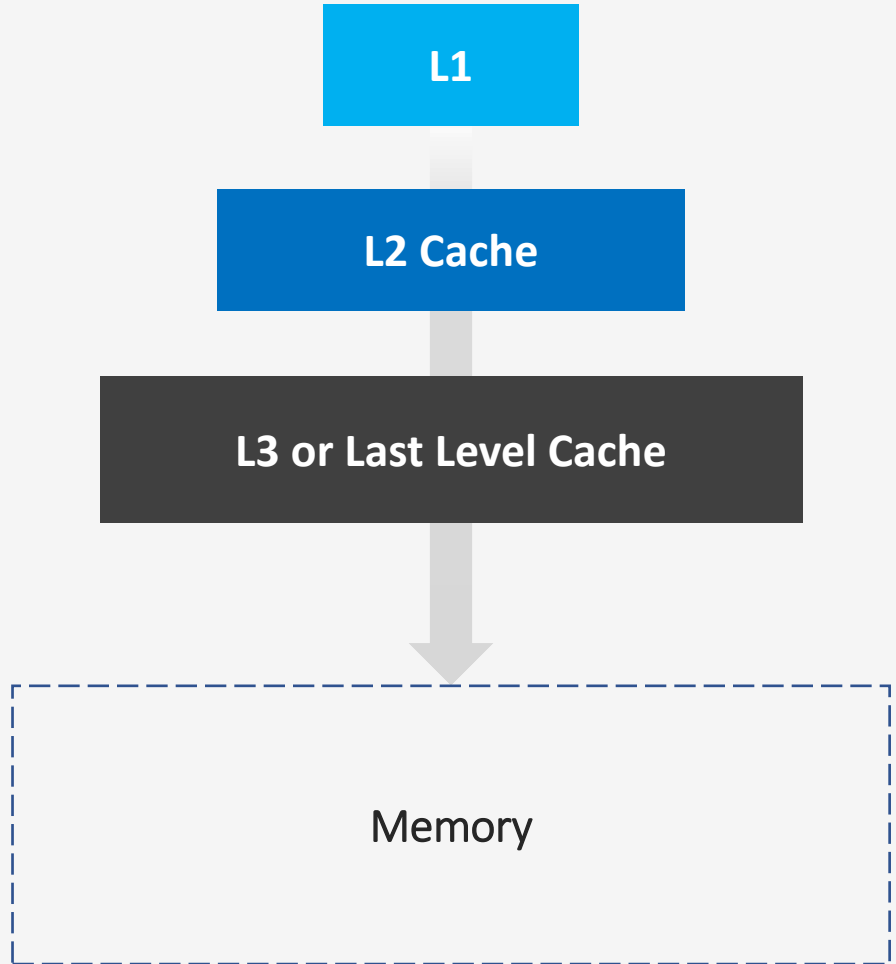
Cache Replacement

LLC Replacement

Final level before data requests move to access Memory

Least Recently Used

This is the most commonly used algorithm for LLC replacement

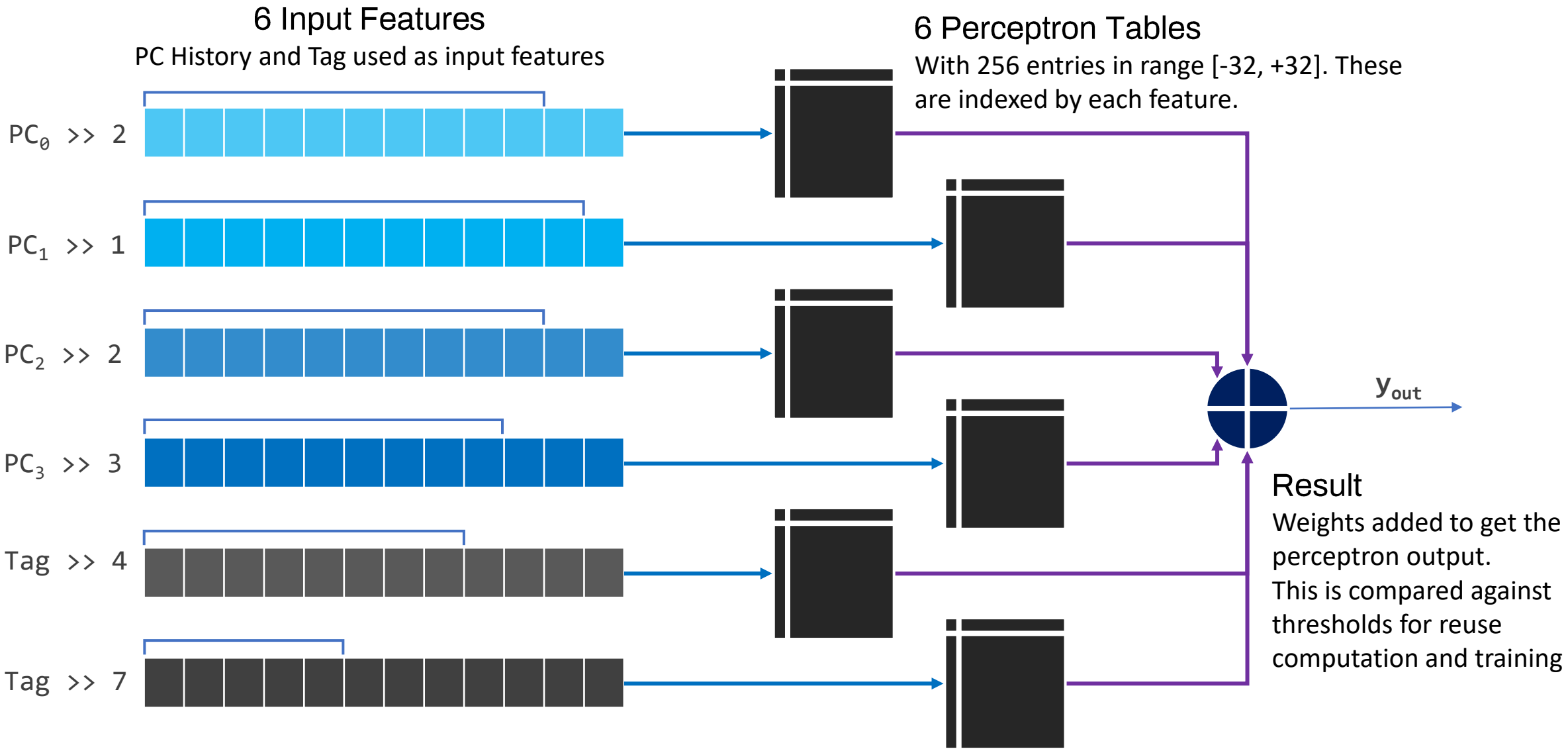


Glider, Hawkeye, SHiP, SRRIP

State of the art LLC replacement policies

Perceptron

Using a single Layer Neural Network to Predict Reuse



Training - Sampler

Additional SRAM used to store samples for perceptron training

64 Sets of 16 Way Set Associative Memory
Every 32th (Blocks/sets in sampler) set is sampled to this memory.

On a hit in the sampler, we train the perceptron to decrement its weights if $y_{out} > -\theta$
On a miss, we find an entry to evict by

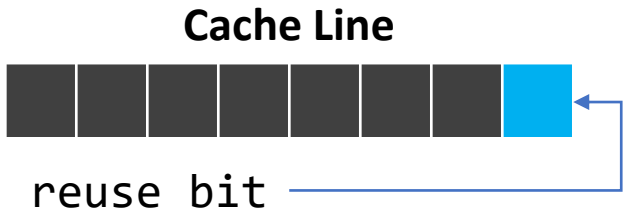
- Fist looking for an invalid block
- Second, a dead block where $y_{out} > \tau_{replace}$, or
- Last, using LRU

Train the perceptron to increment its weights if $y_{out} < -\theta$
The sampler entry is updated with the new set of features and perceptron output (for both hit and miss)

Tag (15 bits)	Y_{out}	Input Features	LRU bits

Perceptron - Eviction

Using the perceptron results to predict blocks less likely to be reused



Thresholds	
θ	74
$\tau_{replace}$	124
τ_{bypass}	3

Reuse Bit

1 bit flag added to each cache line to indicate if it would be reused or not

This is set on each LLC access $y_{out} < \tau_{replace}$, else it is set to false

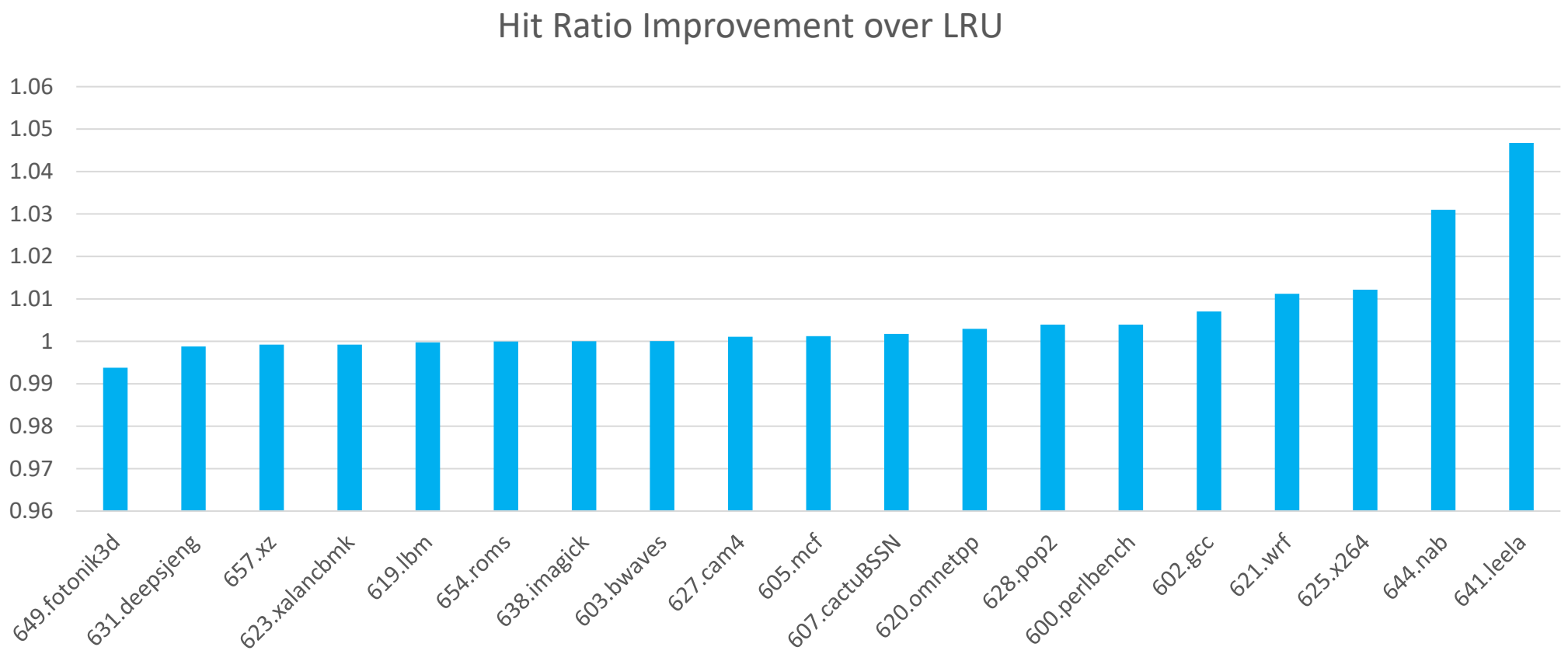
Finding a victim

Consult the perceptron to get y_{out} for incoming address.

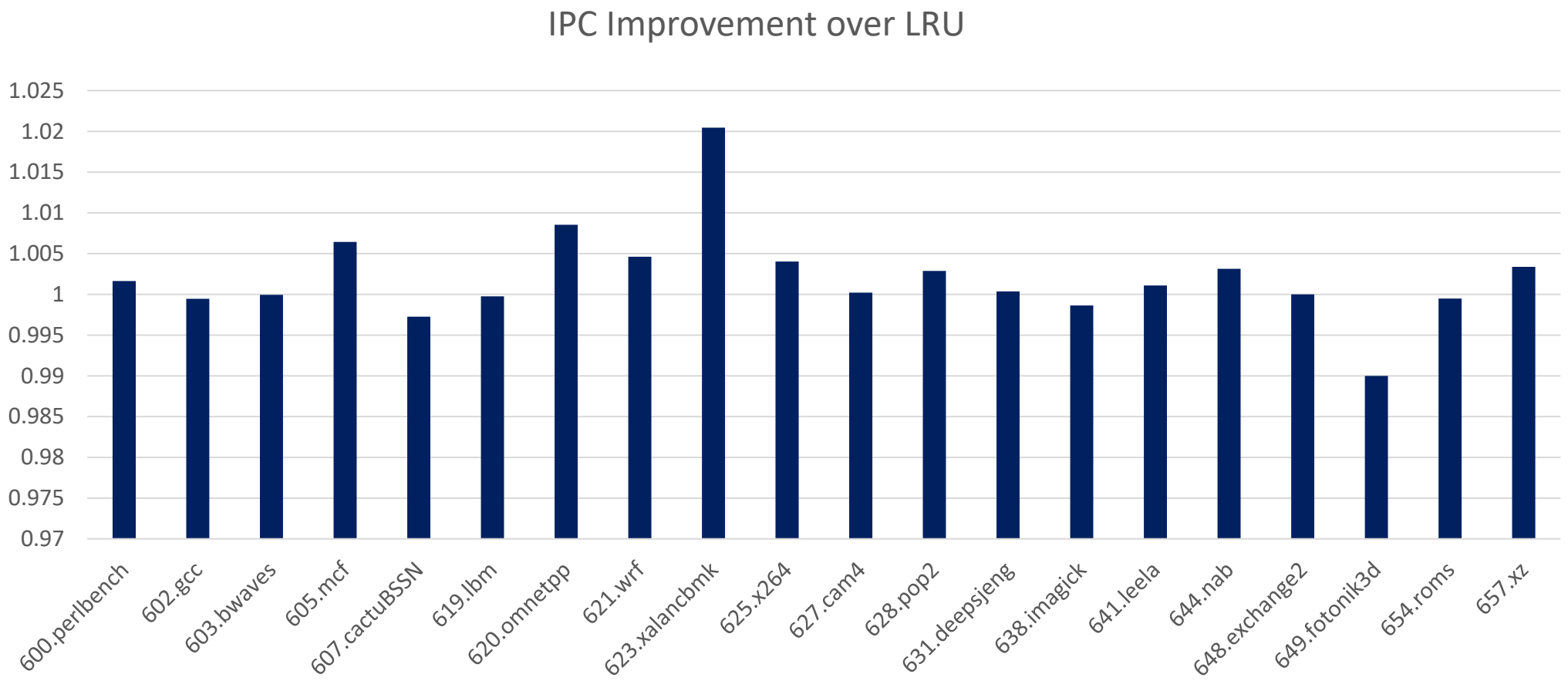
- If the value is higher that the threshold τ_{bypass}, it bypasses the LLC.
- In every other case, we check for a dead block in the set – a block with reuse bit set to 0.
- If that is also not found, evict the least recently used block

Simulation Results

Comparing the perceptron implementation to LRU



The algorithm was created using Champsim and was trained on a single core CPU with 2MB 16-way LLC and was tested on the SPEC 2017 suite if benchmarks



The algorithm was created using Champsim and was trained on a single core CPU with 2MB 16-way LLC and was tested on the SPEC 2017 suite if benchmarks

Reinforcement Learning Replacement

Offline RL based model to train a network to predict reuse of blocks

SPEC2017 Traces

Trace files are fed into a modified version of Champsim to create LLC access traces.
`<IP, address, type>`

Champsim

Cache State Model in Python

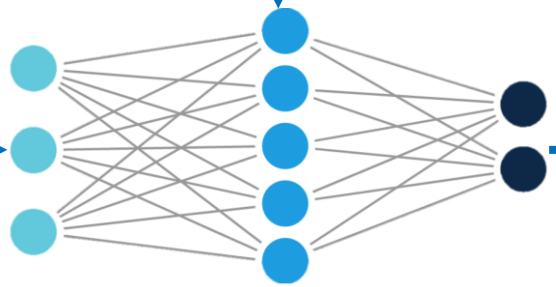
Models the cache state and computes all the parameters required for the input vector. This reads the clear text trace from Champsim

Reward

Based on distance from optimal replacement

Input State

A vector of 214 entries with cache access information, set details and cache line level details (for the current set). This gives a relevant parts of the overall cache for the current request.



Q Value Network

Output

A 16 (number of ways) long vector giving a value to each action to be taken (which line to replace).