Keras2c

A library for converting Keras neural networks to real-time compatible C

Rory Conlin[1], Keith Erickson[2], Joseph Abbate[3], Egemen Kolemen[1,2]

[1] Department of Mechanical and Aerospace Engineering, Princeton University
[2] Princeton Plasma Physics Laboratory
[3] Department of Astrophysical Sciences at Princeton University

Outline

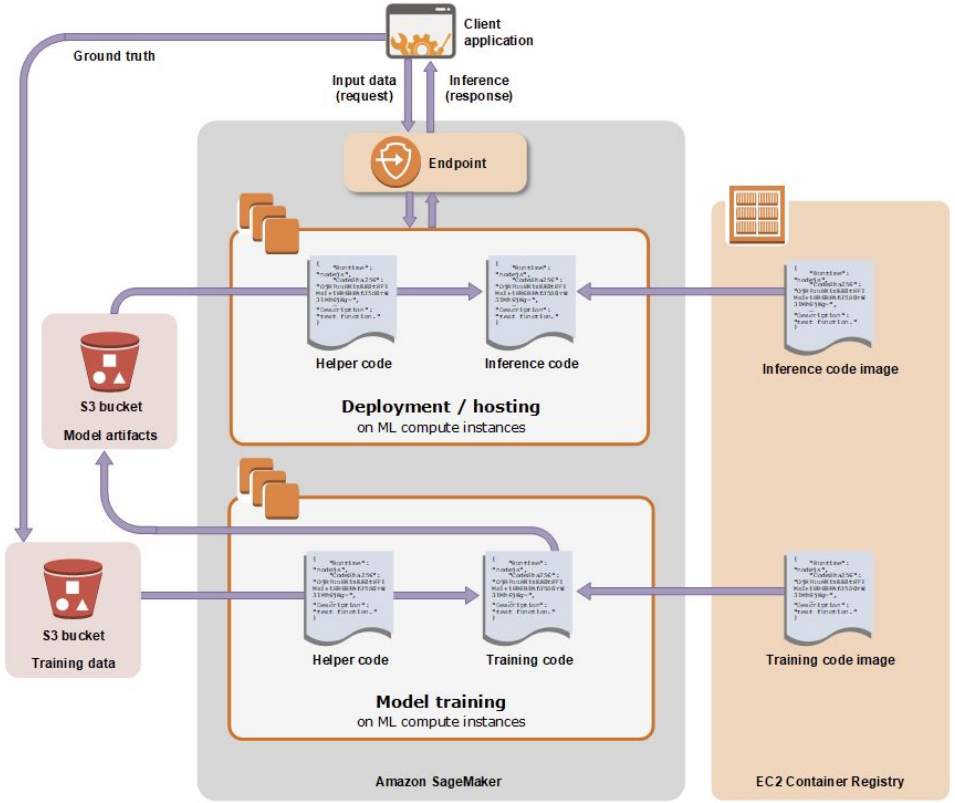
- Use of machine learning (ML) in real-time applications
- Need for real-time friendly way to deploy ML models
- Keras2c basics
- Model parsing and supported features
- C backend
- Automated testing
- Benchmarks
- Real-time application: Plasma Control System on DIII-D Tokamak

Real-time applications of machine learning

- Accelerating first-principles based analysis with data driven model:
  - Boyer et al 2019 *Real-time capable modeling of neutral beam injection on NSTX-U using neural networks*
  - Van De Plassche et al 2020 *Fast modeling of turbulent transport in fusion plasmas using neural networks*
  - Felici et al 2018 *Real-time-capable prediction of temperature and density profiles in a tokamak using RAPTOR and a first-principle-based transport model*
- Purely data driven approach where first-principles are lacking:
  - Kates-harbeck et al 2019 *Predicting disruptive instabilities in controlled fusion plasmas through deep learning*
  - Fu et al 2020 *Machine learning control for disruption and tearing mode avoidance*
- Combined first-principles & data to learn control:
  - Chung et al 2020 *Offline Contextual Bayesian Optimization for Nuclear Fusion*

Deploying machine learning models

- Current method for deploying ML models based around mobile + web applications
  - Amazon SageMaker
  - Oracle GraphPipe
  - Open Neural Network Exchange
- Generally involve communicating with process running on remote server
  - Large latency
  - Non-deterministic behavior
  - **Not safe for real-time applications**



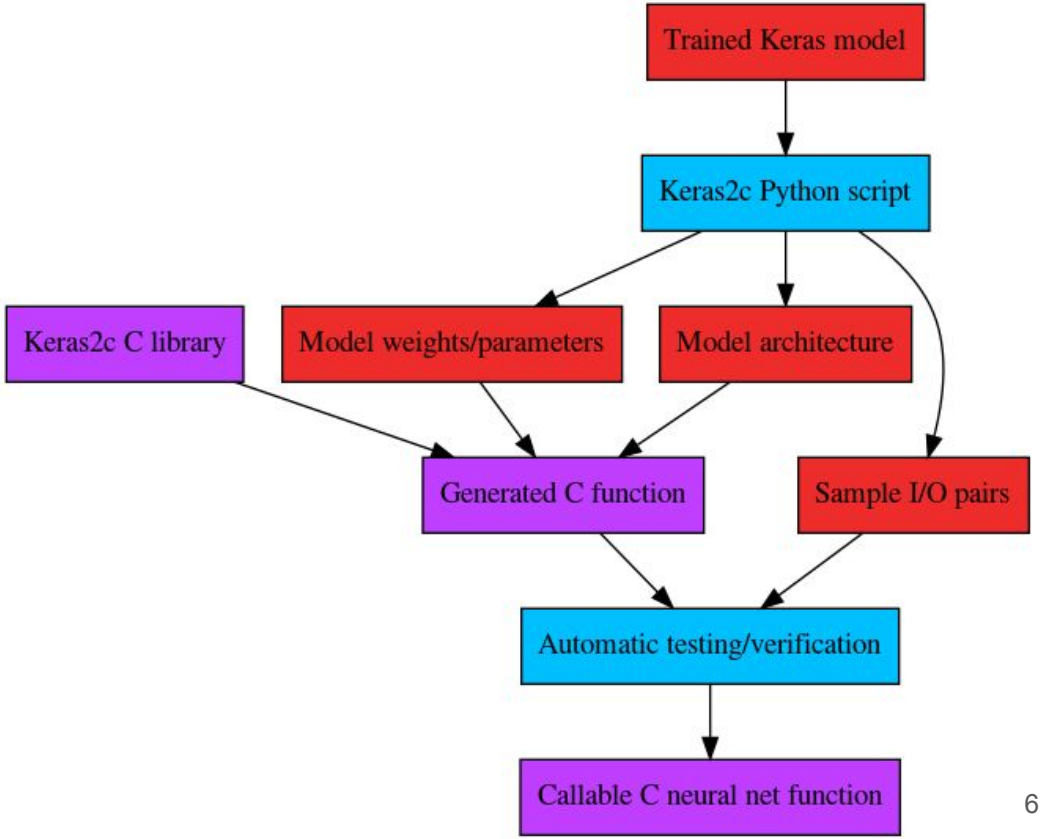
Deploying machine learning models

- Other options designed for mobile & embedded systems:
    - Tensorflow Lite
    - PyTorch TorchScript
  - Limited in what model types they support
  - Often still requires calls to secondary processes
    - Non-deterministic behavior
- TensorFlow C/C++ API
  - Extremely labor intensive to recode entire model by hand
  - Requires large external libraries (~millions SLOC)
  - Generally not safe for real-time

Keras2c: fully automated conversion / code generation

Script/Library for converting Keras neural nets to C functions

- Designed for simplicity and real time applications
- Core functionality only ~1500 lines
- Generates self-contained C function, no external dependencies
- Supports full range of operations & architectures
- Fully automated conversion & testing



Why  Keras ?

- High level API built on TensorFlow
  - “**Deep learning for humans”**
  - User friendly, easy to learn
  - Fast development and training
  - Full feature set for complicated models
  - Most used framework among winning teams on ML competition site Kaggle

Keras2c: neural net is just another function

- Keras API built around “layer” object
  - Each layer transforms input data via standard mathematical functions
    - Dot products, convolutions, sigmoid activation etc
  - Model is built by stacking layers together
  - Not always sequential: can also contain branching & merging layers
- Keras2c follows similar approach
  - Each Keras layer implemented as C function
  - “Model” is just a wrapper function that calls layer functions in the right order with the correct inputs

```
nn_predictor(k2c_tensor * inputs, k2c_tensor * outputs);
```

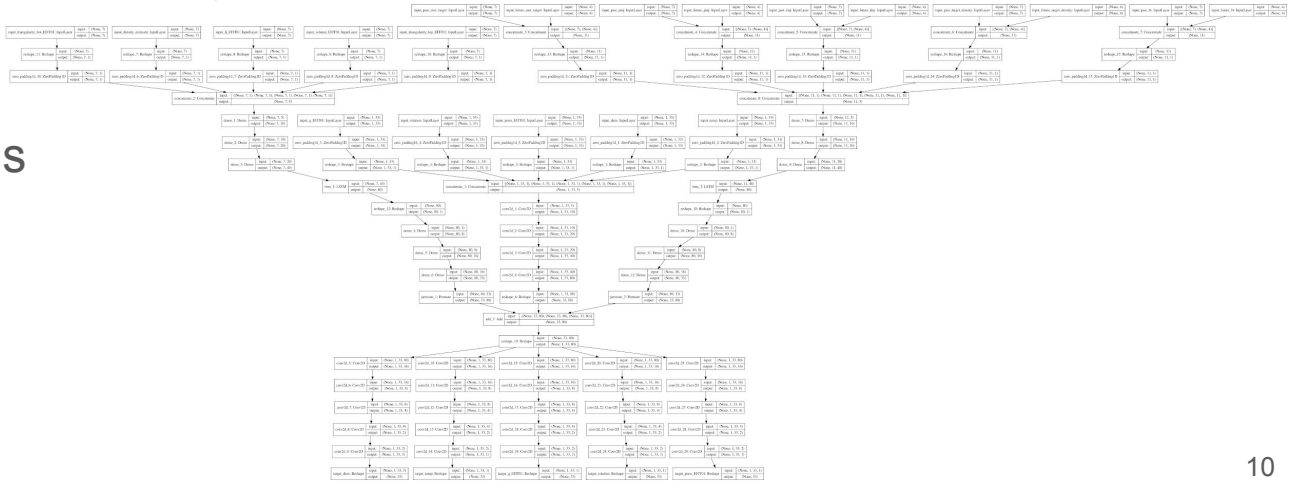
Weights and model parameters automatically parsed

- Python script parses each layer to extract weights and other parameters
  - Eg convolution strides, activation function type
- Generates C code for variables, allocating to either stack or heap
- `keras.tensor` → `k2c_tensor` custom NDarray type

```
struct k2c_tensor{
    float *array;
    size_t ndim;
    size_t numel;
    size_t shape[K2C_MAX_NDIM];
    }
```

Supports complex model architectures

- Keras model consists of layers composed into directed acyclic graph
- Topological sorting algorithm used to flatten graph to linear sequence
- Can handle arbitrarily complicated model structures
  - Recurrent connections
  - Bidirectional / Time distributed layers
  - Shared layers
- Generates C code
  to call layer functions
  in correct order



C backend supports full range of Keras options

- Each Keras layer implemented as pure C function
- Only ~1500 lines
- Supports nearly all Keras layers and options
- Relies only on C standard library

Core	Dense, Reshape, Flatten, Permute, RepeatVector, BatchNormalization, Embedding
Convolution	Convolution(1D/2D/3D,with arbitrary stride, dilation, padding),Cropping(1D/2D/3D), UpSampling (1D/2D/3D), ZeroPadding (1D/2D/3D)
Pooling	MaxPooling(1D/2D),AveragePooling(1D/2D), GlobalMaxPooling (1D/2D/3D),GlobalAveragePooling (1D/2D/3D)
Recurrent	SimpleRNN, GRU, LSTM (statefull or stateless)
Merge	Add, Subtract, Multiply, Average, Maximum,Minimum, Concatenate, Dot
Wrappers	TimeDistributed, Bidirectional
Activation	ReLU, tanh, sigmoid, hard sigmoid, exponential,softplus, softmax, softsign, LeakyReLU, PReLU,ELU, ThresholdedReLU

Automated testing & Extensibility

- During conversion, random inputs are generated and fed through Keras model
- Input/output pairs saved and used to generate C test function
- Test function calls C version of the model with generated inputs and compares outputs to expected values
- Automatically verifies that results match to within user specified tolerance

- Backend can be easily modified to wrap standard linear algebra libraries such as BLAS, LAPACK, MKL etc
- Can also be extended to support custom layer types not included in Keras
  - Only requires definition of the C function, and Python method for parsing

Conversion only requires single command

- From within Python:

```
from keras2c import k2c
k2c(my_model, "my_converted_model")
```

- From command line:

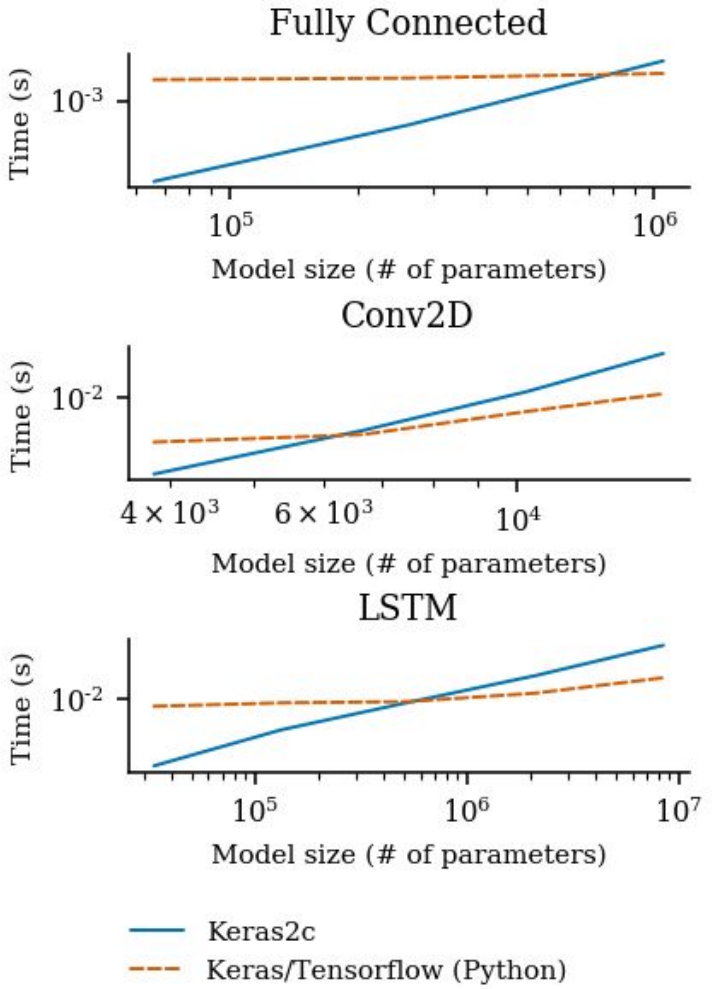
```
python -m keras2c model_path "my_converted_model"
```

- No other user input required
- Generates 3 files:
  - `my_converted_model.c` → source for NN function
  - `my_converted_model.h` → header file with declarations
  - `my_converted_model_test_suite.c` → automated testing to ensure accuracy
- Source / header file can then be used in existing codebase to call neural net function

```
my_converted_model(k2c_tensor * inputs, k2c_tensor * outputs);
```

Comparable speed to optimized TensorFlow

- Backend not currently optimized for speed, yet still outperforms highly optimized TensorFlow backend for many model types
- Dense/Fully Connected and Recurrent models outperform TensorFlow up to 1 million parameters
- Convolutional models outperform TensorFlow up to 5000 parameters



Safe for real-time systems

- All backend and generated code designed to be deterministic and thread-safe
- Non-deterministic function calls (memory allocation, etc) are segmented into dedicated initialization and cleanup routines to be run before and after the real-time portion
- All functions re-entrant, with explicit inputs and outputs and no use of mutable global variables
- Allows multiple calls to functions from different threads safely

Real-time applications: DIII-D Plasma Control

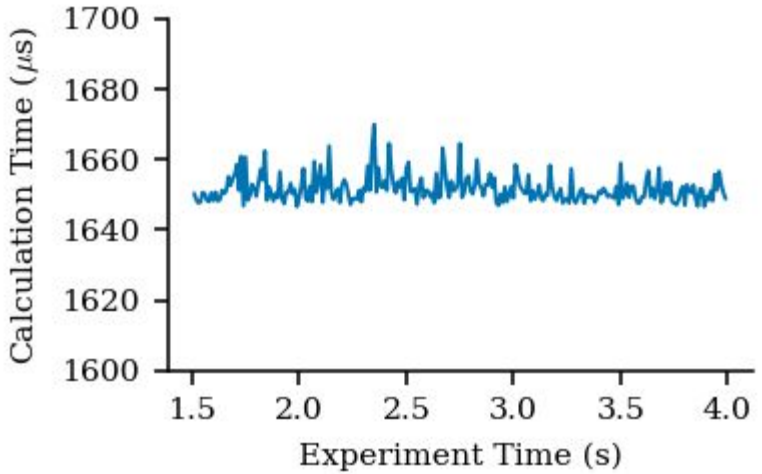
- Tested extensively on the Plasma Control System (PCS) at DIII-D tokamak at the National Fusion Facility operated by General Atomics in San Diego
- PCS: software framework running on GNU/Linux real-time computers connected via an InfiniBand QDR interface
- Operates on microsecond timescales
  - Acquiring data from sensors and diagnostics
  - Calculates monitoring and feedback algorithms
  - Output control commands to actuators on the tokamak device

- In total, PCS runs approximately 50 different algorithms on varying periodics
- Currently 3 algorithms use Keras2c framework to analyze and control the plasma state:
  - Predicting plasma disruptions (*FRNN)*
  - Predicting & controlling neoclassical tearing instabilities (*MLDA)*
  - Predicting & controlling plasma transport (*ETEMP)*
- Other algorithms in development will use Keras2c for controlling plasma divertor and pedestal

Real-time applications: DIII-D Plasma Control

- Example timing shown for neural net predicting plasma transport
  - 30 convolutional layers of varying size
  - 2 recurrent LSTM layers
  - Dozens of reshaping/padding/merging operations
  - Multi-input/multi-output model with branching internal structure
  - Total 45,485 parameters
- Mean time 1.65 ms*
- Worst case jitter 23 μs, rms 3.75 μs

*Also includes time to gather input data from other processes and pre-processing



Summary

- Existing approaches for deploying machine learning models not feasible for low latency, deterministic real-time applications
- Keras2c generates self contained C code, can be easily included in existing systems
- Supports complex model architecture and full range of Keras functionality
- Conversion is fully automated and tests verify accuracy
- Tested for real-time use on DIII-D Plasma Control System
- Fully open source, contributions and improvements welcome
  - https://github.com/f0uriest/keras2c
  - https://f0uriest.github.io/keras2c/
- Publication in review at *Engineering Applications of Artificial Intelligence*

Work supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences, using the DIII-D National Fusion Facility, a DOE Office of Science user facility, under Awards DE-FC02-04ER54698, DE-SC0015878, DE-AR0001166,and Field Work Proposal No. 1903

U.S. DEPARTMENT OF ENERGY | Office of Science

DIII-D NATIONAL FUSION FACILITY SAN DIEGO

References

- Abadi, M.et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. https://www.tensorflow.org/.
- Chollet, F.et al. Keras https://keras.io 2015.
- Hunt, K. J., Sbarbaro, D, Żbikowski, R & Gawthrop, P. J. Neural networks for control systems—a survey. Automatica 28,1083–1112 (1992).
- Jin, L., Li, S., Yu, J. & He, J. Robot manipulator control using neural networks: A survey. Neurocomputing 285,23–34 (2018).
- Liu, W.et al.A survey of deep neural network architectures and their applications. Neurocomputing 234,11–26 (2017).
- Abiodun, O. I.et al.State-of-the-art in artificial neural network applications: A survey. Heliyon 4, e00938 (2018).
- Amazon SageMaker https://aws.amazon.com/sagemaker/ (2020).
- Oracle GraphPipe https://oracle.github.io/graphpipe/ (2020).
- Bai, J., Lu, F., Zhang, K.,et al. ONNX: Open Neural Network Exchange https://github.com/onnx/onnx 2019.
- Ferron, J. R., Penaflor, B., Walker, M. L., Moller, J. & Butner, D. Flexible software architecture for tokamak discharge control systems.Proceedings -Symposium on Fusion Engineering 2,870–873 (1995).
- Hyatt, A. W.et al.Physics operations with the DIII-D plasma controlsystem.IEEE Transactions on Plasma Science38,434–440 (2010).
- Luxon, J. L. A design retrospective of the DIII-D tokamak.Nuclear Fusion42,614–633 (2002).11
- Kahn, A. B. Topological sorting of large networks.Communications of the ACM5,558–562 (1962).
- Abbate J., C. R. & Kolemen, E. Fully Data-Driven Profile Prediction for DIII-D.Nuclear Fusion (In Review).
- Kates-Harbeck, J., Svyatkovskiy, A. & Tang, W. Predicting disruptive in-stabilities in controlled fusion plasmas through deep learning.Nature 568,526–531 (2019).
- Fu, Y.et al.Machine learning control for disruption and tearing mode avoidance.Physics of Plasmas 27(2020).
- Kolemen, E.et al.Initial development of the DIII-D snowflake divertor control.Nuclear Fusion 58,066007 (2018).
- Laggner, F. et al.Real-time pedestal optimization and ELM control with 3D fields and gas flows on DIII-D.Nuclear Fusion 60,076004 (2020).
- Penaflor, B.et al.Extending the capabilities of the DIII-D Plasma Control System for worldwide fusion research collaborations.Fusion engineering and design 84,1484–1487 (2009)