KU LEUVEN

Can deep neural networks learn process model structure?

An assessment framework and analysis

ML4PM

Jari Peeperkorn[1], Seppe vanden Broucke[2,1], and Jochen De Weerdt[1]

[1] Research Center for Information Systems Engineering (LIRIS), KU Leuven, Leuven, Belgium

[2] Department of Business Informatics and Operations Management, Ghent University, Ghent, Belgium

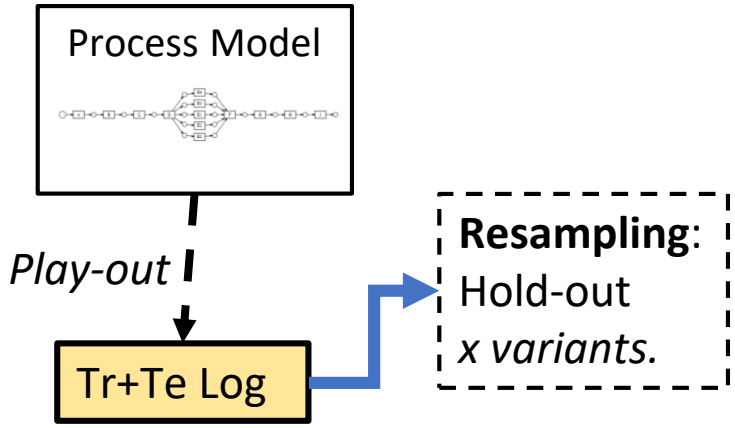
Introduction

- Surge in predictive process monitoring research
  - Focus on Deep Learning techniques: Recurrent neural networks (RNN)
  - Long short-term memory network (LSTM)

- Can LSTMs actually learn process behavior from a (possibly incomplete) set of traces?

- Framework:
  - Variant based resampling procedure → unique control-flow variants
  - Train an RNN on next event prediction
  - Metrics for fitness, precision and generalisation

Framework



Resampling:
Hold-out
x variants.

Idea: isolate correct/wanted behavior, not seen by RNN

Training Log

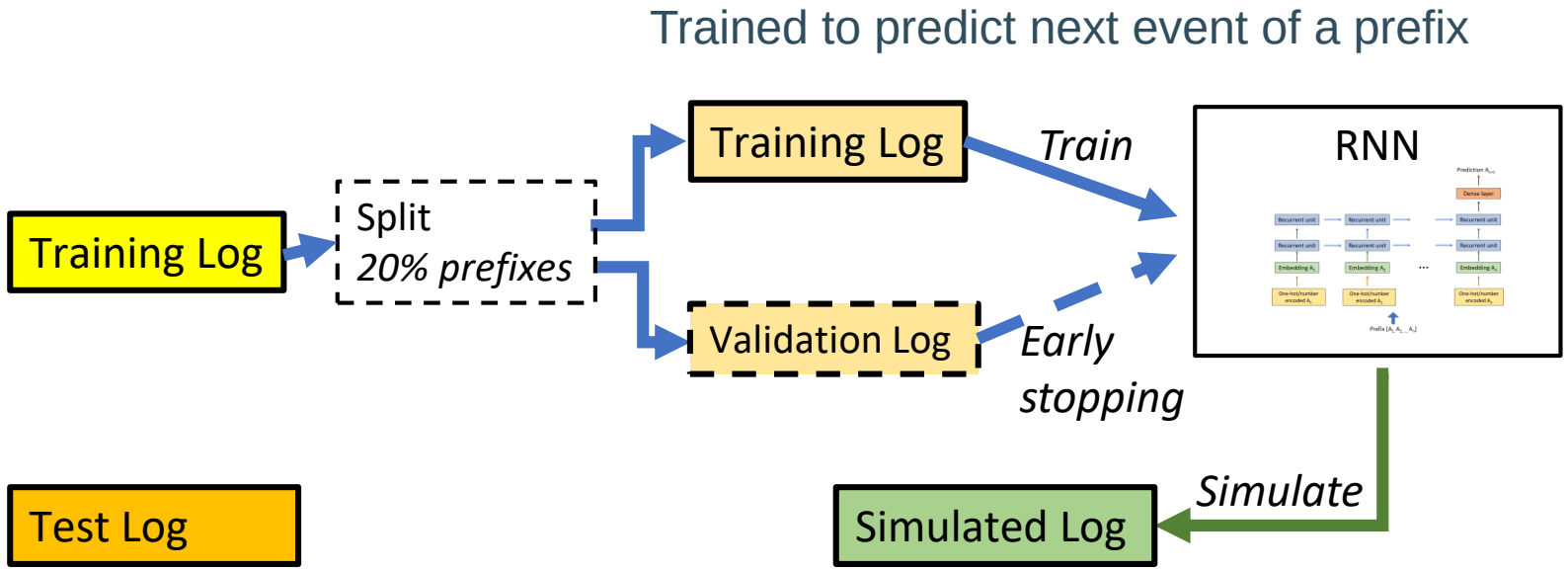
- Converted into prefixes of different sizes
- Leave one variant out cross-validation (LOVOCV)
- Leave 20% out
- …

Test Log

Left out variants

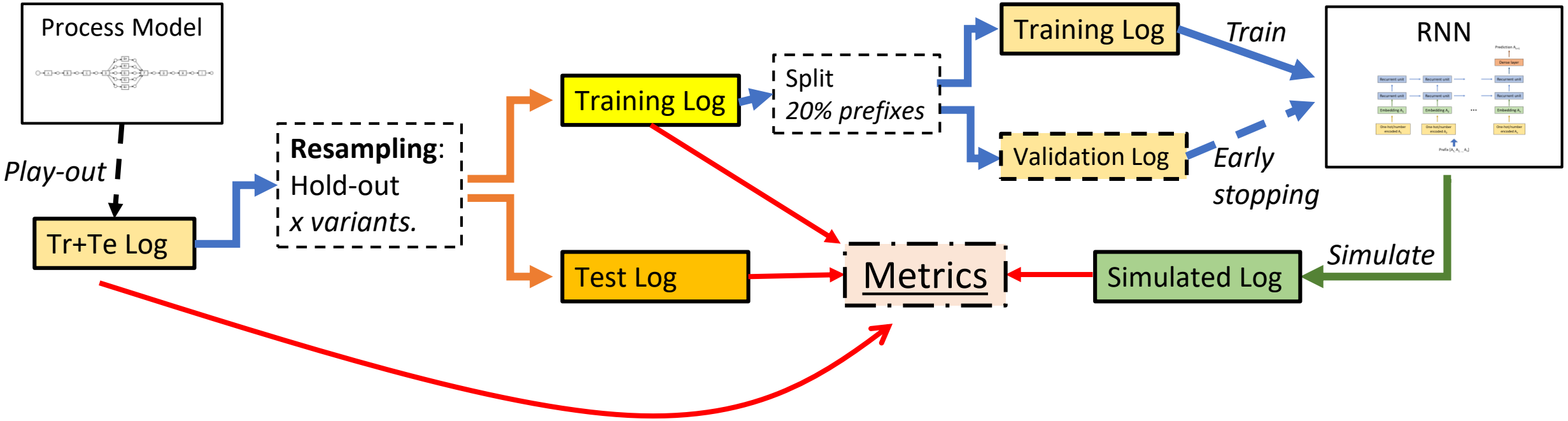
Just in this research setting.

Framework



- Start with a prefix only containing BOS token
- Prefix as input for the RNN
- RNN Outputs a probability for each activity type to be the next event
- Using these probabilities: sample next event and append to prefix
- Repeat until EOS token in reached

Framework



Metrics

$$Fitness = \sum_{v \in Var(Tr)} \frac{Min(Occ(v,Sim),\ Occ(v,Tr))}{|Tr|}$$

- To what extent are all variants present in *Training log* also present in *Simulated log*?
- We want the RNN to learn and be able to replicate all behavior in *Training log*
  - Only *Training log* ↔ Generalisation
- We also expect: $\forall\ v \in Var(Tr): Occ(v,Sim) \approx Occ(v,Tr)$
  - Punish for i.e. under-representation in Simulated Log

Metrics

$$Precision = \sum_{v \in Var(Sim)} \frac{Min(Occ(v,Sim),\ Occ(v,Tr+Te))}{|Sim|}$$

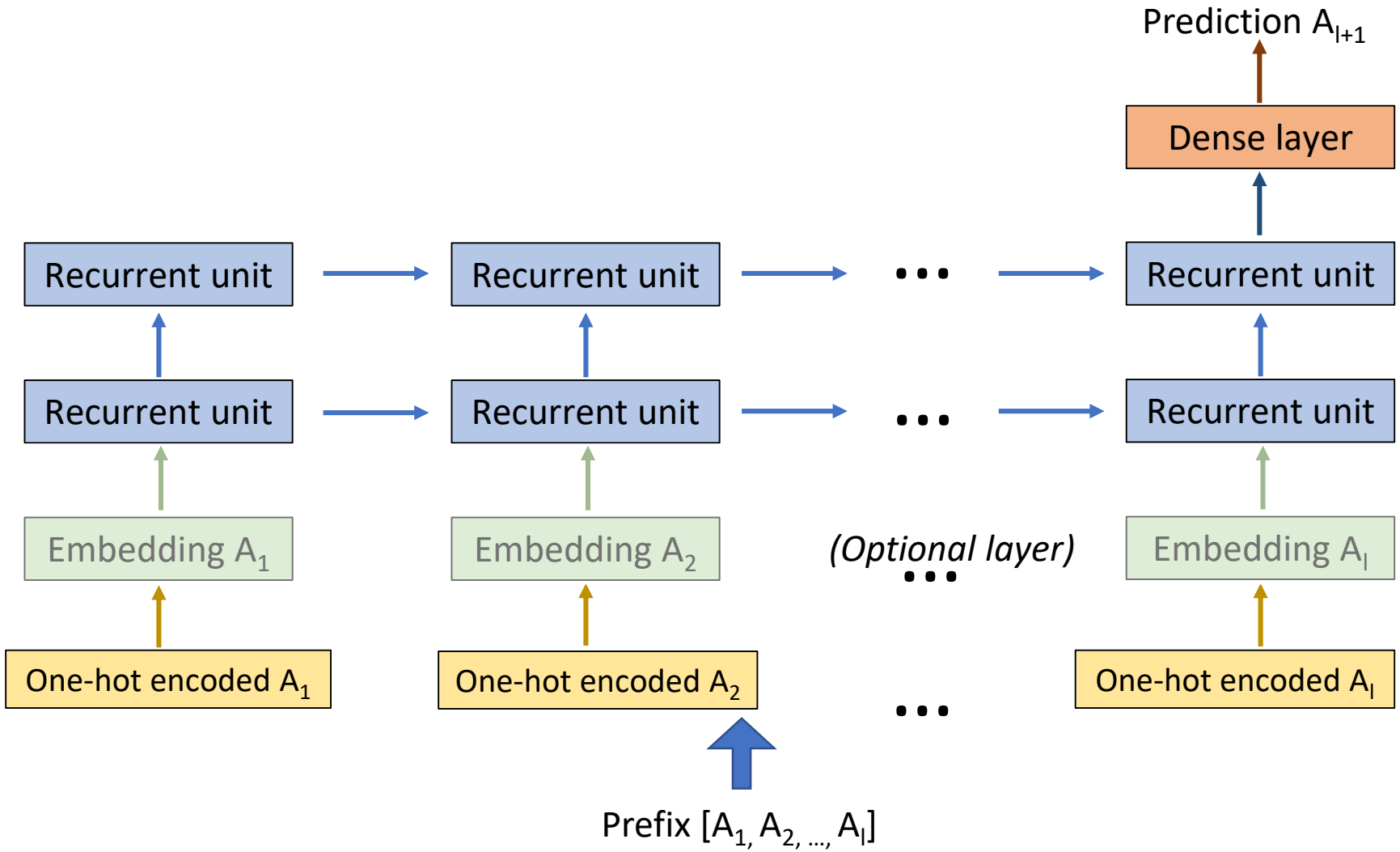
- To what extent are all variants present in *Simulated log* also present in original *Train+Test log*?
- We don't want the RNN to allow for too much extra (wrong) behavior
- We also expect: $\forall\ v \in Var(Sim): Occ(v,Sim) \approx Occ(v,Tr+te)$
  - Punish for i.e. over-representation in Simulated Log

Metrics

$$Generalisation = \sum_{v \in Var(Te)} \frac{Min(\mathrm{Occ}(v, Sim),\ Occ(v, Te))}{|Te|}$$

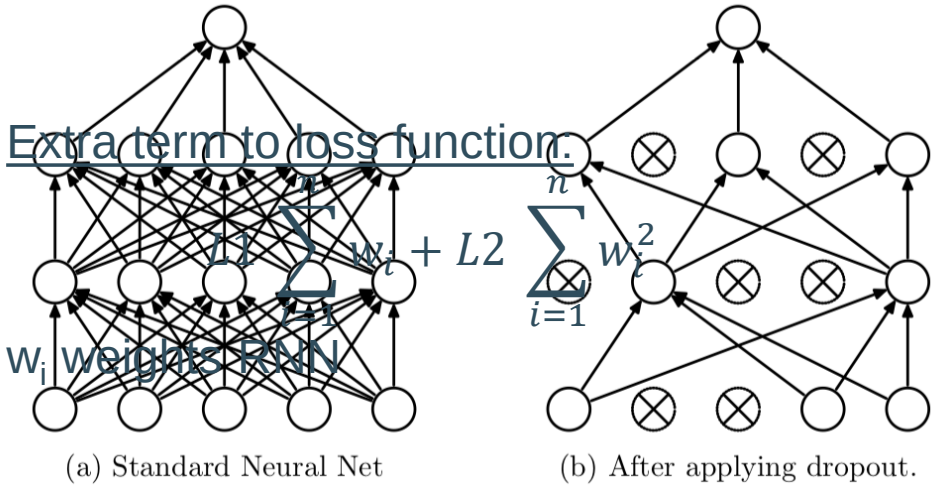
- To what extent are all variants present in the unseen *Test log* also present in the *Simulated log*?
- We want the RNN to learn and reproduce correct but unseen behavior
- We also expect: $\forall\ v \in Var(Te)$: $Occ(v, Sim) \approx Occ(v, Te)$
  - The occurrences in the *Test log* should be reproduced in the *Simulated log*

RNN - LSTM



Hyperparameters

- Number of recurrent layers
- Hidden unit size
- Embedding layer: yes or no?
  - Embedding size
- Dropout rate
- Regularization: L1 and L2
- …



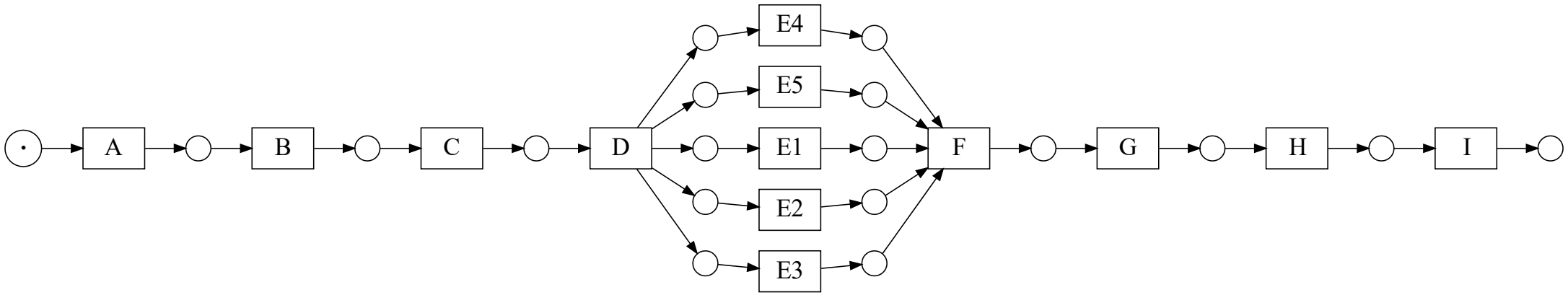
Source: https://ai-pool.com/a/s/dropout-in-deep-learning

Experimental Setting

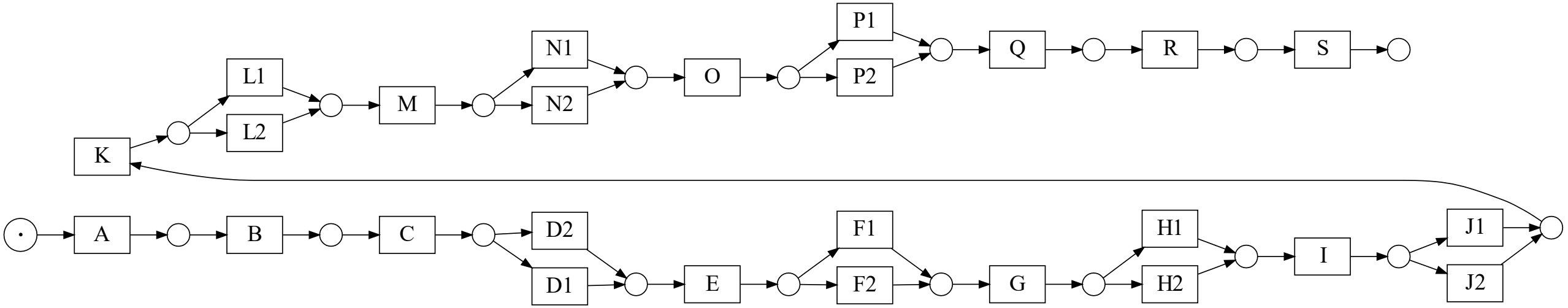
- Hyperparameter search

- Six trivial process models: reflecting essential control-flow constructs
  - Leave-one-variant-out cross-validation (LOVOCV)
    - Over all variants
  - Leave 20% out
    - 3 times, randomly selected variants

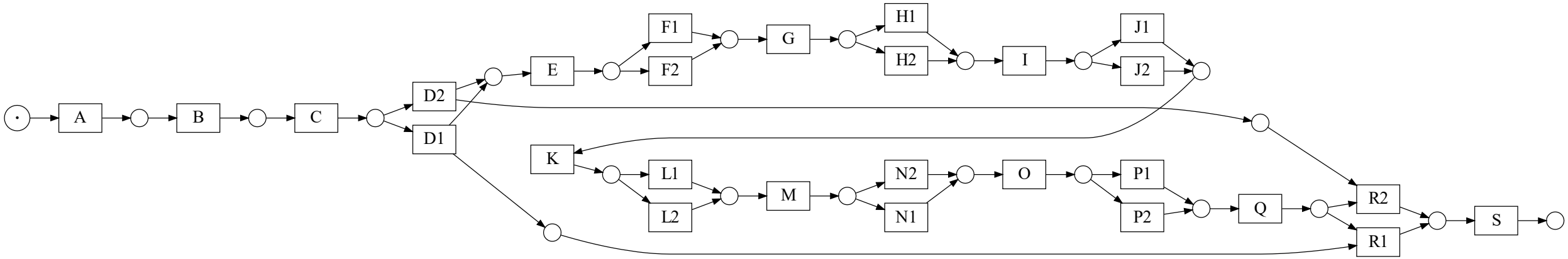
Model 1: parallel model with 120 variants



Model 2: Multiple XOR splits with 128 variants

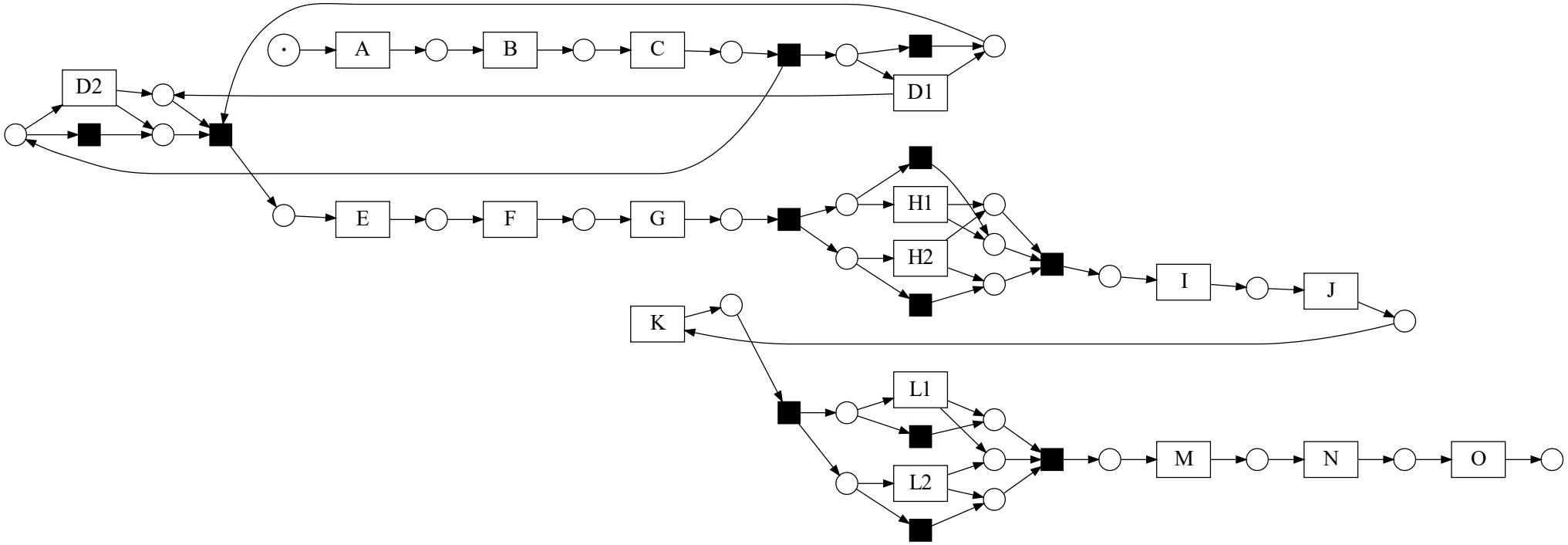


Model 3: multiple XOR splits, and a long-term dependency, 128 variants



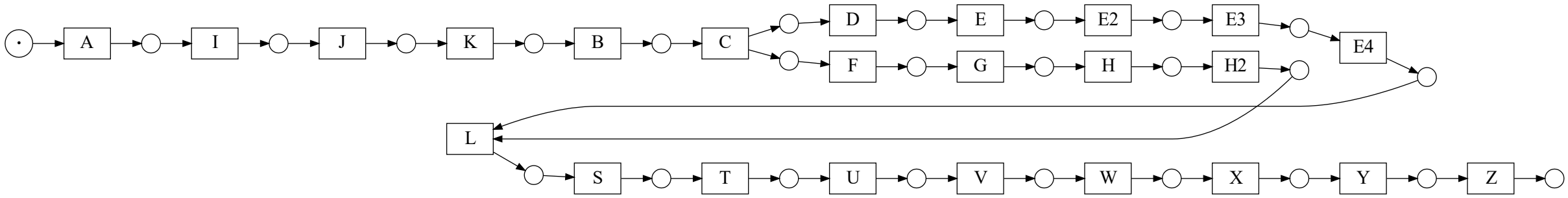
Different frequency!

Model 4: model with multiple IOR splits, 64 variants



Different frequency!

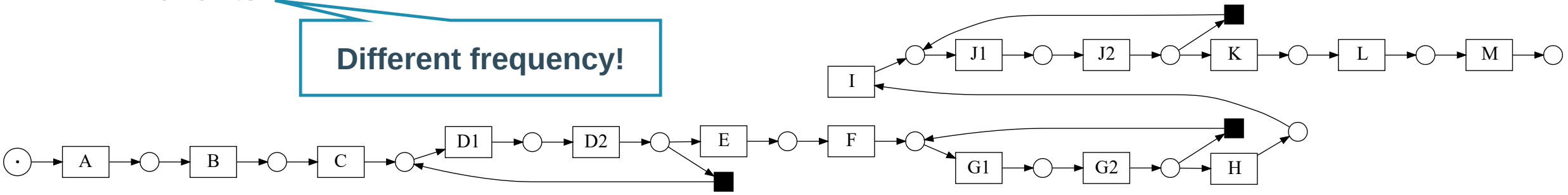
Model 5: model one longer split in parallel, 126 variants



Model 6: Model with 3 different loops (length 2)

Limited marking visit to max. 3
-> 27 variants

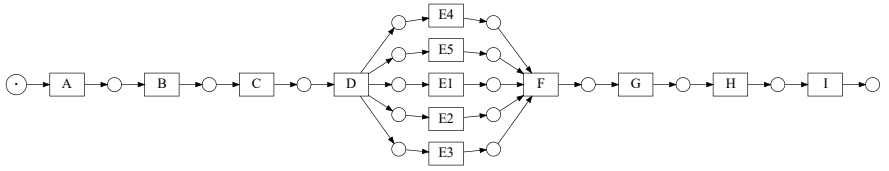
Different frequency!



Hyperparameter search

Hyperparameter	Values
Embedding Layer	Yes, No
Number of LSTM Layers	1, 2
LSTM Layer Size	16, 32, 64
L1 and L2	0.0, 0.00001, 0.0001, 0.001, 0.01
Dropout	0.0, 0.2, 0.4

Table 1: The hyperparameter values used in the grid search.



- LOVOCV on Model 1 (not over all variants, took 8 random variants)
  ⇒ Not necessarily the most optimal parameters for all models/event logs

Hyperparameter search

- Take setting with highest average score (Fitness + Precision + Generalisation)
- One layered LSTM, hidden size 32, L1 and L2 of 0.001 and a dropout of 0.4
- Post hoc analysis
- No or limited overfitting countermeasures leads to low generalisation scores
- If you only optimize for accuracy you see low to no generalisation

Experimental results (LOVOCV)

Model	Prec.	Fit.	Gen.
Model 1	0.94 ± 0.00	0.94 ± 0.00	0.79 ± 0.12
Model 2	0.94 ± 0.00	0.94 ± 0.00	0.92 ± 0.09
Model 3	0.94 ± 0.00	0.94 ± 0.00	0.91 ± 0.10
Model 4	0.95 ± 0.01	0.95 ± 0.00	0.75 ± 0.13
Model 5	0.92 ± 0.01	0.92 ± 0.01	0.68 ± 0.21
Model 6	0.93 ± 0.01	0.93 ± 0.01	0.92 ± 0.11

- Over all variants: procedure repeated as many times as number of variants
  - Each variant is used as *Test log* once
- Lower generalisation scores with parallel-like behaviour
  - Even in this lenient setting
- XOR, loops, long-term dependencies less trouble
- Deviations higher for generalisation

Experimental results (leave 20% out)

Model	Prec.	Fit.	Gen.
Model 1	0.89 ± 0.01	0.93 ± 0.00	0.72 ± 0.04
Model 2	0.92 ± 0.01	0.93 ± 0.01	0.89 ± 0.04
Model 3	0.92 ± 0.02	0.93 ± 0.01	0.85 ± 0.05
Model 4	0.87 ± 0.03	0.94 ± 0.01	0.61 ± 0.14
Model 5	0.84 ± 0.01	0.94 ± 0.01	0.47 ± 0.05
Model 6	0.92 ± 0.01	0.93 ± 0.00	0.83 ± 0.12

- Lower precision and generalisation scores

Conclusion

- Careful tuning of overfitting countermeasures is necessary when RNNs are applied to processes
- Even on simple models problems appear
  - Overfitting, no true generalisation.
  - Parallel behavior
- Lenient evaluation
  - Post hoc checking of optimal parameters
- Real-life processes are orders of magnitude more complex

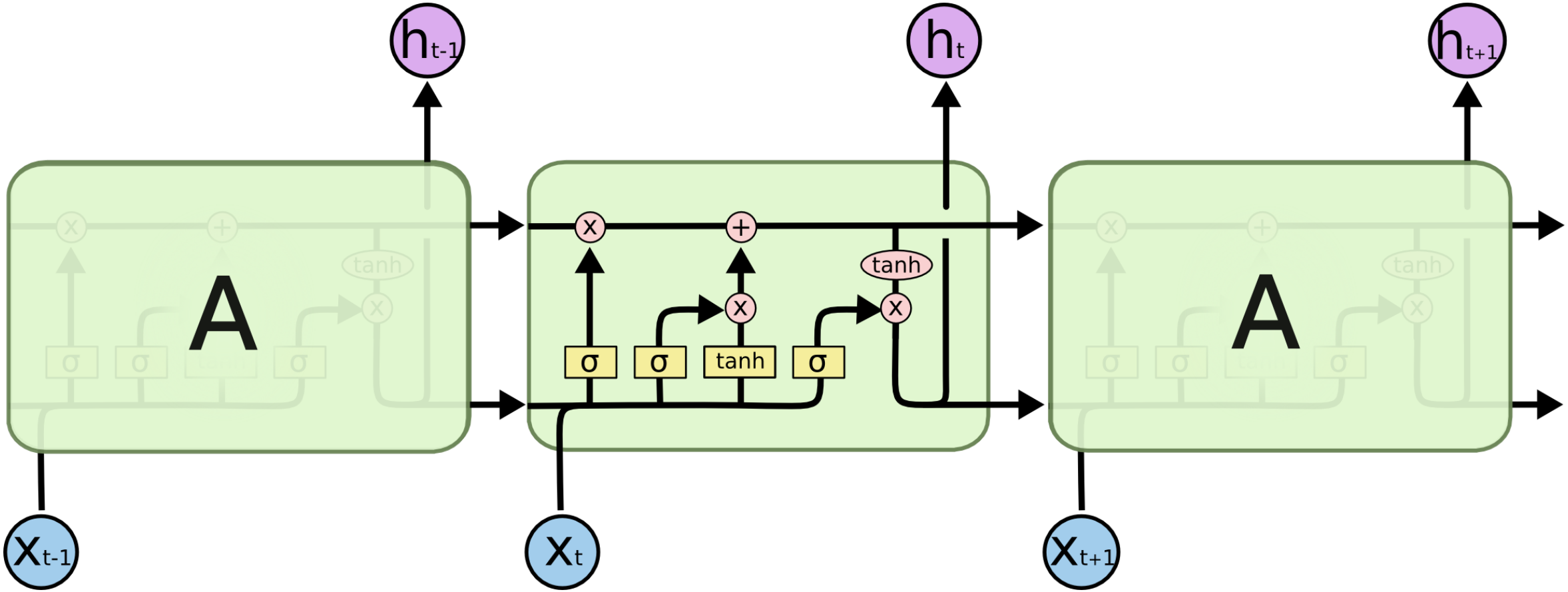
Future work

- More dimensions: timestamps, resource, …
- More synthetic logs
- Real-life logs
- Rigorous theoretical elaboration
  - Comparing with research in different fields
- Expand hyperparameters search
  - Layer normalization, Recurrent batch normalization…
- Other architectures
  - Convolutional Neural Nets, Transformer nets, Generative Adversarial approaches
- Recommendations
  - Sampling training-validation sets
  - Hyperparameter settings

Thank you for your attention

Research Center for Information Systems Engineering (LIRIS), Faculty of Economics and Business

LSTM



Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/