Predicting the Performance of a Computing System with Deep Networks

Mehmet Cengiz – m.cengiz2@newcastle.ac.uk

Matthew Forshaw – matthew.forshaw@newcastle.ac.uk

Amir Atapour-Abarghouei – amir.atapour-abarghouei@durham.ac.uk

Andrew Stephen McGough – stephen.mcgough@newcastle.ac.uk

14th ACM/SPEC International Conference on Performance Engineering (ICPE) / 18.04.2023

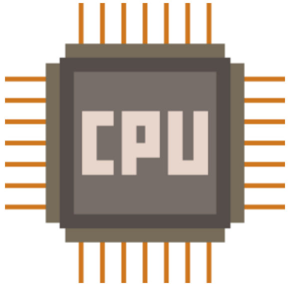
Outline

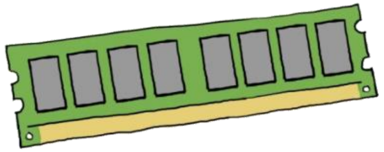
- Problem and approach
- The data
- Data preparation
- Deep Learning models
- Results
- Conclusions

Predicting SPEC CPU 2017 scores for new computers

SPEC CPU 2017 score

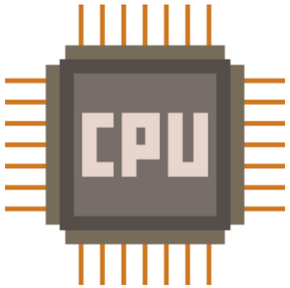
Predicting SPEC CPU 2017 scores for new computers

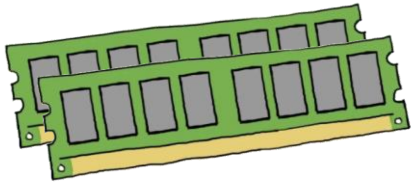




SPEC CPU 2017 score

Predicting SPEC CPU 2017 scores for new computers

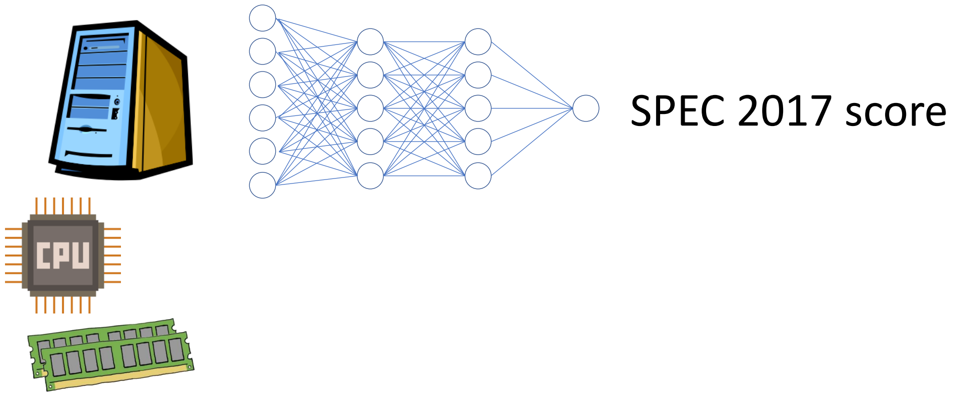




SPEC CPU 2017 score

Filling in the gaps with Machine Learning

- Benchmarking systems is costly
  - Time to conduct tests
  - Financial (hardware + software)
- Machine Learning is promising alternative to building and testing
  - Especially Deep Learning
- We demonstrate the potential of deep learning for predicting performance
  - using Multi-layer Perceptrons and Convolutional Neural Networks



Outline

- Problem and approach
- The data
- Data preparation
- Deep Learning models
- Results
- Conclusions

Features in SPEC CPU 2017 dataset

Data Type	Column
String	Benchmark, Hardware Vendor, System, Processor, CPU(s) Orderable, 1st Level Cache, 2nd Level Cache, 3rd Level Cache, Other Cache, Storage, Operating System, File System, Compiler, License, Tested By, Test Sponsor
Numerical	Peak Result, Base Result, Energy Peak Result, Energy Base Result, # Cores, # Chips, Memory, # Enabled Threads Per Core, Processor MHz
Binary	Parallel
Ternary	Base Pointer Size
Quaternary	Peak Pointer Size
Date (mon-yyyy)	HW Avail, SW Avail, Test Date, Published, Updated
Text	Disclosures

34 attributes / features

Features in SPEC CPU 2017 dataset

Data Type	Column
String	Benchmark, Hardware Vendor, System, Processor, CPU(s) Orderable, 1st Level Cache, 2nd Level Cache, 3rd Level Cache, Other Cache, Storage, Operating System, File System, Compiler, License, Tested By, Test Sponsor
Numerical	Peak Result, Base Result, Energy Peak Result, Energy Base Result, # Cores, # Chips, Memory, # Enabled Threads Per Core, Processor MHz
Binary	Parallel
Ternary	Base Pointer Size
Quaternary	Peak Pointer Size
Date (mon-yyyy)	HW Avail, SW Avail, Test Date, Published, Updated
Text	Disclosures

34 attributes / features

Best result with optimization

Features in SPEC CPU 2017 dataset

Data Type	Column
String	Benchmark, Hardware Vendor, System, Processor, CPU(s) Orderable, 1st Level Cache, 2nd Level Cache, 3rd Level Cache, Other Cache, Storage, Operating System, File System, Compiler, License, Tested By, Test Sponsor
Numerical	Peak Result, Base Result, Energy Peak Result, Energy Base Result, # Cores, # Chips, Memory, # Enabled Threads Per Core, Processor MHz
Binary	Parallel
Ternary	Base Pointer Size
Quaternary	Peak Pointer Size
Date (mon-yyyy)	HW Avail, SW Avail, Test Date, Published, Updated
Text	Disclosures

34 attributes / features

Best result with optimization

Result with no optimization

Features in SPEC CPU 2017 dataset

Data Type	Column
String	Benchmark, Hardware Vendor, System, Processor, CPU(s) Orderable, 1st Level Cache, 2nd Level Cache, 3rd Level Cache, Other Cache, Storage, Operating System, File System, Compiler, License, Tested By, Test Sponsor
Numerical	Peak Result, Base Result, Energy Peak Result, Energy Base Result, # Cores, # Chips, Memory, # Enabled Threads Per Core, Processor MHz
Binary	Parallel
Ternary	Base Pointer Size
Quaternary	Peak Pointer Size
Date (mon-yyyy)	HW Avail, SW Avail, Test Date, Published, Updated
Text	Disclosures

34 attributes / features

Best result with optimization

Result with no optimization

What we'll predict

SPEC CPU 2017 Data example

Benchmark = 'CINT2017',

Hardware Vendor = 'ASUSTeK Computer Inc.',

System = 'ASUS ESC4000A-E10(KRPG-U8) Server System 2.60 GHz, AMD EPYC 7H12',

Peak Result = 9.09,

Base Result = 8.87,

Energy Peak Result = 0.0,

Energy Base Result = 0.0,

Cores = 64,

Chips = 1,

Enabled Threads Per Core = 2,

Processor = 'AMD EPYC 7H12'

Processor MHz = 2600

CPU(s) Orderable = '1 chip',

Parallel = 'Yes',

Base Pointer Size = '64-bit',

Peak Pointer Size = '32/64-bit',

1st Level Cache = '32 KB I + 32 KB D on chip per core',

2nd Level Cache = '512 KB I+D on chip per core',

3rd Level Cache = '256 MB I+D on chip per chip, 16 MB shared / 4 cores',

Other Cache = 'None',

Memory = '512 GB (8 x 64 GB 2Rx4 PC4-3200AA-R)',

Storage = '1 x 480 GB SATA SSD',

Operating System = 'Ubuntu 19.04 (x86_64), Kernel 5.0.0-20-generic',

File System = 'ext4',

Compiler = 'C/C++/Fortran: Version 2.0.0 of AOCC',

HW Avail = 'Jul-2020',

SW Avail = 'Jun-2019',

License = 9016,

Tested By = 'ASUSTeK Computer Inc.',

Test Sponsor = 'ASUSTeK Computer Inc.',

Test Date = 'Jun-2020',

Published = 'Jul-2020',

Updated = 'Jul-2020',

Disclosures = 'HTML CSV PDF PS Text Config'

Outline

- Problem and approach
- The data
- Data preparation
- Deep Learning models
- Results
- Conclusions

Cleaning the data

- Data needs to be very ‘clean’
- ‘1024MB’, ‘1GB’ – convert to same units
- ‘1 CPU’, ‘1 cpu’ – convert to same case
- Base result = ‘0’ – removal of outliers
- ‘1GB’, ‘ 1 GB’, ‘1 GB’ – removal of spurious spaces
- ’1GB’, ‘2GB’, ‘4GB’ – make categorical

- Our reproducibility package contributes code to clean the SPEC CPU 2017 data to support further analyses.

Removal of highly correlated features

- Highly correlated features don't help with producing better results
- And sometimes make things worse
- Kendall's rank correlation used to identify those features > 70% corelated with others
- 7 features removed

Pearson and Spearman gave very similar results

Outline

- Problem and approach
- The data
- Data preparation
- Deep Learning models
- Results
- Conclusions

Challenges

- Choosing the best Neural Network isn’t trivial
- Shape of the network
  - Layers and width
- Types of ‘neurons’
- Activation functions
- Loss function
- Optimizers
- Stride size

Challenges

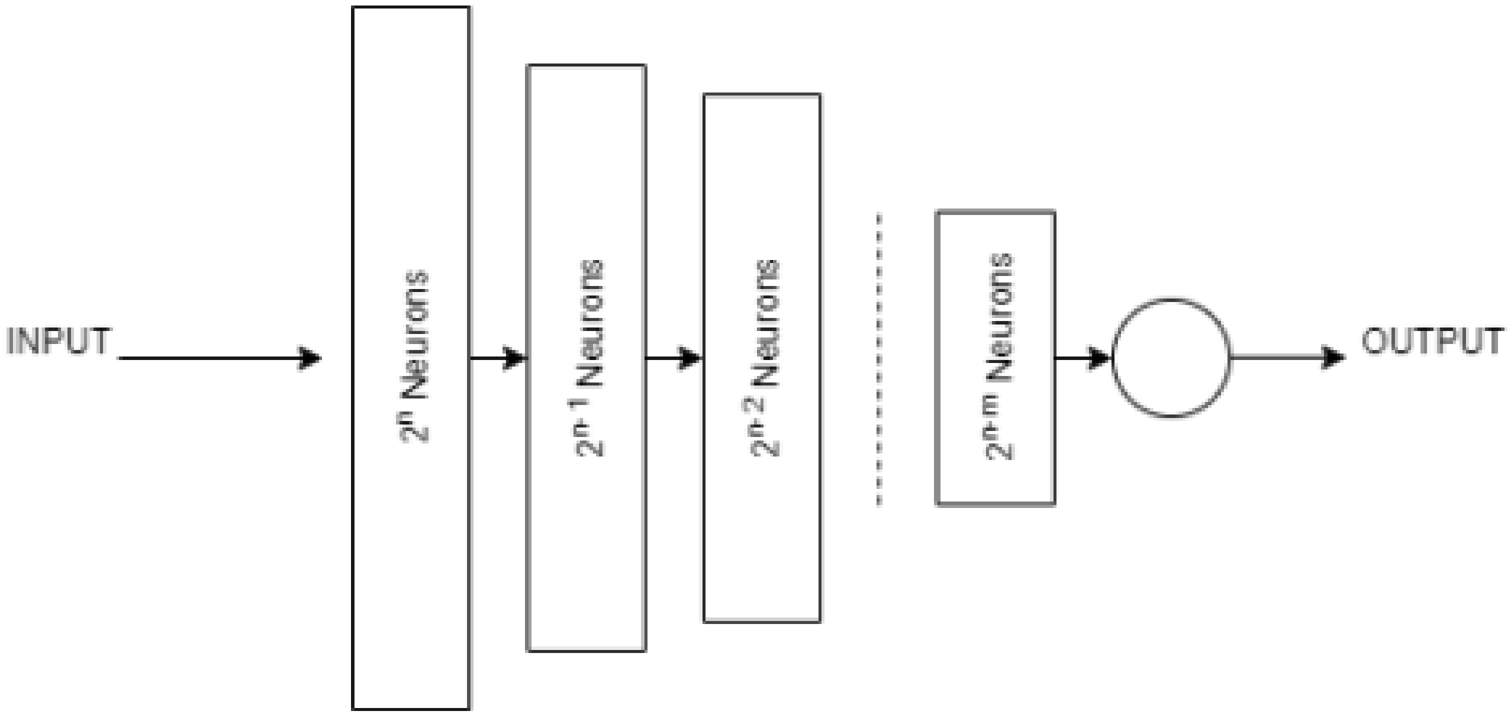
- Choosing the best Neural Network isn't trivial
- Shape of the network
  - Layers and width
- Types of 'neurons'
- Activation functions
- Loss function
- Optimizers
- Stride size
- Epochs

Neural architecture search space

Hyperparameter search space

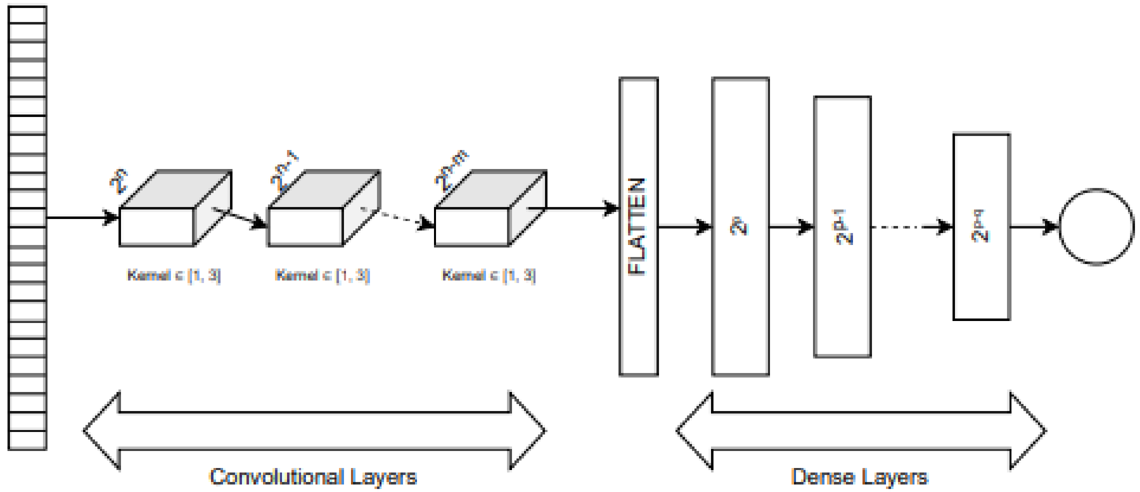
Searching for Neural Network (MLP)

- Fully-Connected Networks: trapezium shaped
  - Number of neurons: From 2^n to 2^{n-m}
    - Range = $n \in [4, \ldots, 11]$, $m \in [1, \ldots, 10]$



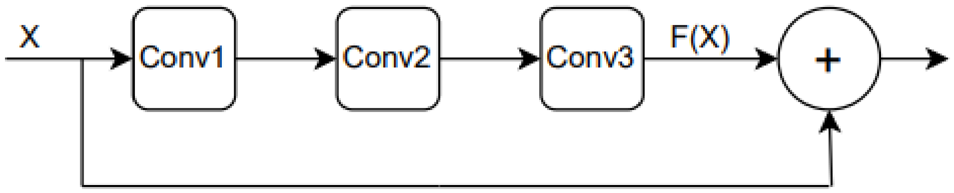
Searching for Neural Network cont. (CNN)

- CNN design: trapezium shaped
  - Number of convolutional layers: From 2^n to 2^{n-m}
    - Range = $n \in [7, \ldots, 11]$, $m \in [4, \ldots, 7]$
    - Kernel $\in [1, 3]$
  - Number of neurons: From 2^p to 2^{p-q}
    - Range = $p \in [7, \ldots, 11]$, $q \in [5, \ldots, 7]$

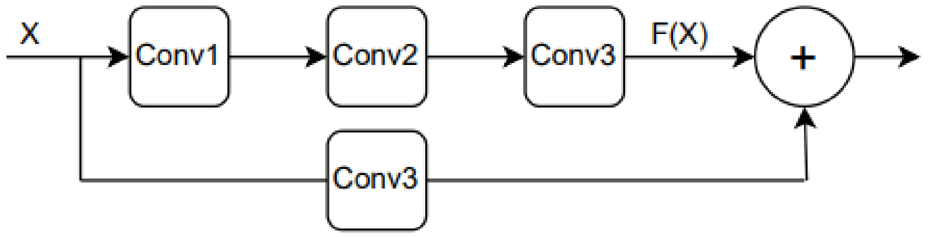


Searching for Neural Network cont. (ResNet Inspired)

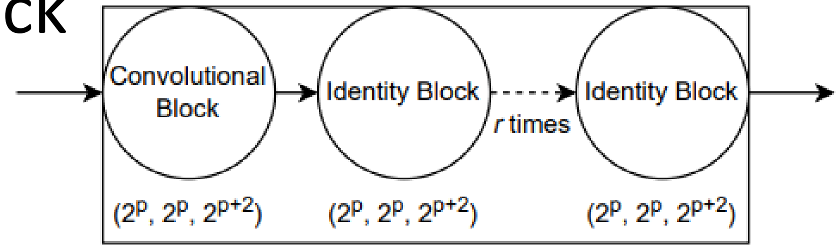
- Identity block



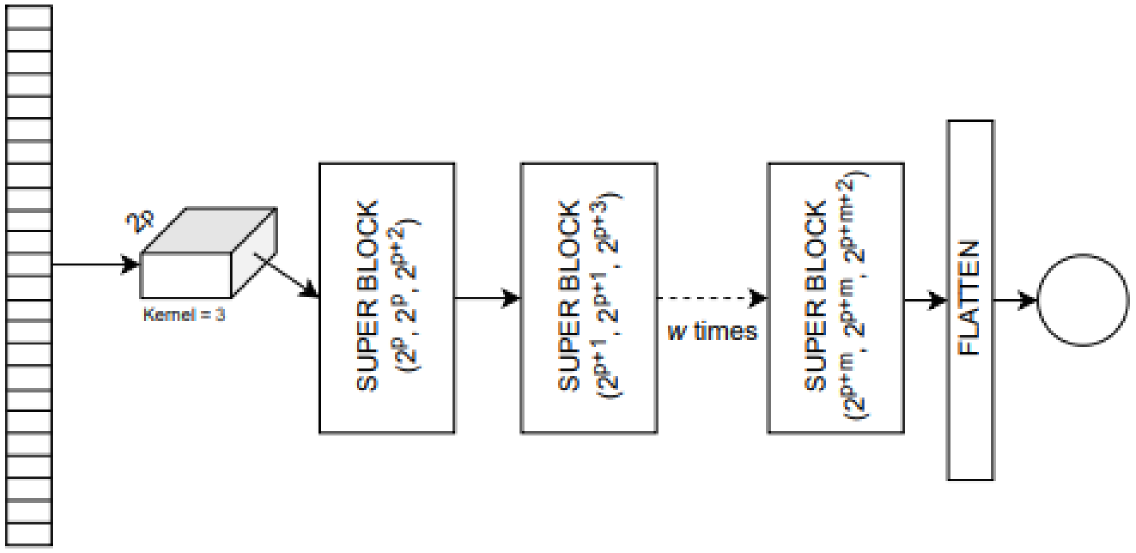
- Convolutional block



- Super block



- Final architecture



Hyperparameter search

- Optimizers: SGD, Adam, Rmsprop
- Loss functions: MAE and MSE
- Activation functions: sigmoid, tanh, ReLU
- Stride size $\in [1, \ldots, 4]$

Outline

- Problem and approach
- The data
- Data preparation
- Deep Learning models
- **Results**
- Conclusions

Metrics

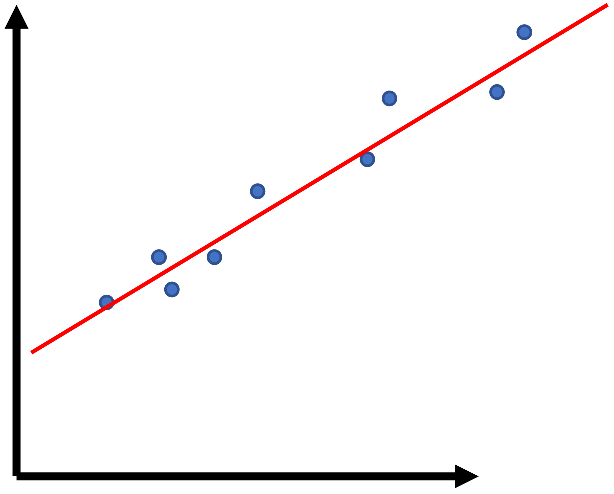
- R^2: strength of the relationship between predictions and actual
  - Closer to 1 is better
- MAE: how big error is between predicted and actual
  - Closer to 0 is better
- MSE: Similar to MAE but more impact from large differences
  - Closer to 0 is better

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - y_i')^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i' - y| \quad MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i' - y_i)^2$$

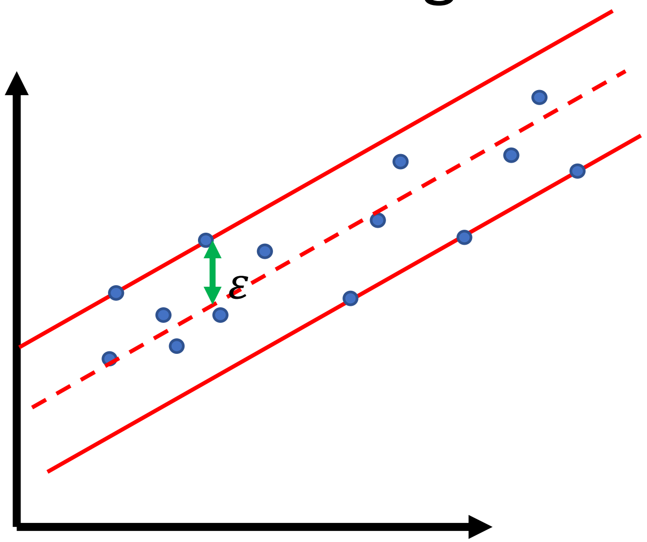
y_i is the true value, y_i' is the predicted value and $\bar{y}_i$ is the mean of all true values

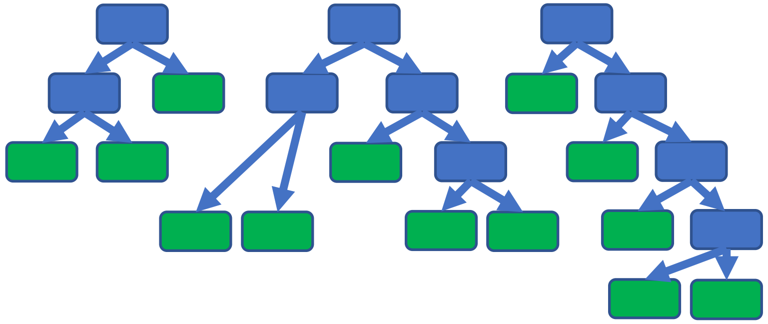
Baseline comparison methods

- Linear Regression



- Support Vector Regression
- Random Forest Regression





Overall Comparison – sorter by R^2

#	Architecture	Loss Fn	Kernel Sizes	Stride Sizes	Number of Filters (m, n)	Neurons in Layers (p, q)	Optimizer	Epochs	R2	MAE	MSE
1	TriCNN	MAE	3	1	(9, 7)	[9, …, 5]	Adam	250	0.98638701	5.67389728	465.3285655
2	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	Adam	250	0.98590661	5.83946465	476.0394343
3	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	Adam	300	0.98579341	5.76197731	494.124225
4	TriCNN	MAE	3	1	(9, 7)	[9, …, 5]	Adam	150	0.98529142	6.25318407	513.9629513
5	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	RmsProp	150	0.98282719	7.14056732	620.2982421
6	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	Adam	200	0.98280914	6.03564805	582.3068145
7	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	Adam	300	0.98278342	5.61076184	582.0247239
8	TriCNN	MAE	3	1	(9, 7)	[9, …, 5]	Adam	300	0.98107176	5.78137347	645.4129883
9	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	RmsProp	250	0.98095925	6.72097815	669.8856237
10	TriCNN	MAE	3	1	(9, 7)	[9, …, 5]	Adam	200	0.98089907	6.32291809	665.1641919
11	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	Adam	150	0.98047251	6.71537772	663.7030719
12	TriCNN	MAE	3	1	(7, 6, 5, 4)	[9, …, 5]	RmsProp	300	0.98038864	6.9974749	653.5821786
~	RF								0.9803076	4.76701531	688.0001262
13	TriCNN	MAE	3	1	(7, 6, 5, 4)	[9, …, 5]	RmsProp	200	0.98002879	7.62788323	684.7595471
14	TriCNN	MAE	2	1	(9, 7)	[11, …, 6]	Adam	150	0.9793459	6.519971	703.0615545
15	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	Adam	100	0.97782539	8.23651529	754.5381605
16	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	Adam	100	0.97748578	7.30871799	757.4994833
17	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	Adam	150	0.97726148	6.65855022	772.0747562
18	TriCNN	MAE	3	1	(7, 6, 5, 4)	[9, …, 5]	RmsProp	250	0.97665471	7.86703389	775.8960386
19	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	RmsProp	250	0.97650919	7.97325412	852.3545636
20	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	RmsProp	300	0.97636563	6.91501173	816.7881606
45	TriMLP	MAE				[11, …, 6]	Adam	250	0.97347275	9.12443258	906.1439402
159	Residual	MAE	Number of Superblocks = (2, 5, 5, 2)		((6, 6, 8), (7, 7, 9), (8, 8, 10), (9, 9, 11))	1	RmsProp	250	0.95007233	10.595069	1006.134564
~	LR								0.52639158	82.4596122	15761.16107
~	SVR								-0.0045634	113.749207	33448.30886

* TriCNN = Trapezium-shaped CNN, RF = Random Forest Regression, TriMLP = Trapezium-shaped MPL, LR = Linear Regression, SVR = Support Vector Regression

Overall Comparison – sorted by MAE

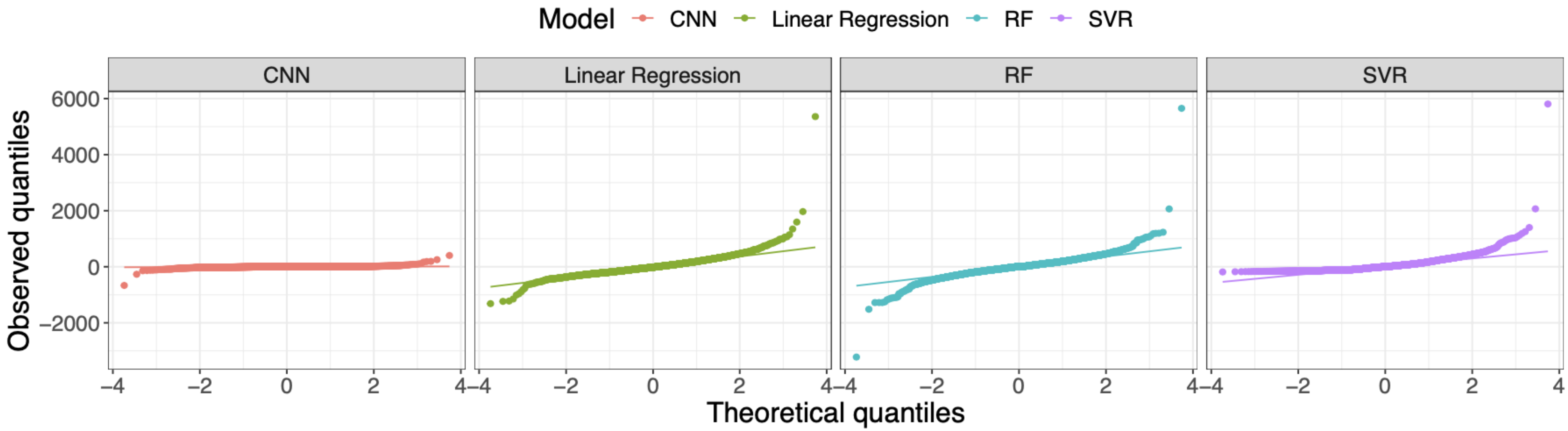
#	Architecture	Loss Fn	Kernel Sizes	Stride Sizes	Number of Filters (m, n)	Neurons in Layers (p, q)	Optimizer	Epochs	R2	MAE	MSE
1	TriCNN	MAE	3	1	(9, 7)	[9, …, 5]	Adam	250	0.98638701	5.67389728	465.3285655
2	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	Adam	250	0.98590661	5.83946465	476.0394343
3	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	Adam	300	0.98579341	5.76197731	494.124225
4	TriCNN	MAE	3	1	(9, 7)	[9, …, 5]	Adam	150	0.98529142	6.25318407	513.9629513
5	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	Adam	300	0.98278342	5.61076184	582.0247239
6	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	Adam	200	0.98280914	6.03564805	582.3068145
7	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	RmsProp	150	0.98282719	7.14056732	620.2982421
8	TriCNN	MAE	3	1	(9, 7)	[9, …, 5]	Adam	300	0.98107176	5.78137347	645.4129883
9	TriCNN	MAE	3	1	(7, 6, 5, 4)	[9, …, 5]	RmsProp	300	0.98038864	6.9974749	653.5821786
10	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	Adam	150	0.98047251	6.71537772	663.7030719
11	TriCNN	MAE	3	1	(9, 7)	[9, …, 5]	Adam	200	0.98089907	6.32291809	665.1641919
12	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	RmsProp	250	0.98095925	6.72097815	669.8856237
13	TriCNN	MAE	3	1	(7, 6, 5, 4)	[9, …, 5]	RmsProp	200	0.98002879	7.62788323	684.7595471
~	RF								0.9803076	4.76701531	688.0001262
14	TriCNN	MAE	2	1	(9, 7)	[11, …, 6]	Adam	150	0.9793459	6.519971	703.0615545
15	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	Adam	100	0.97782539	8.23651529	754.5381605
16	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	Adam	100	0.97748578	7.30871799	757.4994833
17	TriCNN	MAE	3	2	(9, 7, 6, 5, 4)	[9, …, 4]	Adam	150	0.97726148	6.65855022	772.0747562
18	TriCNN	MAE	3	1	(7, 6, 5, 4)	[9, …, 5]	RmsProp	250	0.97665471	7.86703389	775.8960386
19	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	RmsProp	200	0.97613855	7.72461632	807.1294185
20	TriCNN	MAE	3	2	(9, 7)	[9, …, 5]	RmsProp	300	0.97636563	6.91501173	816.7881606
48	TriMLP	MAE				[11, …, 6]	Adam	250	0.97347275	9.12443258	906.1439402
135	Residual	MAE	Number of Superblocks = (2, 5, 5, 2)		((6, 6, 8), (7, 7, 9), (8, 8, 10), (9, 9, 11))	1	RmsProp	250	0.95007233	10.595069	1006.134564
	LR								0.52639158	82.4596122	15761.16107
	SVR								-0.0045634	113.749207	33448.30886

* TriCNN = Trapezium-shaped CNN, RF = Random Forest Regression, TriMLP = Trapezium-shaped MPL, LR = Linear Regression, SVR = Support Vector Regression

Comparison of approaches

Model	Best R^2
Trapezium CNN	0.9864
Random Forest Regression	0.9830
Fully Connected MLP	0.9735
Residual Neural Network	0.9501
Linear Regression	0.5260
Support Vector Regression	-0.0040

How do we do across the range?

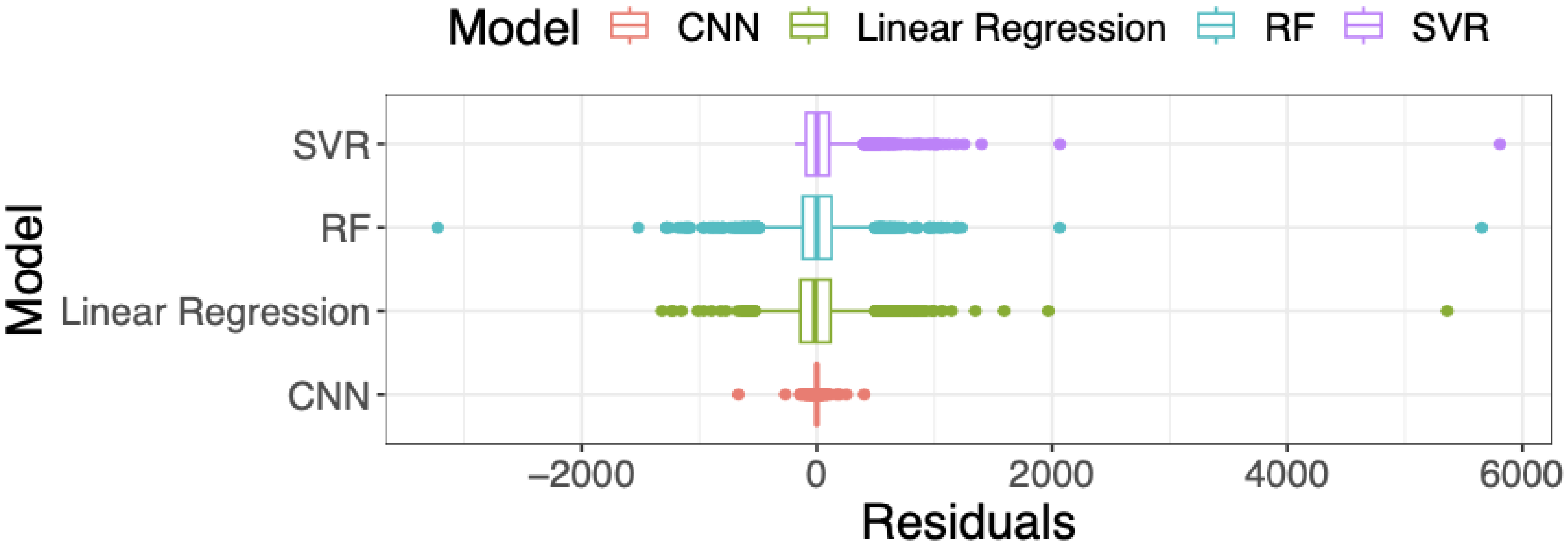


Hyperparameters

- Optimizer: Adam though RMSprop close
- Loss function: MAE, even when metric was MSE
- Activation function: Sigmoid
- Stride: 1 or 2
- Kernel size: normally 3
- Training epochs: Normally 250, though some exceptions

- Though model dependent

Residuals of different models



Threats to Validity

- Limitations:
  - L1: Single benchmark dataset
  - L2: Single expert for data cleaning
- Construct Validity
  - Could also look at predicting energy use
- Internal Validity
  - One researcher cleaned data, though well documented
- External Validity
  - Only done for SPEC 2017
- Reproducibility
  - Code and data is available

Implications

- Can provide more accurate predictions when we can’t do traditional benchmarking methods
- Helps organizations make better decisions when it comes to selecting hardware

Future Research Directions

- More powerful neural network architectures with innovative feature aggregating modules or higher parameter and layer counts could lead to even better performance predictions
- Transfer learning could be used to pre-train the performance prediction system on a larger proxy dataset before fine-tuning it on a benchmark dataset

Conclusions

- Deep learning models have the potential to revolutionize the way we understand computing system performance and make better decisions when it comes to selecting and optimizing hardware based on real-world workloads
- CNN Models produce the best results – though at cost of training time
- RF is close second – but less useful when predicting for novel hardware
- MLP and ResNet-inspired models perform reasonably well, but not as good as others
  - Not worth the extra cost
- Future research could explore more powerful neural network architectures and the effects of transfer learning to further improve performance predictions
- All code and data, available: https://github.com/cengizmehmet/BenchmarkNets

Long term: M.Cengiz2@newcastle.ac.uk
At Conference: Stephen.mcgough@Newcastle.ac.uk