Neural networks in julia

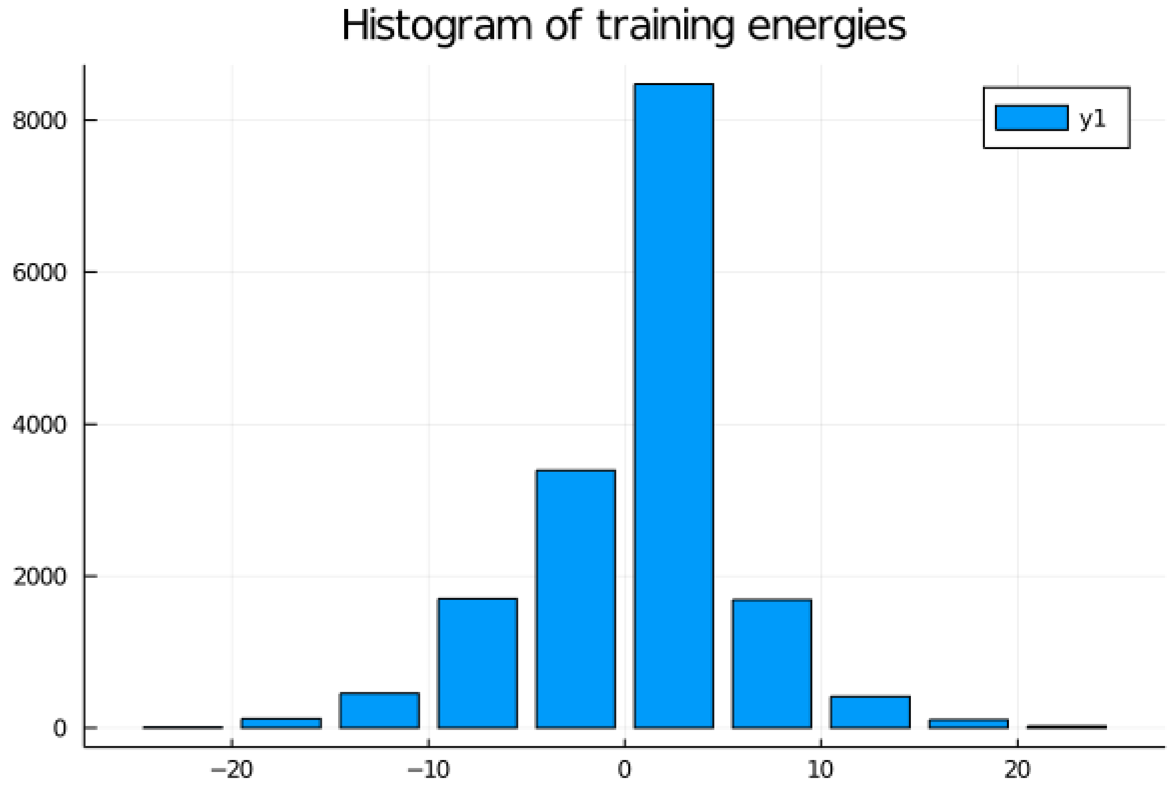
Simple neural net

There is a thing for nets in LaTeX

NN set up

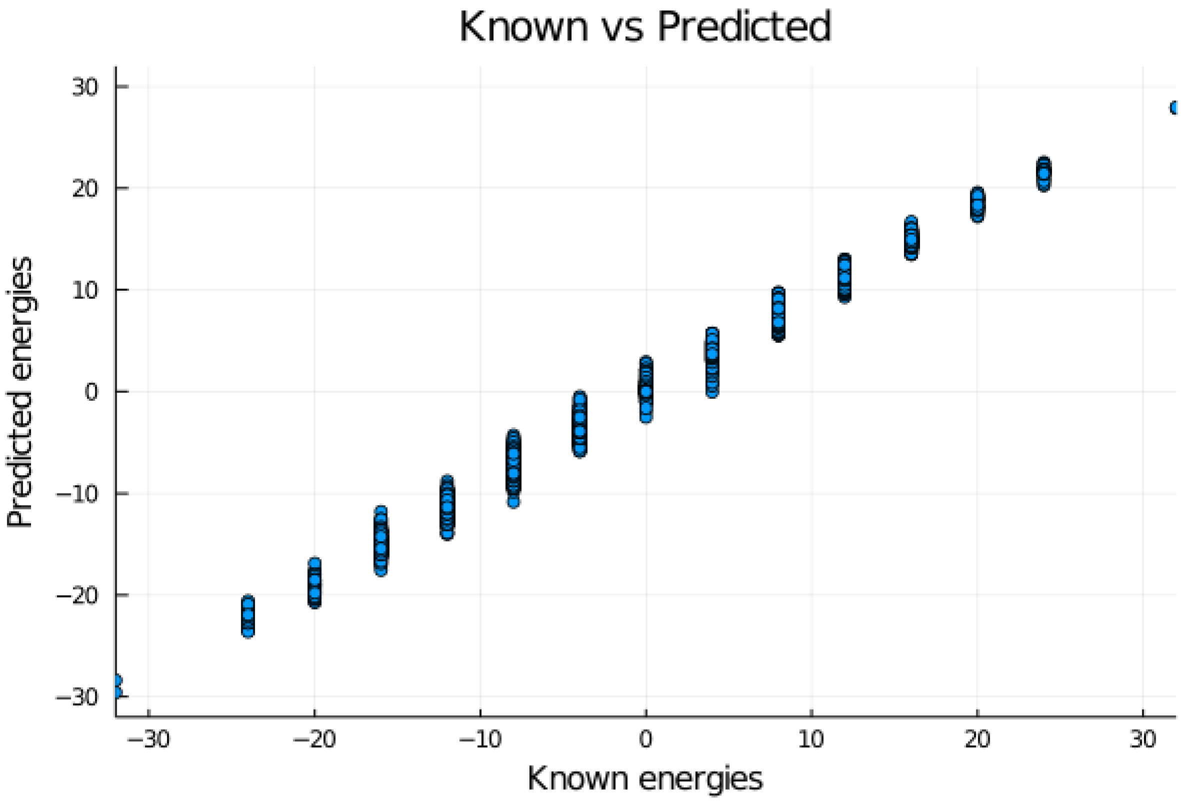
```
Dense(16,32,relu),
    Dense(32, 64, relu),
    Dense(64, 150),
    Dense(150,30),
    Dense(30, 1),
```

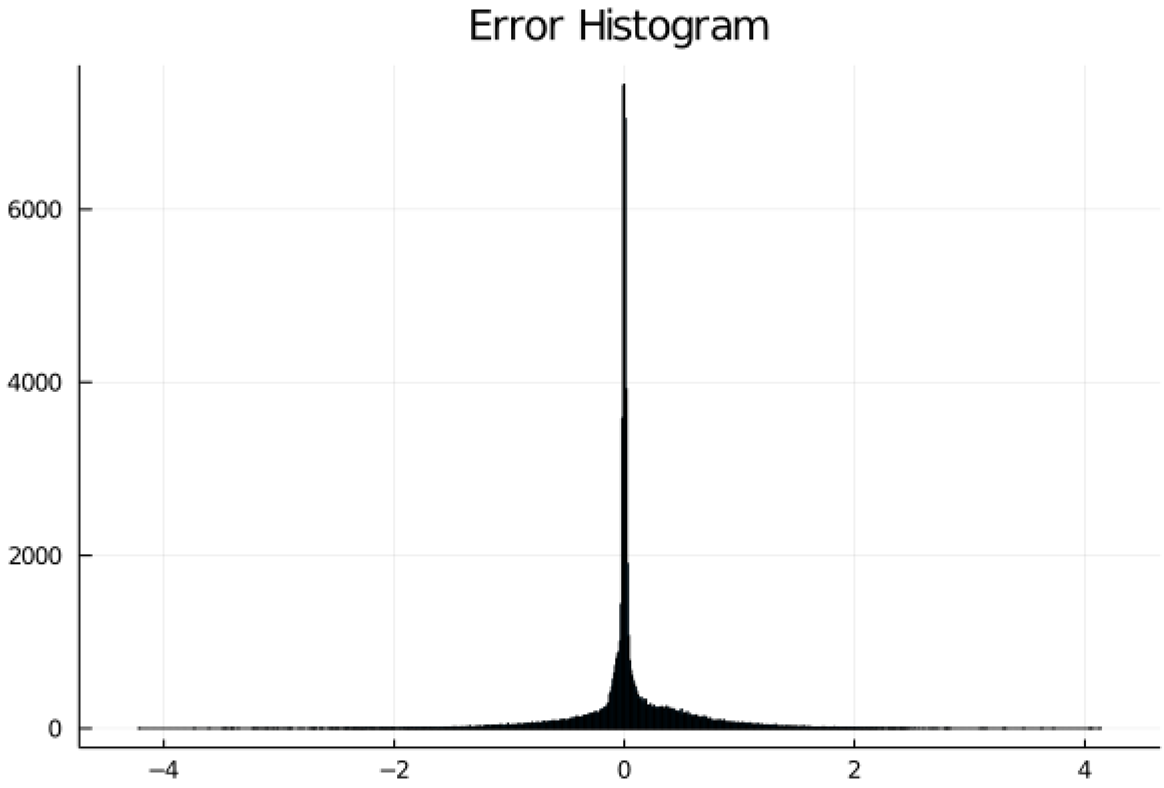
Trained of ~16000 randomly sellected exaples for 20 epochs on the same set

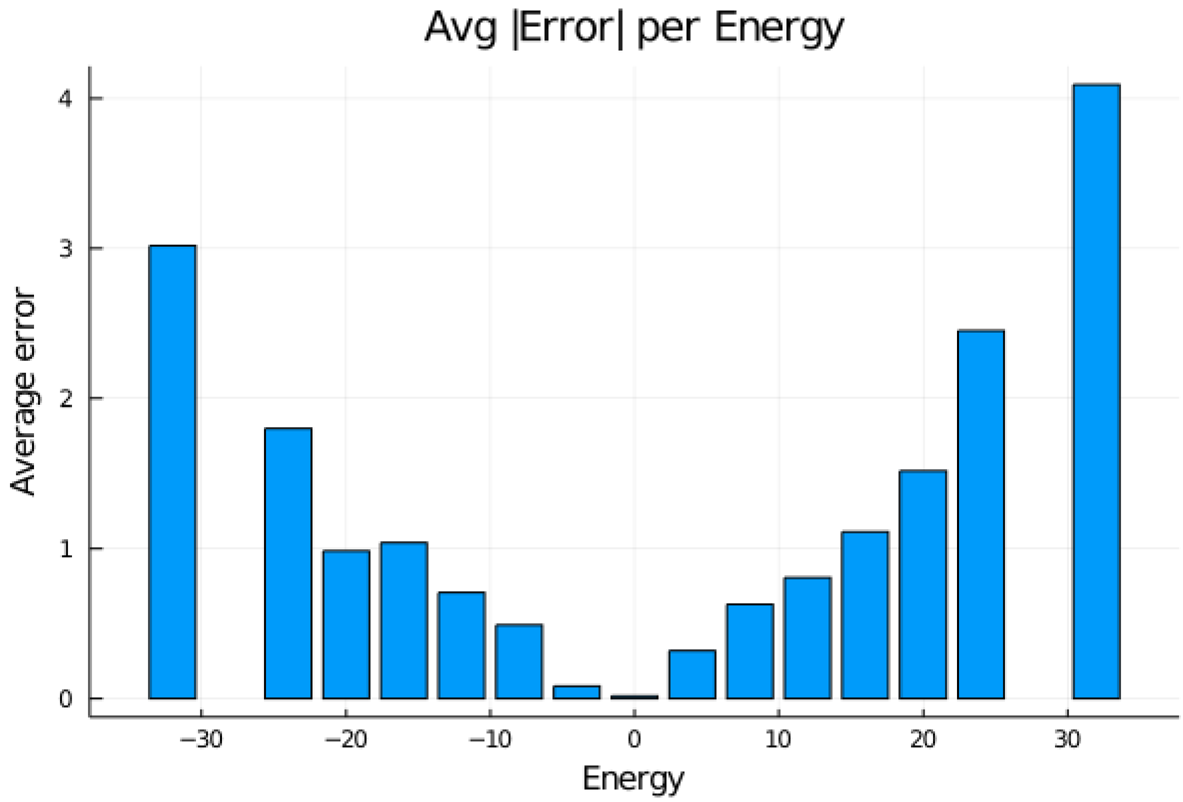


Results

Tested on all available data


Known vs Predicted
Predicted energies
30
20
10
0
−10
−20
−30
−30
−20
−10
0
10
20
30
Known energies


Error Histogram
6000
4000
2000
0
−4
−2
0
2
4

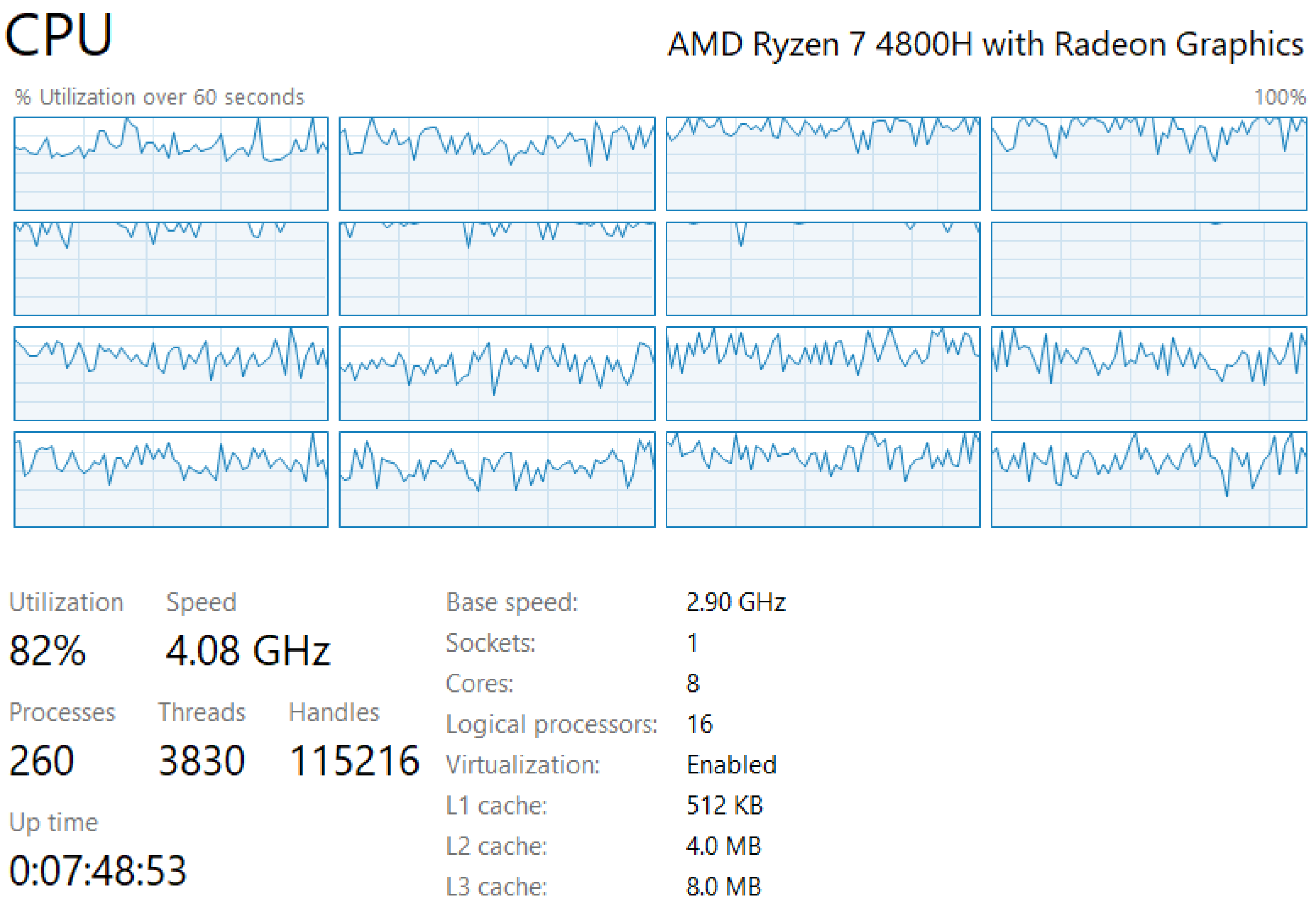


The lopsided error at extrema is explained by 32 or -32 not being included into the training set due to random selection

Noteworthy peculiarity when running

```
model = build_model() |> gpu
@epochs 20 Flux.train!(loss, ps, data, opt )
```

and gpu isn't set up cpu runns very fast i.e.



CPU

AMD Ryzen 7 4800H with Radeon Graphics

Utilization
82%

Speed
4.08 GHz

Processes
260

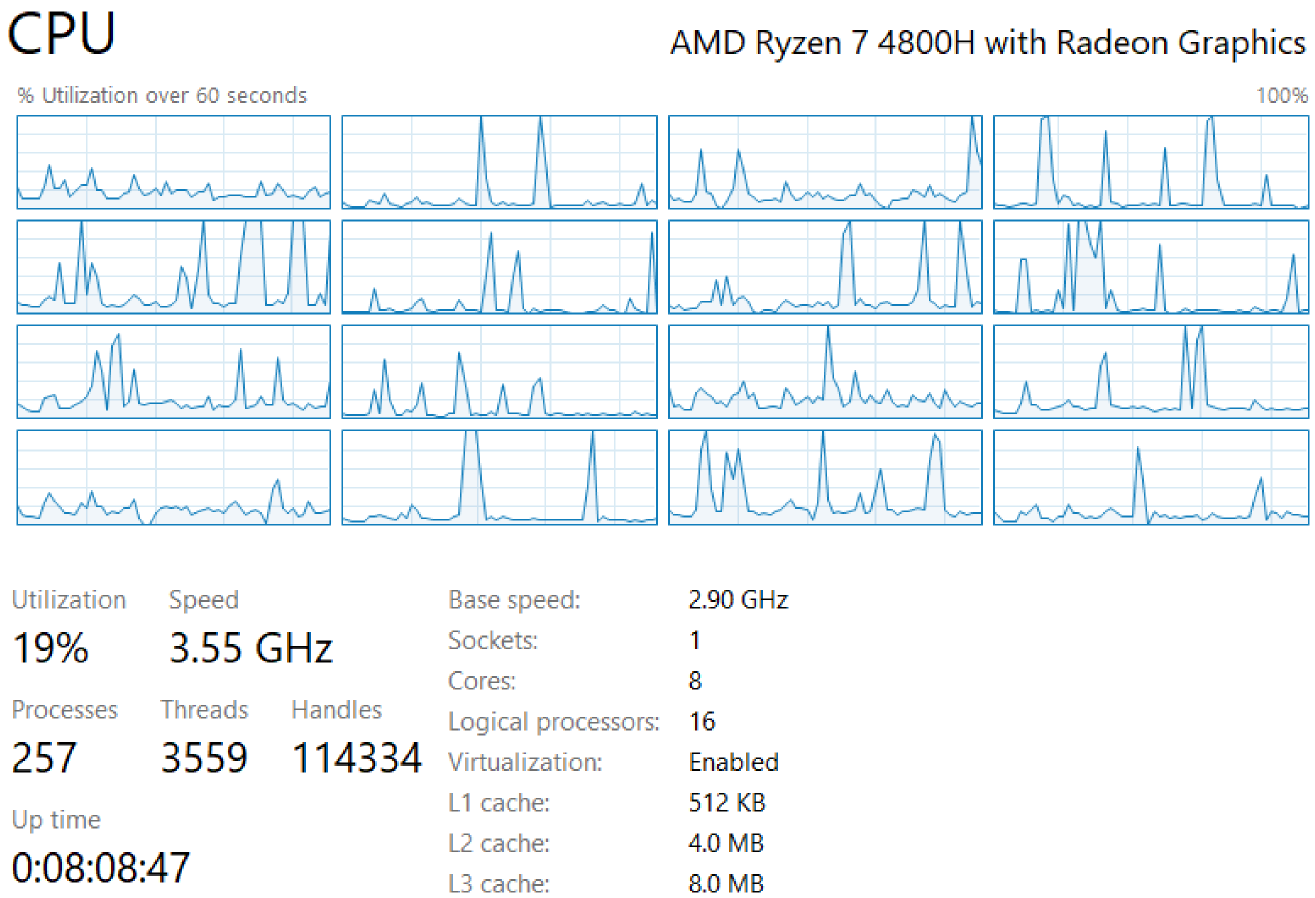
Threads
3830

Handles
115216

Up time
0:07:48:53

Base speed:	2.90 GHz
Sockets:	1
Cores:	8
Logical processors:	16
Virtualization:	Enabled
L1 cache:	512 KB
L2 cache:	4.0 MB
L3 cache:	8.0 MB

This doesn't happen when running a CNN even though the code is virtually the same



CPU
AMD Ryzen 7 4800H with Radeon Graphics
% Utilization over 60 seconds
100%

Utilization 19%
Speed 3.55 GHz
Processes 257
Threads 3559
Handles 114334
Up time 0:08:08:47

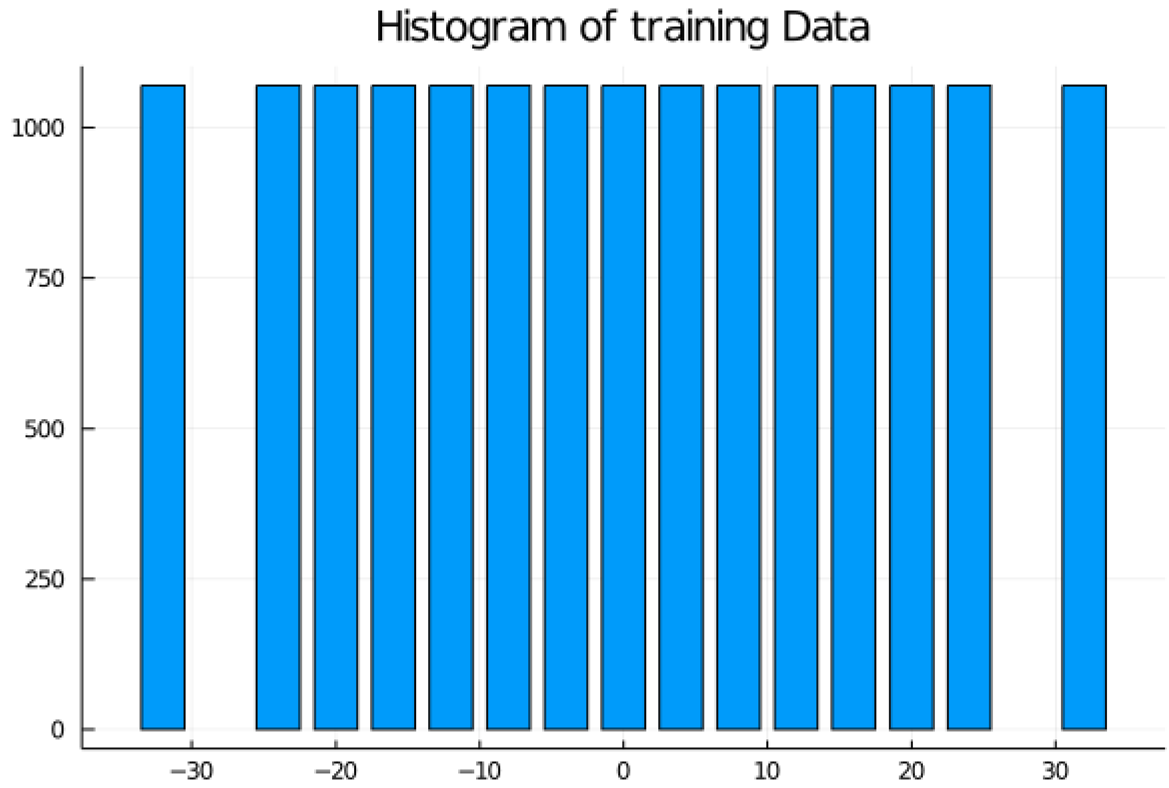
Base speed: 2.90 GHz
Sockets: 1
Cores: 8
Logical processors: 16
Virtualization: Enabled
L1 cache: 512 KB
L2 cache: 4.0 MB
L3 cache: 8.0 MB

CNN

```
    Conv((2,2),1=>5,relu),
    Conv((2,2), 5=>3,pad=(1,1), relu),
    Conv((2,2), 3=>3,pad=(1,1), relu),
    Conv((2,2), 3=>5,relu),
    Conv((2,2), 5=>3,pad=(1,1), relu),
    Conv((2,2), 3=>3,pad=(1,1), relu),
    Conv((2,2), 3=>5,relu),
    Flux.flatten,
    Dense(125, 1),
```

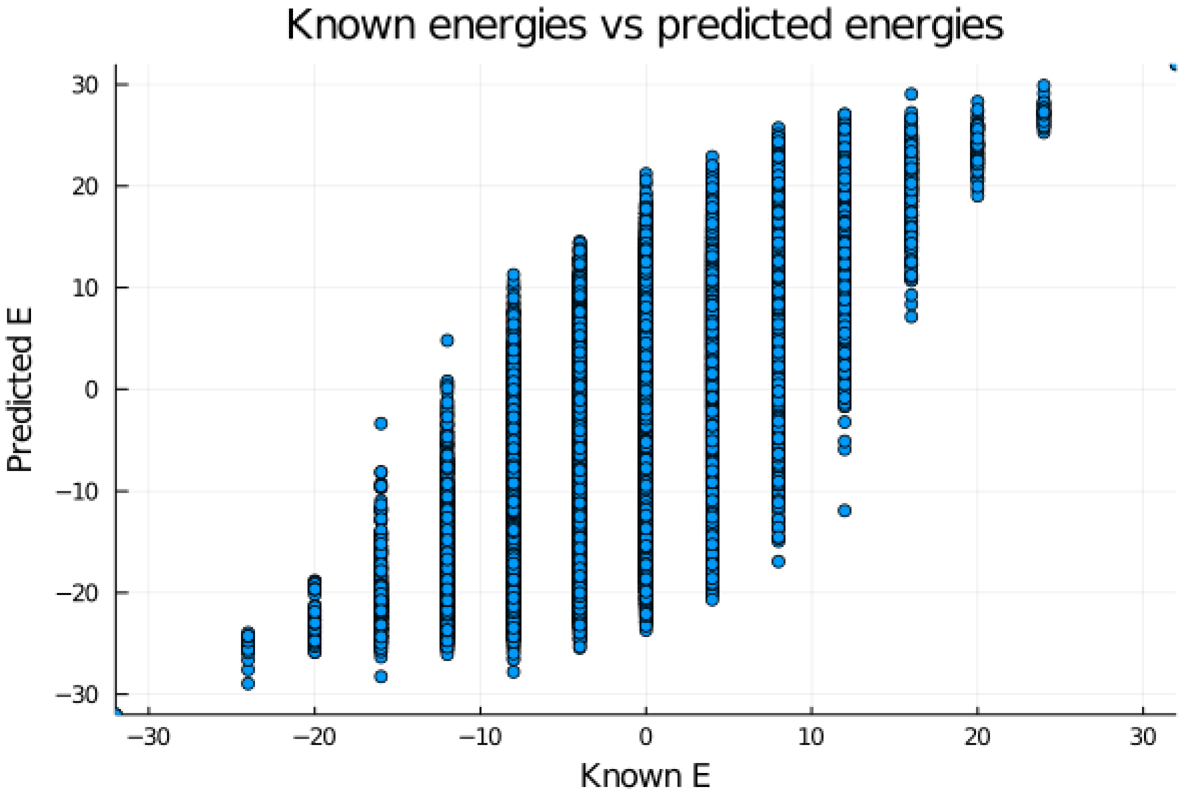
Trained on **Flat** energy distribution distribution of data

for 20 epochs

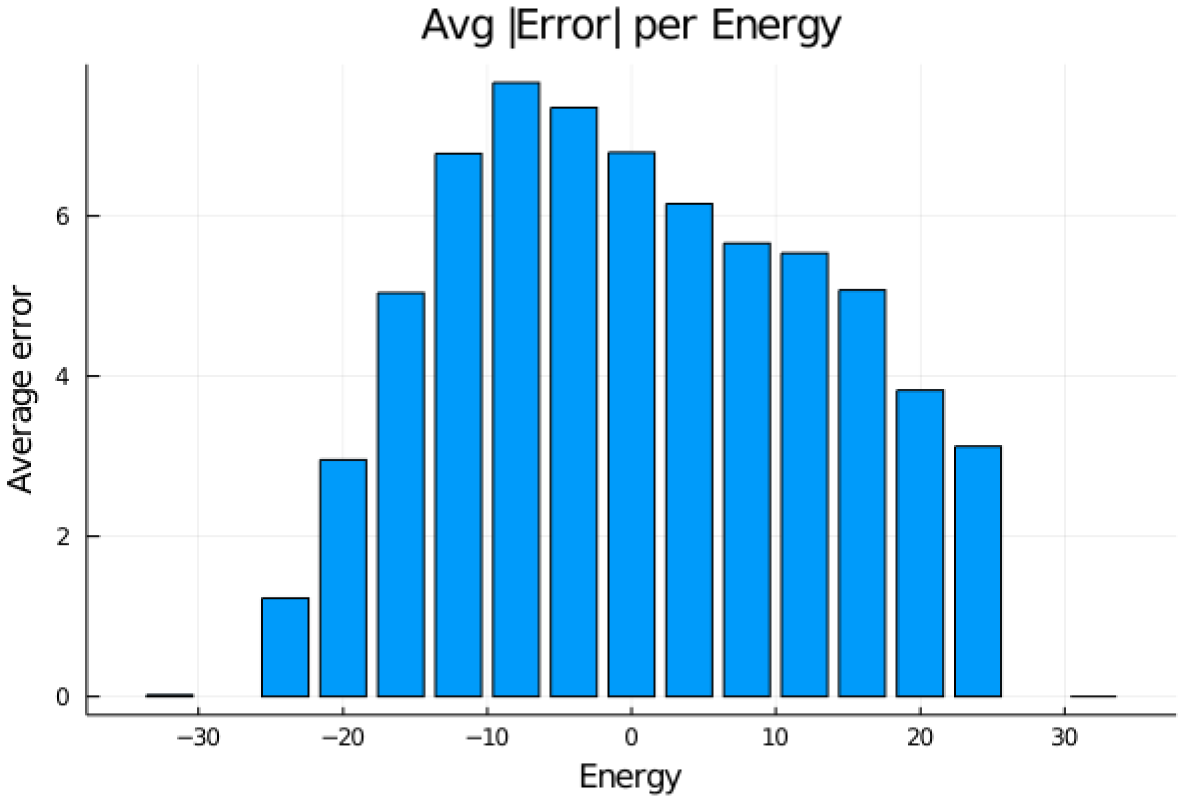


When evaluated over all data

the results are much worse



There is clearly less error on the boundareis where there were a lot of examples 32 and -32



Then repeated over **Random** data distribution

Finally, NN on random vs NN on random vs CNN on random vs CNN on flat