HiPAS GridLAB-D

Software Implementation Presentation (May 2020)

EPC 17-043 GLOW

EPC 17-046 HiPAS

EPC 17-047 OpenFIDO

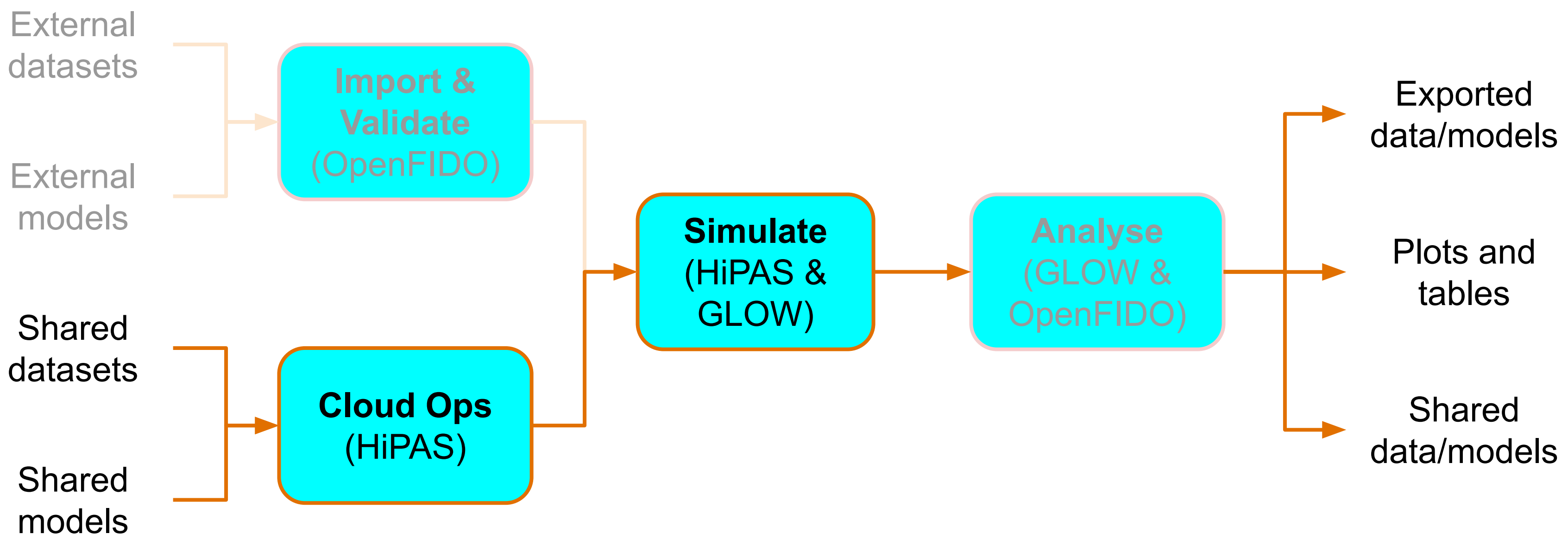
This presentation was prepared with funding from the California Energy Commission under grant EPC-17-046. SLAC National Accelerator Laboratory is operated for the US Department of Energy by Stanford University under Contract No. DE-AC02-76SF00515

U.S. DEPARTMENT OF ENERGY

Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Project Focus Area



Project Objectives

Enhance GridLAB-D to support principle California use-cases

1. Integration Capacity Analysis - Integrate improvements from GLOW (CEC)
2. Resilience - Integrate improvements from GRIP (DOE)
3. Tariff design - Integrate improvements from Powernet With Market (CEC)
4. Electrification - Enable decarbonization simulations

HiPAS Capabilities Currently in Development

SLAC

High-performance simulation

- Machine learning powerflow solver

Improved data processing tools

- General file input/output (“any input, any output, anytime, anywhere”)
- High-performance database module (InfluxDB)

New simulation modules/classes

- Industrial loads (NAICS facilities from NERC LMFT)
- Residential loads (RBSA data, multi-family residences)
- Commercial loads (CEUS data, retail/office/schools/health/etc. buildings)
- Revenue module (tariff and billing)

Highlights: Cloud deployment

Github deployment

- Project repository: http://source.gridlabd.us/
- Integrated online documentation at http://docs.gridlabd.us/

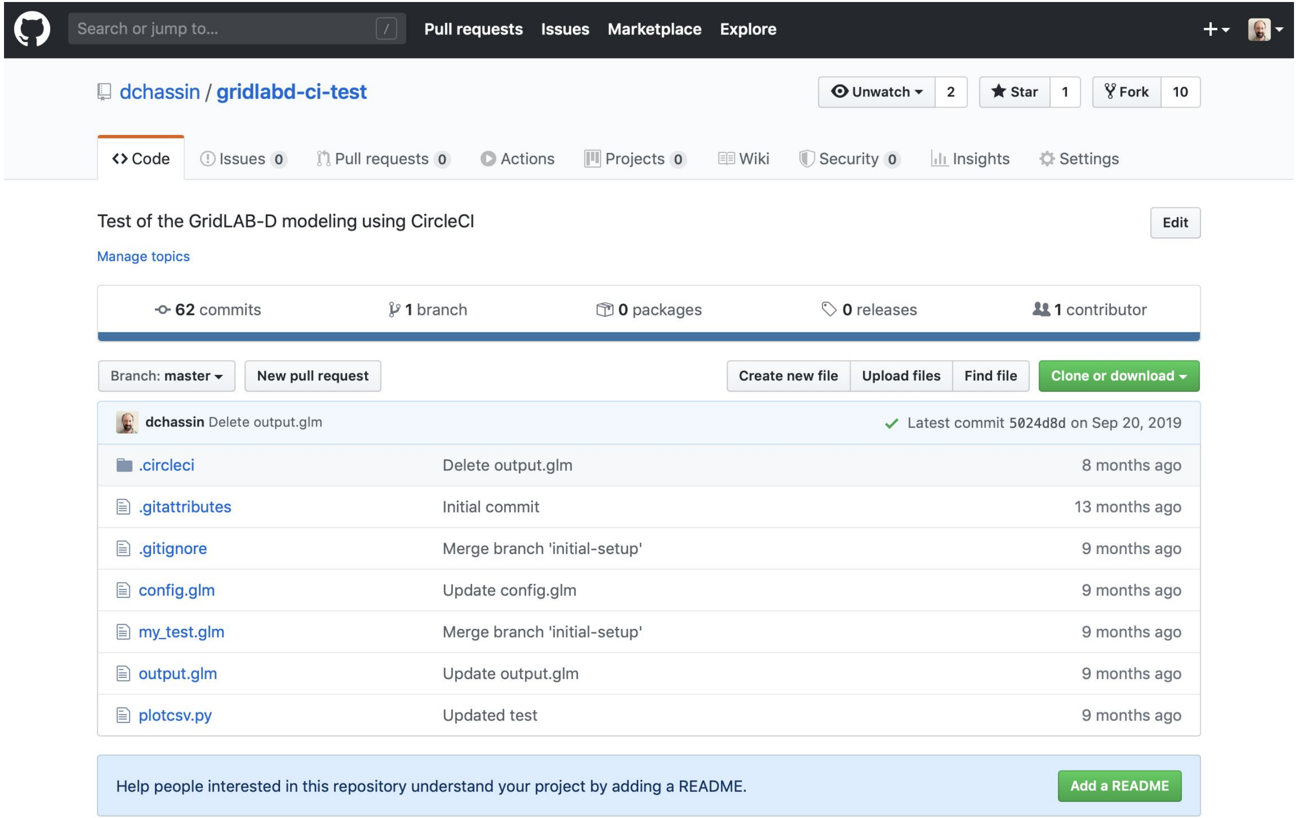
Docker containers maintained/updated automatically

- Docker hub repository located at http://docker.gridlabd.us/

CircleCI free tier operations

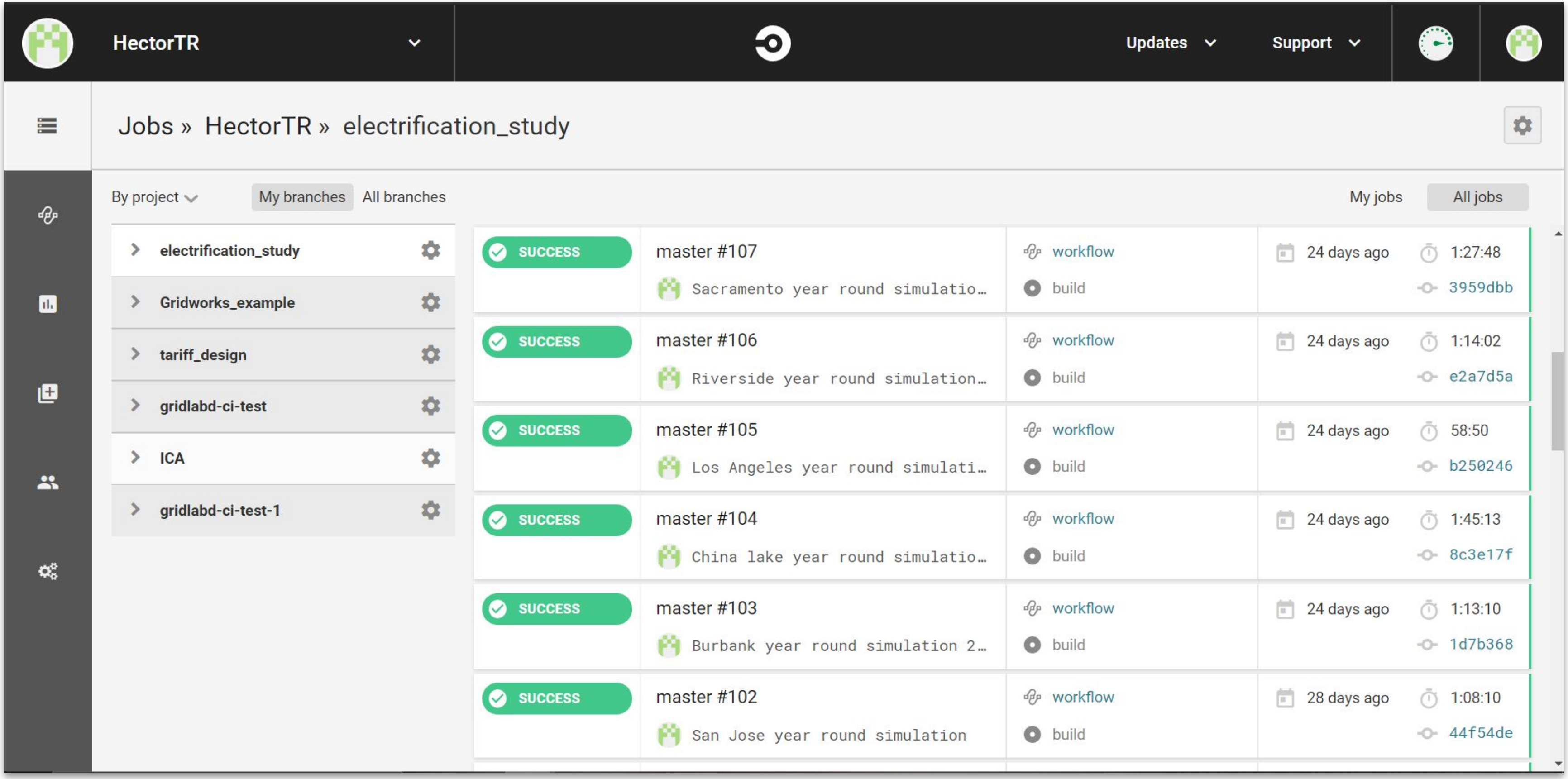
- GitHub projects can run HiPAS GridLAB-D on CircleCI for free
  - Latest master release: *slacgismo/gridlabd:latest*
  - Release candidate: *slacgismo/gridlabd:develop*
  - Tagged version: *slacgismo/gridlabd:beauharnois-03*

Cloud operations: GitHub projects with GridLAB-D



Cloud operations: Running GridLAB-D in CircleCI

Results: Electrification Study



Next Steps: Deployment and operations infrastructure

- **Restructuring of GridLAB-D resources in GitHub**
  - Weather data: http://weather.gridlabd.us/ (done)
  - Component libraries: http://library.gridlabd.us/ (in progress)
  - Project templates: http://template.gridlabd.us/ (coming soon)
  - Examples solution: http://example.gridlabd.us/ (coming soon)
  - Training videos: http://training.gridlabd.us/ (coming soon)
  - Tutorials videos: http://tutorials.gridlabd.us/ (coming soon)
  - Online documentation: http://docs.gridlabd.us/ (done)
  - Reference models: http://models.gridlabd.us/ (in progress)
  - Docker images: http://docker.gridlabd.us/ (done)
- **Security/scaling AWS CloudFront (https support)**

Highlights: Machine Learning Powerflow

Powerflow simulation in GridLAB-D

- 3-phase, unbalanced, quasi-steady power flow
- Finds voltages given power injections

Standard approach

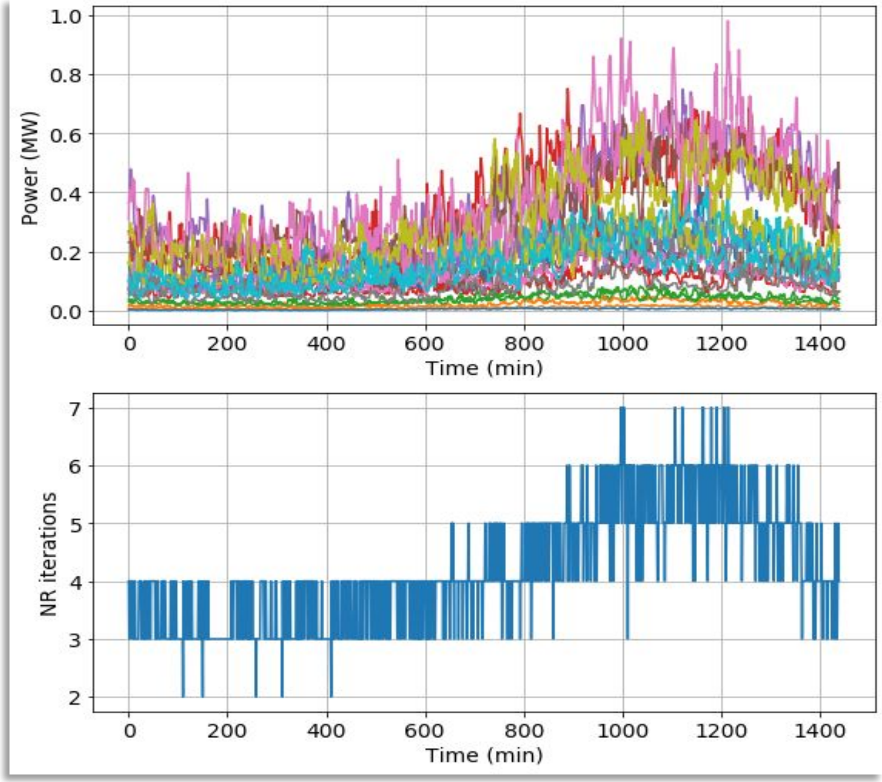
- Newton Raphson with the current injection method
- Computationally expensive for large networks

Machine learning approach

- Convolutional neural network (CNN)
- Training using previous NR solutions

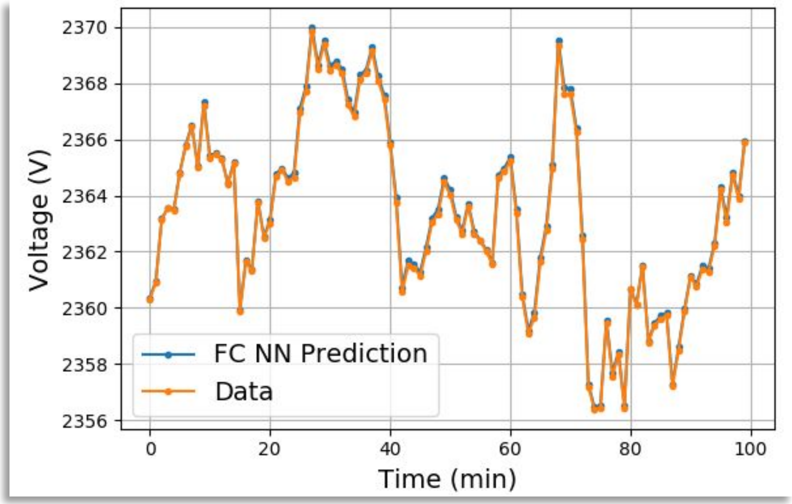
Solution performance and validation

- Training duration ~ 1 week of simulation
- Voltage error typically < 0.05%



Results: ML powerflow solver prediction accuracy

Voltage magnitude prediction for the IEEE 123 bus network (bus 92 phase A) using a fully-connected neural network.



Test errors for the convolutional neural network using *tanh* activation.

Power network	Test error
IEEE 4	1.0e-4
IEEE 13	1.7e-4
IEEE 123	9.9e-5
R1-12.47-3	2.6e-4
R2-12.47-2	4.8e-4

Notes

- Accuracy achieved by the convolutional model is suitable for many simulation applications
- Optimal hyperparameters can be estimated based on known network characteristics (e.g. size)

Next steps: Machine learning powerflow solver

Explore improved CNN architecture

- Network model contain useful topology information

Optimize training duration

- Automatically detect when training is done/required

Integrate ML solver into GridLAB-D solvers

- Need to know when to fall back to NR solver
- Opportunity to hot-start NR solver using ML result

Evaluate performance of deployed solver

- NR solver performance may be affected by ML solver

TAC Feedback

Are any new use cases emerging that we consider?

- PSPS impact analysis is an emerging use-case

What is your view on cloud operations?

- Utility comfort with cloud operations increasing steadily
- Enable support of multiple cloud vendors

Thank You

Contact: dchassin@slac.stanford.edu

ML Powerflow: Case Studies

Populate grid models with residential load data

- Individual house end-use metering dataset

Generate training, validation, and test sets

- NR-based GridLAB-D powerflow solver input & output
- IEEE and PNNL taxonomy feeders

Analyze various loading conditions

- Power injection magnitude
- Power factor
- Load composition (voltage dependency)

Power network models vary in terms of

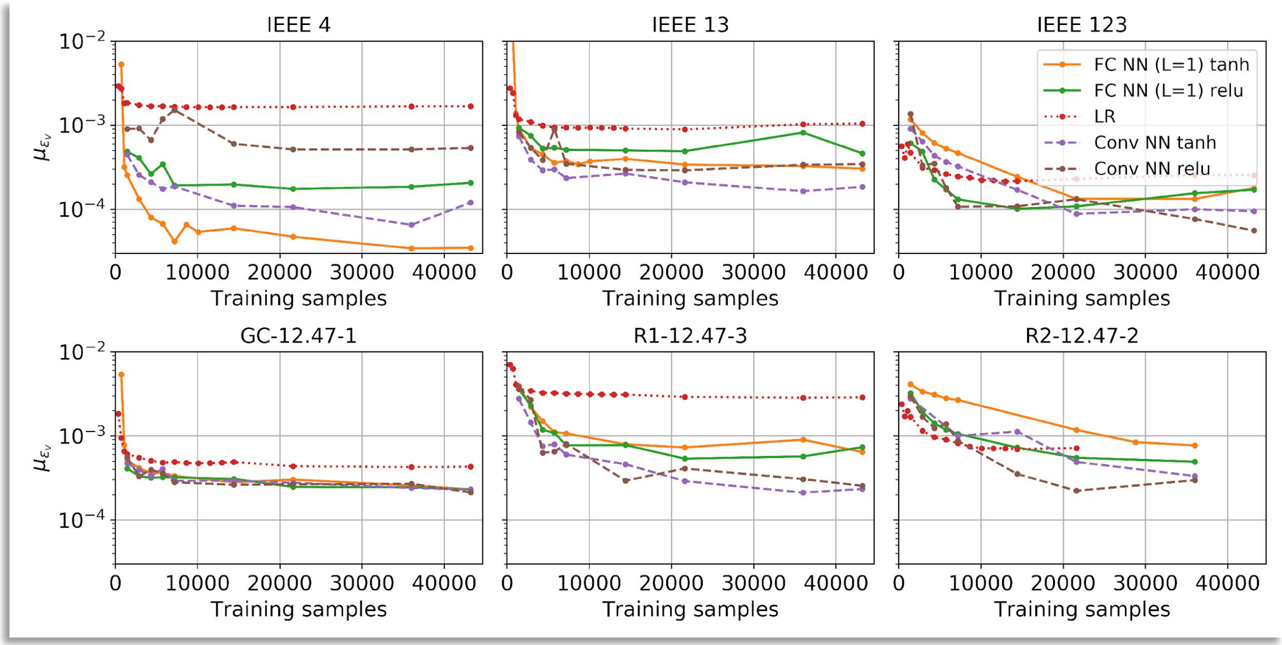
- Input and output dimension
- Linearity

Power network dimensions

Network	Voltage dimension	Power dimension
IEEE 4	12	3
IEEE 13	48	22
IEEE 123	402	95
GC-12.47-1	108	9
R1-12.47-3	297	37
R2-12.47-2	2553	214

ML Powerflow: Performance Results

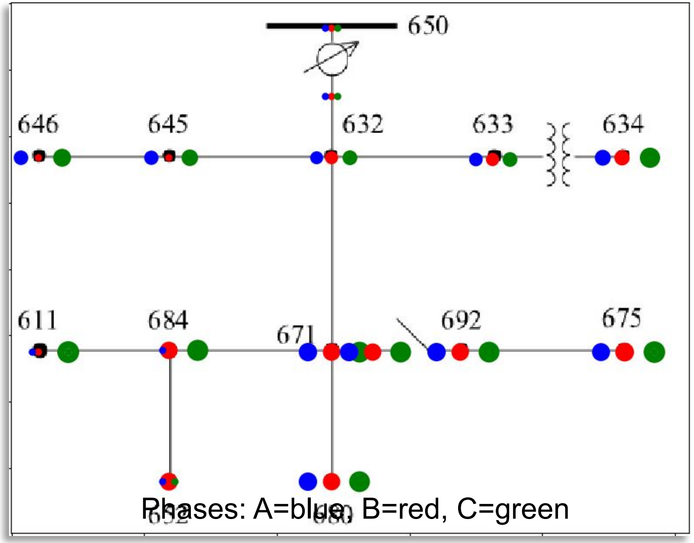
Voltage magnitude errors



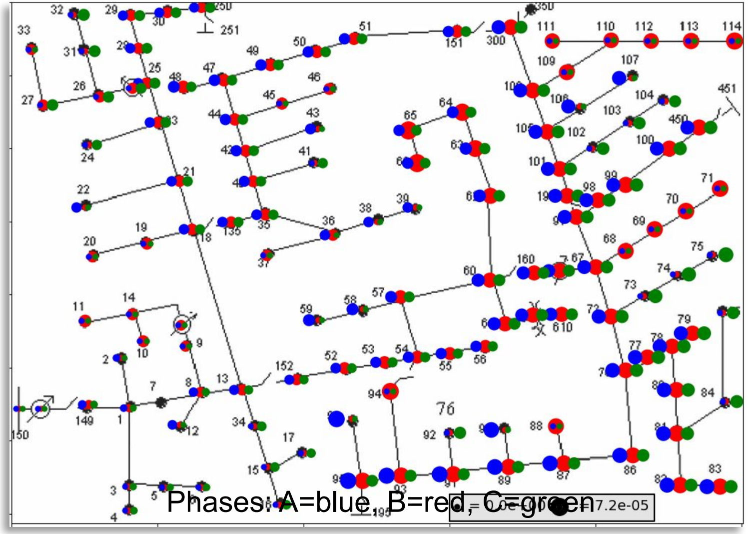
- The convolution model outperforms other models for most networks and is more scalable with network size
- Training data requirements depend on both the network size, characteristics, and how much the network is loaded
- Tanh NN activation generally results in the best and most consistent performance

ML Powerflow: Spatial Dependency of Errors

IEEE 13 bus network



IEEE 123 bus network



- Marker size indicates prediction error magnitude
- Voltage magnitude prediction errors increase further from the slack bus where there is larger deviation from the nominal voltage

Results – Computational Cost

Linear Regression

- Fast training time (<15 s on personal laptop)
- Small training set requirements (low memory requirements)

Neural network approaches

- Training time is 1-3 orders of magnitude larger than linear regression
  - Additional computational gains may be achieved using improved optimization and weight initialization methods
- Larger training set requirements (larger memory requirements) compared to linear regression
  - Training set size could potentially be minimized through better design of the training set (the sampling of power injections)

Linear regression training time (using a laptop Intel Core i7 processor)