Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Έγκαιρη Προανάκτηση Δεδομένων στην Κρυφή Μνήμη με Χρήση Τεχνικών Μηχανικής Μάθησης

Διπλωματική Εργασία

Δημήτριος Στυλιαράς

Summary

- **Prefetching**: Reduction in CPU idling from memory stalls
- Study of **traditional prefetchers** for different benchmark suites shows **no universal solution** to the prefetching problem for varying memory access patterns
- **Neural Networks** are a promising solution to pattern prediction that could be used in **memory prefetching**
- We implement an **LSTM** model and experiment with complementary use of traditional prefetchers in the **Last Level Cache.**
- An important factor in our model is **timeliness**, where our analysis shows the significant impact in the **performance** of Prefetchers
- Experimentation shows promising use of **Neural Networks in Computer Architecture** alongside **traditional techniques**.
- Further study of similar methods and techniques with more **exploration** in model parameters is needed.

Contents

Prefetching

- Prediction and transfer of data to faster memory levels before their use
- **Data** and Instruction Prefetching
- Software and **Hardware**

Characteristics:

- Prefetch Data
- Timing of Prefetches
- Operation Level and Placement
- Prediction Model

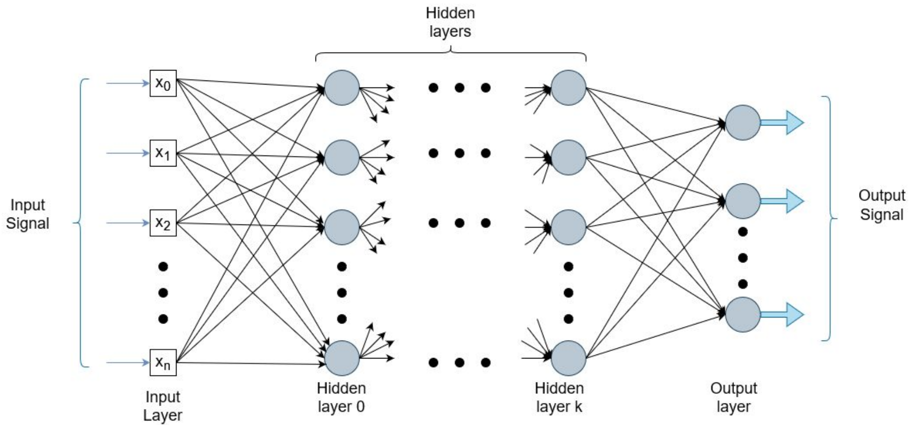
Challenges:

- Cache Pollution
- Efficiency issues
- Implementation factors

Machine Learning

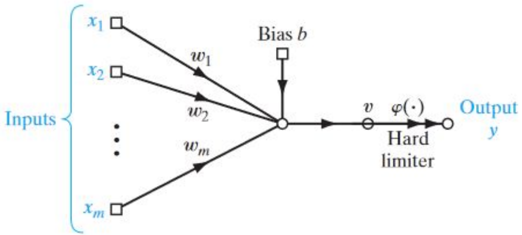
- Supervised
  - Regression
  - Classification
- Unsupervised
- Reinforcement

Multilayer Perceptron



Neural Networks

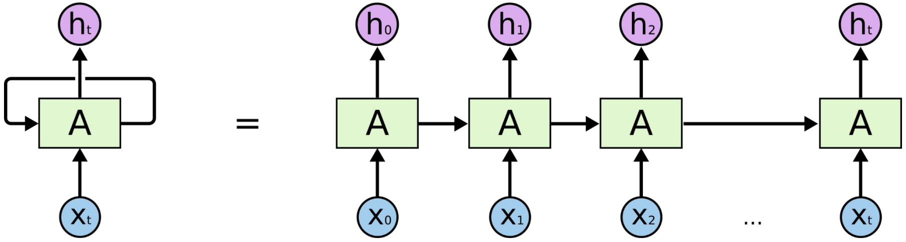
Perceptron



Key components

- Activation Function
- Loss Function
- Training Algorithm
  - Backward Propagation
  - Iterative Algorithms(Gradient Descent)

Recurrent Neural Networks



RNN unrolled over time

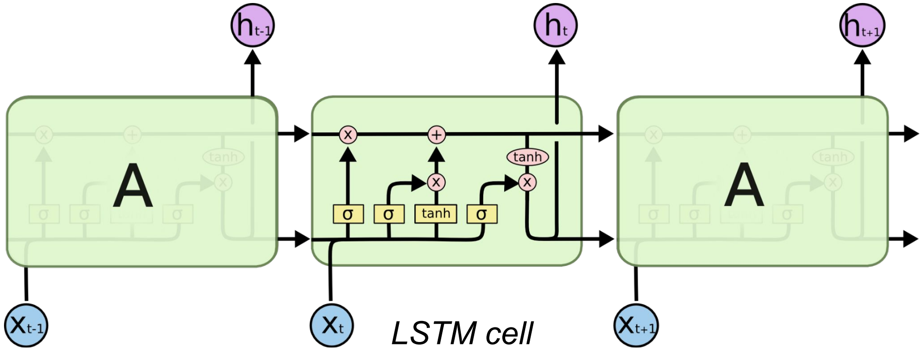
Types

- Vanilla RNNs
- Long Short-Term Memory RNNs
- Gated Recurrent Units
- Bidirectional RNNs

Sequential Data

- Time Series Prediction
- Machine Translation
- NLP Next Word Prediction

Long Short-Term Memory Recurrent Neural Network

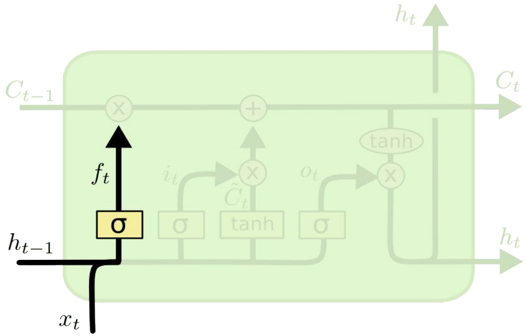


Advantages

- Counter Vanishing Gradient
- Keep dependance in large sequences

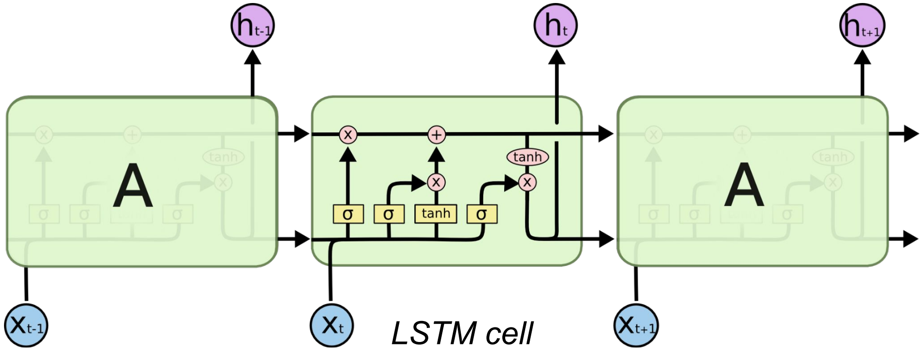
Components

- Forget Gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Long Short-Term Memory Recurrent Neural Network

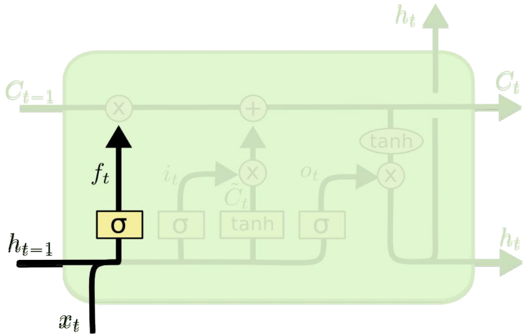


Advantages

- Counter Vanishing Gradient
- Keep dependance in large sequences

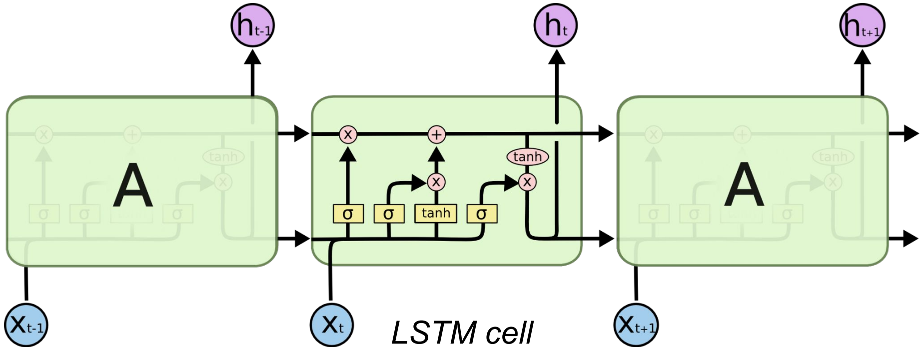
Components

- Forget Gate
- Update Gate



$$
\begin{aligned}
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right) \\
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)
\end{aligned}
$$

Long Short-Term Memory Recurrent Neural Network

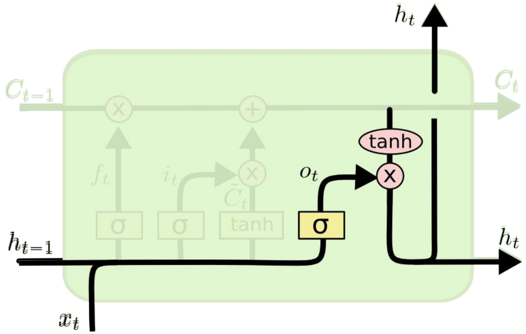


Advantages

- Counter Vanishing Gradient
- Keep dependance in large sequences

Components

- Forget Gate
- Update Gate
- Output Gate

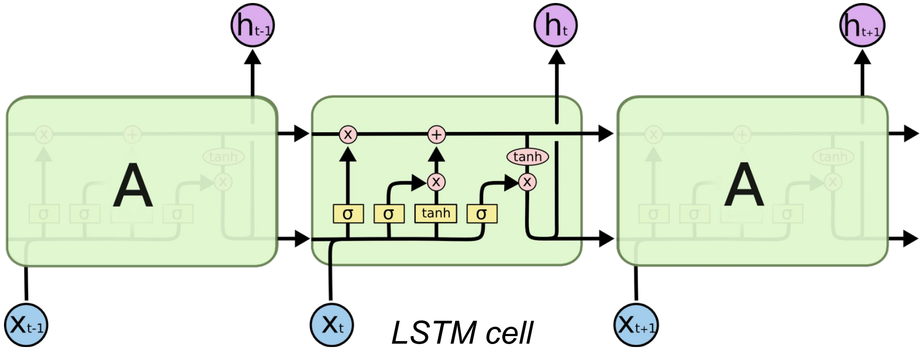


$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$h_t = o_t * \tanh(C_t)$$

Long Short-Term Memory Recurrent Neural Network

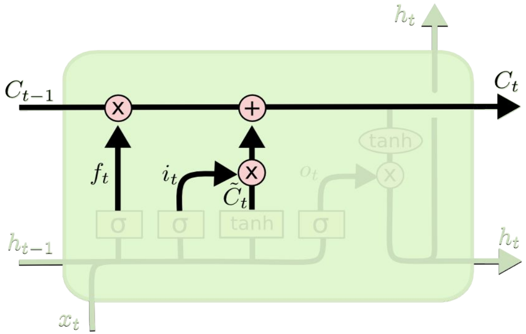


Advantages

- Counter Vanishing Gradient
- Keep dependance in large sequences

Components

- Forget Gate
- Update Gate
- Output Gate



Cell State

$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t] + b_o\right)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Non Machine Learning Prefetchers

Next Line Prefetcher: Simple prediction of next address prediction

Best Offset Prefetcher

[HPCA '16]

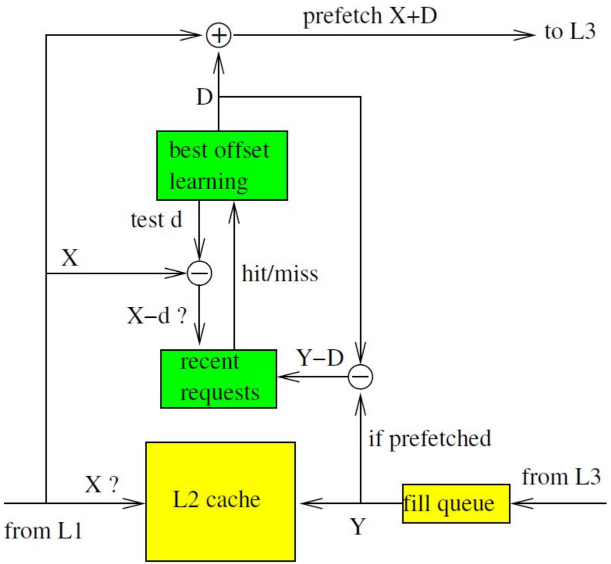
Focuses on timeliness

Recent prefetch addresses Table

Offset List and Score Tables

Dynamic offset selection for prefetching

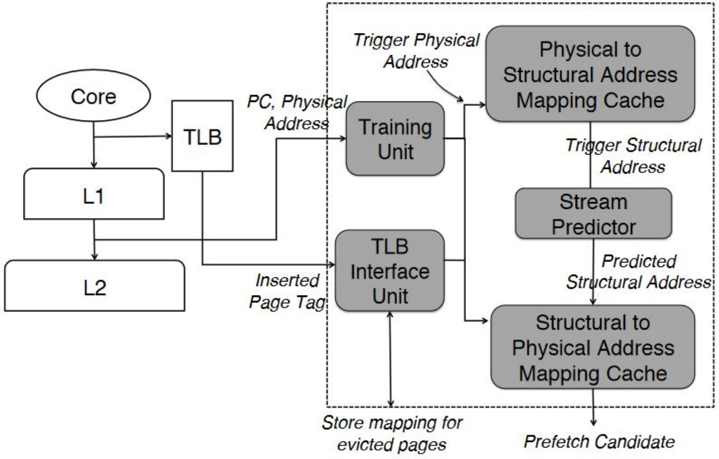
Scoring offsets by testing prefetch timing on recent history



Irregular Stream Buffer
[MICRO '13]

Maps physical addresses

Learns address temporal correlation



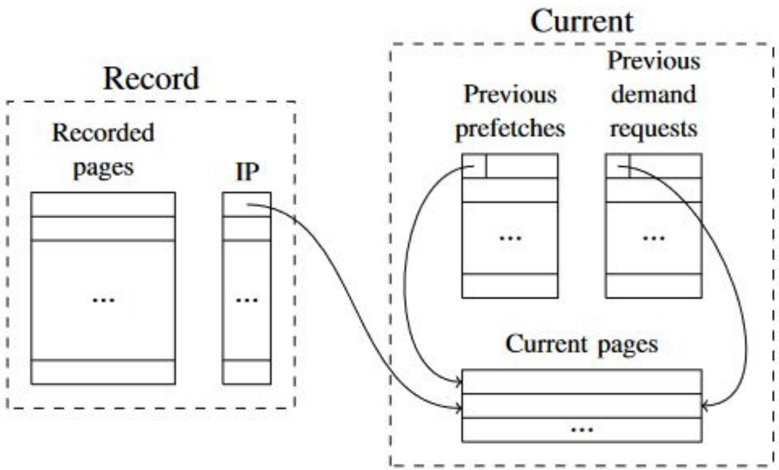
BLUE [ISCA '21]

Focus on Timeliness

Extension of Berti for across page prediction

Entangling Prefetcher and Next Line

Berti

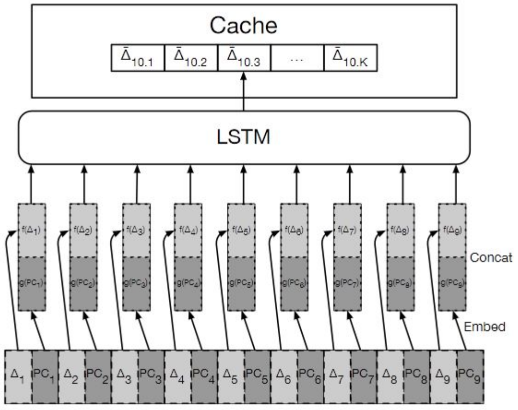


ML-based Prefetchers

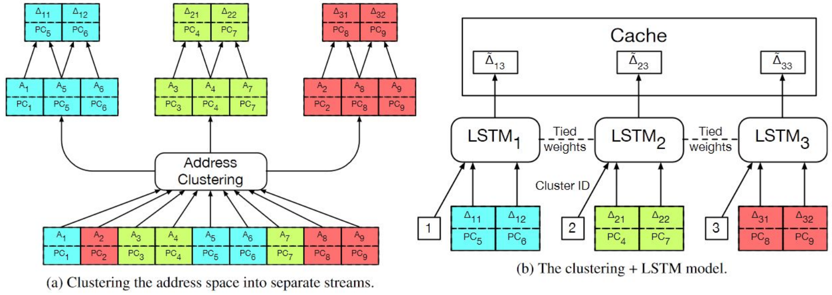
Learning Memory Access Patterns

[ICML '18]

- Vocabulary of most common memory address deltas
- One hot encoded
- PC + deltas inputs
- LSTM Model
- Prefetching Degree of 10
- Only Accuracy
- Two models



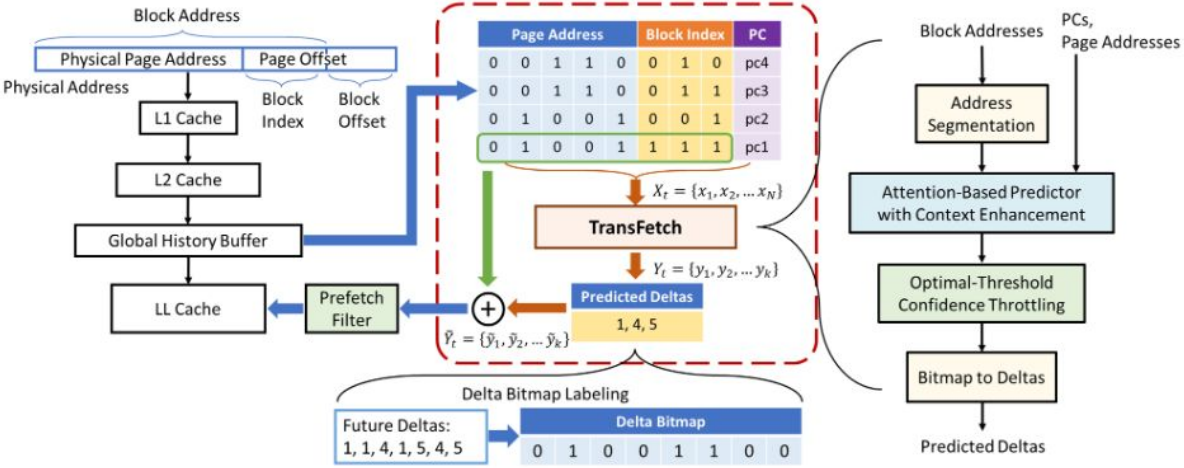
Embedding LSTM



Clustering and multiple LSTMs for each cluster

Transfetch

[CF '22]



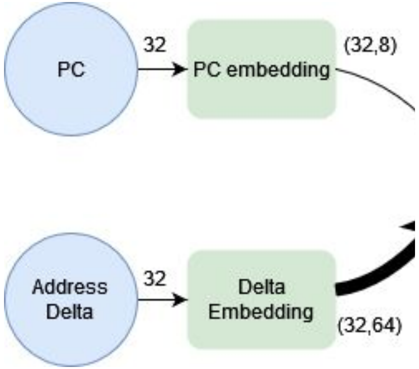
Address segmentation

Transformer

Attention Mechanism

PCs and Page distances as Context

Our proposed LSTM Prefetcher Model



PC Embedding Size = 8

Delta Embedding Size = 64

Sequence Size = 32

Two layer LSTM

LSTM Size = 64

Linear Layer Size = 30000

Using the Model

- **Offline** Training and Generation
- Training Dataset **randomly sampled** from the run
- **ADAM optimizer**
- Data Organization with **pandas** and custom **Dataloader** of **Pytorch**
- Prediction Labels derived from statistical analysis of the 30000 most **common Deltas**
- Use of the model as **Stateless LSTM** between batches
- Experimentation with **additional simultaneous** prefetching from **non ML prefetchers**

Methodology

- Oracle Prefetching
- Delta Analysis
- Parameter Tuning

Tools

- Pytorch framework
- ChampSim Simulator
- SPEC06, SPEC17 and GAP benchmark suites

Prefetcher Characteristics

- Last Level Cache traces
- PC-Based
- Address Deltas and PCs as features
- Static Distance in Prefetches
- Across Page Physical Address
- 2 Degree Prefetching
- Main Metric is IPC

Prefetcher Setups

Individual prefetchers

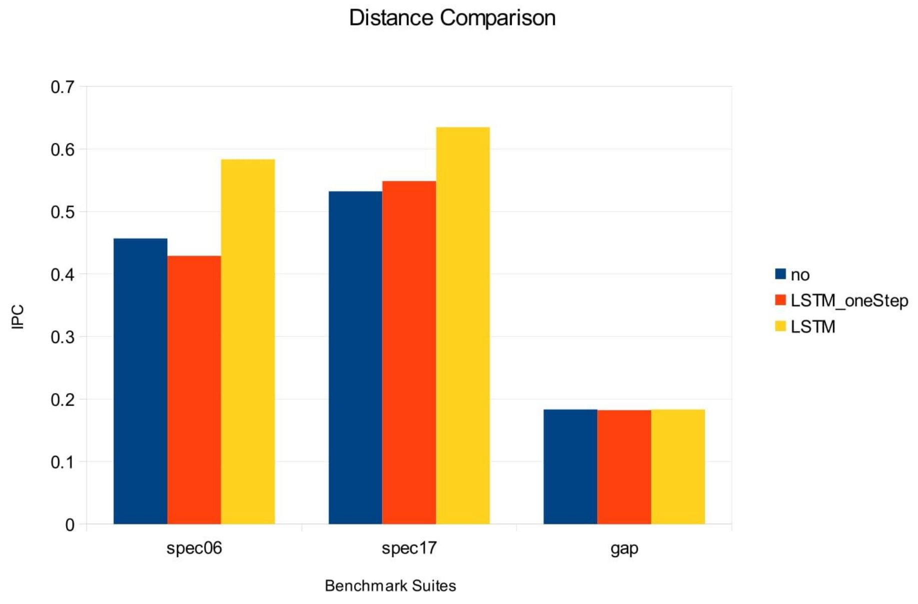
- Baseline (no)
- Next Line (next_line)
- Best Offset Prefetcher (bo)
- Irregular Stream Buffer (sisb)
- TransFetch
- Our model (LSTM)

Combined Prefetchers

- Irregular Stream Buffer with Best Offset Prefetcher (sisb_bo)
- Our model with Next Line (LSTM_next)
- Our model with Best Offset Prefetcher (LSTM_bo)
- Our model with Irregular Stream Buffer (LSTM_sisb)

The evaluation of the above setups was mainly based on IPC and IPC Improvement, with other metrics being for assistance in the designing process

Distance Results



Comparison of baseline(no), our model with prediction distance 1(LSTM_oneStep), and our model with optimal Step prediction(LSTM) on Geometric Means of IPCs on SPEC06, SPEC17, GAP

OneStep Model is worse in all Benchmark Suites.

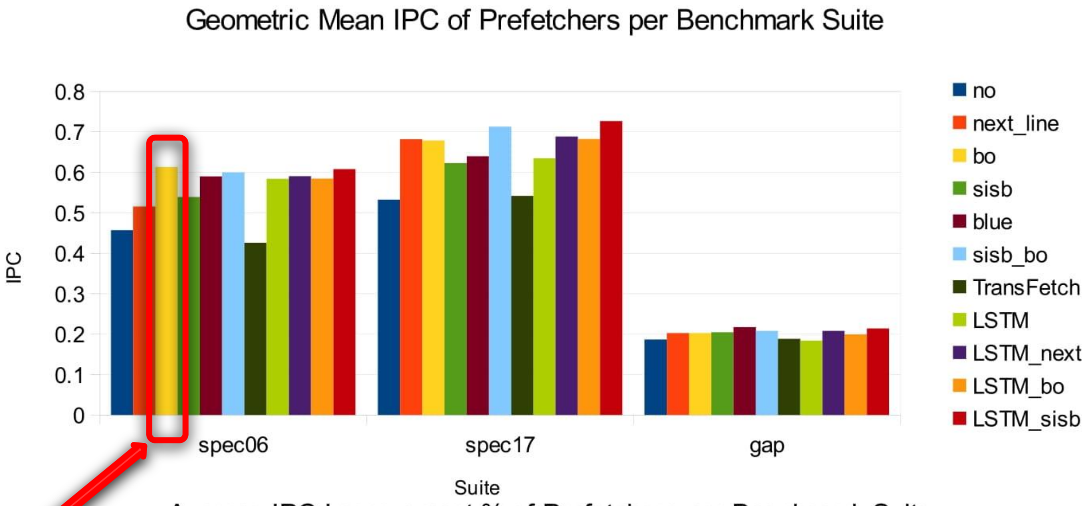
SPEC06: one Step hurts performance, while LSTM improves LSTM over baseline

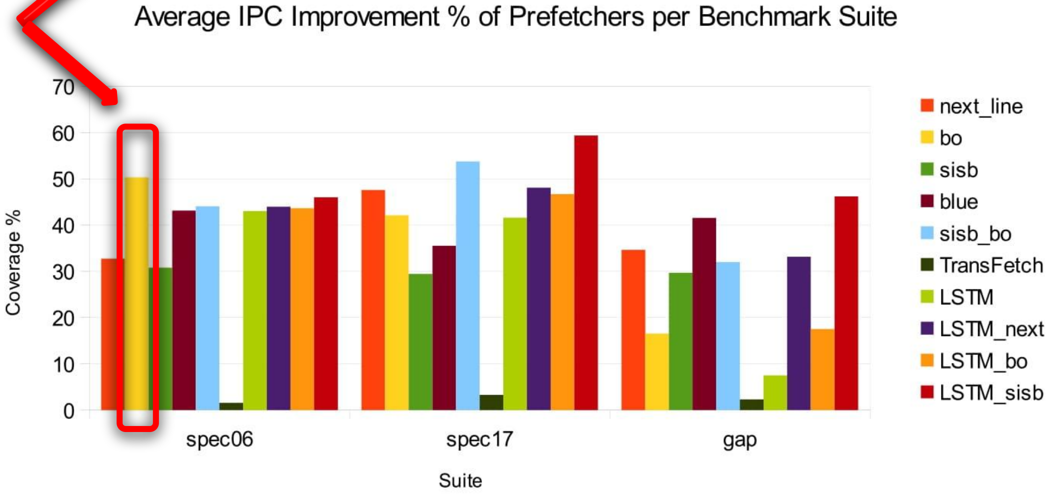
SPEC17: both improve the baseline, but LSTM has significant advantage

GAP: both slightly hurt the performance with LSTM being a bit better

Results

SPEC06 we have the BOP performing the best with our LSTM_sisb being second and TransFetch is the worst

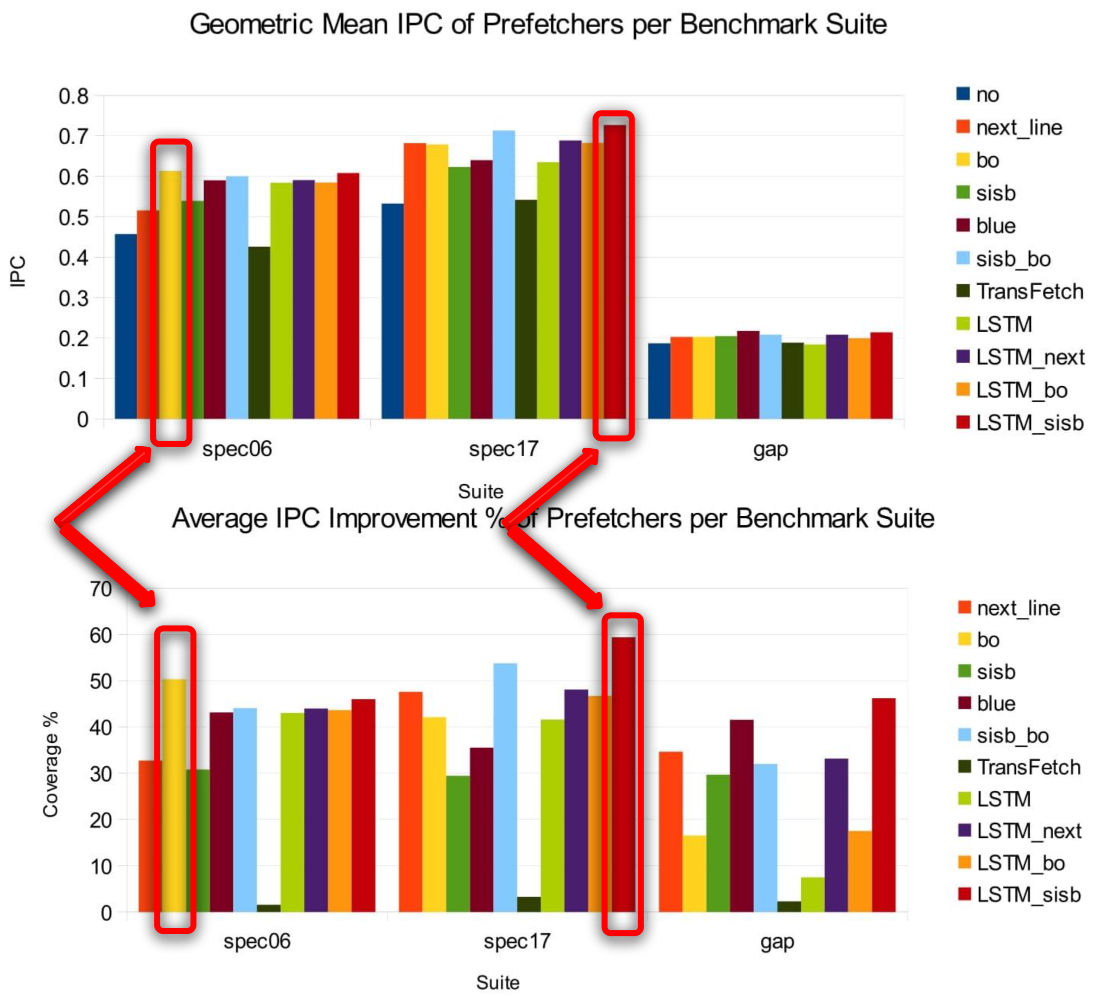




Results

SPEC06 we have the BOP performing the best with our LSTM_sisb being second and TransFetch is the worst

SPEC17 we have the LSTM_sisb performing the best followed by sisb_bo and TransFetch being the worst

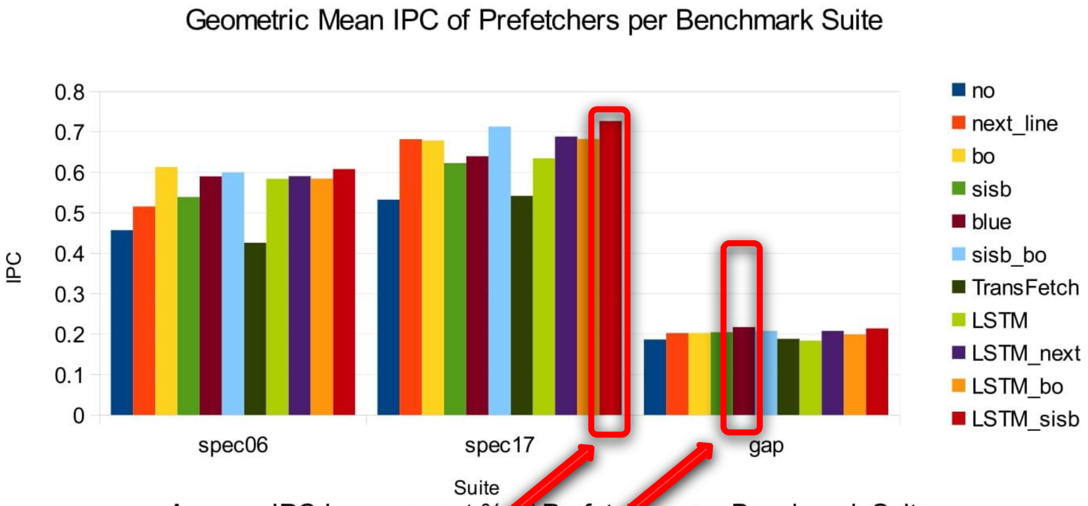


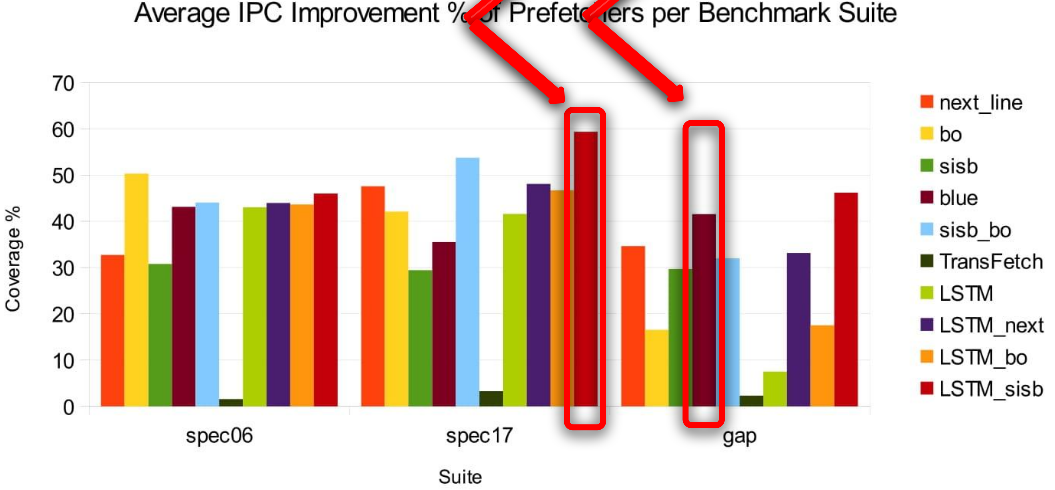
Results IPC

SPEC06 we have the BOP performing the best with our LSTM_sisb being second and TransFetch is the worst

SPEC17 we have the LSTM_sisb performing the best followed by sisb_bo and TransFetch being the worst

GAP we have the blue is performing the best followed by LSTM_sisb and LSTM being the worst.



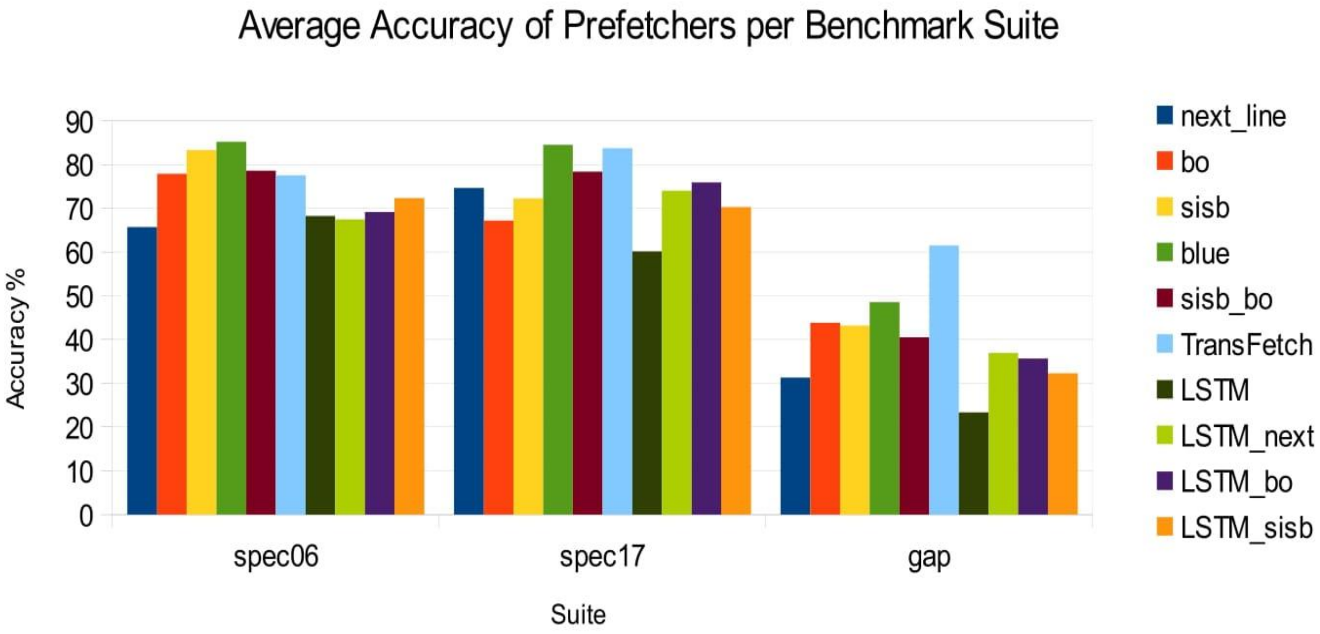


Results Accuracy

SPEC Generally high accuracy

LSTM Lowest Accuracy

Blue and TransFetch Higher Everywhere

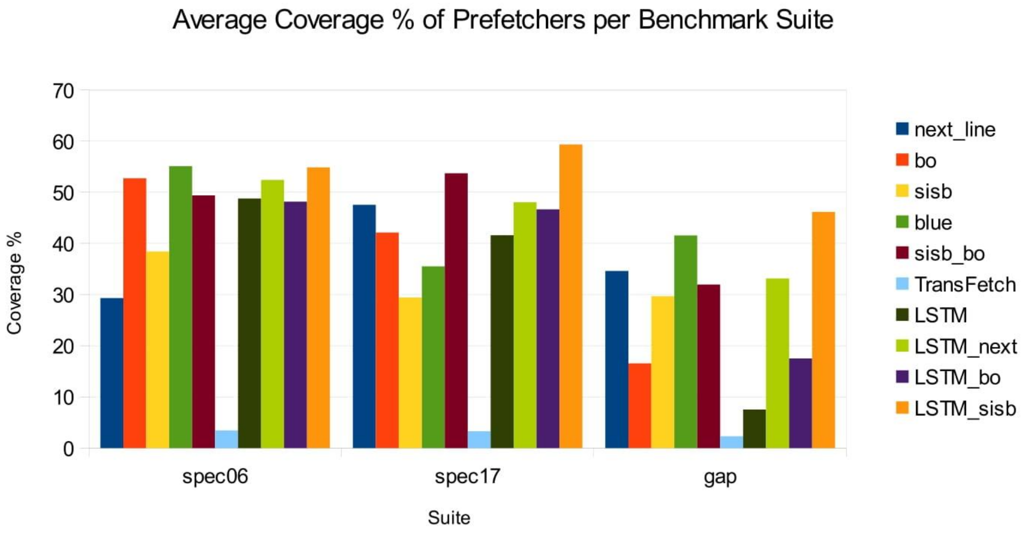


Results Coverage

TransFetch way lower Coverage

LSTM_sisb consistently high coverage everywhere

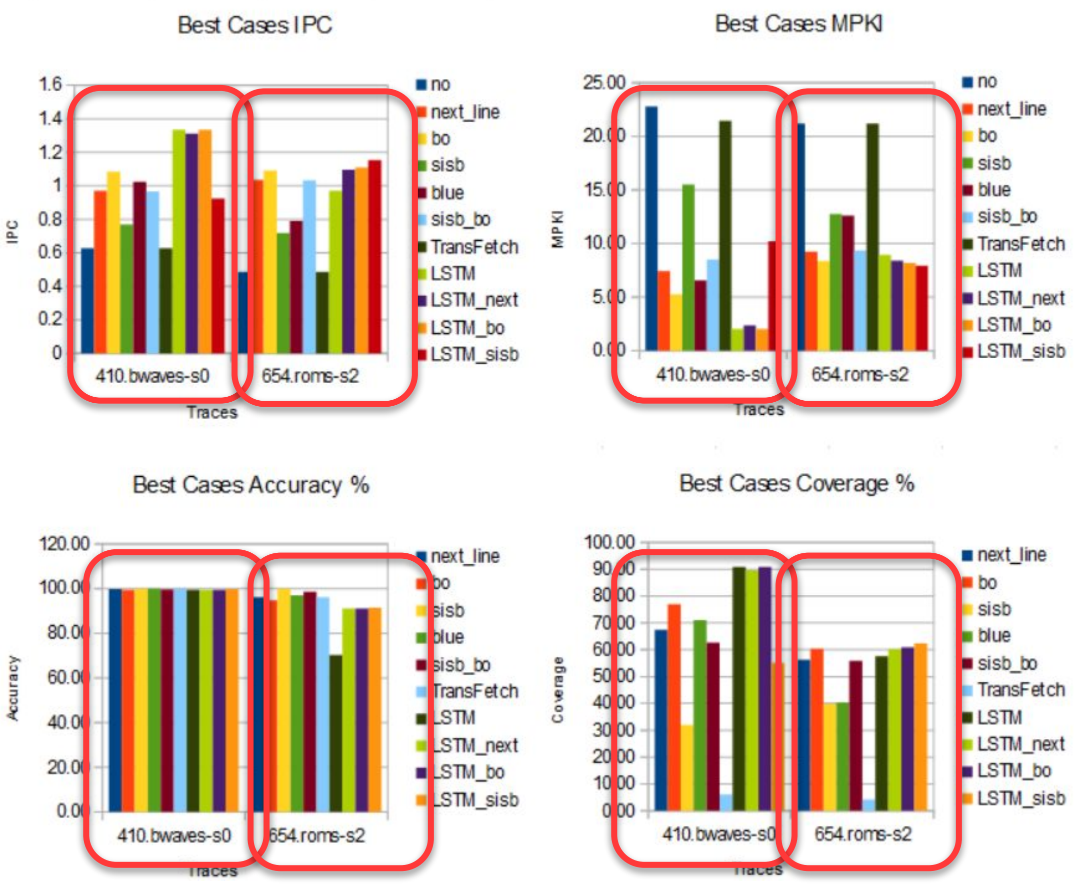
Blue better in SPEC06 and second in GAP but low coverage in SPEC17



Best cases

410.bwaves-s0 has plain LSTM and LSTM_bo as the best performance even with higher coverage and reduction of misses by a large number

654.roms-s2 has similar characteristics with the difference that LSTM_sisb is better than just the LSTM



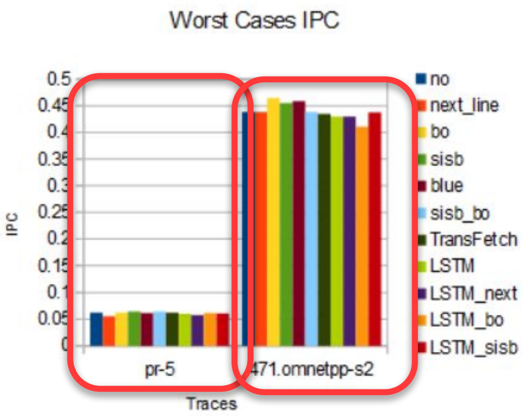
Worst cases

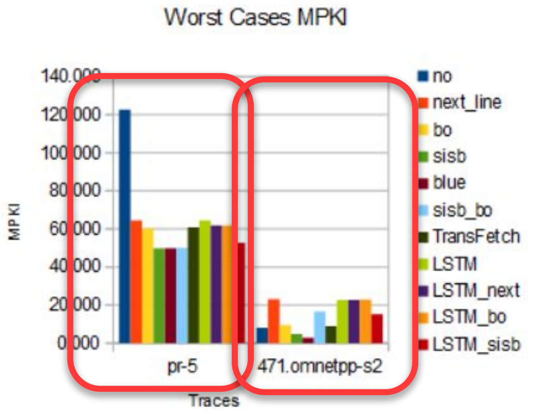
pr-5 has low accuracy, coverage and IPC in general only sisb performs positively.

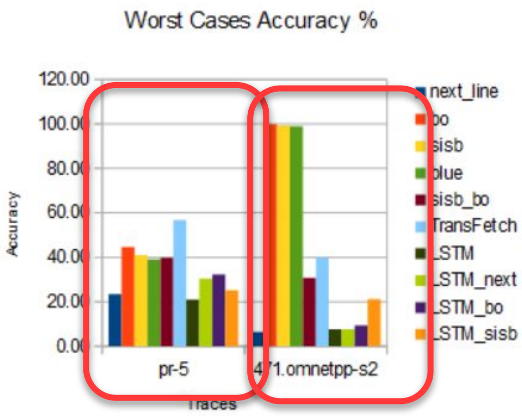
It has a large number of different deltas with the most common covering lower percentage compared to other benchmarks.

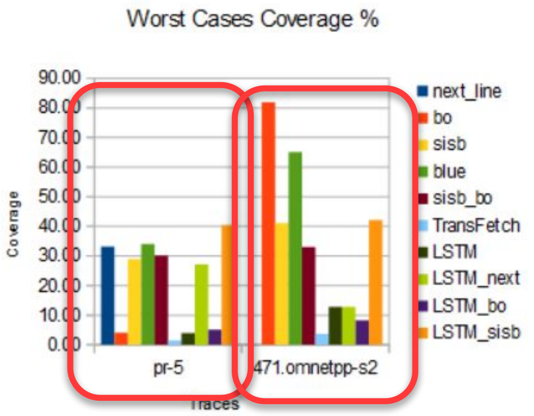
471.omnetpp-s2 has BOP as the best with blue following. Our models suffer in accuracy compared to the other ones.

Here all LSTMs except LSTM_sisb, increase the number of misses.









Conclusion

- **Prefetching** counters CPU **idle time** waiting for memory
- Simple and complex algorithms based on application and resources work as prefetchers
- **Neural Networks and Machine Learning** seem promising to a better and more **universal** solution
- **Address sequence** prediction similar to **NLP Next word prediction**
- **PC-based, address delta classicification** on **Last Level Cache** in **physical** address space
- Results show better performance with **combined** use of **traditional** prefetchers and our **LSTM** model

Future Work

- **Online training** and generation
- **Sampling** Method or adaptation for training in the beginning
- Different Delta **encoding**
- Experimentation with **additional features** as input
- **Virtual Address** application

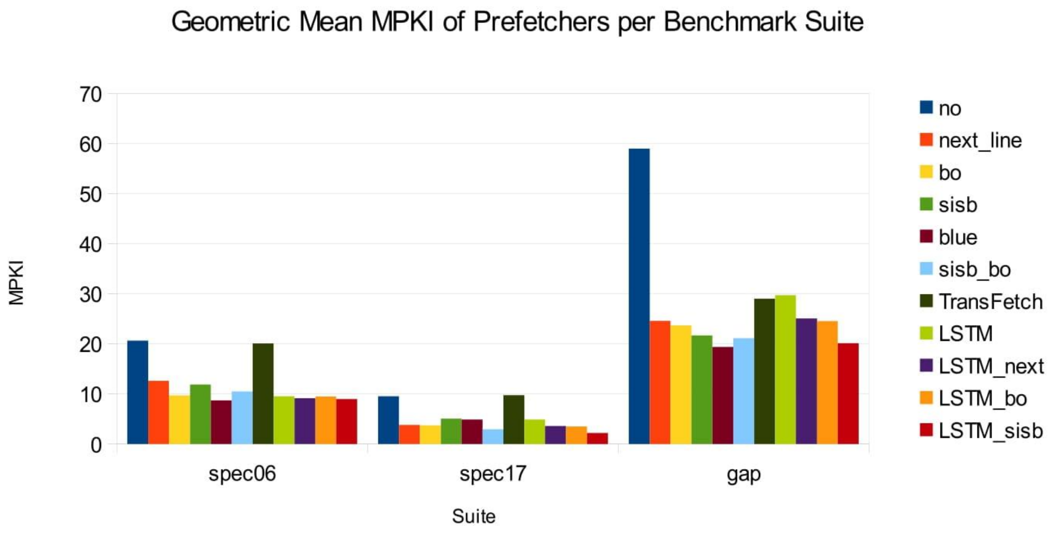
Questions?

Results MPKI

SPEC06 TransFetch worst, blue lowering number of misses

SPEC17 LSTM_sisb and sisb_bo are the best and TransFetch the worst

GAP has high number of MPKI, with the greater reduction from blue and LSTM_sisb με χειρότερη το LSTM



Oracle Stats

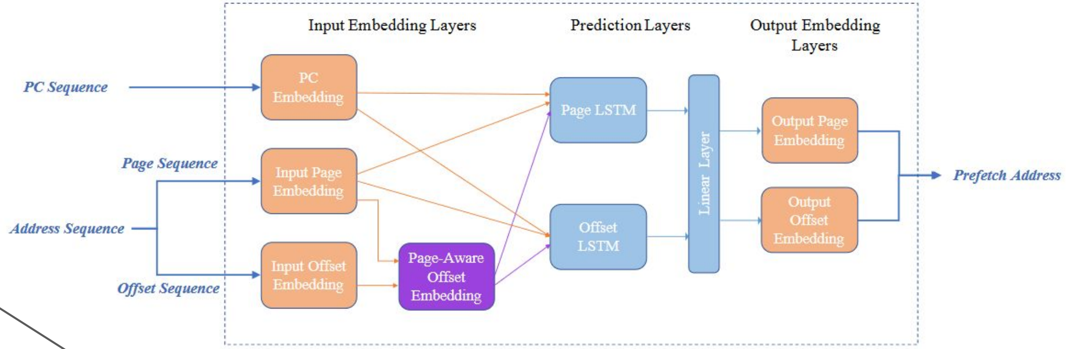
	Global Distance			Per PC distance		
Suite	Step	IPC	IPC Improvement %	Step	IPC	IPC Improvement %
SPEC06	13.48	0.7289969748	73.86	7.29	0.7261998886	73.26
SPEC17	-	-	-	4.51	0.7453825265	45.72
GAP	24.26	0.3360217918	71.11	8.05	0.2759496478	40.32

Delta Label and Coverage

	Global Delta Calculation			Per PC Delta Calculation		
Suite	**Step**	**No of Deltas**	**Filter Coverage %**	**Step**	**No of Deltas**	**Filter Coverage %**
SPEC06	13.48	3415580.48	71.32	7.29	482760.86	87.31
SPEC17	-	-	-	4.51	2351659.70	80.51
GAP	24.26	3386725.16	62.40	8.05	724414.00	86.39

Machine Learning Prefetchers

Voyager



Pythia