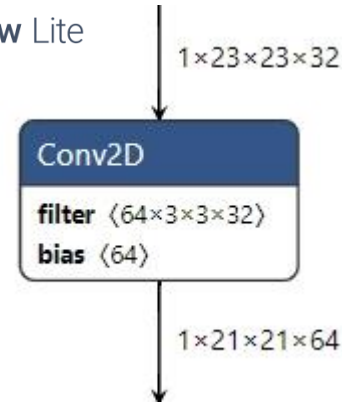
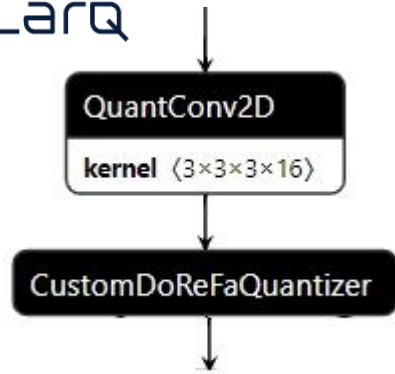


# Heterogeneity: DL Differences

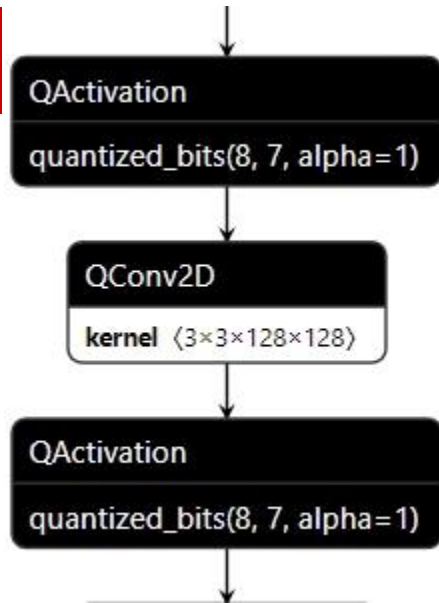
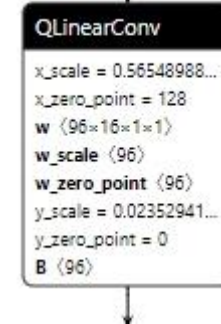
- Availability of operators: not all operators are available in all frameworks
- Attributes: different attributes, different meaning, different default values
- Data layout, i.e., channel first vs. channel last
- Quantization, i.e., available precisions to represent data



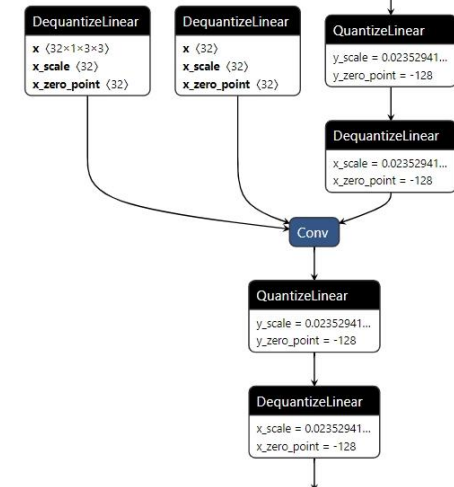
# Heterogeneity: DL Formats



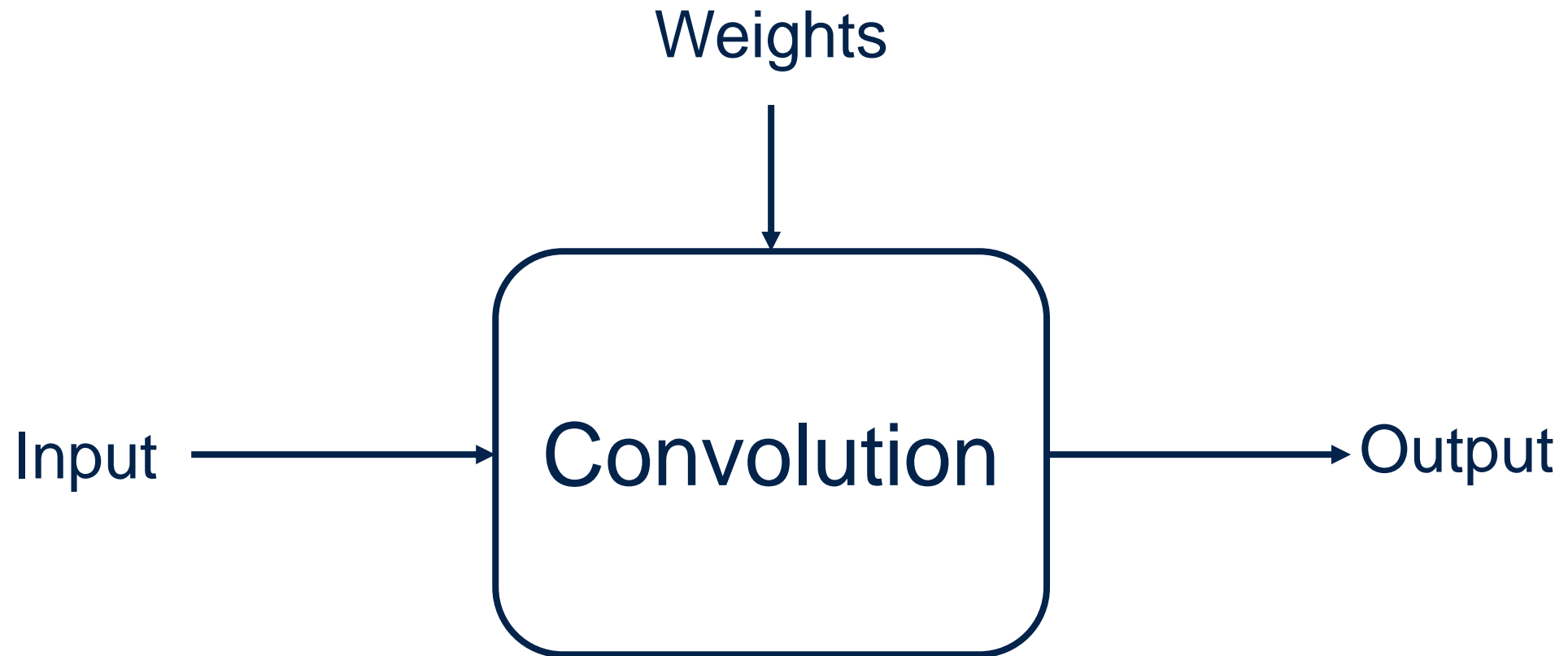
ONNX



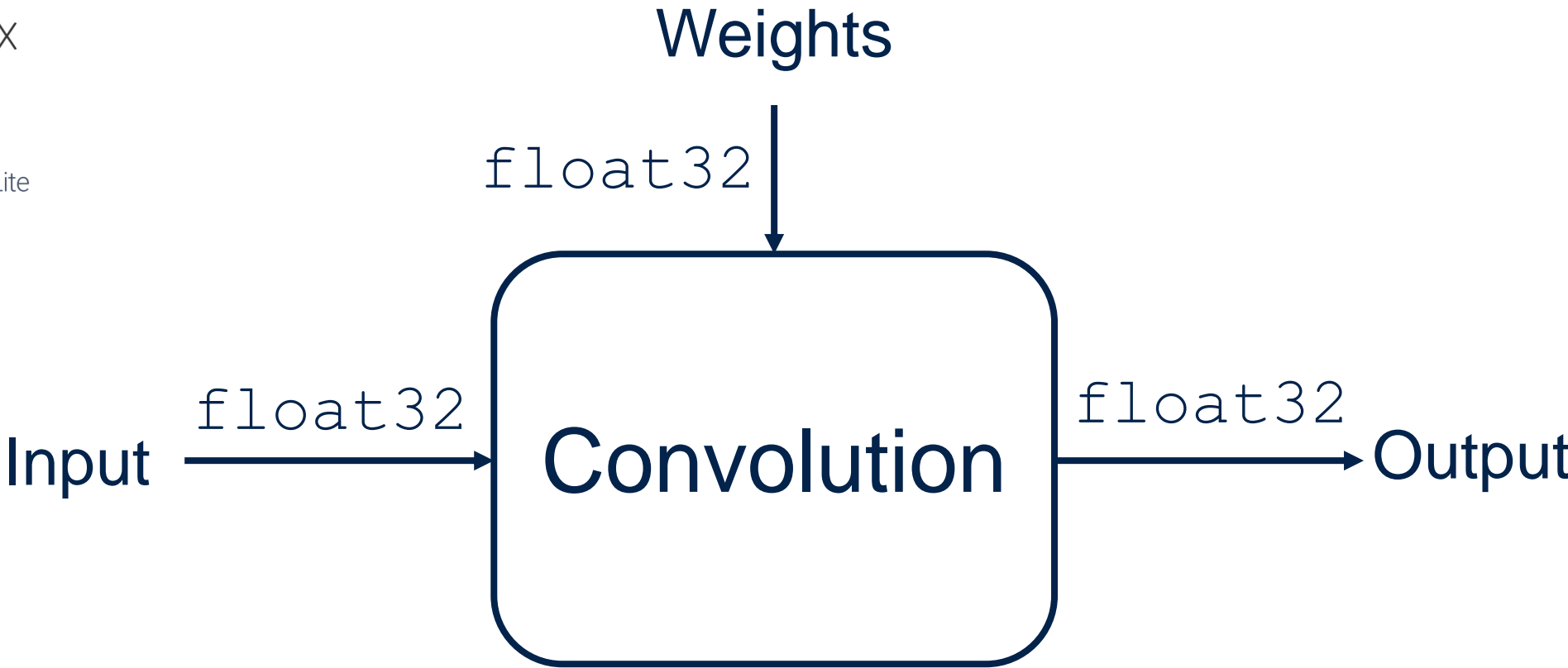
ONNX



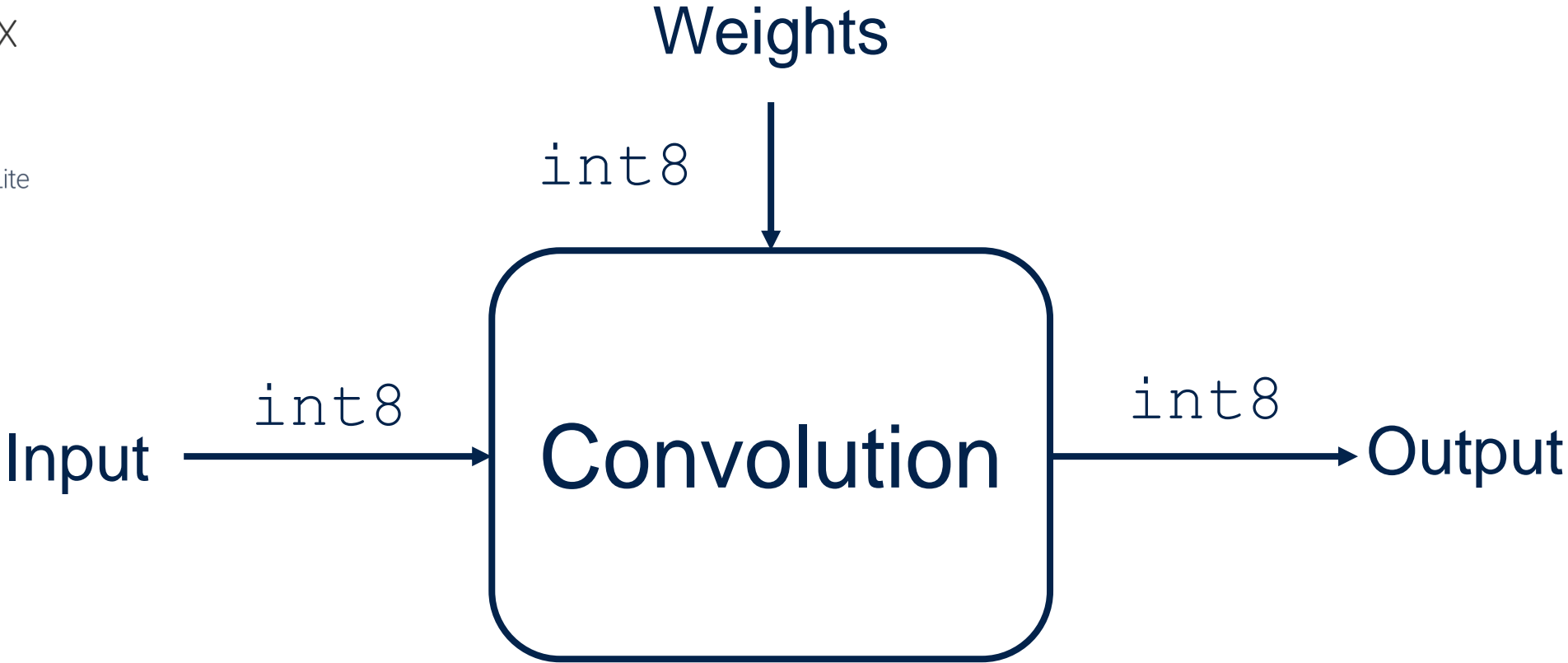
# Heterogeneity: Quantization



# Heterogeneity: Quantization

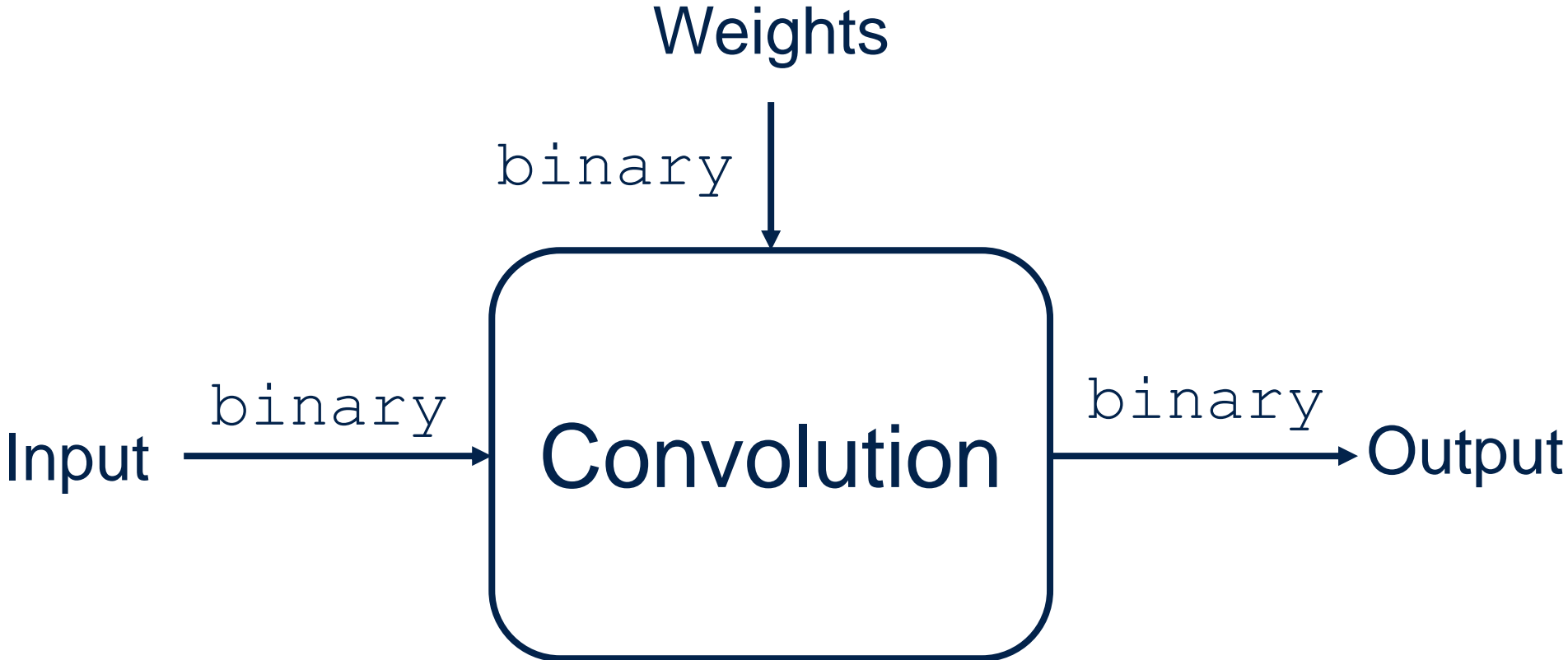


# Heterogeneity: Quantization

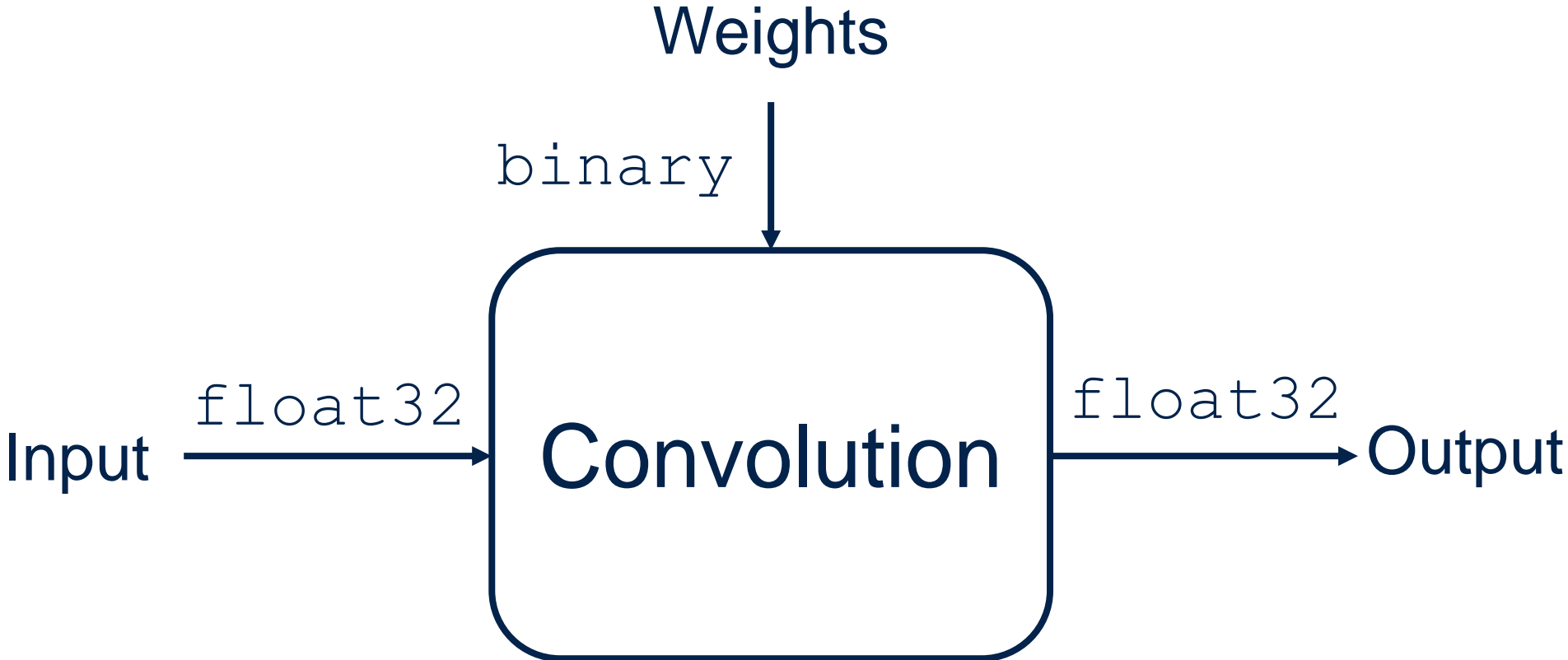




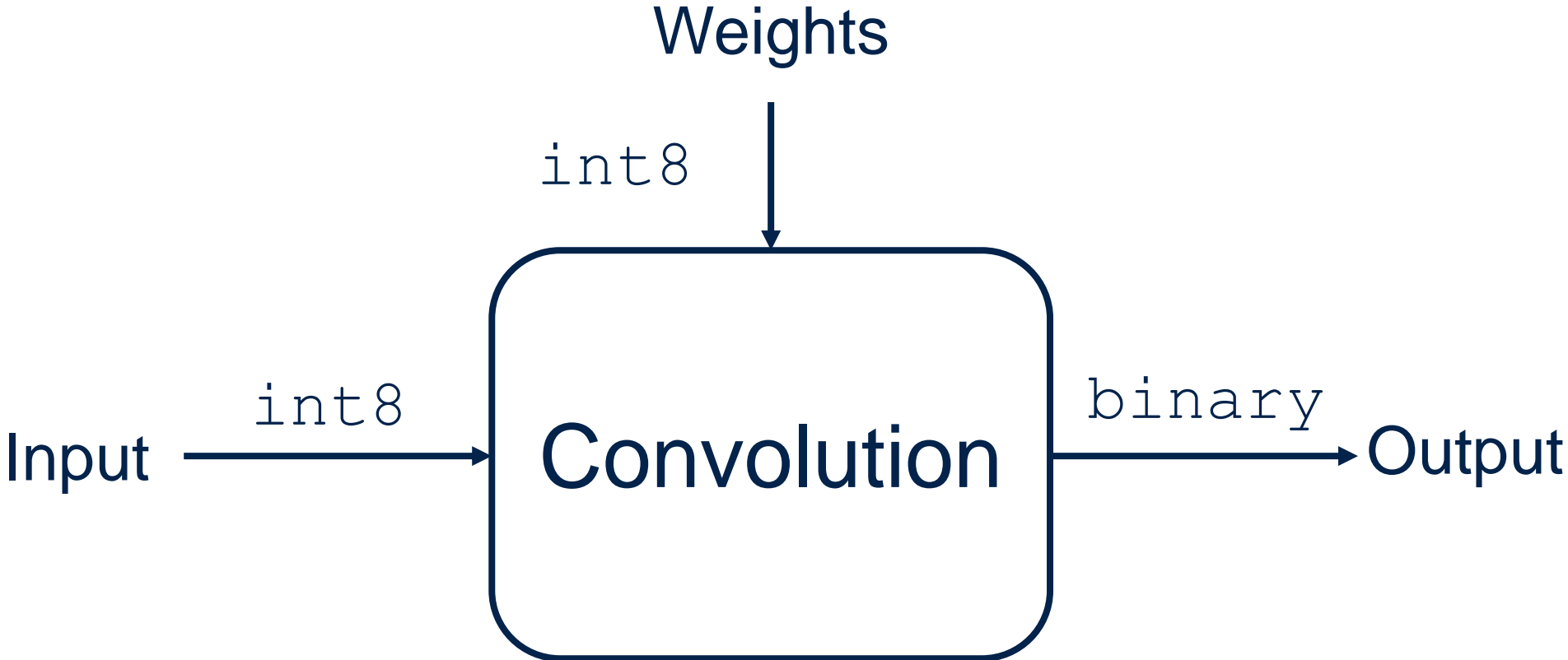
# Heterogeneity: Quantization



# Heterogeneity: Quantization



# Heterogeneity: Quantization



# Heterogeneity: Quantization

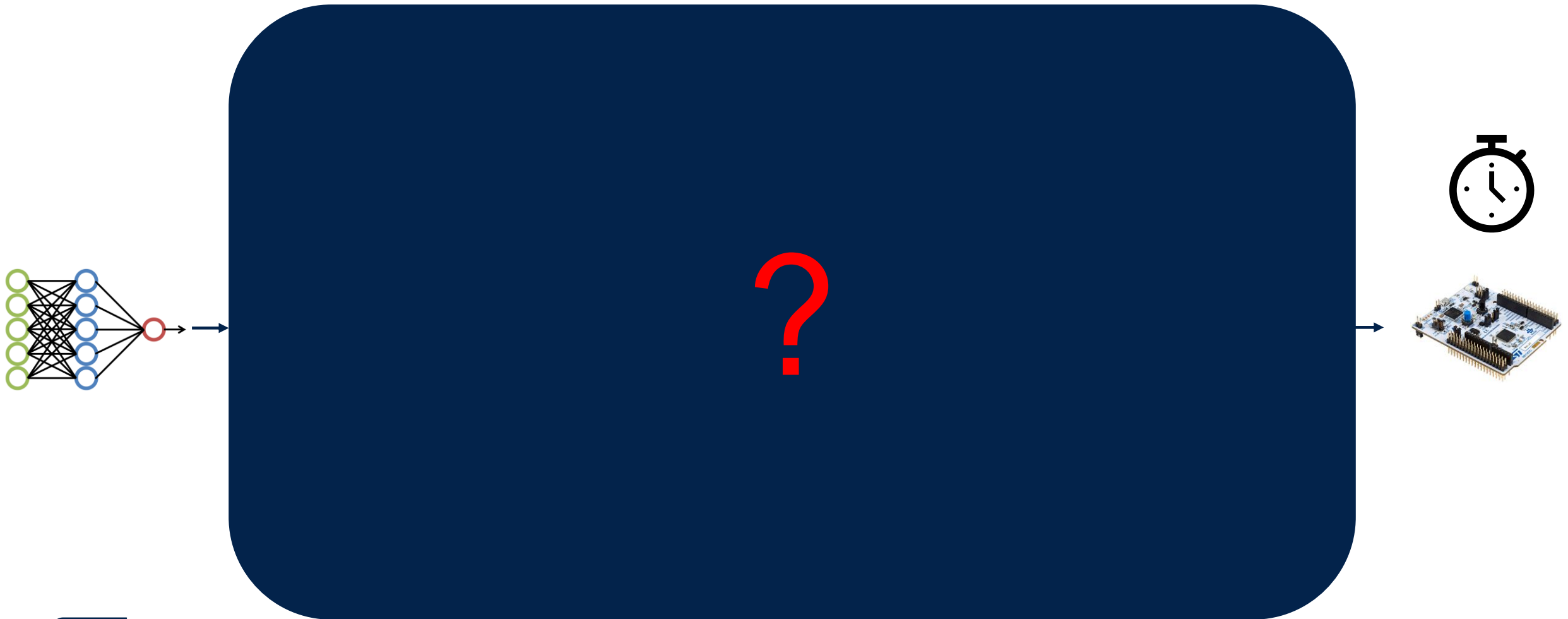


Input

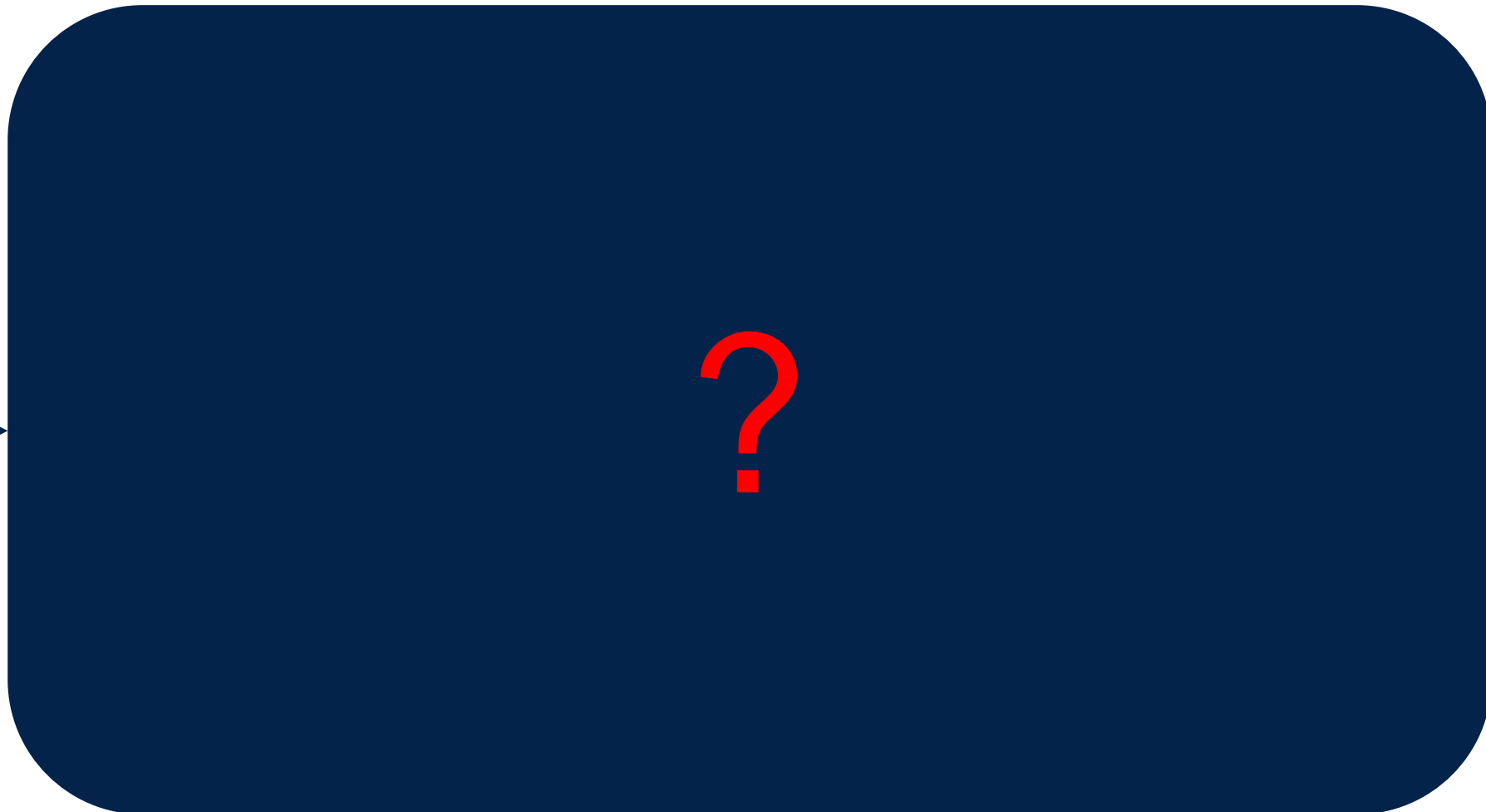
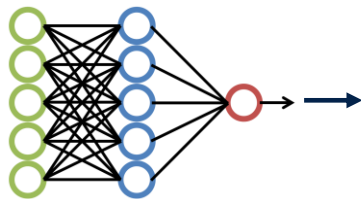
Limiting to binary, int8,  
float32 there are  
 $3^3=27$  combinations  
to be supported for a  
single layer

Output

# Removing Heterogeneity



# Removing Heterogeneity



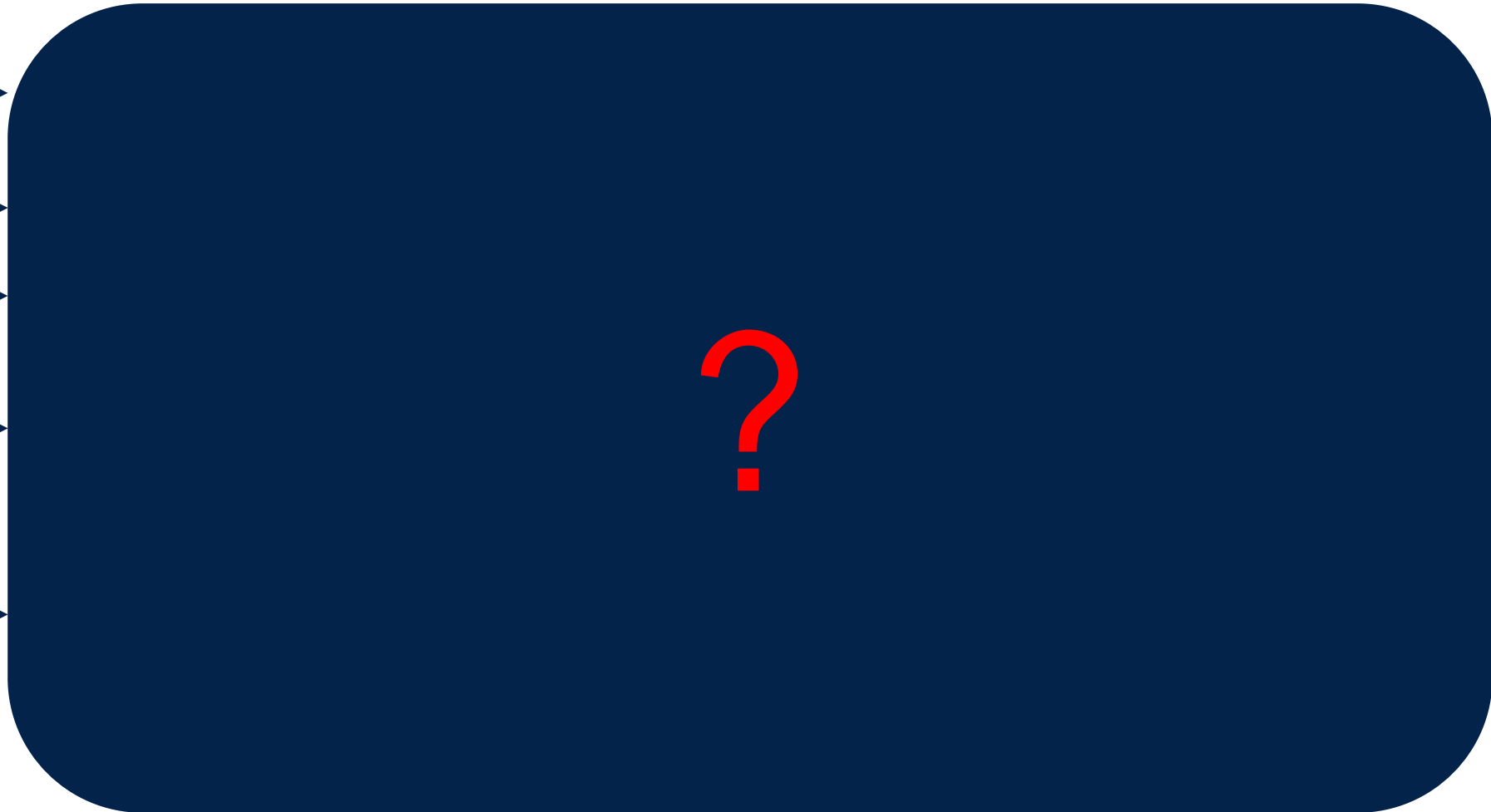
# Removing Heterogeneity

 Keras →

 Larq →

 ONNX →

 TensorFlow Lite →



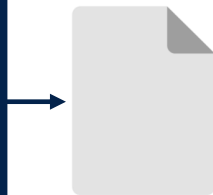
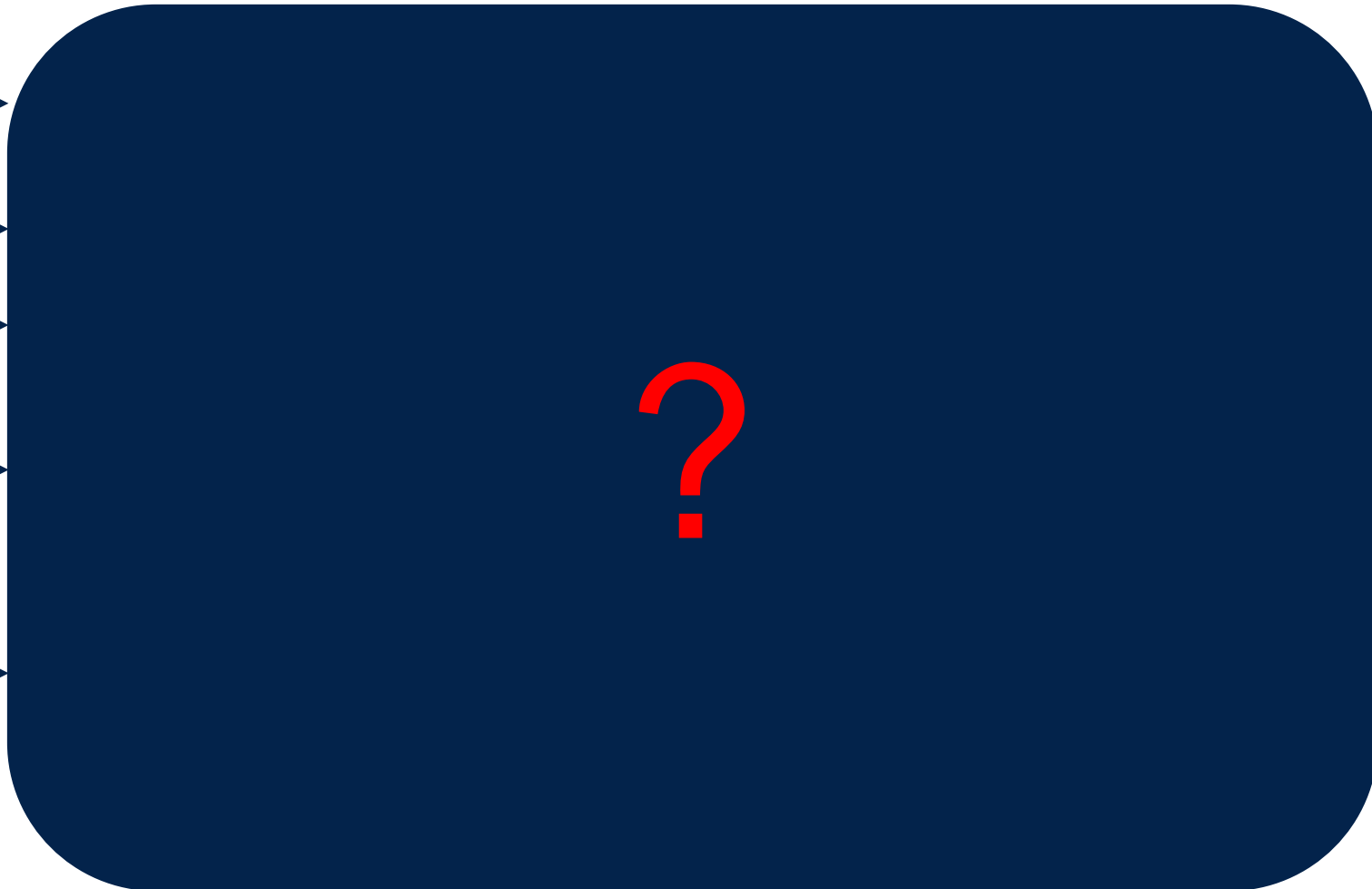
# Removing Heterogeneity

 Keras →

 Larq →

 ONNX →

 TensorFlow Lite →

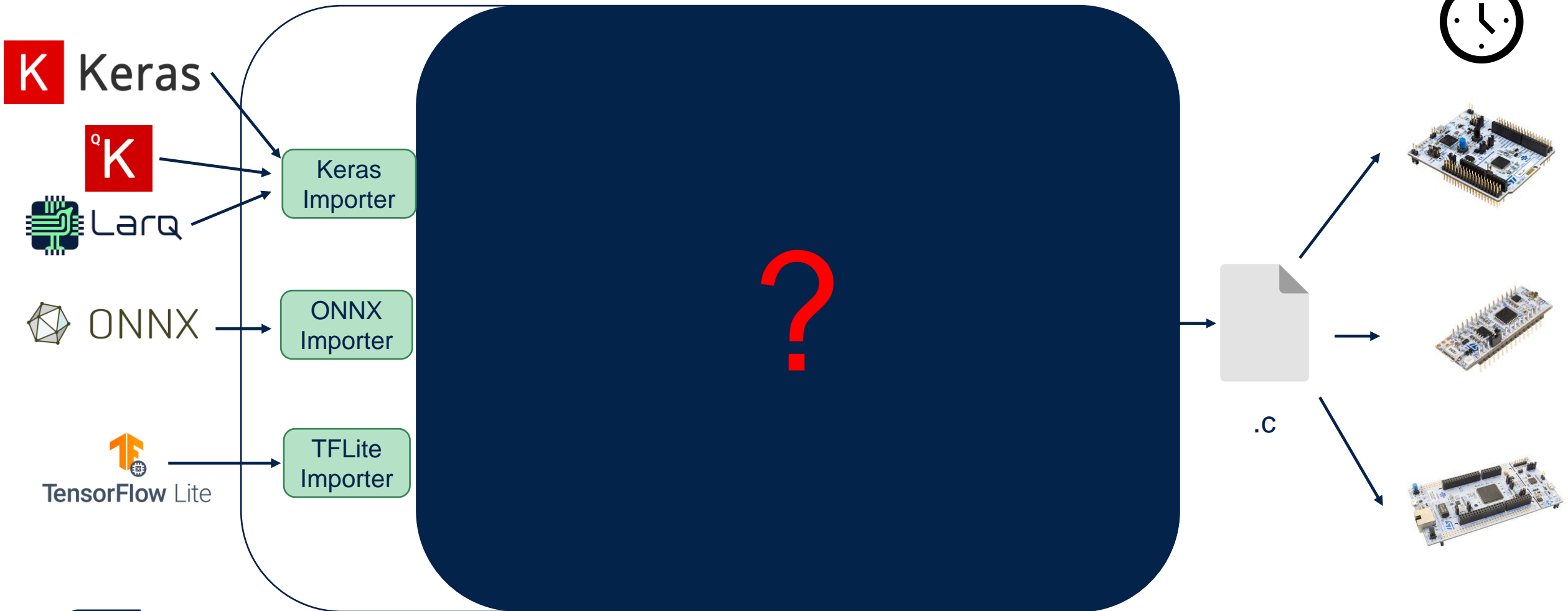


.C

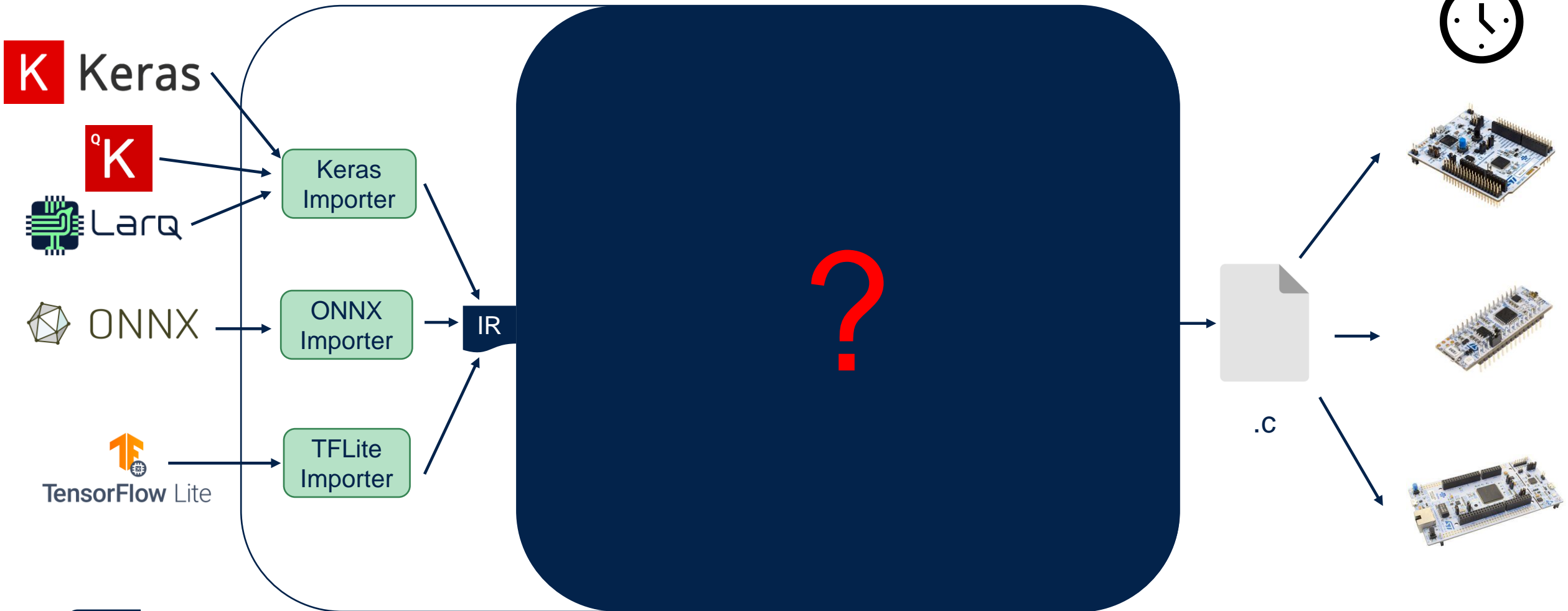




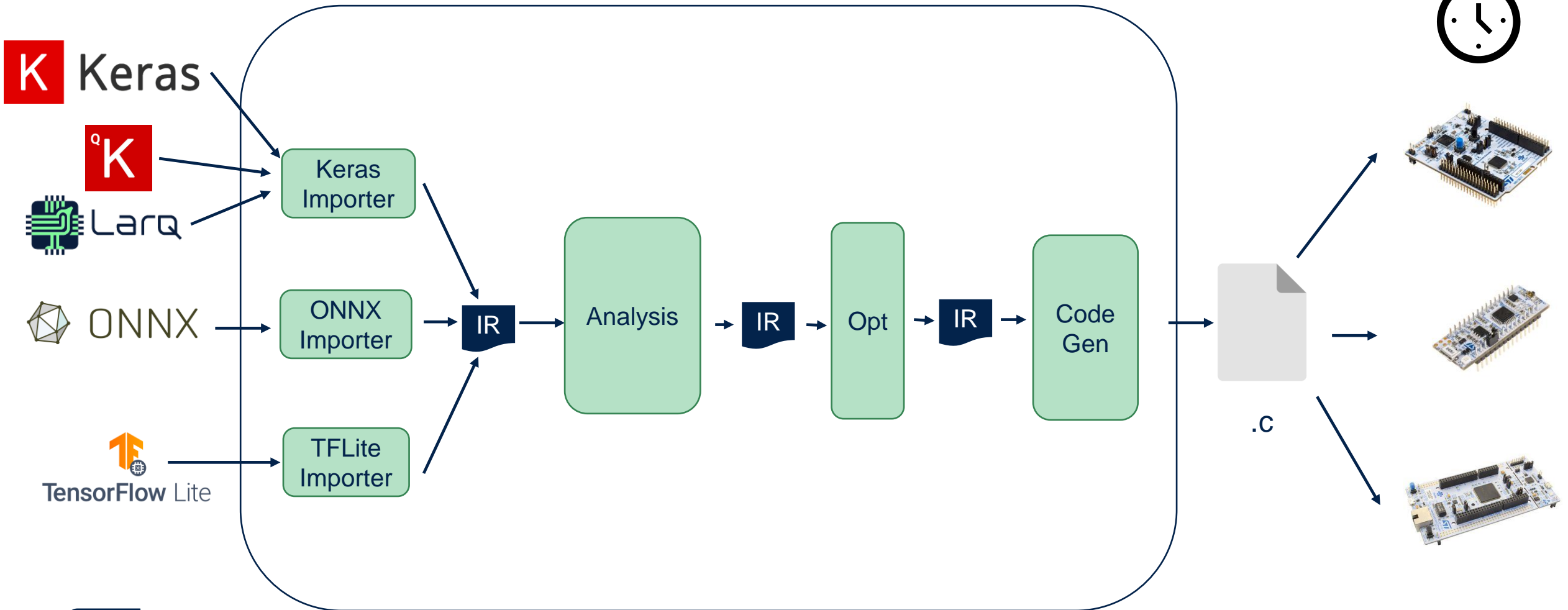
# Removing Heterogeneity



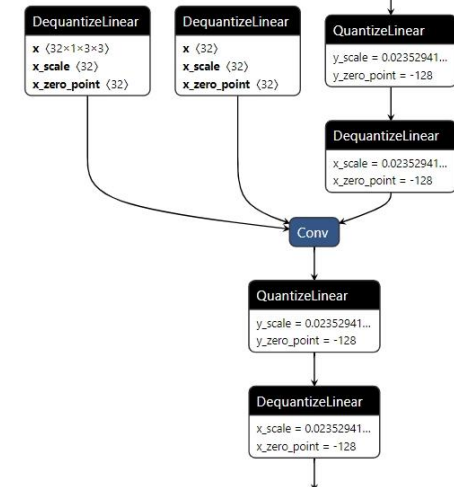
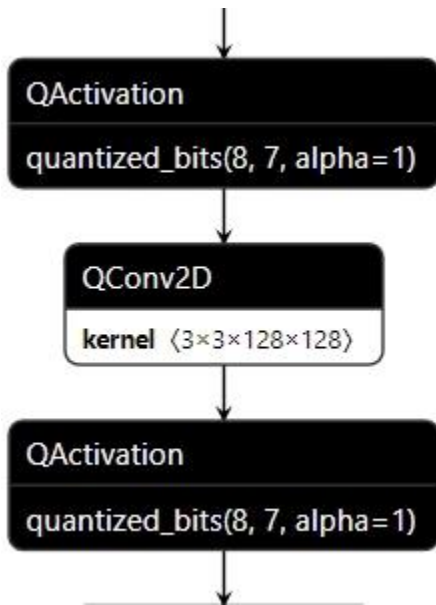
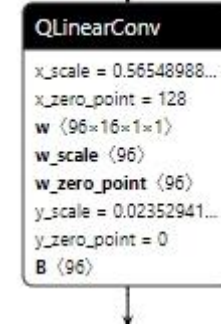
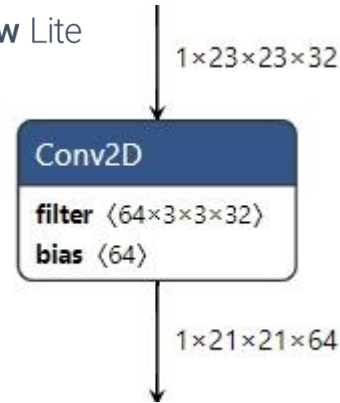
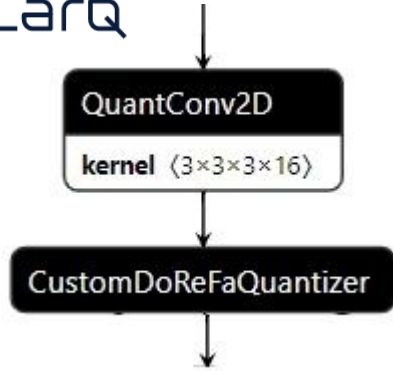
# Removing Heterogeneity



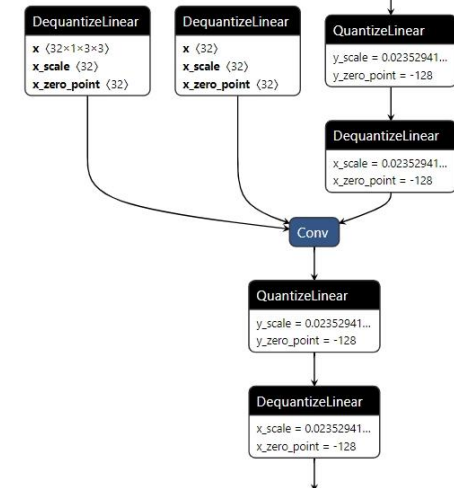
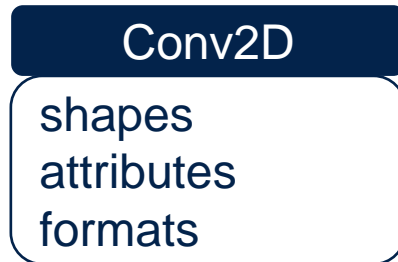
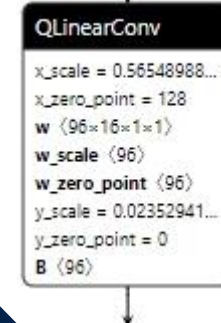
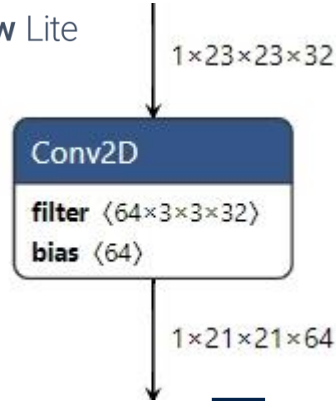
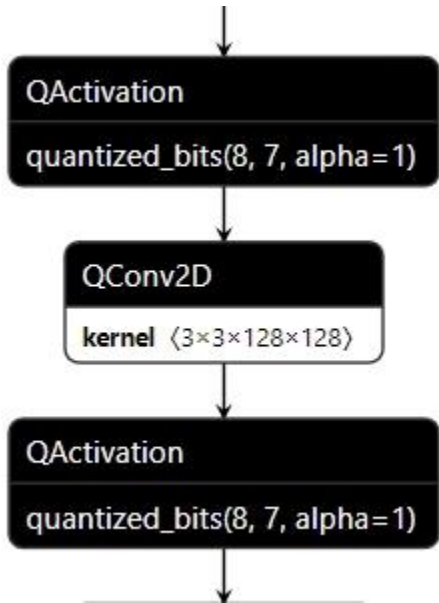
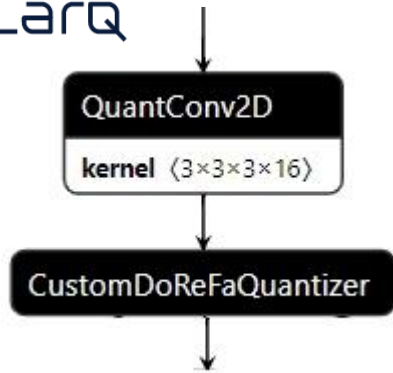
# Removing Heterogeneity



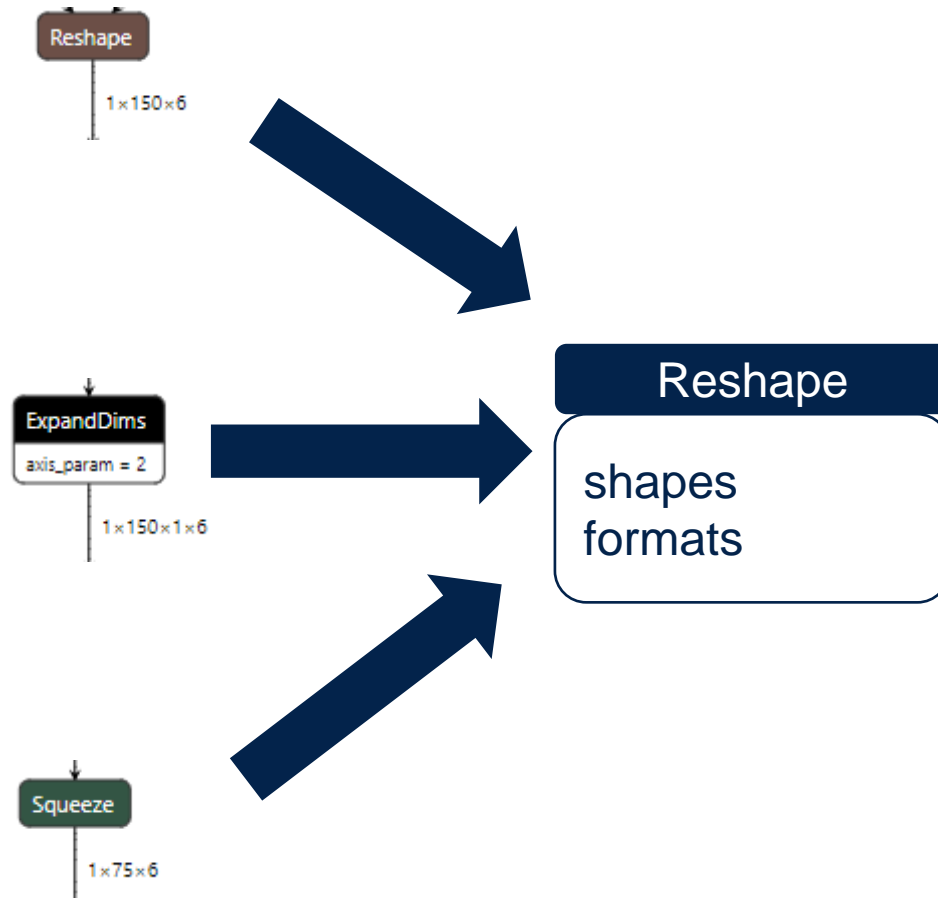
# Common Intermediate Representation



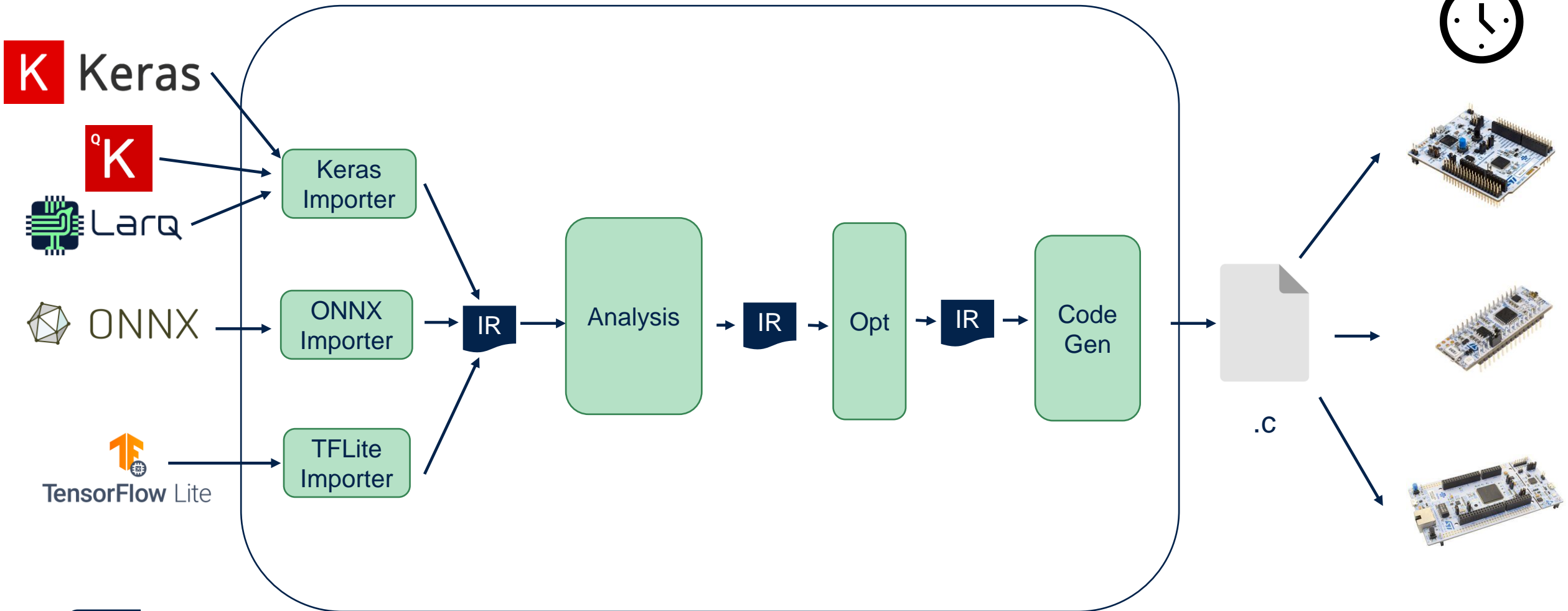
# Common Intermediate Representation



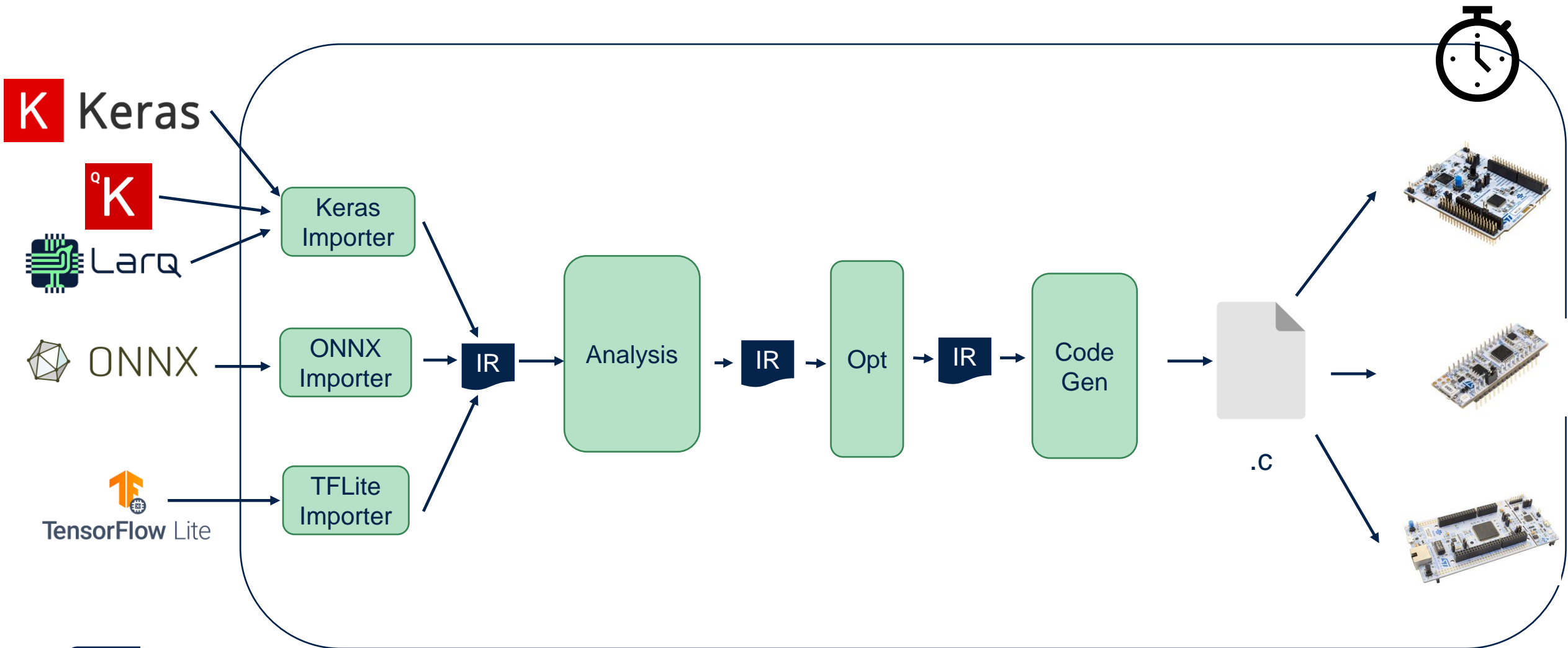
# Common Intermediate Representation



# STM32Cube.AI Developer Cloud

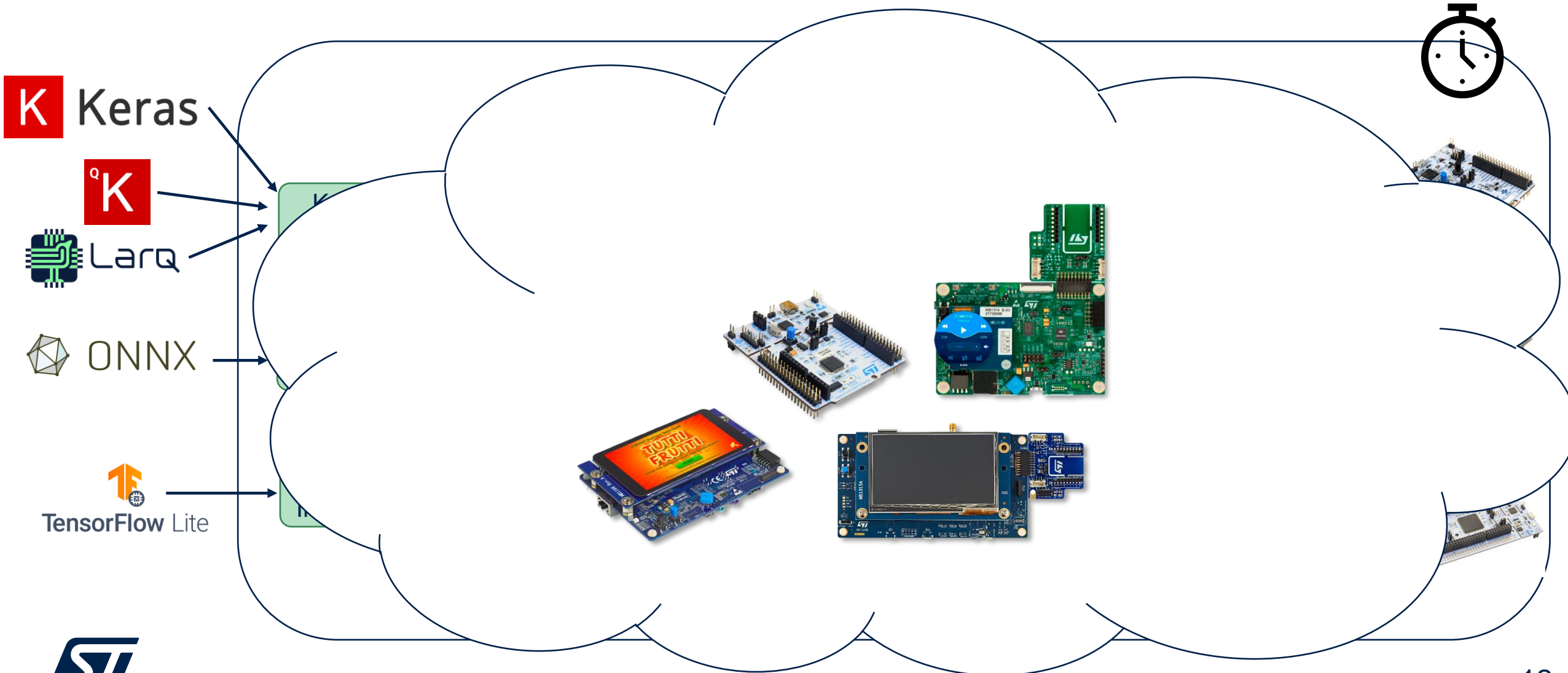


# STM32Cube.AI Developer Cloud





# STM32Cube.AI Developer Cloud



# STM32Cube.AI Developer Cloud

The screenshot displays the STM32Cube.AI Developer Cloud interface. At the top, there is a navigation bar with the STM logo and a 'Sign in' link. The main heading is 'OPTIMIZE YOUR TRAINED NEURAL NETWORK', followed by a sub-heading: 'Optimize and measure performance of your Artificial Intelligence library for STM32 MCUs'. Below this, a yellow 'START NOW' button is visible. The workflow is divided into three main stages:

- Load your trained Neural Network model**: This stage offers the option to 'or pick one from STM32 model zoo (AI models library)'. It features logos for 'learn', 'Keras', 'PyTorch', 'TensorFlow', 'MATLAB', and 'ONNX'.
- Optimize and benchmark your NN model**: This stage is powered by 'STM32Cube.AI' and 'STM32Cube.AI Developer Cloud'. It includes an 'Online platform', 'REST API', and 'Benchmarking tool'. The core process is a 'Neural Network model converter and benchmarking for STM32 MCU', which utilizes a 'Graph optimizer', 'Quantizer', and 'Memory optimizer'.
- Generate optimized code for STM32**: This final stage produces 'Optimized model code in multiple format according to your need'. The outputs include 'STM32 C-code', 'STM32 Firmware', 'STM32Cube MX IOC file', and 'STM32CubeIDE Project'.

At the bottom of the interface, there are links for 'STM32 Model Zoo' and 'More AI Solutions for STM32'.



# Our technology starts with You



Find out more at [www.st.com](http://www.st.com)

© STMicroelectronics - All rights reserved.

ST logo is a trademark or a registered trademark of STMicroelectronics International NV or its affiliates in the EU and/or other countries.

For additional information about ST trademarks, please refer to [www.st.com/trademarks](http://www.st.com/trademarks).

All other product or service names are the property of their respective owners.



life.augmented