QLKNN-hyper-10D in integrated modelling

Introducing QLKNN-hyper-10D and family

Karel van de Plassche[1]

J. Citrin[1], C. Bourdelle[2], Y. Camenen[3], F. J. Casson[4], F. Felici[5], P. Horn[1,6]
A. Ho[1], M. Marin[1], S. Van Mulders[5], F. Koechl[4], and JET Contributors*

and countless Master students

[1] *DIFFER, PO Box 6336, 5600 HH Eindhoven, The Netherlands*
[2] *CEA, IRFM, F-13108 Saint-Paul-lez-Durance, France*
[3] *CNRS, Aix-Marseille Univ., PIIM UMR7345, Marseille, France*
[4] *CCFE, Culham Science Centre, OX14 3DB, Abingdon, UK*
[5] *EPFL-SPC, CH-1015, Lausanne, Switzerland*
[6] *CASA, Eindhoven University of Technology, Eindhoven, The Netherlands*

* *See the author list of E. Joffrin et al. 2019 Nucl. Fusion* ***59*** *112021*

September 16th, 2020

Section A

QLKNN in JINTRAC and RAPTOR; PoP2020

QuaLiKiz: fast and accurate reduced turbulence model

- Reduced quasi-linear gyrokinetic code
- Multiple examples of agreement with experiments[1,2,3,4,5,6]
- 6 orders of magnitude faster than nonlinear calculations
  - Still in agreement with nonlinear flux [7,8]
- 10 CPU seconds to calculate turbulent fluxes at a single radial position
- QuaLiKiz + JINTRAC: $\mathcal{O}(24)$ *hrs* per second of JET

Open source: see http://qualikiz.com for more information

[1] C. Bourdelle PPCF 2016
[2] J. Citrin PPCF 2017
[3] S. Breton NF 2018
[4] O. Linder NF 2019
[5] A. Ho NF 2019
[6] F. Casson NF 2020
[7] A. Casati NF 2009
[8] J. Citrin NF 2012

Faster modelling by replacing QuaLiKiz ...

Local dimensionless plasma parameters

- R/L_{T_e}
- R/L_{T_i}
- R/L_n
- q
- $\hat{s}$
- r/R
- T_i/T_e
- ν^*
- Z_{eff}
- (γ_E)
- $\vdots$

10^1 CPUs*

QuaLiKiz

- Heat flux $q_{i,e}$
- Particle flux $\Gamma_{i,e}$
- Momentum flux Π_i

* Depends on ”physics”. It’s 1 (no rotation) to 4 (rotation) seconds per radial point per wavenumber

... by fast Neural Network

Local dimensionless plasma parameters

R/L_{T_e}

R/L_{T_i}

R/L_n

q

$\hat{s}$

r/R

T_i/T_e

ν^*

Z_{eff}

(γ_E)

10^{-2} – 10^{-5} CPUs

Neural network(s)

In GyroBohm units

Heat flux $q_{i,e}$

Particle flux $\Gamma_{i,e}$

Jacobians $\partial q_{i,e}/\partial \left(R/L_{T_e}, ..., \gamma_E\right)$

Large dataset of 3e8 QuaLiKiz points has been generated

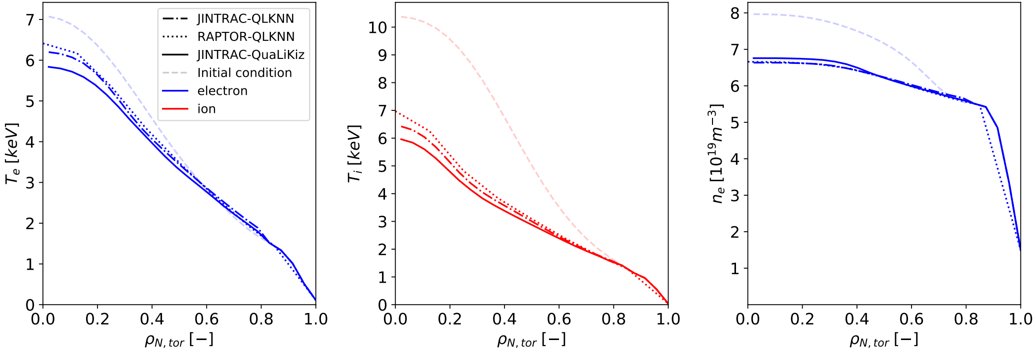
Dataset spans wide core-relevant regime, and is freely available on Zenodo: doi.org/10.5281/zenodo.3497065, online visualization

variable	# points	min	max
$k_\theta \rho_s \leq 2$	10	0.1	2
$k_\theta \rho_s > 2$	8	3.5	36
R/L_{T_e}	12	0	14
R/L_{T_i}	12	0	14
R/L_n	12	-5	6
q	10	0.66	15
$\hat{s}$	10	-1	5
r/R	8	0.03	0.33
T_i/T_e	7	0.25	2.5
ν^*	6	1×10^{-5}	1
Z_{eff}	5	1	3
Total flux calculations	3×10^8	$\approx 1.3\,\mathrm{MCPUh}$	

Benchmark: good match between QLKNN and QuaLiKiz

Simplified physics case

Based on high performance baseline JET92436 [A. Ho *et al.* NF 2019]

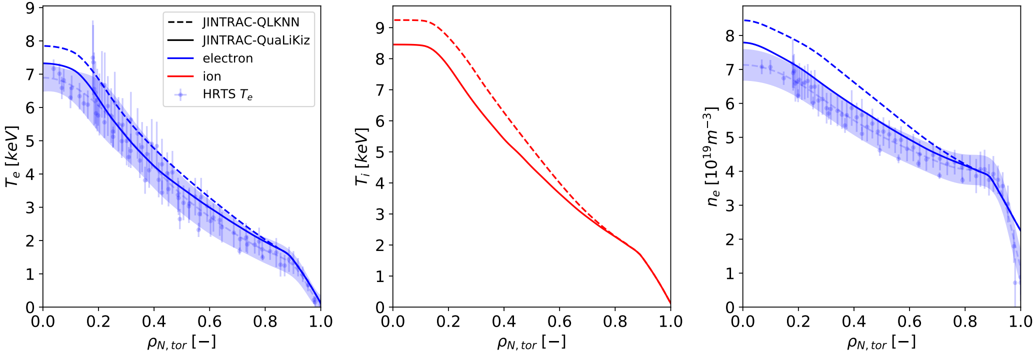


JINTRAC-RAPTOR-QLKNN benchmark at t=22.75s [K.L. van de Plassche et al. PoP 2020]

No full QuaLiKiz surrogate, but most important features are captured

Full physics case, based on high performance hybrid [Casson IAEA 2018]

Parameters not included in QLKNN (e.g. γ_E, α, $R/L_{T_{i,imp}}$) can play a role; still good match!



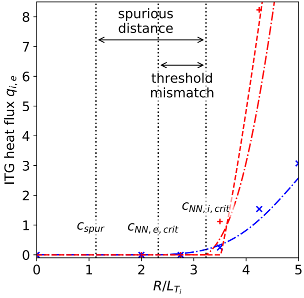
JINTRAC-QLKNN benchmark at t=22.75s [K.L. van de Plassche et al. PoP 2020]

Essential: Capture physics in the surrogate model

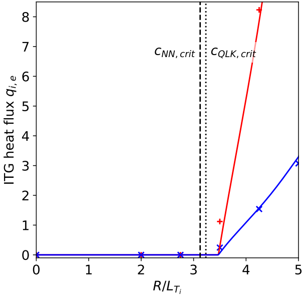
Same threshold for all transport channels ($q_{i,e}, \Gamma_e$) essential

- Global regression measures less important than local features
- Global RMS error weak indicator of surrogate model quality

Essential for integrated modelling: Include this in surrogate model.



Bad fit

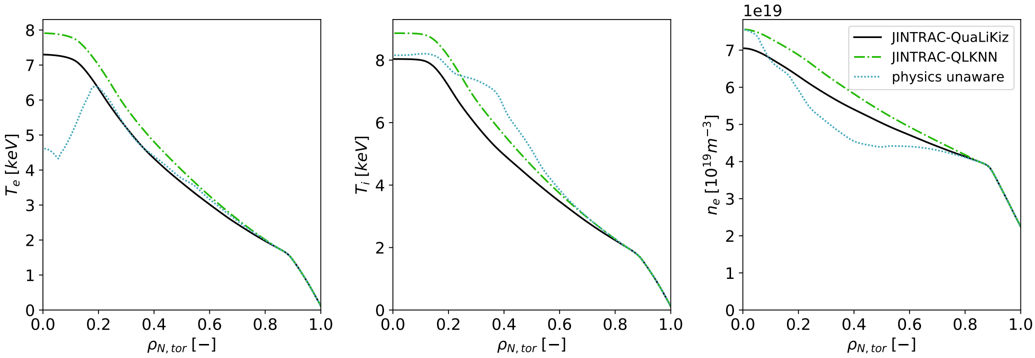


Good fit

Example of matching ITG temperature gradient critical thresholds

Physics-unaware: Less quality for the same fit quality!

Full physics case, based on high performance hybrid [Casson IAEA 2018]
NO special cost function, NO special train targets, *same RMS of fit!* (see QLKNN-hyper structure)



JINTRAC-QLKNN-hyper-10D vs a network trained on the full fluxes at t=22.75s [K.L. van de Plassche et al. PoP 2020]

QLKNN enables modelling of transport in seconds

- QLKNN integrated in RAPTOR
  - ND-in, ND-out QLKNN + Jacobians
  - $\sim$270ms per profile (9 radial points, spline base)
  - $\sim$**7s per second** *of JET (dynamic, flattop)*
  - WIP: MKL and MPI acceleration
- QLKNN integrated in JINTRAC
  - ND-in, ND-out QLKNN
  - $\sim$5ms per profile (25 radial points, finite differences)
  - $\sim$**15s per second** *of JET (dynamic, flattop)*
  - WIP: MKL acceleration
- Simulations at unprecedented speeds: ***3-5 orders of magnitude*** over QuaLiKiz+JINTRAC

QLKNN ready for testing

- QLKNN integrated in RAPTOR
  - QLKNN-hyper-10D version ready for users
- QLKNN integrated in JINTRAC
  - QLKNN-hyper-10D version ready for users, drop in for QuaLiKiz (`usenn=1`)
- QLKNN integrated in METIS
  - QLKNN-hyper-10D integrated since 07 Aug 2020, did not try myself!
- QLKNN soon to be available in ETSv6 (hopefully..)
- QLKNN transport model freely available in standalone
  - Visit qualikiz.com
  - Fortran, Python and MATLAB available: wikis/QLKNN/QLKNN-hyper
  - Small caveat, developer-oriented interface, user-friendliness WIP
- Paper available in Physics of Plasmas: doi/10.1063/1.5134126

Section B

Application of PoP QLKNN to ITER cases

Testing QLKNN in new regimes

Training set was designed to have a wide validity regime

- Chosen on our estimate of large-aspect ratio tokamak parameters
- Worked out of the box for JET

HOWEVER, there is no thing like testing!

- Neural network models extrapolate poorly outside training set regimes
  - QLKNN-hyper-10D, sacrifices robustness for speed
- Our experience might be biased
  - BUT: We did consult other experts, thanks C. Bourdelle and Y. Camenen!
- Our ”physics informed” strategy *could* be robust to extrapolation
  - BUT: Always check if we are extrapolating a posteriori

Advantage of theory-based NNs: Improving validity regime

We can check how well the shots match the QLKNN assumptions a-priori

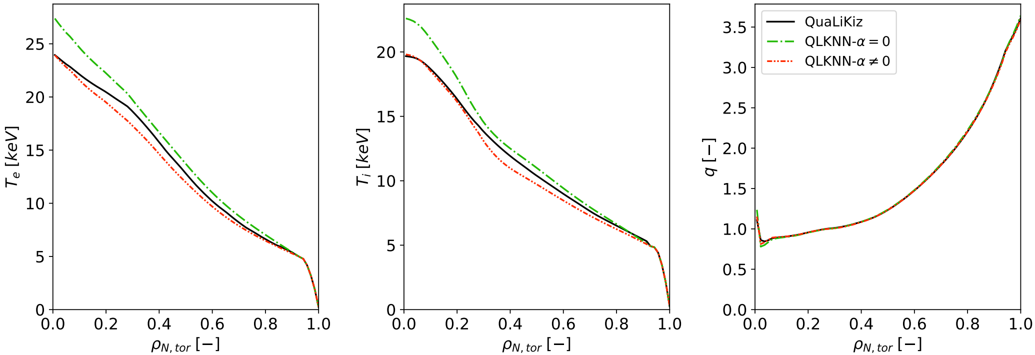
- Full QuaLiKiz uses $s - \alpha$ geometry
- QLKNN-hyper has $\alpha = 0$ in training set, bad assumption for some shots!
- Challenge: Extending training set with α would take multi-MCPUh
- Solution: Capture effect of α by modifying $\hat{s}$ input
- Rule-of-thumb based on initial dedicated QLK scans: $\hat{s}_{eff} = \hat{s} - {}^{\alpha}/_{2}$

Speed of QLKNN-hyper now being leveraged for off-line scenario optimization.
Proof-of-principle with EC deposition q-profile optimization for ITER hybrid scenario.

Full optimization takes only few minutes *[S. Van Mulders, F. Felici et al.]*

New regime: ITER baseline

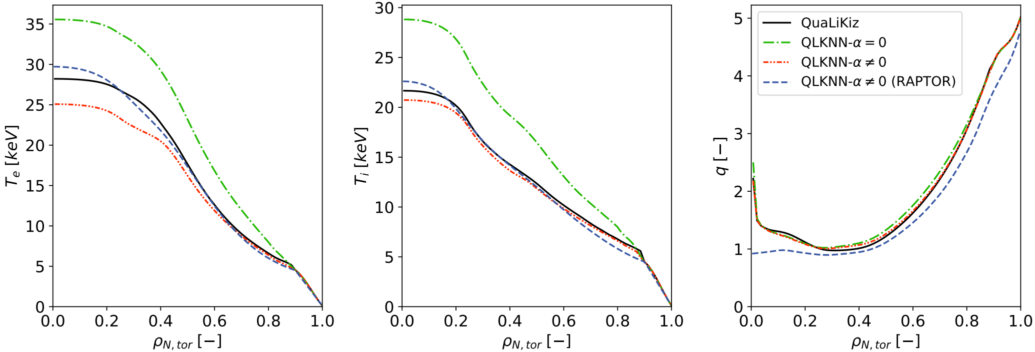
Very encouraging early results, similar performance as for JET in PoP2020
Based on [F. Koechl *et al.* IAEA 2018, P. Mantica *et al.* PPCF 2020]



QLKNN-hyper-$\alpha = 0$ on ITER hybrid with $\alpha \neq 0$ rule-of-thumb $\hat{s}_{eff} = \hat{s} - {}^{\alpha}/_{2}$

New regime: ITER hybrid

Reference run for WIP optimization exercise. α rule important for hybrid scenario due to large $\hat{s}$ and large α. Based on [J. Citrin *et al.* NF 2010]



QLKNN-hyper-$\alpha = 0$ on ITER hybrid with $\alpha \neq 0$ rule-of-thumb $\hat{s}_{eff} = \hat{s} - \alpha/2$ in RAPTOR optimalization [S. Van Mulders et al.]

Section C

Planned extensions to QLKNN model

QLKNN family: Investigate NN train strategies

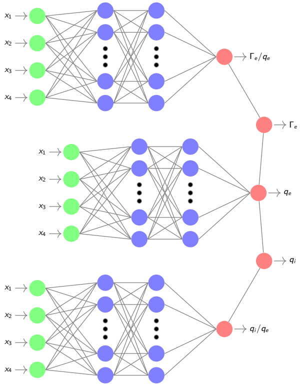
Designing machine learning models is an art in itself

- Depending on strategy, model might not train
  - Millions of free parameters
  - NN training is a highly-dimensional non-convex optimization
- Design decisions might influence applicably of resulting model
- Different use cases might have different demands

Solution: The QLKNN family, extending QLKNN-hyper-10D

- Investigate behaviour of different strategies
- Mixed strategies possible, but not treated here

Include physics in network structure: QLKNN-hyper



General unconstrained 'combined NN'

For ITG general input $\mathbf{x}_g$ and special input x_s

$$\mathbf{x}_g \in \{R/L_{T_e}, R/L_n, q, ..., Z_{eff}\}$$
$$x_s \in \{R/L_{T_i}\}$$

QLKNN-hyper

$$q_e(\mathbf{x}_g, x_s) = NN_{q_e/q_i}(\mathbf{x}_g, x_s) * NN_{q_i}(\mathbf{x}_g, x_s)$$
$$q_i(\mathbf{x}_g, x_s) = NN_{q_i}(\mathbf{x}_g, x_s)$$
$$\Gamma_e(\mathbf{x}_g, x_s) = NN_{\Gamma_e/q_i}(\mathbf{x}_g, x_s) * NN_{q_i}(\mathbf{x}_g, x_s)$$

QLKNN family: Include relevant physics in different ways

- We want to approximate the full QuaLiKiz model *including isotopes, impurities*
  - Challenge: Needs larger dataset, make sure QuaLiKiz is okay
  - Solution: Extend dataset $\Rightarrow$ QLKNN-hyper-*11D*
- We want to include physics rigorously
  - Solution: Include physics in network architecture itself: *QLKNN-HornNet* (P. Horn *et al.*, in preparation)
  - Solves challenge: Combing QLKNN-hyper nets compounds errors
  - Solves challenge: QLKNN-hyper training strategy allows for non-physical freedom
  - WIP: Currently being tested in JINTRAC
- We want to approximate the full QuaLiKiz model
  - Challenge: Hypercube dataset scales poorly with input dimensionality, multiple TCPUh!
  - Solution: Base on experimental data $\Rightarrow$ *QLKNN-jetexp* (A. Ho *et al.*, on EuroFusion pinboard#2118)
  - New challenge: Need large database of profiles of different machines
  - WIP: Publishing paper for JINTRAC and JET, testing ASDEX dataset creation

QLKNN-hyper-11D: Extend to isotopes and impurities

- Extend dataset with impurity density gradients
- Includes updates to physics and numerics, see QuaLiKiz-2.8.0
- Final testing in progress before launching run
- Data generation and managing pipeline ready to go

variable	# points	min	max
$k_\theta \rho_s \leq 2$	10	0.1	2
$k_\theta \rho_s > 2$	8	3.5	36
R/L_{T_e}	11	0	14
R/L_{T_i}	11	0	14
R/L_{n_e}	11	-5	5
$R/L_{n_{i,0}}$	12	-15	15
q	9	0.66	10
$\hat{s}$	9	-1	4
r/R	8	0.1	0.9
T_i/T_e	7	0.25	2.5
ν^*	5	0	0.1
n_i/n_e	4	0	0.3
Total flux calculations	2×10^9	$\approx 8\,\mathrm{MCPUh}$	

QLKNN family: Include relevant physics in different ways

- We want to approximate the full QuaLiKiz model *including isotopes, impurities*
  - Challenge: Needs larger dataset, make sure QuaLiKiz is okay
  - Solution: Extend dataset ⇒ QLKNN-hyper-*11D*
- We want to include physics rigorously
  - Solution: Include physics in network architecture itself: *QLKNN-HornNet* (P. Horn *et al.*, in preparation)
  - Solves challenge: Combing QLKNN-hyper nets compounds errors
  - Solves challenge: QLKNN-hyper training strategy allows for non-physical freedom
  - WIP: Currently being tested in JINTRAC
- We want to approximate the full QuaLiKiz model
  - Challenge: Hypercube dataset scales poorly with input dimensionality, multiple TCPUh!
  - Solution: Base on experimental data ⇒ *QLKNN-jetexp* (A. Ho *et al.*, on EuroFusion pinboard#2118)
  - New challenge: Need large database of profiles of different machines
  - WIP: Publishing paper for JINTRAC and JET, testing ASDEX dataset creation

QLKNN family: Include relevant physics in different ways

- We want to approximate the full QuaLiKiz model *including isotopes, impurities*
  - Challenge: Needs larger dataset, make sure QuaLiKiz is okay
  - Solution: Extend dataset ⇒ QLKNN-hyper-*11D*
- We want to include physics rigorously
  - Solution: Include physics in network architecture itself: *QLKNN-HornNet* (P. Horn *et al.*, in preparation)
  - Solves challenge: Combing QLKNN-hyper nets compounds errors
  - Solves challenge: QLKNN-hyper training strategy allows for non-physical freedom
  - WIP: Currently being tested in JINTRAC
- We want to approximate the full QuaLiKiz model
  - Challenge: Hypercube dataset scales poorly with input dimensionality, multiple TCPUh!
  - Solution: Base on experimental data ⇒ *QLKNN-jetexp* (A. Ho *et al.*, on EuroFusion pinboard#2118)
  - New challenge: Need large database of profiles of different machines
  - WIP: Publishing paper for JINTRAC and JET, testing ASDEX dataset creation

Include physics in network structure: QLKNN-HornNet

New work by P. Horn, K.L. van de Plassche *et al.*

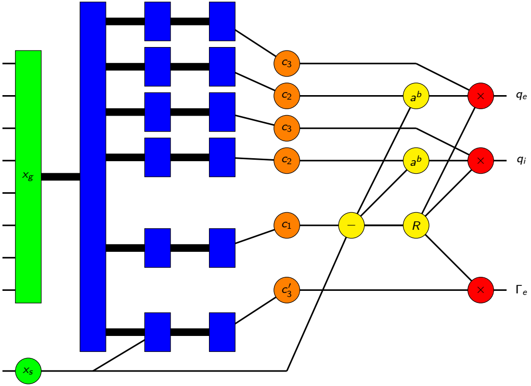
- Inspired by late-fusion techniques in QLKNN-hyper-$\alpha = 0$ [F. Felici *et al.*] on ITER hybrid with $\alpha \neq 0$ rule-of-thumb $\hat{s}_{eff} = \hat{s} - {}^{\alpha}/_{2}$ in RAPTOR [S. Van Mulders *et al.*]

Strategy: Force a Critical Gradient Model *directly into network architecture*

- Pro: Still general, but smooth 1st derivative of IO mapping
- Pro: Simple input/output derivatives
- Pro: Very sharp turbulent threshold

Combining CGM and NNs: The power of general NN tools

NN training techniques can be applied to arbitrary functions. So we can include a Critial Gradient Model as such:



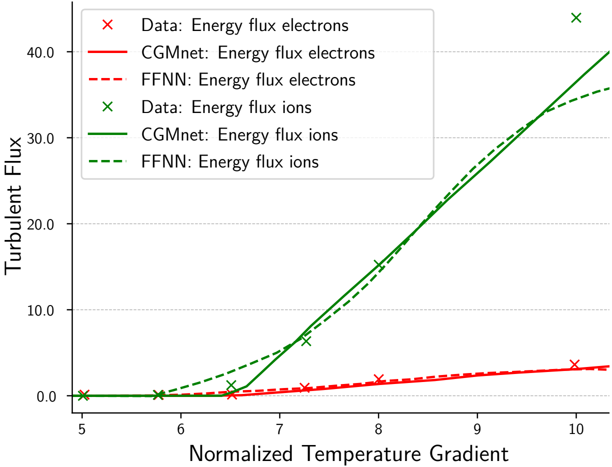
QLKNN-HornNet

$$q_e(\mathbf{x}_g, x_s) = c_{3,e} R(x_s - c_1)(|x_s - c_1|)^{c_{2,e}}$$
$$q_i(\mathbf{x}_g, x_s) = c_{3,i} R(x_s - c_1)(|x_s - c_1|)^{c_{2,i}}$$
$$\Gamma_e = NN(\mathbf{x}_g, x_s) R(x_s - c_1)$$

"slope" $c_{3,[i,e]}(\mathbf{x}_g)$
"threshold location" $c_1(\mathbf{x}_g)$
"bendiness" $c_{2,[i,e]}(\mathbf{x}_g)$
network output $NN(\mathbf{x}_g, x_s)$
Rectifier $R(\cdot)$

Gives a well-constrained neural network model

Standalone QLKNN-HornNet performs well



- Shown in [P. Horn *et al.* MSc. thesis]
- Shows "bendiness" of QLKNN-hyper
- Warning: Bad slice of QLKNN-hyper, good slice of QLKNN-HornNet
- WIP: Testing in JINTRAC. Paper to be submitted

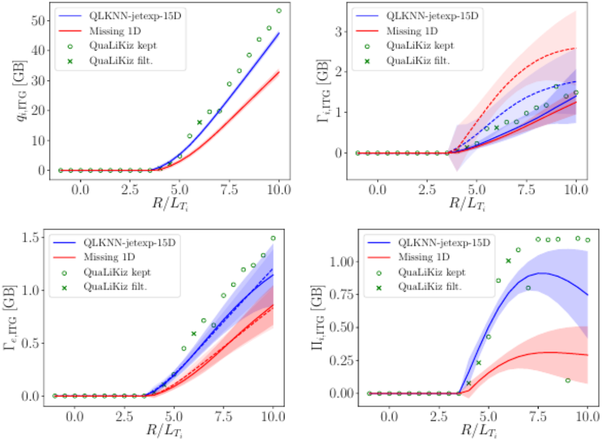
QLKNN family: Include relevant physics in different ways

- We want to approximate the full QuaLiKiz model *including isotopes, impurities*
  - Challenge: Needs larger dataset, make sure QuaLiKiz is okay
  - Solution: Extend dataset ⇒ QLKNN-hyper-*11D*
- We want to include physics rigorously
  - Solution: Include physics in network architecture itself: *QLKNN-HornNet* (P. Horn *et al.*, in preparation)
  - Solves challenge: Combing QLKNN-hyper nets compounds errors
  - Solves challenge: QLKNN-hyper training strategy allows for non-physical freedom
  - WIP: Currently being tested in JINTRAC
- We want to approximate the full QuaLiKiz model
  - Challenge: Hypercube dataset scales poorly with input dimensionality, multiple TCPUh!
  - Solution: Base on experimental data ⇒ *QLKNN-jetexp* (A. Ho *et al.*, on EuroFusion pinboard#2118)
  - New challenge: Need large database of profiles of different machines
  - WIP: Publishing paper for JINTRAC and JET, testing ASDEX dataset creation

QLKNN family: Include relevant physics in different ways

- We want to approximate the full QuaLiKiz model *including isotopes, impurities*
  - Challenge: Needs larger dataset, make sure QuaLiKiz is okay
  - Solution: Extend dataset $\Rightarrow$ QLKNN-hyper-*11D*
- We want to include physics rigorously
  - Solution: Include physics in network architecture itself: *QLKNN-HornNet* (P. Horn *et al.*, in preparation)
  - Solves challenge: Combing QLKNN-hyper nets compounds errors
  - Solves challenge: QLKNN-hyper training strategy allows for non-physical freedom
  - WIP: Currently being tested in JINTRAC
- We want to approximate the full QuaLiKiz model
  - Challenge: Hypercube dataset scales poorly with input dimensionality, multiple TCPUh!
  - Solution: Base on experimental data $\Rightarrow$ *QLKNN-jetexp* (A. Ho *et al.*, on EuroFusion pinboard#2118)
  - New challenge: Need large database of profiles of different machines
  - WIP: Publishing paper for JINTRAC and JET, testing ASDEX dataset creation

QLKNN-jetexp-15D highlight



Showing the necessity of proper data analytics and physics background *[A. Ho et al.: Neural network surrogate of QuaLiKiz using JET experimental data to populate training space]*

Non-planned extensions to QLKNN model

- I just want a non-physics informed QLKNN!
  - Solution: Do a naive fit $\Rightarrow$ *QLKNN-fullflux*
  - Con: No longer under development
  - Con: Your integrated modelling will be of poor quality
- I do not trust your methodology!
  - Solution: Train yourself! Data here: doi.org/10.5281/zenodo.3497065
  - Con: Might be more work than you signed up for

QLKNN: Integration in other frameworks

QLKNN-hyper-10D integrated in various levels of matureness in multiple transport frameworks

- ETS/TCI: P. Strand, K.L. van de Plassche *et al.*
- ASTRA: M. Bergmann *et al.*
- METIS: JF. Artaud *et al.*
- IMAS: CS. Byun *et al.*

Section D
Wrap-up

1. QLKNN in JINTRAC and RAPTOR; PoP2020
2. Application of PoP QLKNN to ITER cases
3. Planned extensions to QLKNN model
4. Wrap-up

Conclusion

QLKNN-hyper-10D model:

- is *ready for exploitation* in RAPTOR and JINTRAC
- is being integrated in *multiple frameworks*
- enables QuaLiKiz approximation *3-5 orders of magnitude faster*

Improvements to QuaLiKiz and its family of surrogate models is ongoing

Backup

Wide parameter ranges

variable																		
$k_\theta \rho_s$	0.1	0.175	0.25	0.325	0.4	0.5	0.6	0.8	1	2	3.5	6	10.5	15	19.5	24	30	36
$-\frac{R}{T_i}\frac{\mathrm{d}T_i}{\mathrm{d}r}$	0	2	2.75	3.5	4.25	5	5.75	6.5	7.25	8	10	14						
$-\frac{R}{T_e}\frac{\mathrm{d}T_e}{\mathrm{d}r}$	0	2	2.75	3.5	4.25	5	5.75	6.5	7.25	8	10	14						
$-\frac{R}{n}\frac{\mathrm{d}n}{\mathrm{d}r}$	-5	-3	-1	0	0.5	1	1.5	2	2.5	3	4	6						
q_x	0.66	1	1.5	2	2.5	3	4	5	10	15								
$\hat{s}$	-1	0.1	0.4	0.7	1	1.5	2	2.75	3.5	5								
r/R	0.03	0.07	0.11	0.15	0.19	0.23	0.28	0.33										
$\frac{T_i}{T_e}$	0.25	0.5	0.75	1	1.33	1.66	2.5											
ν^*	1.00E-05	1.00E-03	1.00E-02	7.00E-02	3.50E-01	1												
Z_{eff}	1	1.3	1.7	2.2	3													
Total	3×10^8	$\approx 1.3\,\mathrm{MCPUh}$																

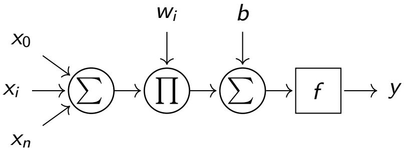
QuaLiKiz simplifications

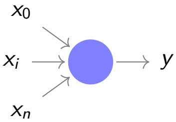
- Low Mach number
- Electrostatic limit
- Eigenfunction assumed to be a (shifted) Gaussian, with width and shift solved from a high-frequency expansion of the dispersion relation
- Shifted circle geometry ($s - \alpha$) with small inverse aspect ratio
- For passing particles: since $v_{\parallel} >> v_{\perp}$, a pitch angle averaged transit frequency as well as curvature and ∇B drift frequencies are performed.
- For trapped particles: since the bounce frequency is larger than ω, a bounced average is performed.
- Only collisions for trapped electrons are taken into account using a Krook type collision operator.
- Particle trajectories are strongly passing or strongly trapped

Simple building block: single neural network neuron

Nonlinear activation function f (e.g. tanh) and weights w_i and biases b that are free and optimized (training)

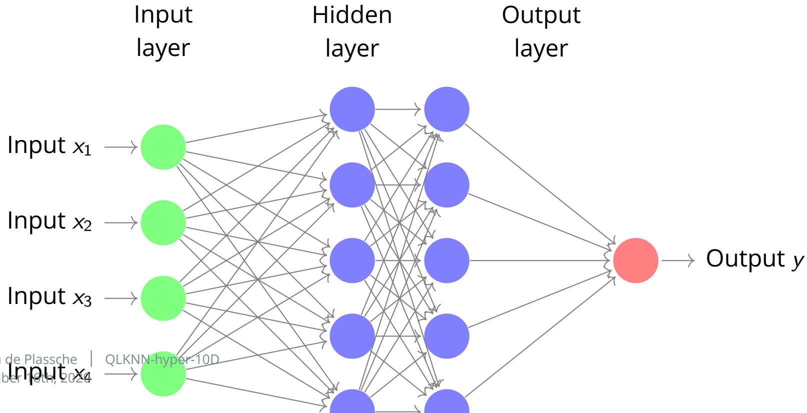
$$y = f\left(\sum_{i=0}^{n} w_i x_i + b\right)$$





Universal approximator: Feed-Forward Neural Network

- Approximate any function up to arbitrary error
- Analytical: Jacobians 'for free' - implicit solvers and trajectory optimization
- Many free parameters, which are optimized/*trained*



Edge-focussed dataset generated (J. van Galen MSc)

variable	# points	min	max
$k_\theta \rho_s$	18	0.1	36
$-\frac{R}{T_{i,e}} \frac{\mathrm{d}T_{i,e}}{\mathrm{d}r}$	11	0	150
$-\frac{R}{n} \frac{\mathrm{d}n}{\mathrm{d}r}$	9	0	30
q_x	10	2	30
$\hat{s}$	10	1	20
r/R	1	0.95	0.95
$\frac{T_i}{T_e}$	1	1	1
ν^*	10	.1	.99
Z_{eff}	6	1	4
Total	8×10^5		

Solving threshold matching: leading-flux fitting

Thoroughly tested and working

Mode	Leading flux	'div's
ETG	q_e	q_i/q_e
ITG	q_i	$q_e/q_i, \Gamma_e/q_i, \Gamma_i/q_i, D_e/q_i, V_i/q_i$ ...
TEM	q_e	$q_i/q_e, \Gamma_e/q_e, \Gamma_i/q_e, D_e/q_e, V_i/q_e$ ...

Then i.e.

$$q_{e,ETG} * \frac{q_{i,ETG}}{q_{e,ETG}} = q_{i,ETG}$$

is perfectly threshold matched

Solving popback and threshold: cost function

Tested and working, giving reduction in popback. Similar for 7D and 9D.

$$C = C_{good} + C_{regu} + \underbrace{C_{stab}}_{new}$$

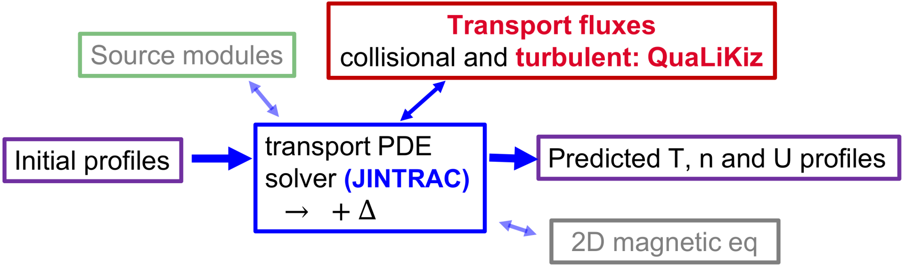
$$C_{good} = \left.\begin{cases} \frac{1}{n}\sum_{i=1}^{n}(QLK_i - NN_i)^2, & \text{if } QLK_i \neq 0 \\ 0, & \text{if } QLK_i = 0 \end{cases}\right\} \text{Sharp threshold}$$

$$C_{regu} = \lambda_{L2} \sum_{i=1}^{k} w_i^2$$

$$C_{stab} = \left.\begin{cases} 0, & \text{if } QLK_i \neq 0 \\ \frac{\lambda_{stab}}{n}\sum_{i=1}^{n} NN_i - c_{stab}, & \text{if } QLK_i = 0 \end{cases}\right\} \text{Reduce popback}$$

Integrated modelling: pathway to full-device modelling

Solver for transport equations coupled with models for sources, sinks, different fluxes etc.



[J. Citrin: Frontiers and interfaces York 2018]

ExB suppression rule

'rotation rule' scales **all fluxes** with $f_{vic}(q, \hat{s}, \epsilon)$ and rotationless growth rate γ_0 which is predicted by a NN.

- Based on new linear GENE scans of new linear-GENE scans of q, ϵ, and $\hat{s}$
- Includes ExB stabilisation and PVG destabilisation

$$
\begin{aligned}
f_{vicnum} &= c_1 q + c_2 * \hat{s} + c_3/\epsilon - c_4 \\
f_{vic} &= \max(1 + f_{vicnum}\gamma_E/\gamma_0, 0) \\
q_{i/e,ITG/TEM} &= f_{vic} * q_{i/e,ITG/TEM} \\
D_{i/e,ITG/TEM} &= f_{vic} * D_{i/e,ITG/TEM} \\
V_{i/e,ITG/TEM} &= f_{vic} * V_{i/e,ITG/TEM} \\
\Gamma_{i/e,ITG/TEM} &= f_{vic} * \Gamma_{i/e,ITG/TEM}
\end{aligned}
$$

QLKNN-10D drivers

Development (analysis, training, etc) in Python. NN prediction drivers in:

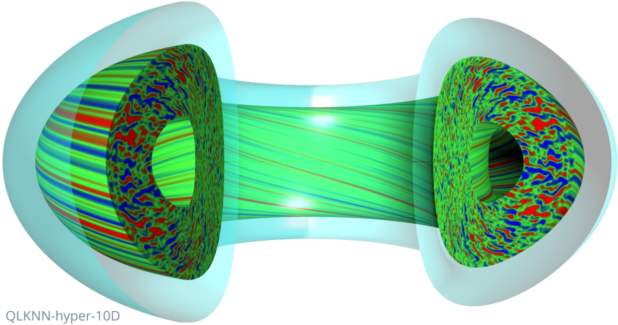
- Pure-python: https://github.com/Karel-van-de-Plassche/QLKNN-develop
- Fortran (MATLAB+Python wrappers): https://github.com/QuaLiKiz-group/QLKNN-fortran
- Network weights and biases: https://gitlab.com/qualikiz-group/QLKNN-hyper

Turbulent transport == fusion performance

Temperature and density influence the fusion power.

$$P_{fusion} \propto n^2 T^2$$

Gradients drive turbulent heat flux, limiting achievable temperatures



[GYRO code: General Atomics Gyromovies]

Tokamak core transport: 1D power balance

Dominated by turbulent particle flux Γ_s and heat flux q_s.

$$\frac{\partial n_s}{\partial t} + \frac{\partial}{\partial r}\left(\Gamma_s\right) = S_s$$
$$\frac{3}{2}\frac{\partial P_s}{\partial t} + \frac{\partial}{\partial r}\left(q_s\right) = Q_s$$

Relatively simple problem IF one knows the turbulent fluxes!

Tokamak core transport: Vlasov equation

3D spatial + 3D velocity + time, coupled partial differential equations...

$$\left[\frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla + \frac{q_s}{m_s}\left(\mathbf{E} + \frac{\mathbf{v}}{c} \times \mathbf{B}\right) \cdot \frac{\partial}{\partial \mathbf{v}}\right] f_s = C_s f_s + S_s$$

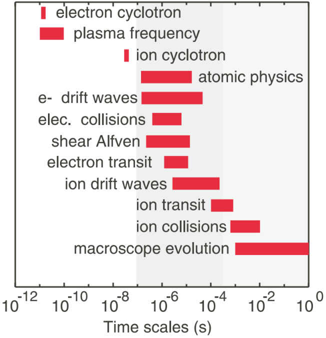
$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}$$

$$\nabla \times \mathbf{E} = -\frac{1}{c}\frac{\partial \mathbf{B}}{\partial t}$$

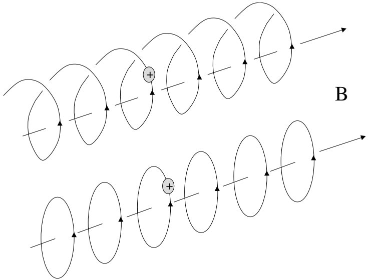
$$\nabla \times \mathbf{B} = \frac{1}{c}\left(4\pi \mathbf{J} + \frac{\partial \mathbf{E}}{\partial t}\right)$$

Tokamak core transport: Gyrokinetics

First simplification, average over gyrofrequency $\rightarrow$ gyrokinetics



[Rognlien PPCF 2005]



[Ethier JPCS 2005]

Still takes $\mathcal{O}(3 \times 10^{10})$s for a single timeslice (without collisions)...

Tokamak core transport: quasi-linear reduced gyrokinetics

"Bath of linear-like fluctuations whose amplitude and wavenumber spectra are set by nonlinear physics."

- Fluctuations are relatively small $\mathcal{O}(10\%) \rightarrow \partial n \ll n$
- Linearized Vlasov equation, only ∂f
- Fluxes set by non-linear scaling rule, derived from nonlinear codes
- Many, many more assumptions $\rightarrow$ QuaLiKiz!

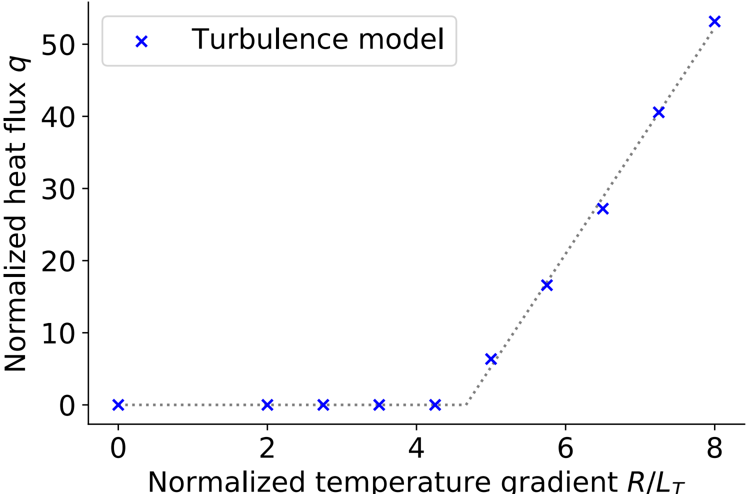
QuaLiKiz is fast, but still too slow

Gyrokinetic Simulation	$\mathcal{O}(\mathit{time})$ single profile (s)
Full 5D code (adiabatic electrons)	3×10^{10}
QuaLiKiz	1×10^{2}
Feed-Forward Neural Network	1×10^{-3}
Real-time controller	$\mathcal{O}(1 \times 10^{-3})$

The trick: use a NN as a surrogate model for QuaLiKiz.

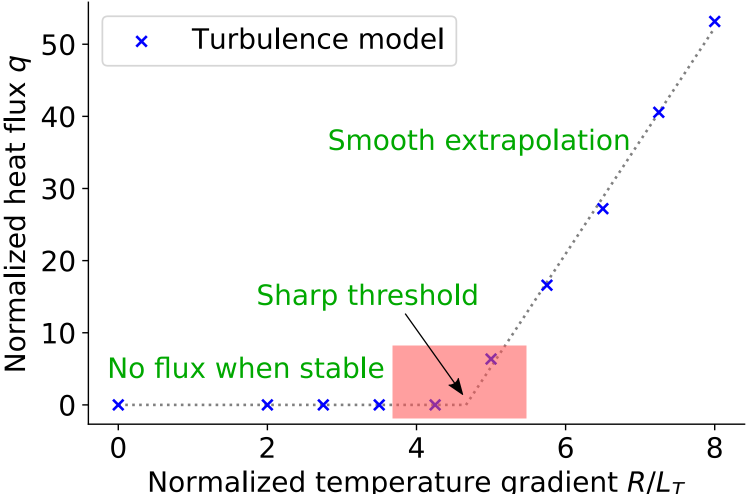
Capture of non-smooth behaviour essential

(earlier version of QLKNN)



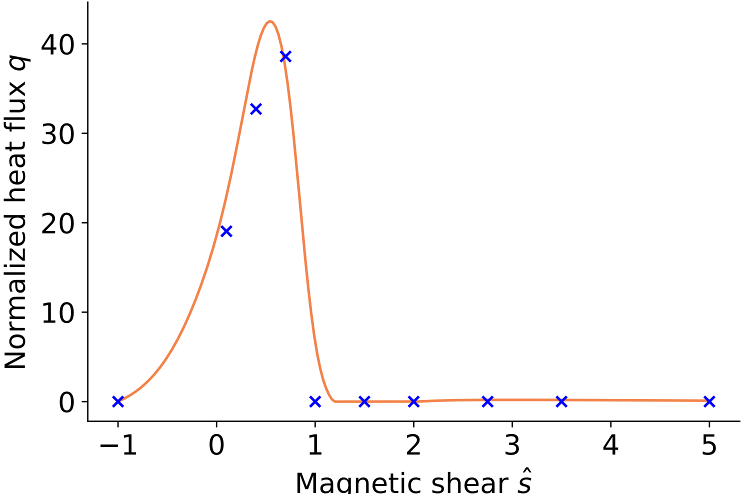
Capture of non-smooth behaviour essential

(earlier version of QLKNN)



Turbulence has very nonlinear bahaviour in 9D

(earlier version of QLKNN)



9D NNs trained and embedded in transport frameworks

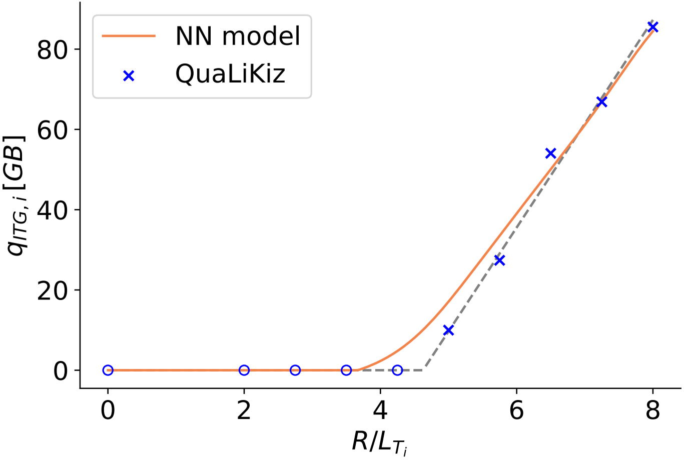
- 9D Neural Networks trained
  - Trained on three main instabilities: ETG, ITG, TEM
  - Trained for heat and particle fluxes
  - Ion and electron threshold matched
- NNs embedded in transport model RAPTOR (RApid Plasma Transport simulatOR) [9]
- Several measures of goodness related to underlying physics found
  - Capture of instability threshold (thresh mis)
  - Match of ion and electron particle and heat threshold (thresh mismatch)
  - Stability in expected stable region (pop)
  - Smoothness of unstable region prediction (wobble)

[9] F. Felici PPCF 2012

51/31

Neural network ’3D in, 1D out’ slice showcase

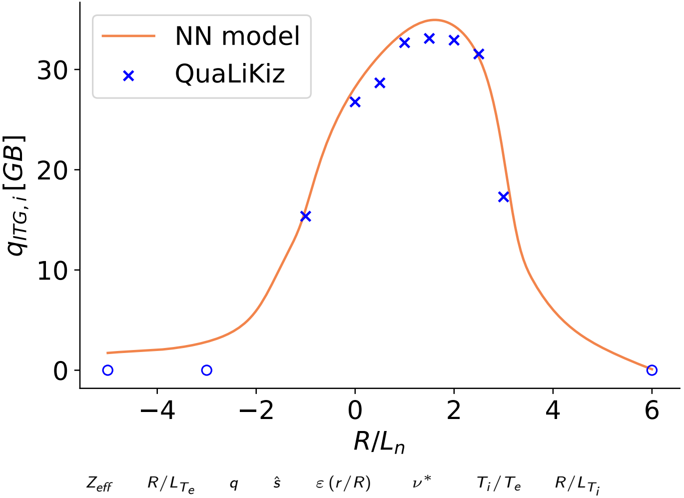
(earlier version of QLKNN)



Z_{eff} R/L_{T_e} q $\hat{s}$ $\varepsilon\,(r/R)$ ν^* T_i/T_e R/L_n

Neural network ’3D in, 1D out’ slice showcase

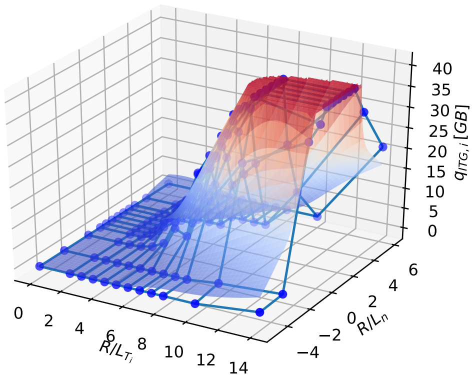
(earlier version of QLKNN)



Neural network '3D in, 1D out' slice showcase

(earlier version of QLKNN)

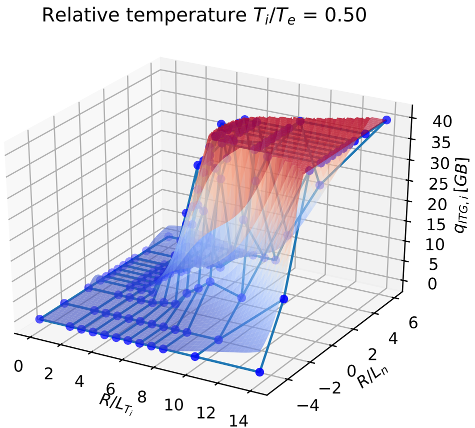
Relative temperature T_i/T_e = 0.25



Z_{eff}	R/L_{T_e}	q	$\hat{s}$	$\varepsilon\,(r/R)$	ν^*
1	5.75	3	.4	.11	10^{-2}

Neural network '3D in, 1D out' slice showcase

(earlier version of QLKNN)

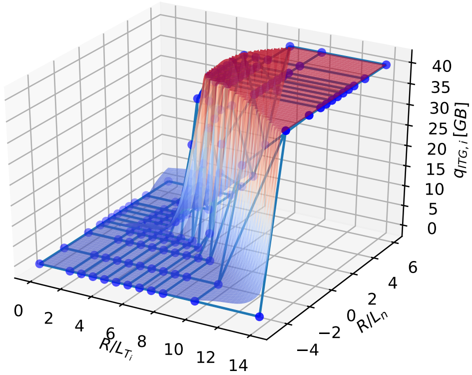


Z_{eff}	R/L_{T_e}	q	$\hat{s}$	$\varepsilon\,(r/R)$	ν^*
1	5.75	3	.4	.11	10^{-2}

Neural network ’3D in, 1D out’ slice showcase

(earlier version of QLKNN)

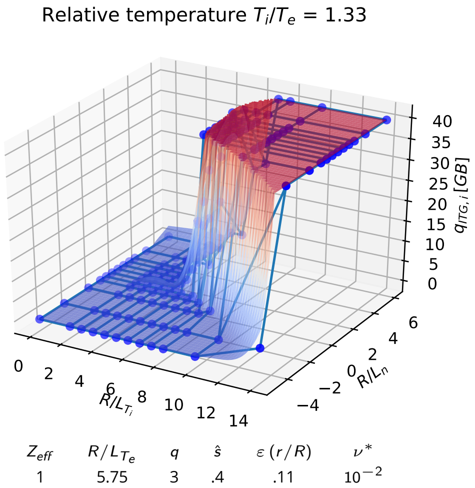
Relative temperature T_i/T_e = 1.00



Z_{eff}	R/L_{T_e}	q	$\hat{s}$	$\varepsilon\,(r/R)$	ν^*
1	5.75	3	.4	.11	10^{-2}

Neural network ’3D in, 1D out’ slice showcase

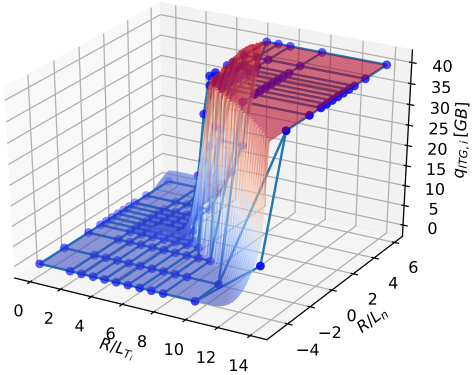
(earlier version of QLKNN)



Neural network ’3D in, 1D out’ slice showcase

(earlier version of QLKNN)

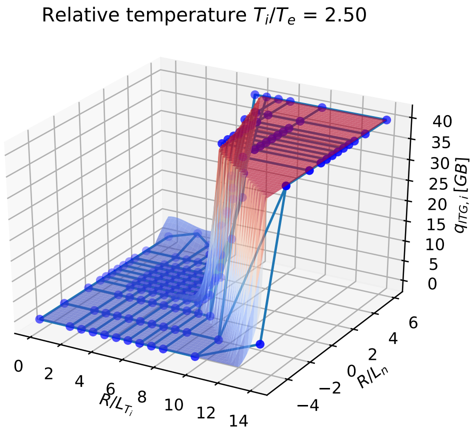
Relative temperature T_i/T_e = 1.66



Z_{eff}	R/L_{T_e}	q	$\hat{s}$	$\varepsilon\,(r/R)$	ν^*
1	5.75	3	.4	.11	10^{-2}

Neural network '3D in, 1D out' slice showcase

(earlier version of QLKNN)



Z_{eff}	R/L_{T_e}	q	$\hat{s}$	$\varepsilon\ (r/R)$	ν^*
1	5.75	3	.4	.11	10^{-2}

Lessons learned: Tensorflow is optimized for deep learning

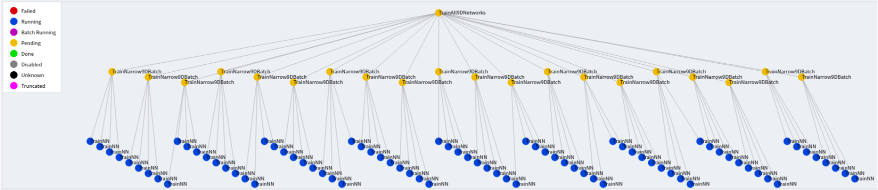
Sometimes surprising bottlenecks, probably more so for higher-level codes!
Runtime for one epoch, 2.10^7 samples.

Machine	Backprop [ms]	CPU shuffle [ms]	dataset API shuffle [ms]
CPU (4x3 GHz)	260	100	200
GPU Quadro K420	190	100	200
GPU Quadro K620	125	100	200
GPU Tesla P100	7	100	200

My choice of tools

- Plotting: bokeh, seaborn, matplotlib
- Machine Learning: TensorFlow (keras?)
- Automation: Luigi
- Speed: Intel MKL (C/F)

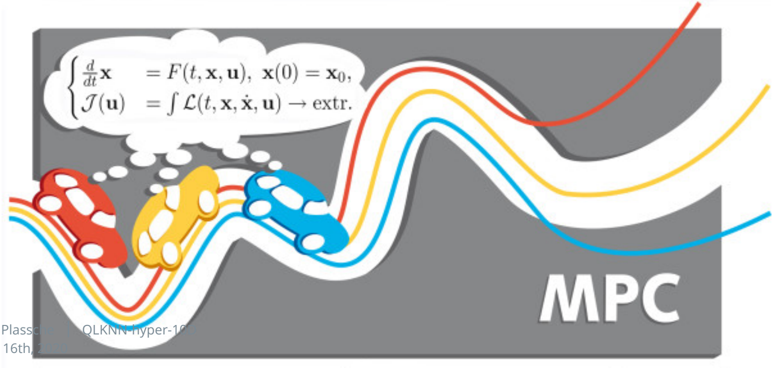
Bokeh Tensorflow



Fast modelling is essential for future tokamaks

Fast modelling enables:

- Optimizing and analysing current experiments
- Extrapolation to future devices
- Real-time control of plasma parameters



Large 9D database generated to train NNs

variable	# points	min	max
$k_\theta \rho_s$	18	0.1	36
$-\frac{R}{T_e}\frac{\mathrm{d}T_e}{\mathrm{d}r}$	12	0	14
$-\frac{R}{T_i}\frac{\mathrm{d}T_i}{\mathrm{d}r}$	12	0	14
$-\frac{R}{n}\frac{\mathrm{d}n}{\mathrm{d}r}$	12	-5	6
q_x	10	0.66	15
$\hat{s}$	10	-1	5
$^r/_R$	8	0.03	0.33
$\frac{T_i}{T_e}$	7	0.25	2.5
ν^*	6	1×10^{-5}	1
Z_{eff}	5	1	3
Total	3×10^8	$\approx 1.3\,\mathrm{MCPUh}$	

Generated using the QuaLiKiz-pythontools

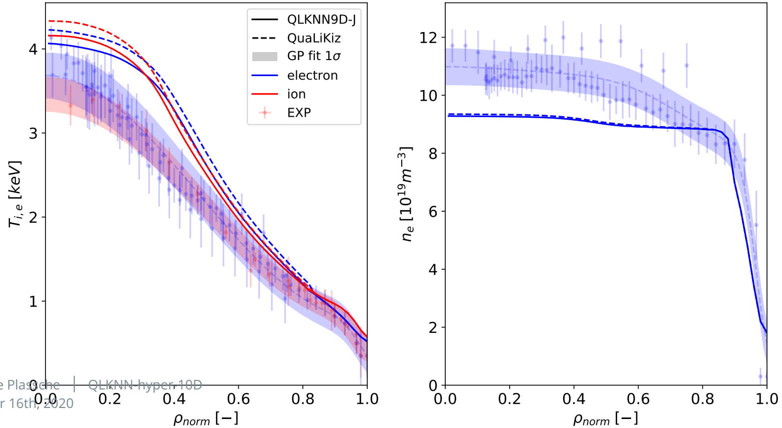
Conclusion

We have achieved

- Easily extendable gyrokinetic neural network training
- On the largest gyrokinetic database to date
- Accurately predicting turbulent heat transport fluxes
- Enabling unprecedented fast tokamak scenario modelling

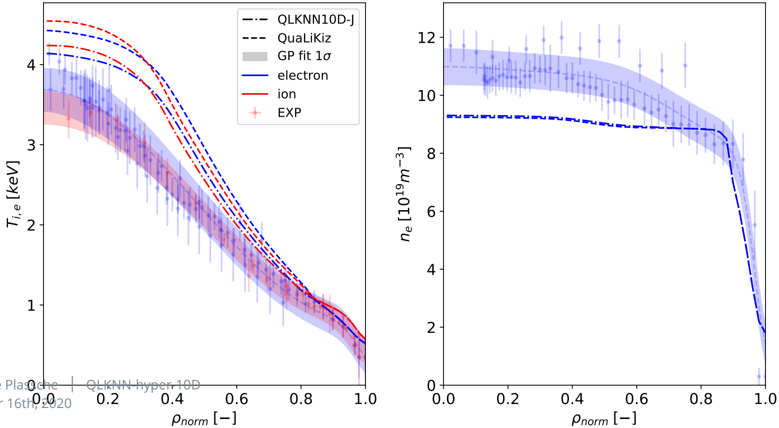
JET73342 no rotation

High collisionality case (known underprediction n_e QuaLiKiz)
[Baoicchi PPCF 2015, Citrin NF 2015]



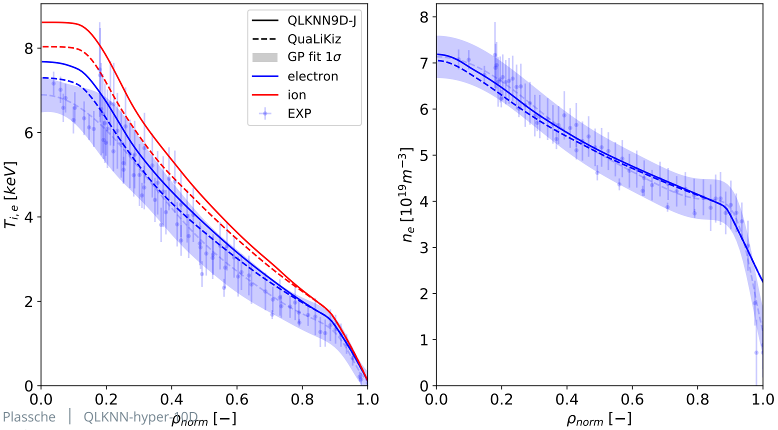
JET73342 with rotation

High collisionality case (known underprediction n_e QuaLiKiz)
[Baoicchi PPCF 2015, Citrin NF 2015]



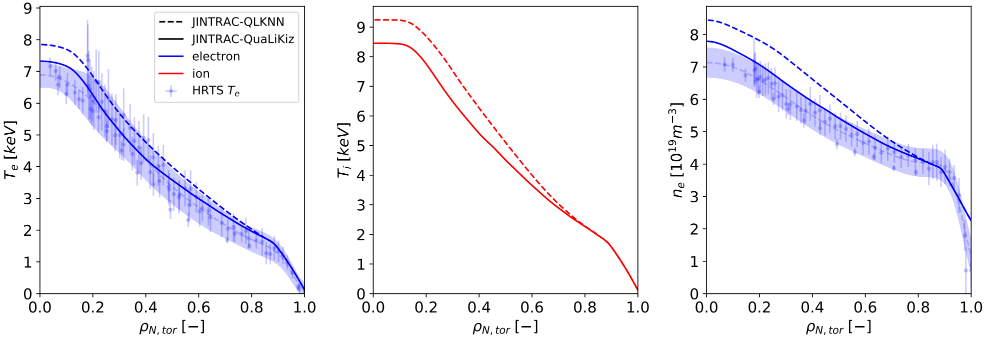
JET92398 no rotation

High performance hybrid [Casson IAEA 2018]



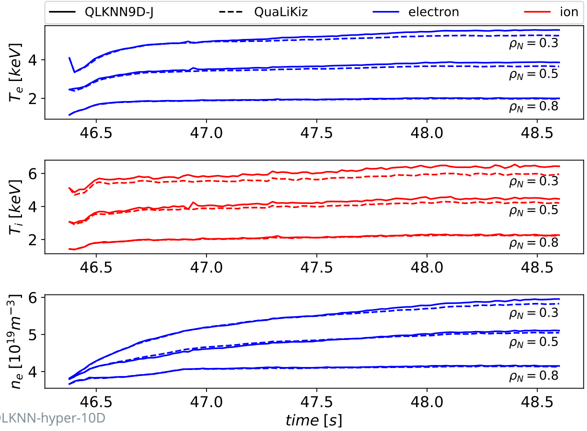
JET92398 with rotation

High performance hybrid [Casson IAEA 2018]



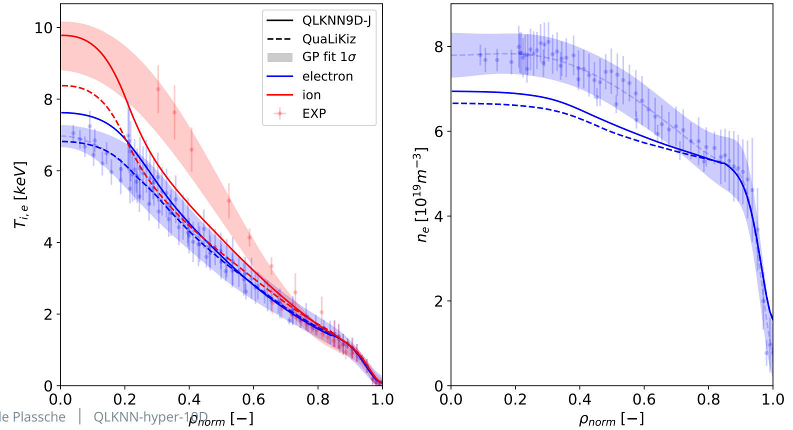
JET92398 no rotation

High performance hybrid [Casson IAEA 2018]



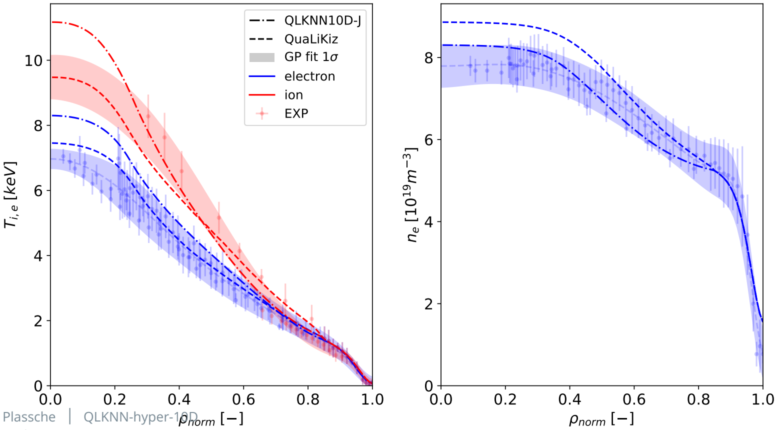
JET92436 no rotation

High performance baseline [Ho NF 2019]



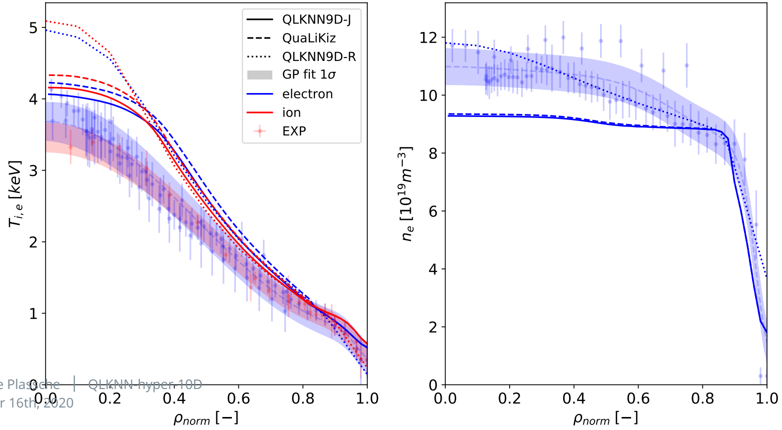
JET92436 with rotation

High performance baseline [Ho NF 2019]



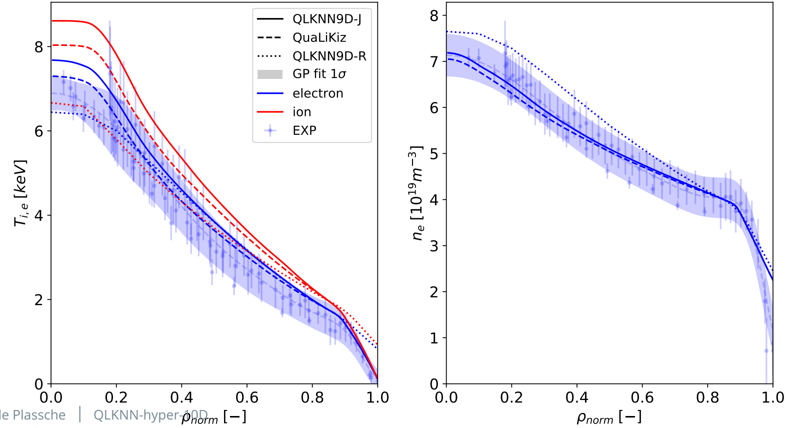
JET73342 no rotation

High collisionality case (known underprediction n_e QuaLiKiz)
[Baoicchi PPCF 2015, Citrin NF 2015]



JET92398 no rotation

High performance hybrid [Casson IAEA 2018]



JET92436 no rotation

High performance baseline [Ho NF 2019]