ICLR 2022

Parallel Training of GRU Networks with a Multi-Grid Solver for Long Sequences

Gordon Euhyun Moon[1], Eric C. Cyr[2]

[1]Korea Aerospace University
[2]Sandia National Laboratories

Sandia National Laboratories

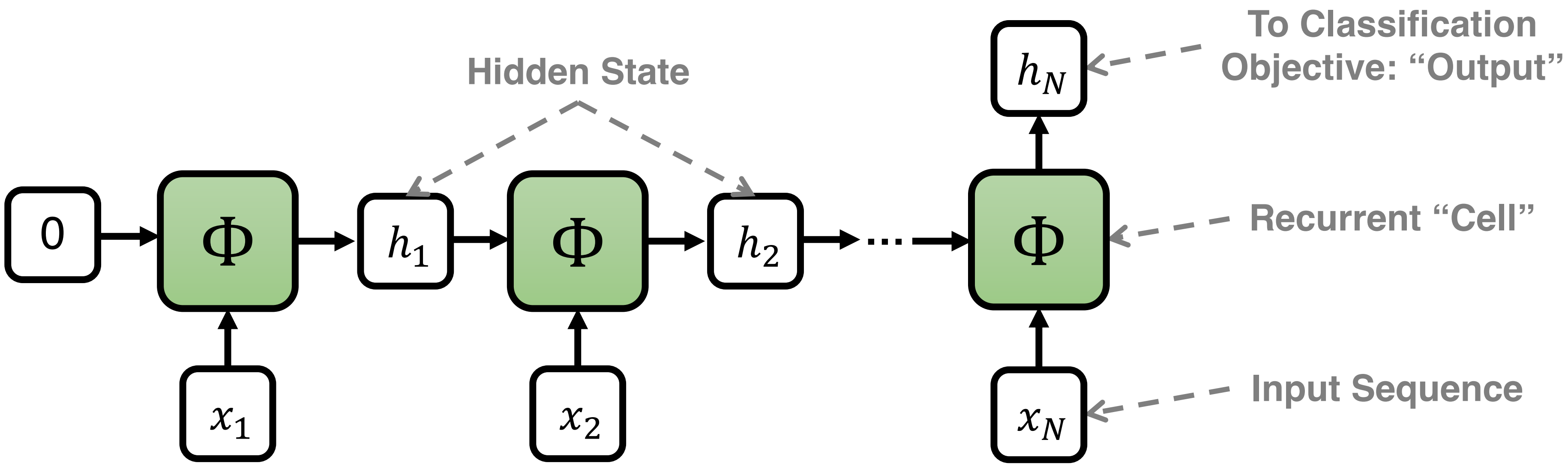
KOREA AEROSPACE UNIVERSITY

Recurrent Neural Networks (RNNs)

- **Problem: Classify a sequence, e.g. learn the mapping**

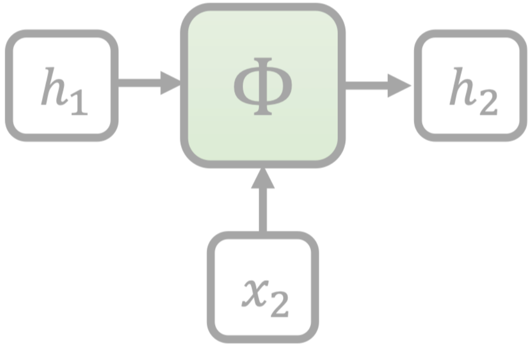
$$\Phi(\underbrace{x_1, x_2, \dots, x_N}_{\text{Sequence of N items}}) \rightarrow \underbrace{\{1, C\}}_{\text{One of C classes}}$$

- **Solution: Recurrent Neural Networks**
  - **Learn a neural network 'Φ' to produce a classifier**



Gated Recurrent Units (GRUs)

- **LSTMs and GRUs are two trainable types of RNNs**
  - Historically RNNs are unstable to train
  - "memory" remembers important features in the sequence
  - "forget" gate eliminates some redundant/irrelevant from the sequence
- **Classic GRU:**
  - $\boldsymbol{h}_*$: hidden state
  - $\boldsymbol{x}_*$: input sequence
  - $\boldsymbol{W}_*$ and $\boldsymbol{b}_*$: learnable network parameters

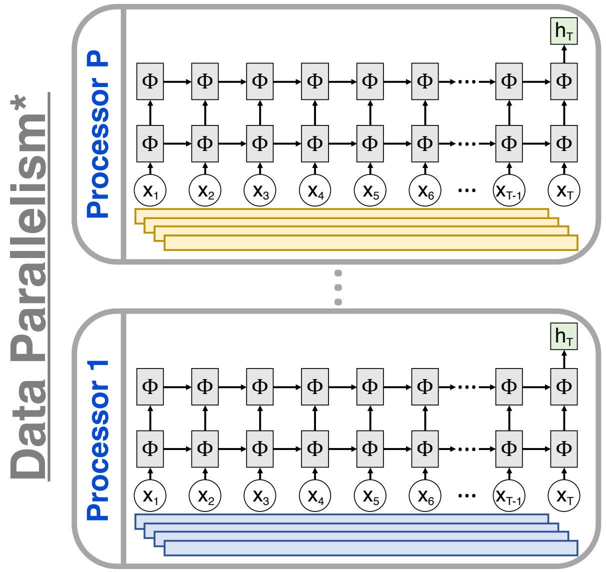


$$\left.\begin{aligned} r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \\ z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \\ n_t &= \varphi(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{t-1} + b_{hn})) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot n_t \end{aligned}\right\} h_t = \Phi(x_t, h_{t-1}; \xi)$$

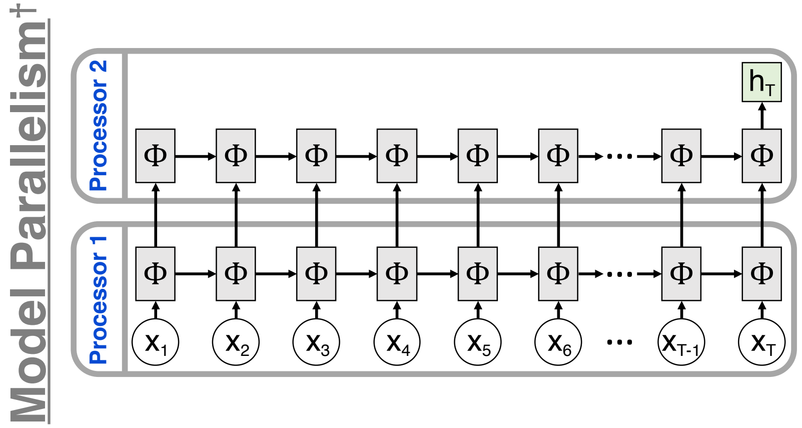
Hadamard Product

Use Parallelisms to Accelerate Training of RNNs

- **Data Parallelism for RNNs**
  - Distribute a batch of sequences over processors
  - Problem:
    - The accuracy of mini-batch stochastic gradient descent (SGD) degrades as the size of mini-batch increases



- **Model Parallelism for RNNs**
  - Distribute layers across multiple processors
  - Problem:
    - The number of usable processors is limited to the number of layers



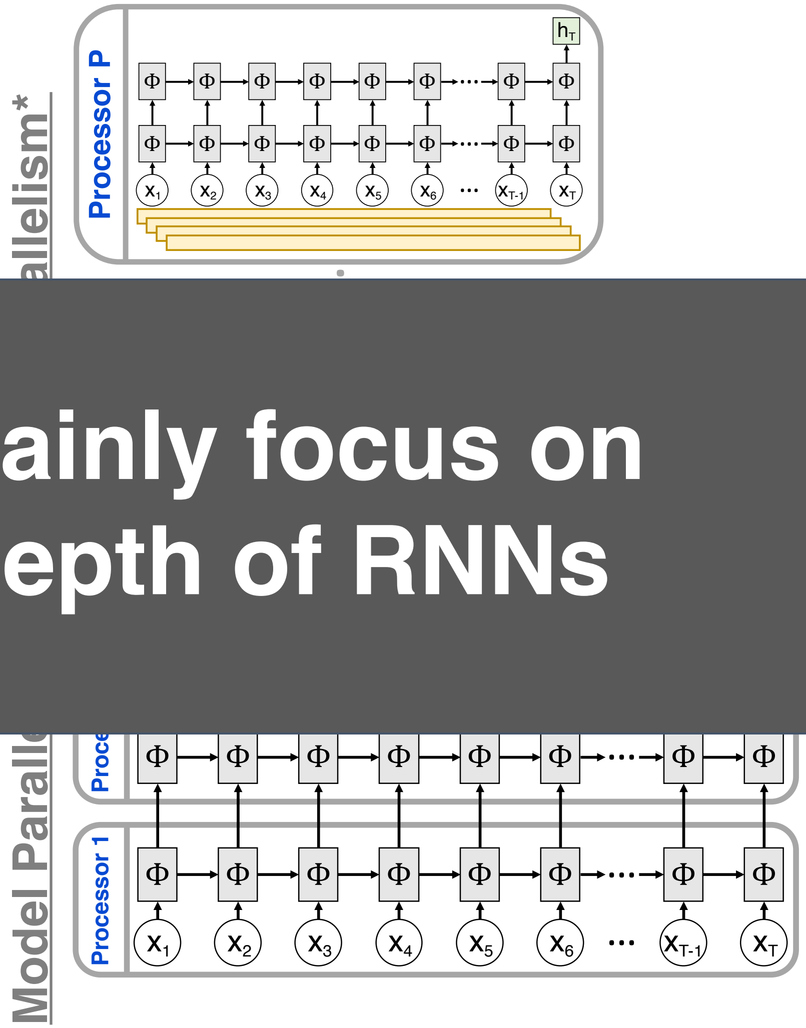
*You et al. “Large-Batch Training for LSTM and Beyond” SC 2019
†Wu et al. “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation” arXiv 2016

Use Parallelisms to Accelerate Training of RNNs

- **Data Parallelism for RNNs**
  - **Distribute a batch of sequences over processors**

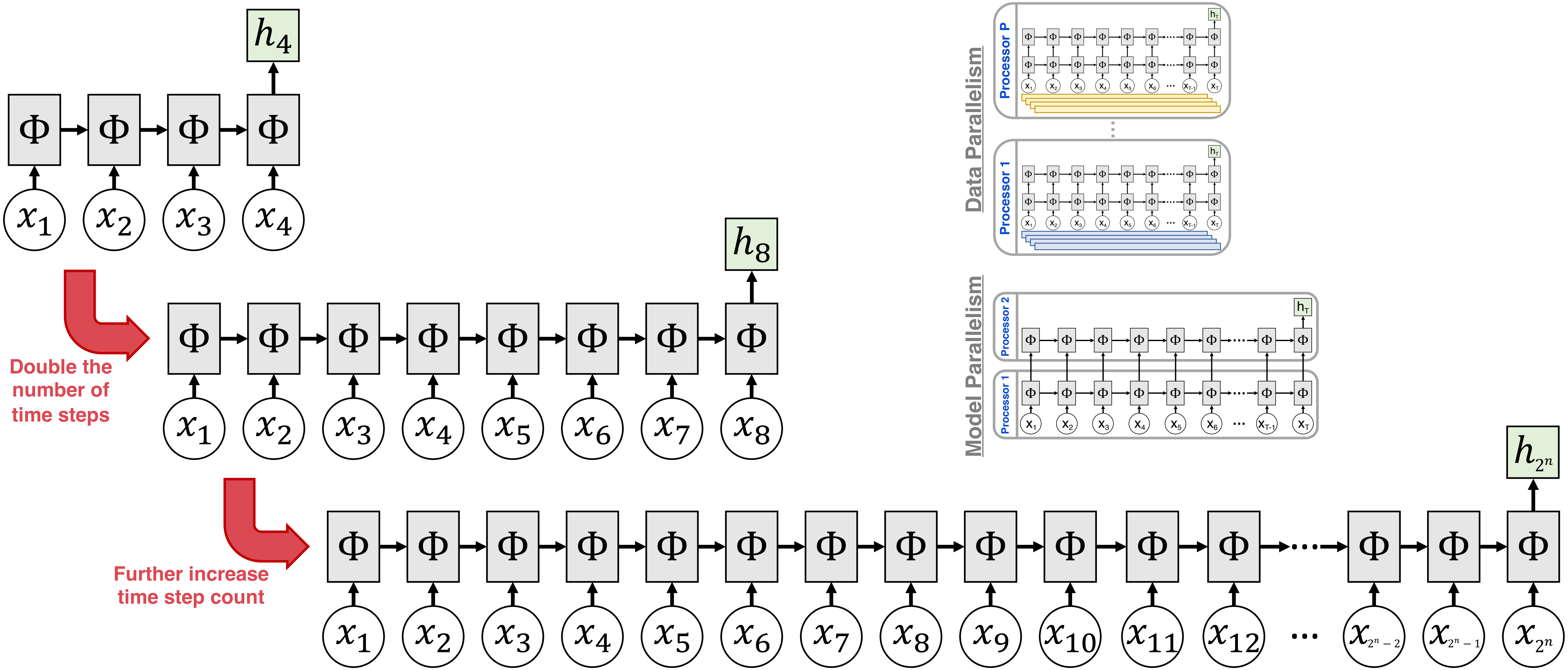
Current parallelisms for RNNs mainly focus on increasing size of dataset and depth of RNNs

  - **Problem:**
    - **The number of usable processors is limited to the number of layers**

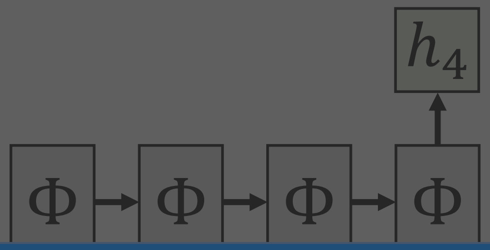


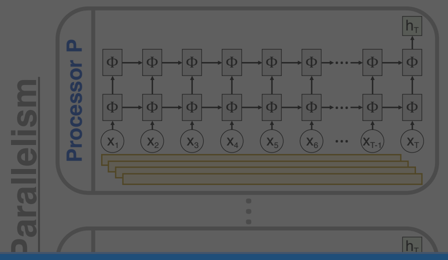
*You et al. “Large-Batch Training for LSTM and Beyond” SC 2019
†Wu et al. “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation” arXiv 2016

What About Sequence Length?

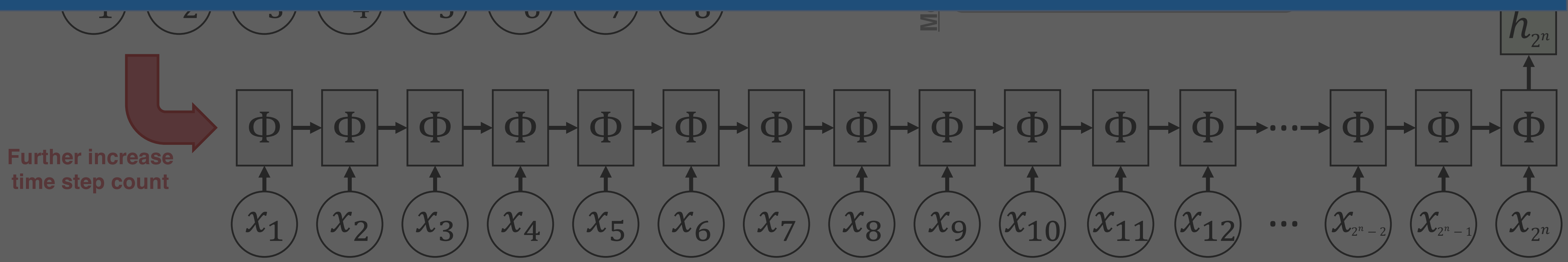


What About Sequence Length?

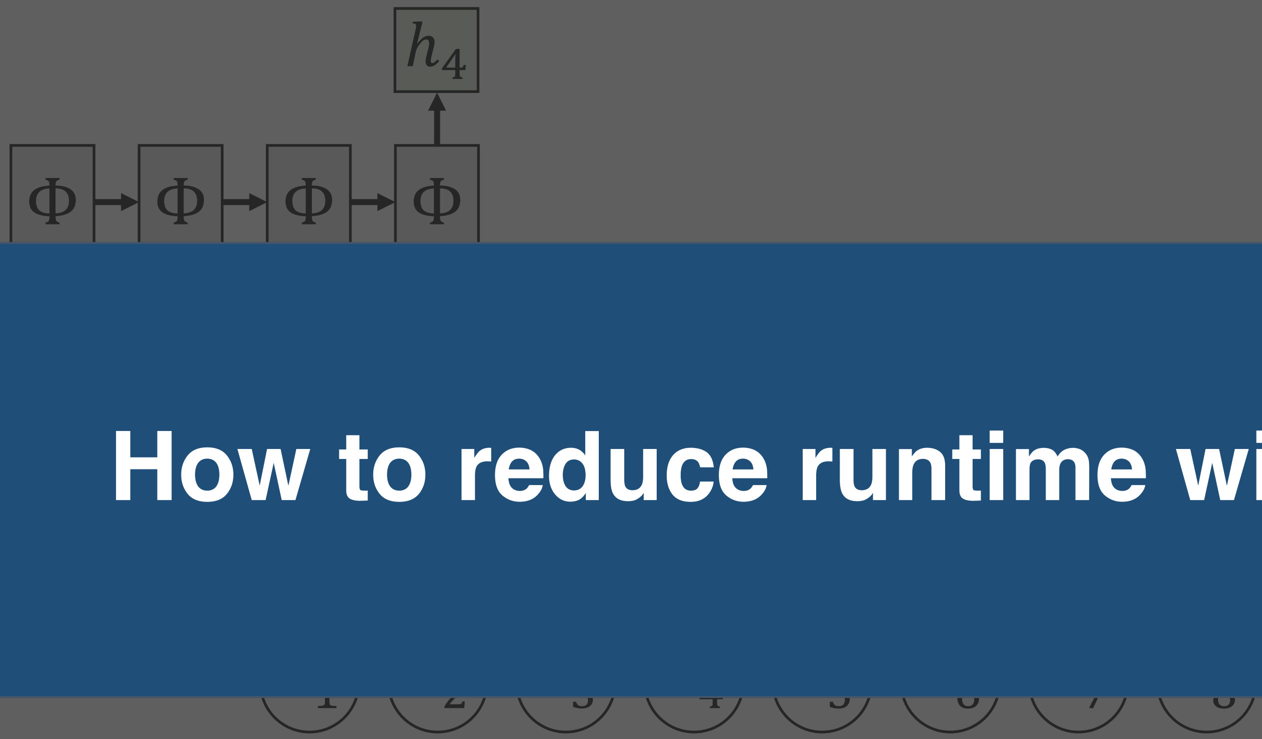


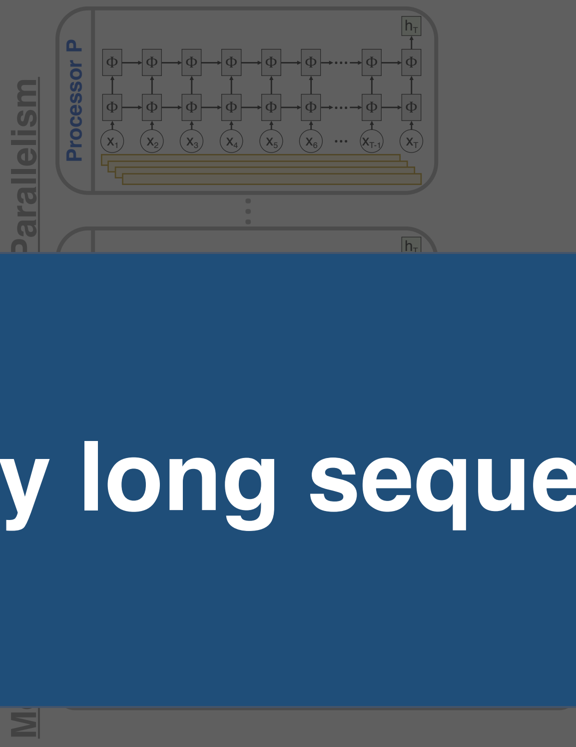


The training execution time of current parallelisms increases with the sequence length!

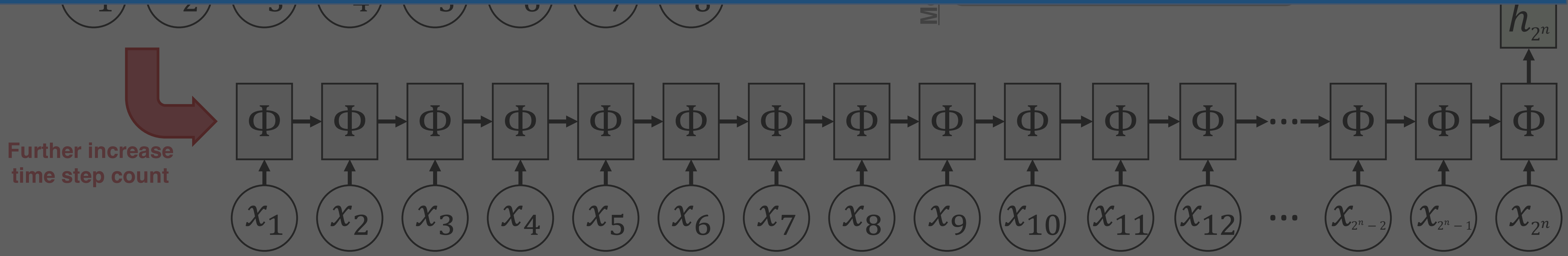


What About Sequence Length?

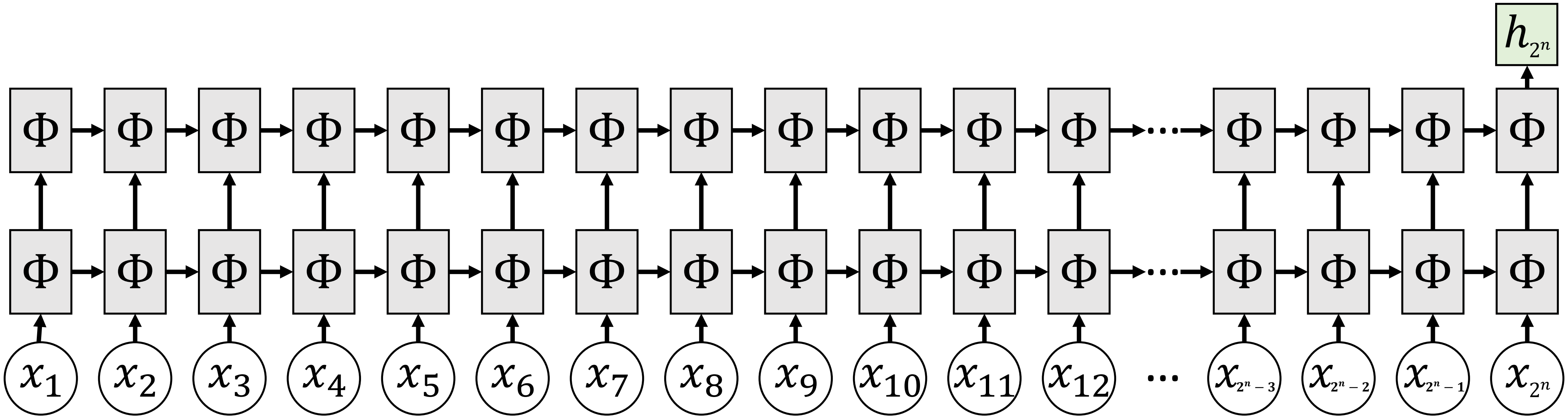




How to reduce runtime with very long sequences?



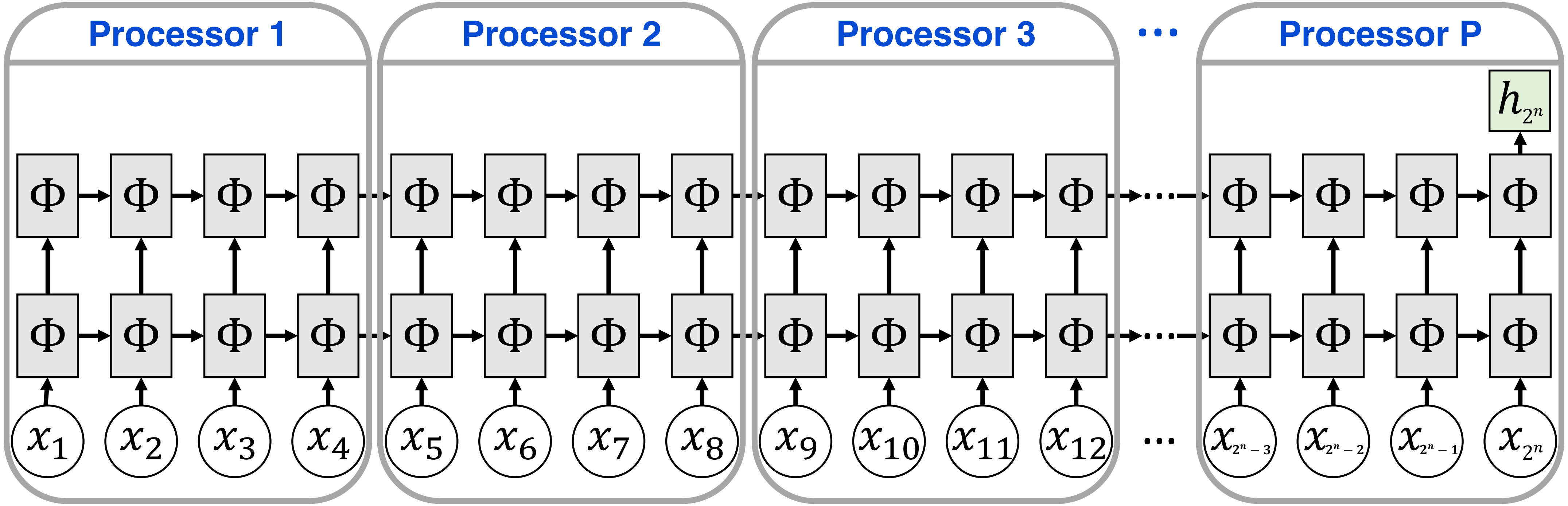
Our Approach



where
2^n = sequence length

Our Approach: Parallel-in-Time Training

- Distribute shorter sub-sequences across multiple processors
- MGRIT algorithm enables accelerated parallel training of RNNs on longer sequences → permitting growth in time dimension



where
2^n = sequence length
P = $2^n/4$ = number of processors

GRUs to ODEs

- We rewrite the update with a time step update (assume $\gamma = 1$)

$$\begin{aligned} h_t = \Phi_\gamma(x_t, h_{t-1}; \xi) &= z_t \odot h_{t-1} + (1 - z_t) \odot n_t \\ &= h_{t-1} + \gamma(-(1 - z_t) \odot h_{t-1} + (1 - z_t) \odot n_t) \end{aligned}$$

- Taking $\gamma \to 0$, we arrive at an ODE form

$$\partial_t h_t = \underbrace{-(1 - z_t) \odot h_t}_{\text{Stiff mode: Collapsing onto multi-rate asymptotic (this is a dissipation term!)}} + \underbrace{(1 - z_t) \odot n_t}_{\text{Introduction of new sequence information}}$$

Implicit GRUs

- **Stiff mode suggests a problem for traditional GRU's with large γ:**
  - **This will be a problem for coarse grids in parallel-in-time!**
  - **In the Neural ODE case, we took bigger time steps on coarse grids**
  - **Here, we will take $\gamma = 1$, coarse grid will likely be unstable!**

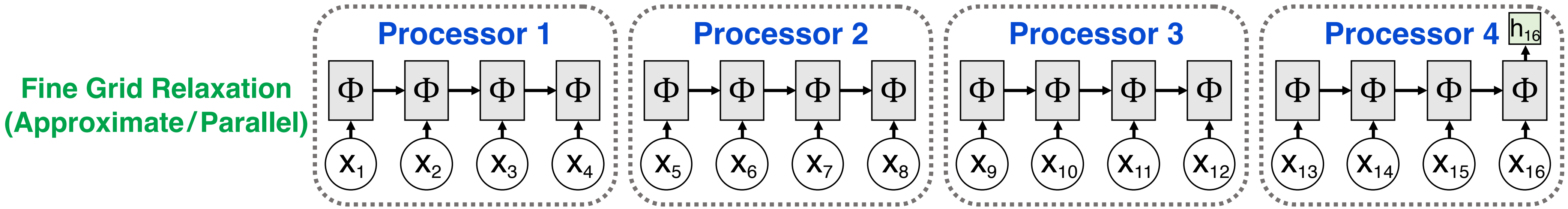
- **Remedy: a new "Implicit GRU", default to $\gamma = 1$:**

$$\left(1 + \gamma(1 - z_t)\right) \odot h_t = h_{t-1} + \gamma(1 - z_t) \odot n_t$$

  - **Because stiff mode is implicit, this new formulation will be stable for "large" γ**
  - **We will leverage this in a MGRIT solver**

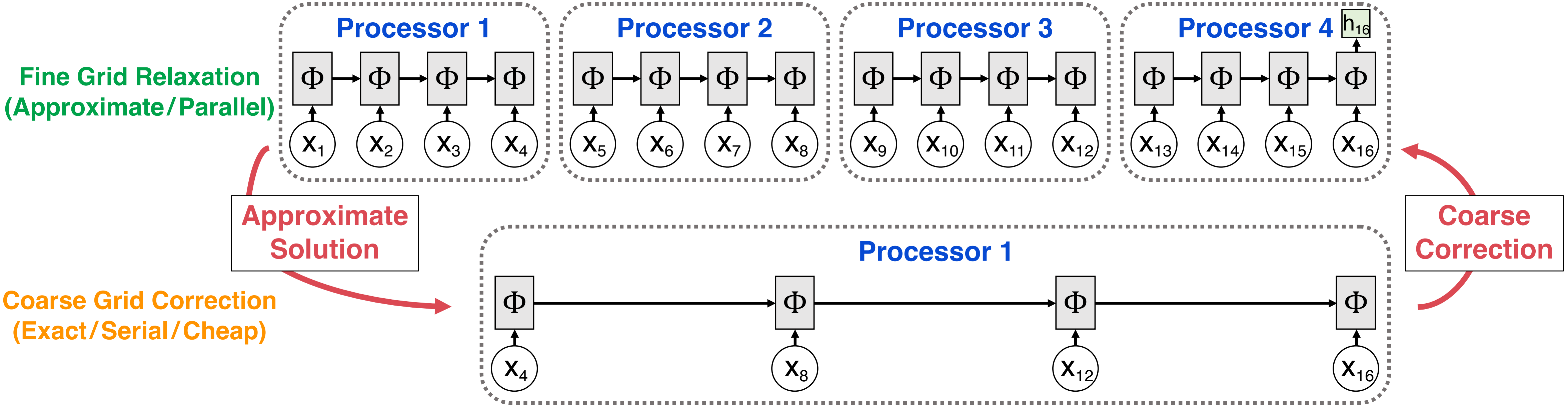
Parallel-in-Time Training – A Multigrid Approach

- **Multi-grid algorithm uses "divide and conquer" approach to inference**
  - **"Fine Grid Relaxation": Fixes local errors between time steps – embarrassingly parallel**



Parallel-in-Time Training – A Multigrid Approach

- Multi-grid algorithm uses "divide and conquer" approach to inference
  - "Fine Grid Relaxation": Fixes local errors between time steps – embarrassingly parallel



  - "Coarse Grid Correction": Fixes global errors – serial inference on smaller network

Multigrid is applied for both forward and back propagation

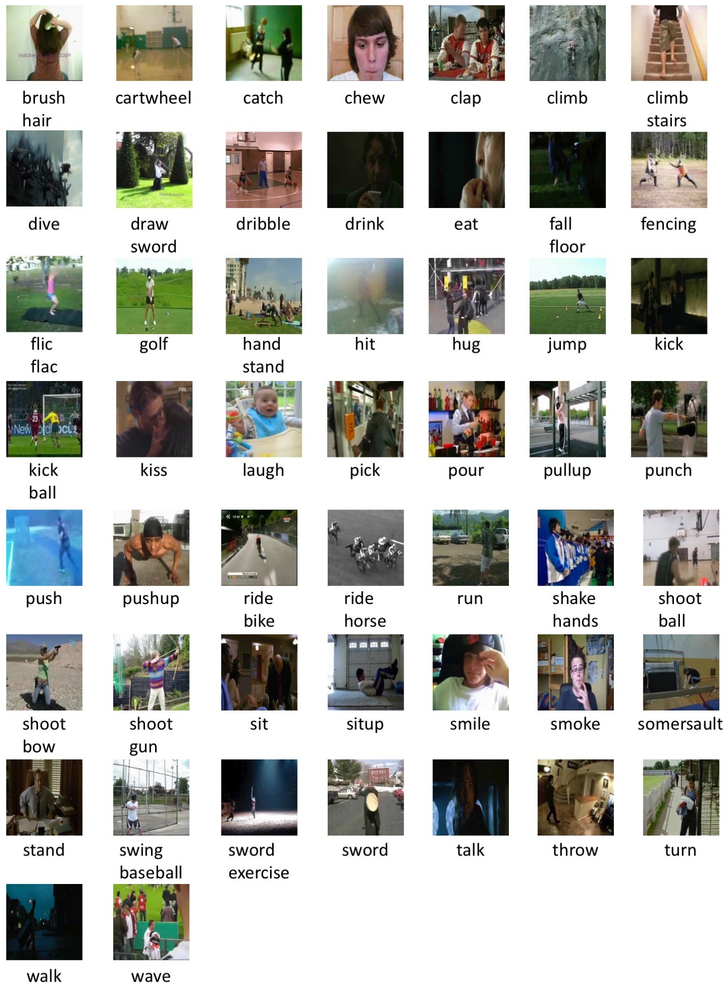
HMDB51: A Large Human Motion Database

- **Task: Classify human activity in each video**

- **Full Database:**
  - ~6700 clips distributed in 51 classes
  - Train/Test split: 6053/673
  - Frame count ranges from 20 to more than 200
  - We truncate/pad each video to 128 frames
  - We use 240×240 pixels in each frame

- **Subset Database (we also run this):**
  - 6 classes: `chew, eat, jump, run, sit, walk`
  - Train/Test split: 1157/129
  - Frame count ranges from 20 to more than 200
  - We truncate/pad each video to 128 frames
  - We use 240×240 pixels in each frame

Examples of the 51 Action Classes



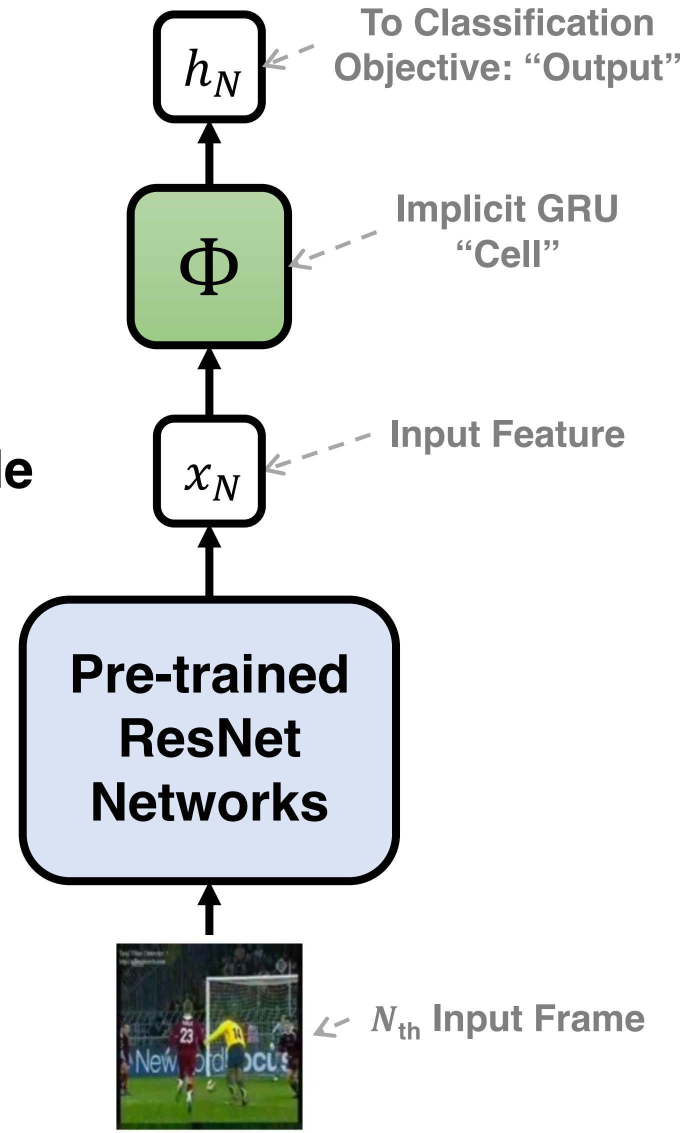
Kuehne et al. “HMDB: A Large Video Database for Human Motion Recognition” ICCV 2011
https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/

Video Classification with Implicit GRU

- **Implicit GRU with ResNet Preprocessor:**
  - Pre-trained ResNet18, 34, and 50 networks are used for generating low-dimensional input features for the GRU
    - E.g., ResNet18 computes 1000 features per frame

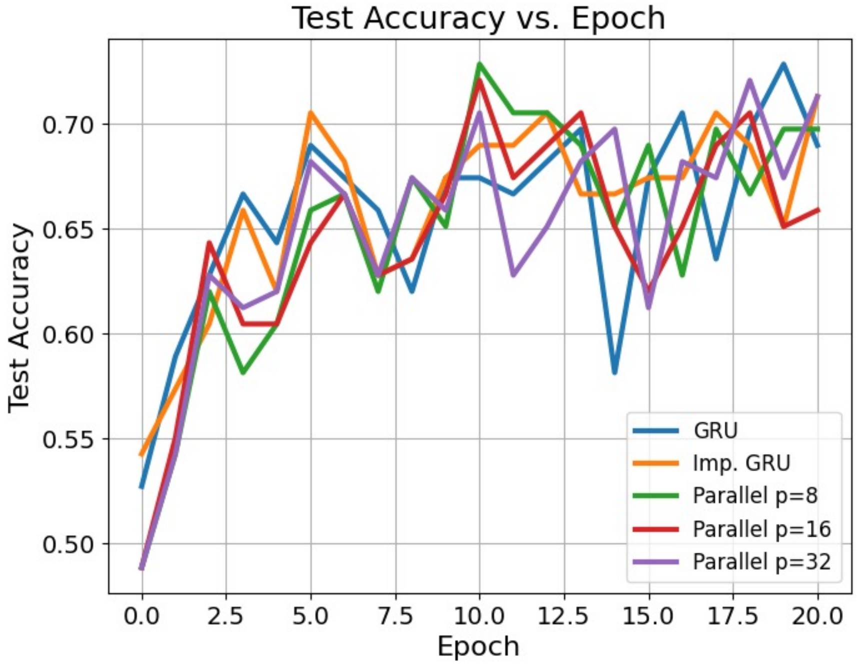
- **Computing Platform (Sandia's Attaway machine):**
  - 2.3 GHz Intel Xeon, 2 Sockets, 18 Cores each: 36 cores per node
  - Run with 9 OpenMP threads per MPI rank (4 ranks per node)

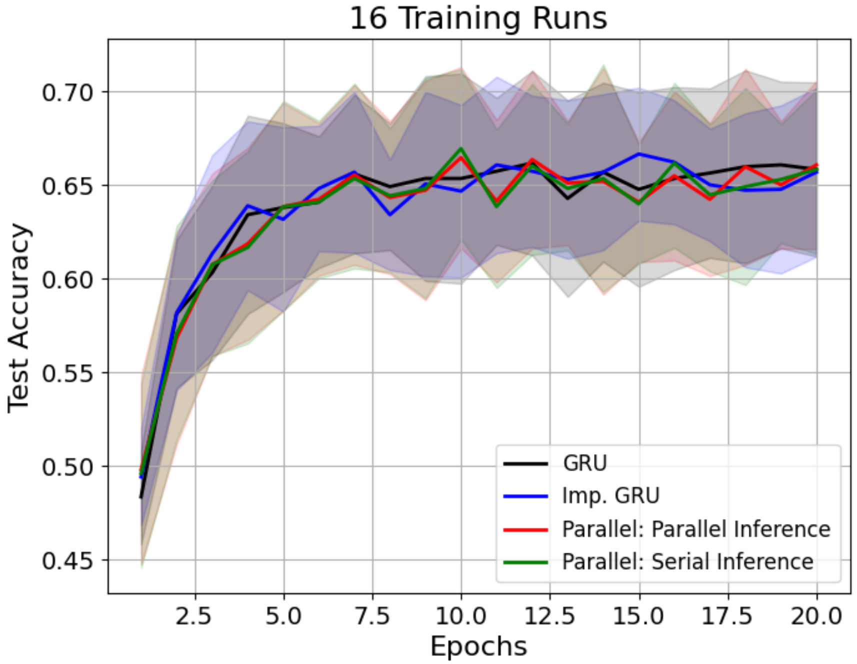
- **Training Details:**
  - Batch size of 100
  - ADAM optimizer with learning rate of 10^{-3}
  - ResNet18, 34, and 50 networks are not applied on coarse grids
  - Input image feature is computed once



Performance Comparison: Test Accuracy

- **Comparison of Classic, Implicit, and Parallel GRU**



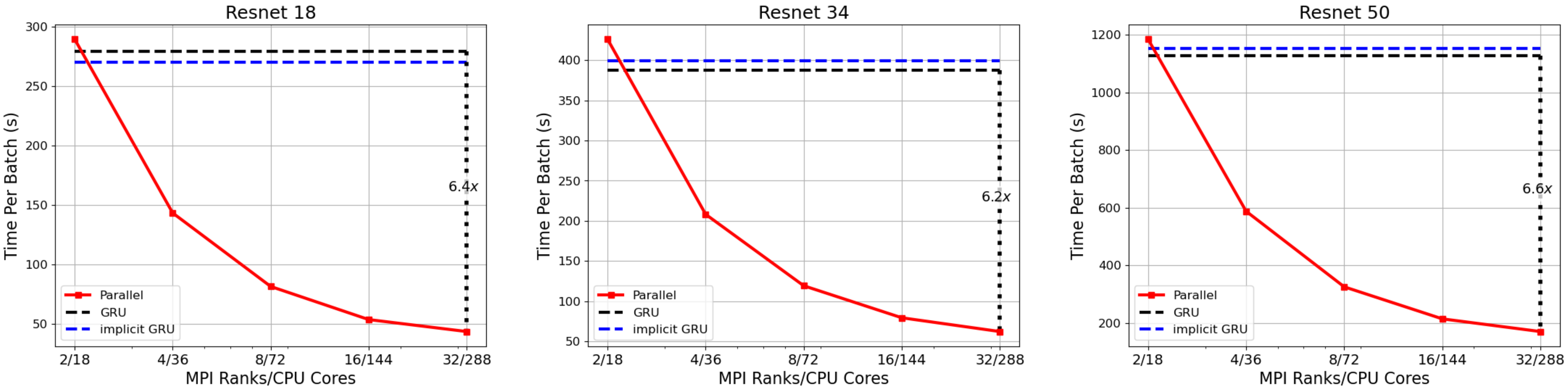


The higher the better

Serial Classic, Serial Implicit, and Parallel Implicit GRU make little difference in test accuracy

Performance Comparison: Speedup

- **Comparison of Classic, Implicit, and Parallel GRU with different preprocessors**

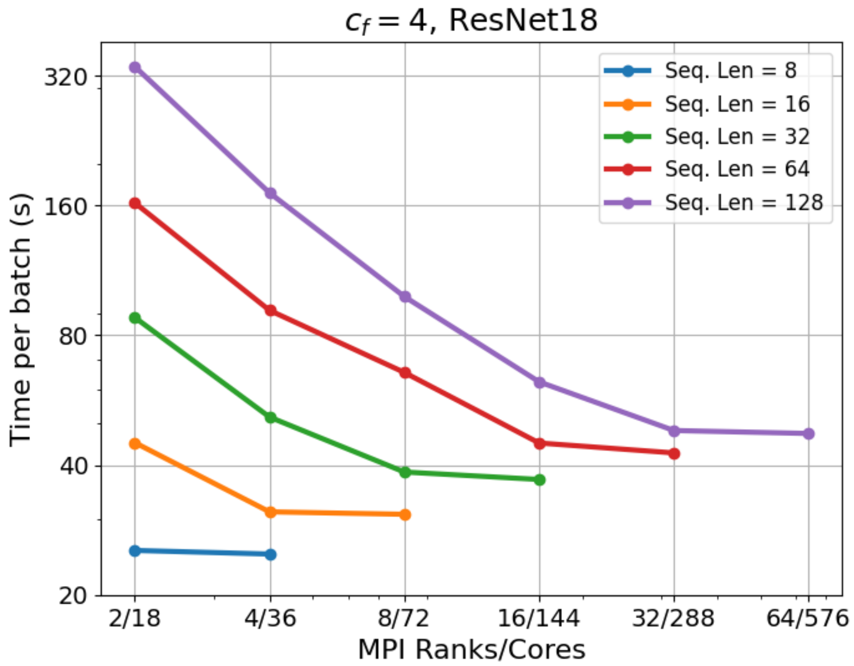


The lower the better

Parallel Implicit GRU achieves up to 6.6× speedup over the Classic GRU

Performance Comparison: Speedup

- **Scalability with long sequences**



The lower the better

Parallel Implicit GRU achieves additional speedup as the sequence length increases

Summary and Conclusions

- **Presented a parallel-in-time algorithm for training GRU**
  - **Current parallelisms mitigate increased data size and network depth of GRU**
  - **We developed new GRU parallel training procedure**
    - **Increased runtimes with sequence length can be ameliorated by parallel-in-time approach**
    - **We traded inexactness for performance with multigrid algorithms**
  - **Developing new parallel training algorithm is imperative for different types of DEEPER deep neural networks**

Please check out our paper to learn more about this work!

U.S. DEPARTMENT OF ENERGY | Office of Science

Thanks to the DOE Office of Science ASCR Early Career Research Program for supporting this work!