Tencent AI Lab

Efficient Phase-Functioned Real-time Character Control in Mobile Games: A TVM Enabled Approach

Haidong Lan, Wenxi Zhu, Qian Qiu, Dou Wu, Honglin Zhu, Jingjing Zhao, Xinghui Fu, Minwen Deng, Jintao Meng

Tencent AI Lab

2022-8-12

zjianzheng(草子健)

Agenda

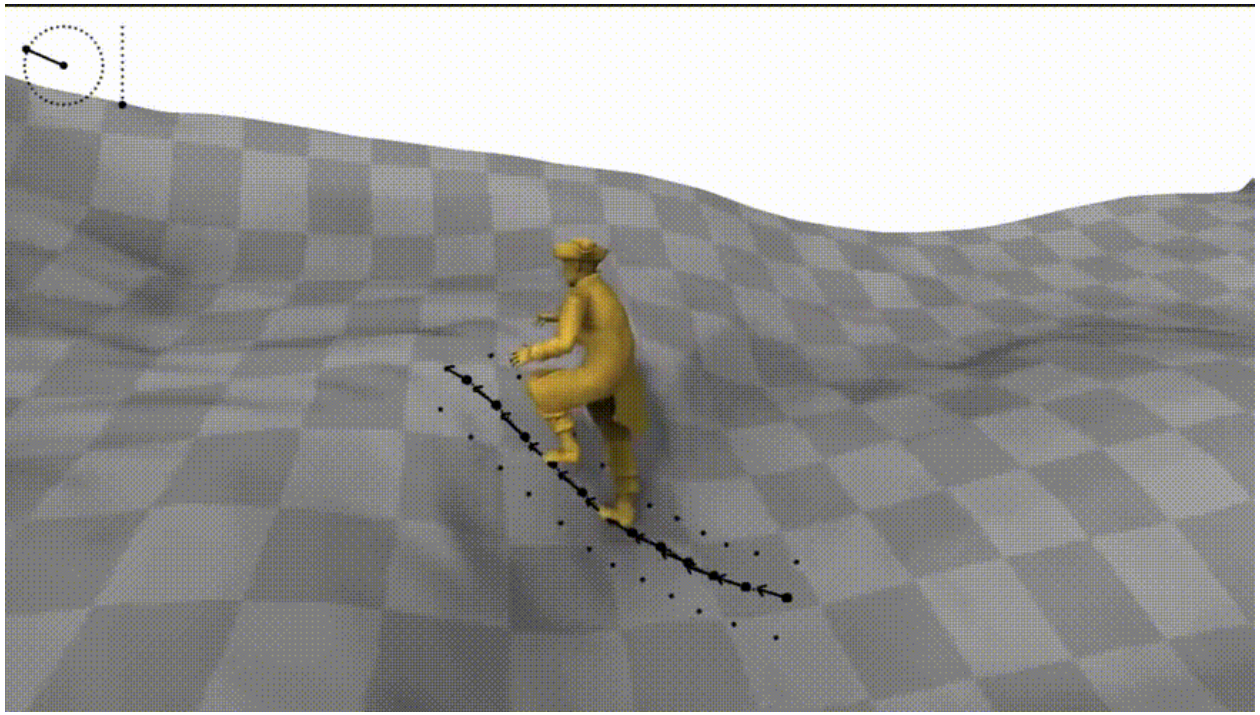
Tencent AI Lab

- Background
- Methods
- Performance analysis
- Lessons learnt

zijianzheng(草子健)

Background – Game character control

Tencent AI Lab



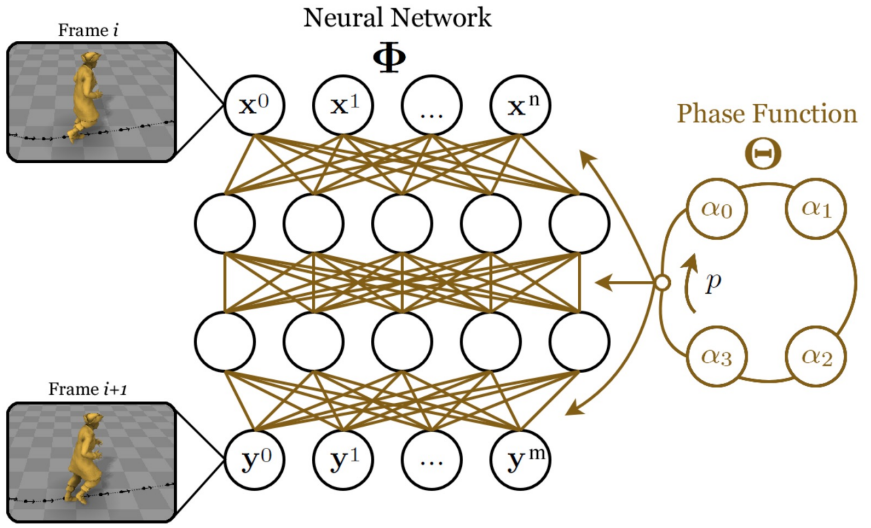
Real-time game character control algorithm

- Predict the anchor point of the game character in the next frame
- Embedded in mobile game system for real-time computation
- Required latency: within 2ms for 8 characters
- Deploy on Android/iOS, verification on macOS/Windows

Video taken from https://dl.acm.org/doi/10.1145/3072959.3073663, supplement materials.

zijianzheng(草子健)

Background – PFNN



Typical NN layer
Y = W * X

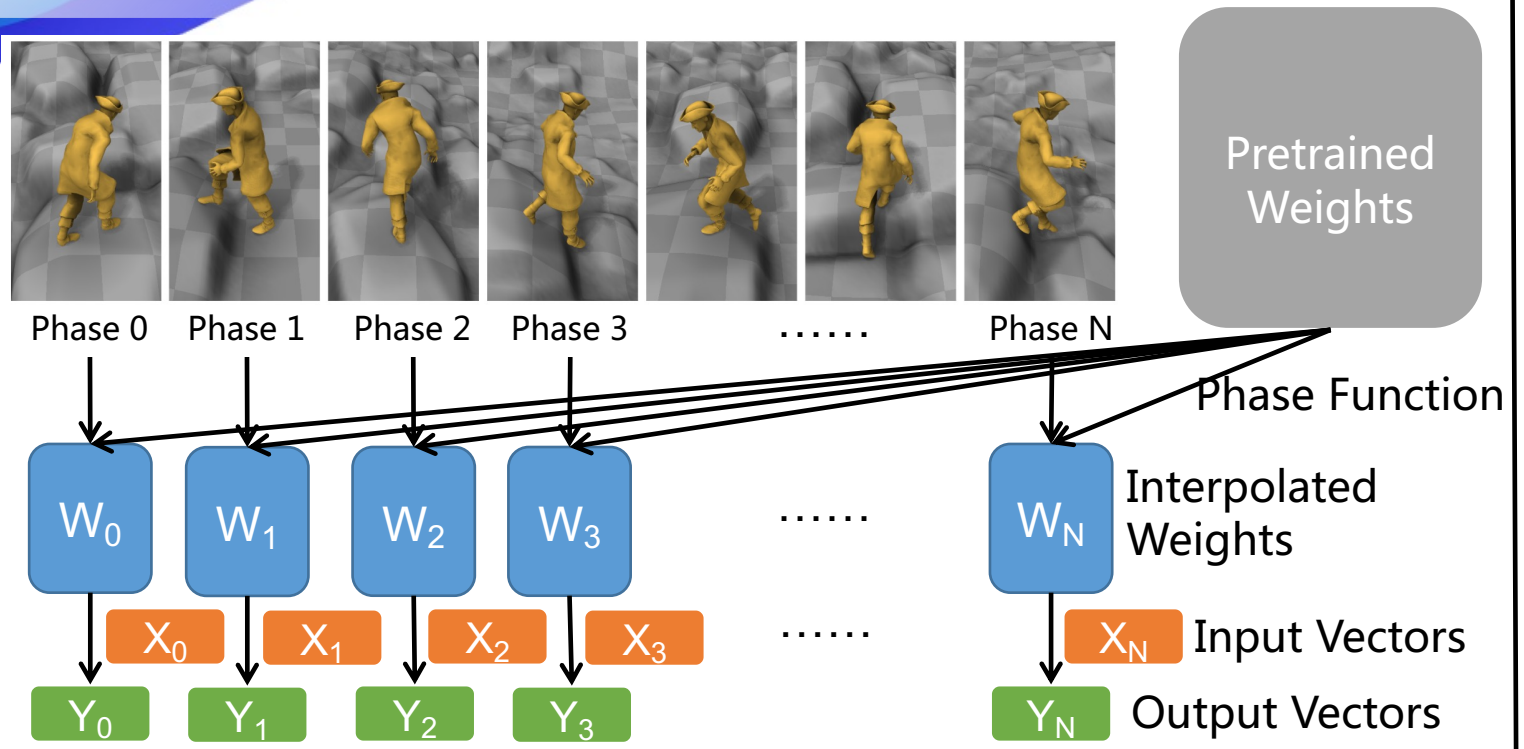
PFNN layer
Y = Φ (W, p) * X

Algorithm

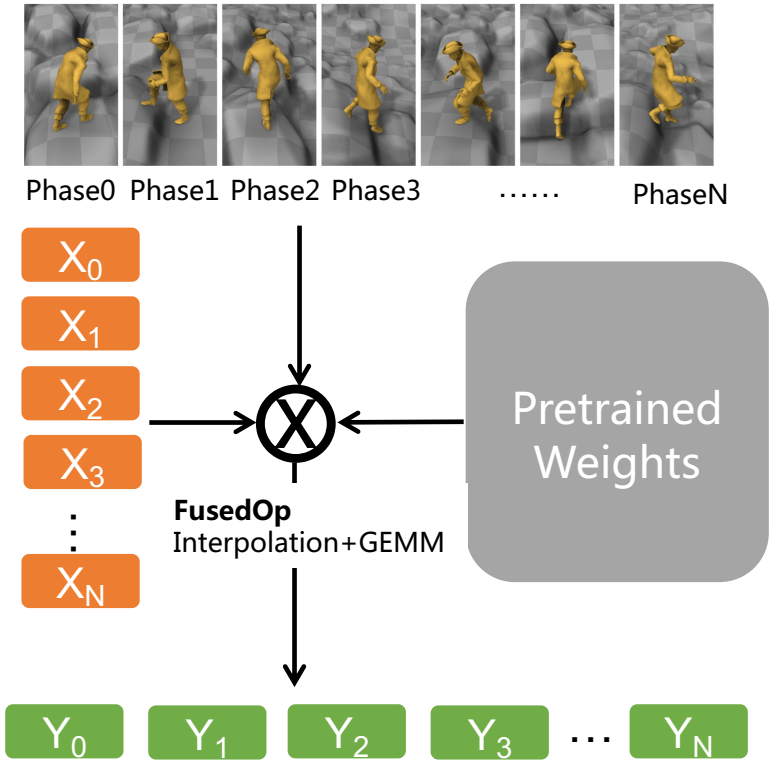
- PFNN (Phased-Functioned Neural Network)
- Phase value corresponds to game character status
- Architecture: fully connected layers where weight matrices are interpolated by a phase function
- The interpolation overhead is noticeable

Reference: Holden D, Komura T, Saito J. Phase-functioned neural networks for character control[J]. ACM Transactions on Graphics (TOG), 2017, 36(4): 1-13.

Background – PFNN in real game scenario



(a) Existing libraries can only compute sequentially for N characters due to distinct phase values



(b) Fused operators can efficiently compute for all characters in a batch

Technical Difficulties

- Different characters cannot share identical phase values and weight matrices.
- Strict requirement on inference latency: within 2ms for 8 characters on mobile.
- Agile development: dev in 2 weeks for desktop and mobile.

Optimization Oppotunities

- Interpoation (phase) function is element-wise, fusible with matrix multiplications

Reference: Holden D, Komura T, Saito J. Phase-functioned neural networks for character control[J]. ACM Transactions on Graphics (TOG), 2017, 36(4): 1-13.

Methods – Fused Operator

Tencent AI Lab

C: #Characters

Algorithm 1 Non-fused Pseudocode for Equation 5

```
for c = 1 to C do
  for k = 1 to K do
    for n = 1 to N do
      S = 0
      for i = 1 to 4 do
        S = S + W_{c,i} · α_i(k, n)
      end for
      I_c(k, n) = S
    end for
  end for
end for
for c = 1 to C do
  Below is equivalent to a GEMV call.
  for n = 1 to N do
    S = 0
    for k = 1 to K do
      S = S + I_c(k, n) · X_c(k)
    end for
    Φ(c, n) = S
  end for
end for
```

Phase Function

Vector-Matrix Multiplication

Algorithm 2 Fused Pseudocode for Equation 5

```
for n = 1 to N do
  for c = 1 to C do
    S = 0
    for k = 1 to K do
      T = 0
      for i = 1 to 4 do
        T = T + W_{c,i} · α_i(k, n)
      end for
      S = S + T · X_c(k)
    end for
    Φ(c, n) = S
  end for
end for
```

PF-GEMM

Observation 1 : The fused operator reduces $\mathbf{I_c}$ to local variables.
Deduction 1 : No need to separately store interpolated matrices.

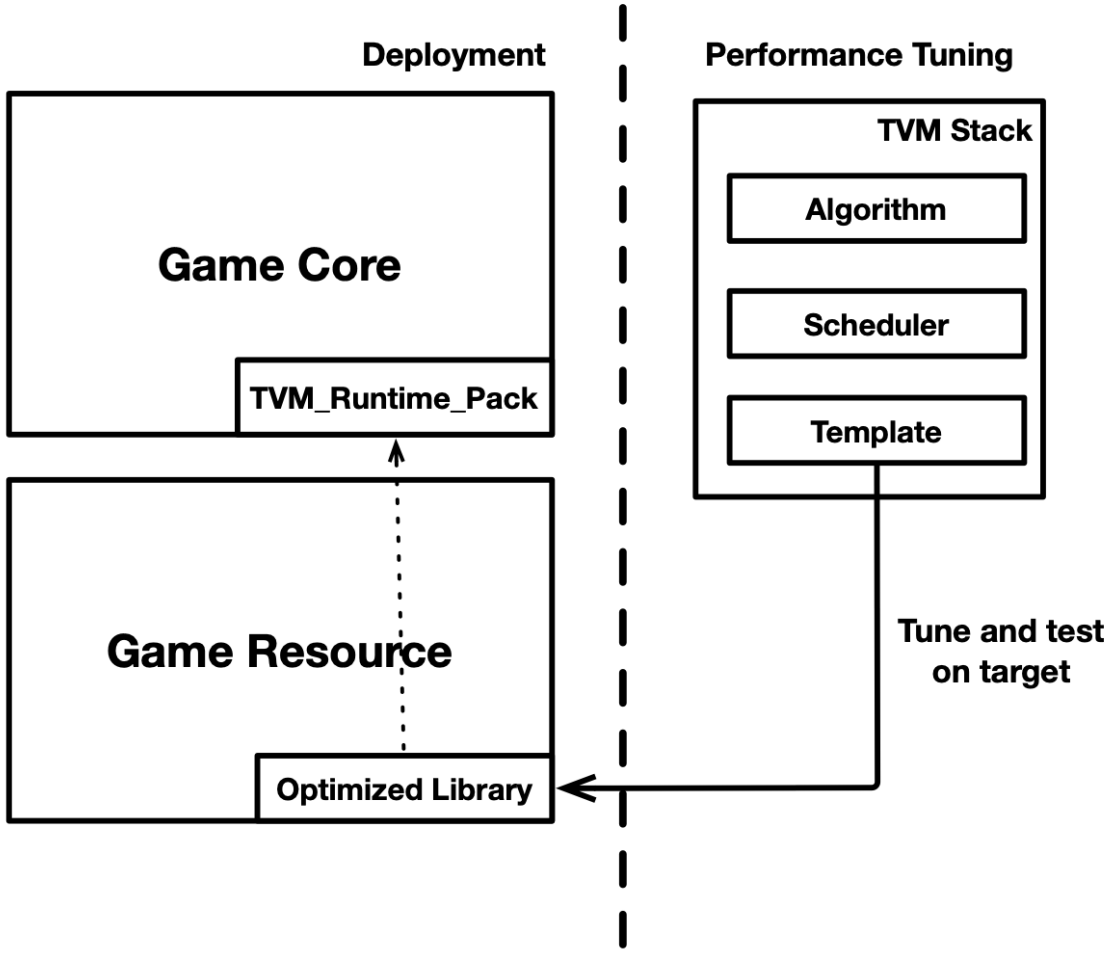
Observation 2 : The loop order of fused operator is identical with a matrix multiplication of shape (C, K) x (K, N)
Deduction 2 : The optimization techniques for fused operator can be borrowed from GEMM optimizations.

zjianzheng(草子健)

Methods – Integration

Tencent AI Lab

TVM Workflow

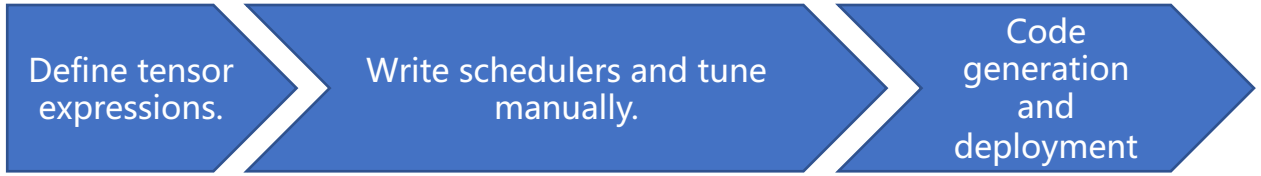


zijianzheng(草子健)

Methods – TVM approaches

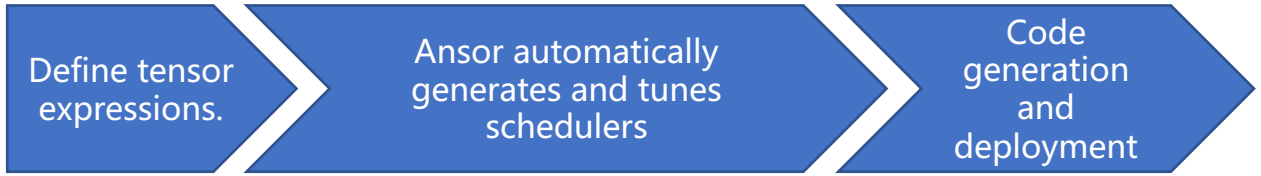
Tencent AI Lab

- **TVM with manual scheduler**
  - Manually write algorithms and schedulers.
  - TVM acts as cross-platform code generator.
  - Requires HPC knowledge but we are cool.



- **AutoTVM with external kernels**
  - Manually written inner kernels (for every hardware) and scheduler templates.
  - TVM tunes cache block parameters and generates code for outer loops.
  - Best performance with long search time.

- **Ansor: end-to-end TVM approach**
  - We only defined the algorithm.
  - Quickly converged to descent results.



Performance analysis

Eigen (baseline BLAS library)

- Cannot fuse operators.
- High latency when multiple game characters are present.

TVM with Hand Schedulers

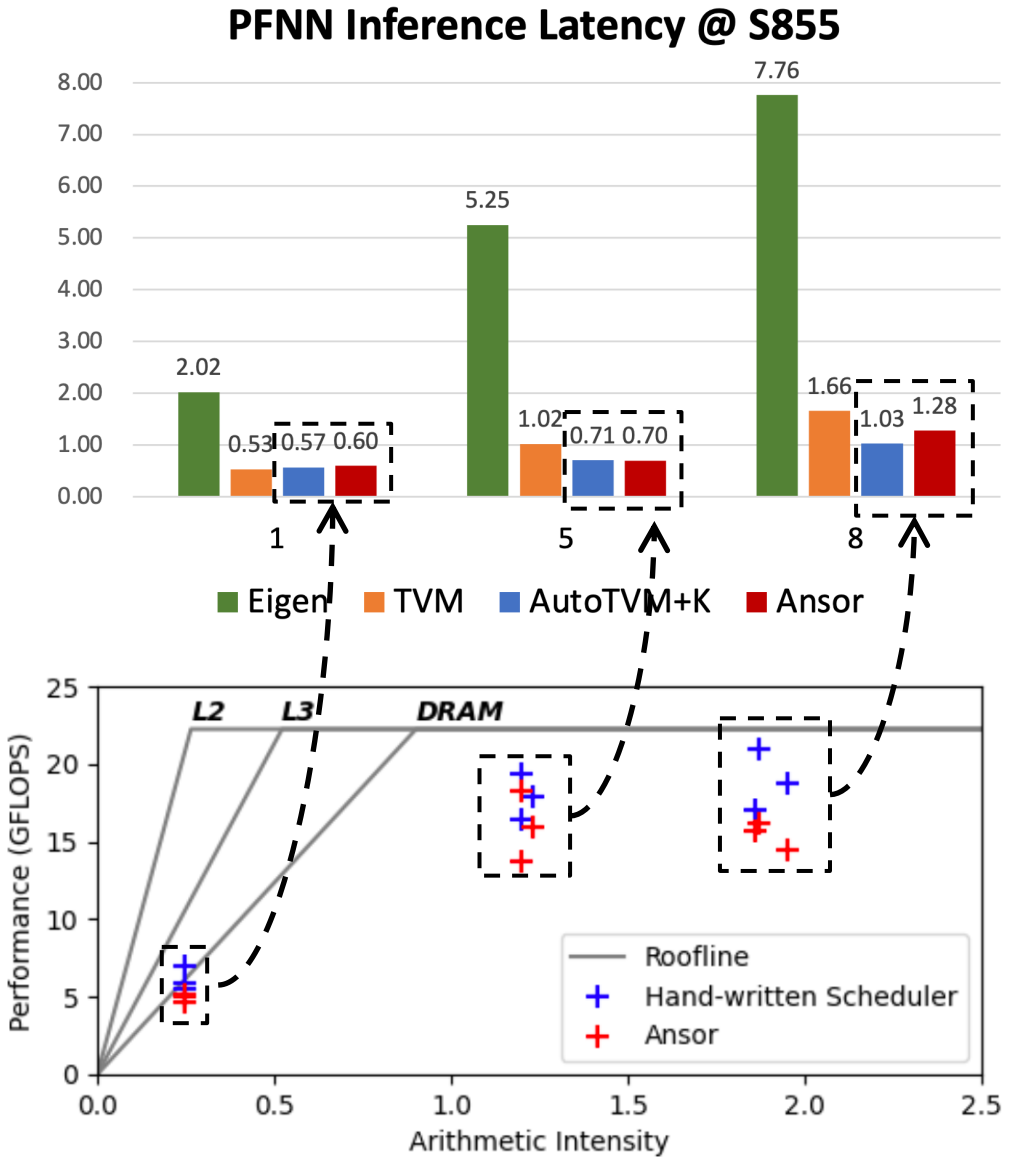
- Our TVM baseline.
- Developed in 2 hours for x86 and Arm processors.
- Parameters are not carefully tuned.
- Met our business goals.

AutoTVM with external inner kernels

- Extremely high performance (30% faster than hand schedulers).
- AutoTVM searches for best cache block size, loop order and register block size.
- We spent three weeks on development and tuning.

Ansor

- Close to AutoTVM approach but converged much faster (~2 hrs vs days).
- Consumes least human efforts.



Performance assessment conducted on a single big core on Qualcomm Snapdragon 855

zljianzheng(郑子健)

Lessons learnt – What went right?

Tencent AI Lab

Performance

- Operator fusion makes a big difference.
- Great performance even with roughly tuned schedulers.
- AutoTVM can effectively tune hand-written kernels.
- Ansor exhibits amazing performance in an end-to-end workflow.

Usability

- Packed runtime is easy to integrate and easy to customize.
- Cross-platform versions are easy to maintain.

zijianzheng(草子健)

Lessons learnt – What went wrong?

Tencent AI Lab

Performance

- LLVM has some corner cases in vectorization (will file an issue for this).
- Better cooperation of hand-written inner kernels and auto scheduling is important to HPC guys.

  (TensorIR is magic! We want more documents!)
- AutoTVM cannot converge stably, we made lots of trials to achieve best performance.

Usability

- Static library needs some hack to work. We exported static library in order to accommodate to all IDE versions.
- **Why TVM cannot be installed through pip or anaconda?**
- Shapes are always assumed to be static such that we cannot adjust character number dynamically.

Tencent AI Lab

Thank You – Q&A