Streaming Adaptation of Deep Forecasting Models using Adaptive Recurrent Units

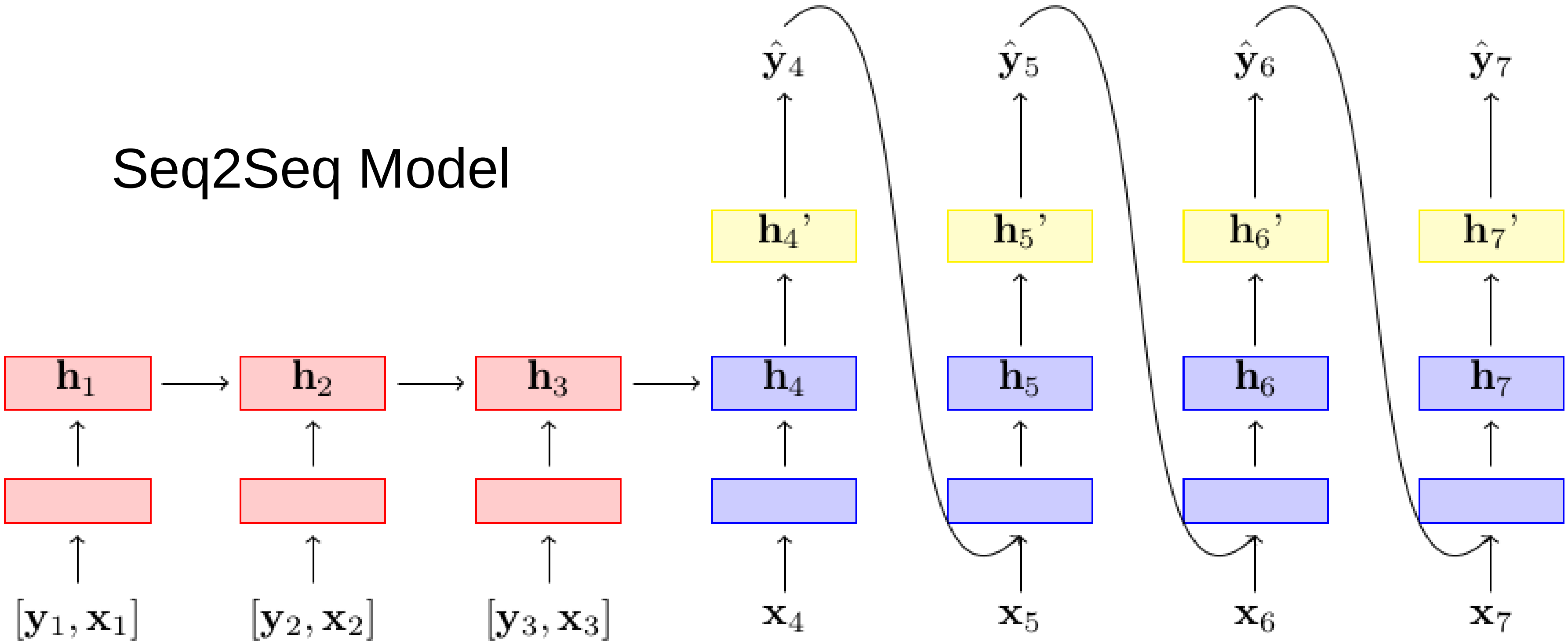
Prathamesh Deshpande

Timeseries Forecasting

- Given a history of values of a variable of interest, predict its future values
  - Forecasting product sales.
  - Forecasting traffic congestions at a location.
- Challenges -
  - Forecast for multiple timeseries: forecast sales of all products a company makes.
  - Forecast congestions at all locations in a city.
  - Long term forecasting is another challenge

RNN based Global Models

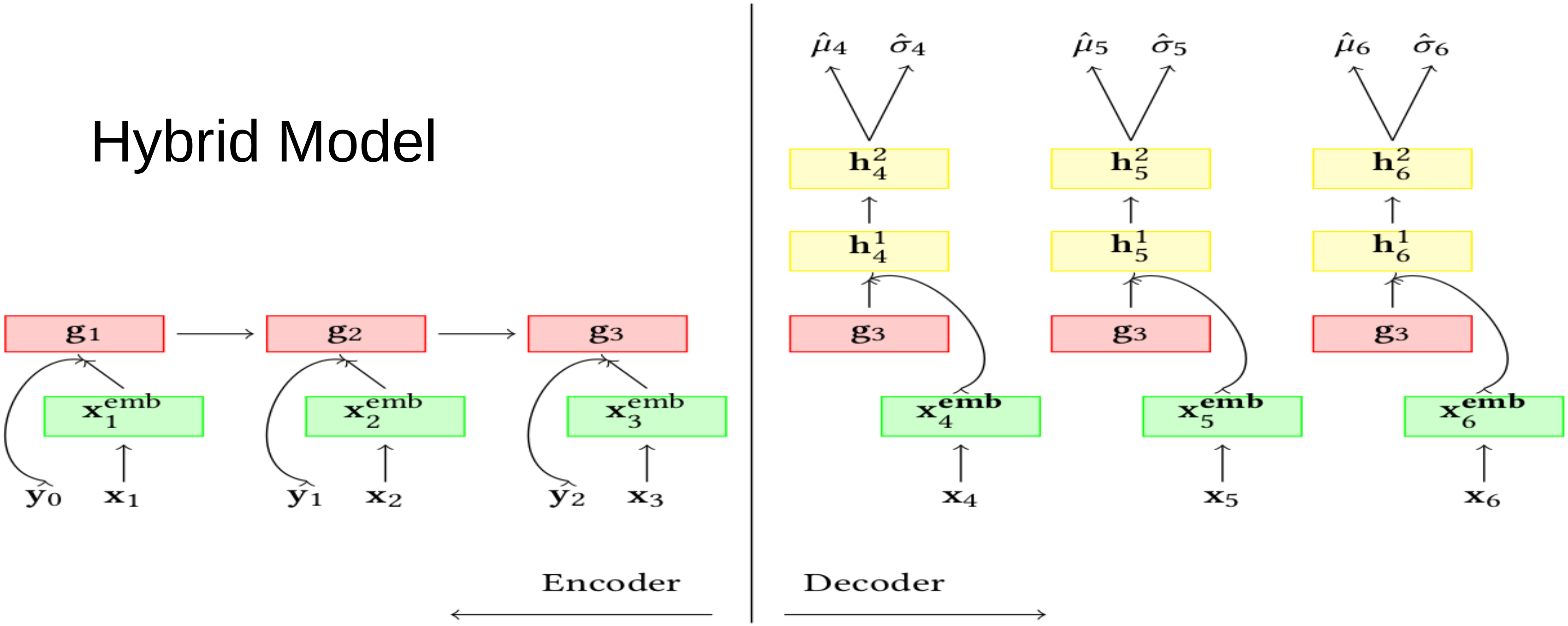
Seq2Seq Model



Encoder size = 3, Decoder size = 4

RNN based Global Models

Hybrid Model



Encoder size = 3, Decoder size = 4

- Predicts outputs at all decoder timesteps together.

Global Models and its Challenges

- Useful to capture information common across timeseries.
- Local information about outputs *y* captured in RNN state
  - Capacity limited by state size
  - Even harder when timeseries are heterogeneous

Solution: Local Adaptation

Local / Domain Adaptation

- Setup – Multiple tasks $T_1, T_2, \ldots, T_N \sim p(T)$ drawn from a task distribution.
- Objective – Train a shared model with parameters θ such that
  - for a new task T_i , it can update the parameters to θ_i by looking at only few instance of T_i

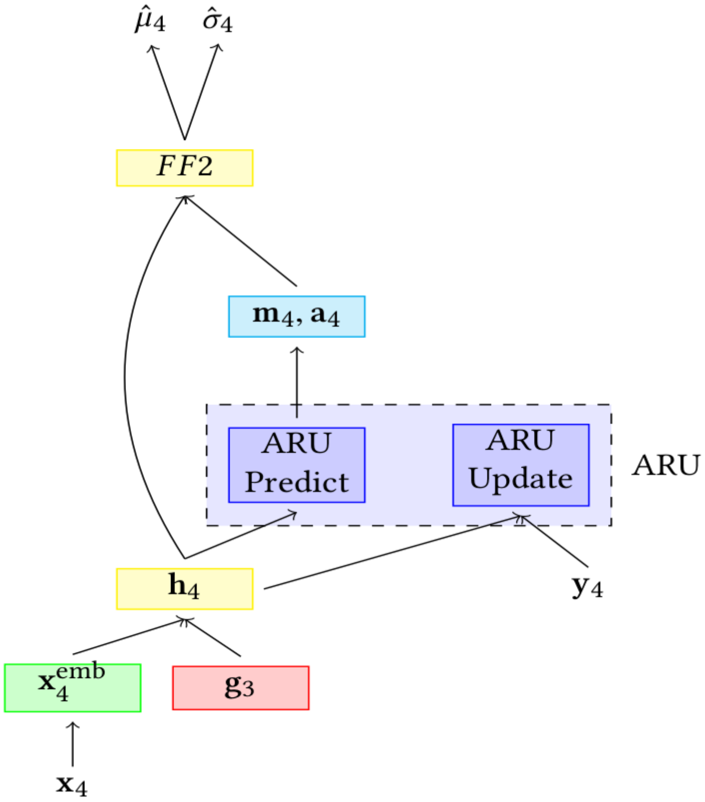
Domain Adaptation can be used for Timeseries Forecasting.

Adaptive Recurrent Unit (ARU)

- Exploits closed form solution of least squares.
- No need to train local parameters through gradient updates.
- Makes fully local predictions.
- Output of ARU can be easily combined with global model
  - Provides fully local signals to global model.
  - Does not affect dynamics of global model.
- ARU state maintained for each timeseries.

The ARU RNN

- Given a decoder input *x* , ARU returns a fully local prediction of output.
- Local prediction is combined with RNN state and passed to next layers.
- Because ARU is closed form, gradient flow is stopped at ARU cell.



The ARU States and Equations

- ARU states are sufficient statistics required to evaluate closed form solution.
- maintained online, updates as timeseries unfolds through time axis.

Global Model

$$\mathbf{g}_T^i = RNN([y_{t-1}^i, \mathbf{x}_t^i] : t = 1 \ldots T | \theta_{enc})$$
$$\mathbf{h}_t^i = FF([\mathbf{g}_T^i, \mathbf{x}_t^i] : t = T+1 \ldots T+K | \theta_{dec})$$
$$\mu_t^i = \theta_\mu[\mathbf{h}_t^i, 1], \quad \sigma_t^i = \log(1 + \exp(\theta_\sigma[\mathbf{h}_t^i, 1]))$$

ARU States

$$\mathbf{sxx}_t^i = \alpha\, \mathbf{sxx}_{t-1}^i + [\mathbf{h}_t^i\ 1]^{\mathrm{T}}[\mathbf{h}_t^i\ 1]$$
$$\mathbf{sxy}_t^i = \alpha\, \mathbf{sxy}_{t-1}^i + [\mathbf{h}_t^i\ 1]^{\mathrm{T}}(\mathbf{y}_t^i)$$

Local Prediction

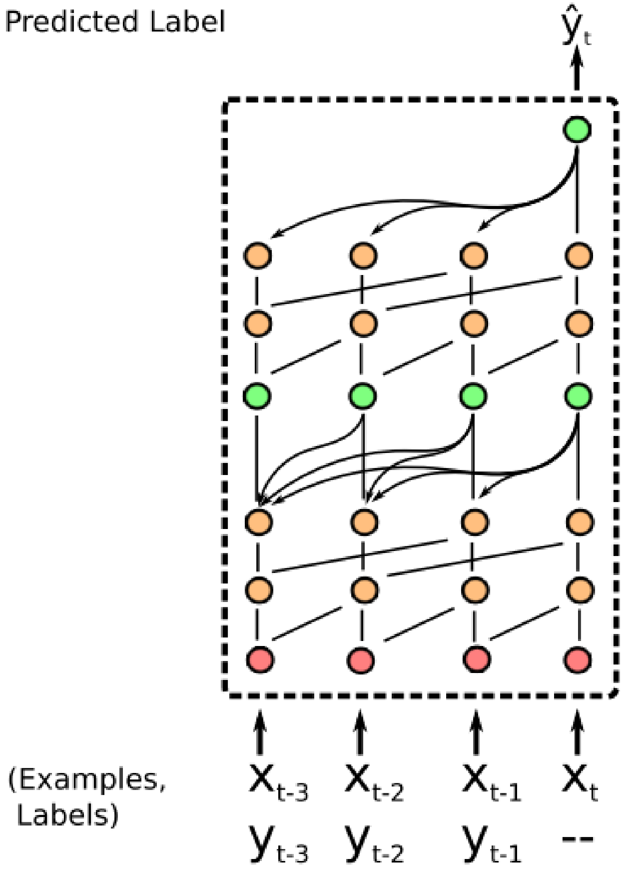
$$\boldsymbol{\theta}_{t,\mu}^i = (\mathbf{sxx}_t^i + \lambda I)^{-1} \mathbf{sxy}_t^i,$$
$$\mathbf{m}_t^i = \boldsymbol{\theta}_{t,\mu}^i [\mathbf{h}_t^i\, 1],$$

Final Prediction

$$\mu_t^i = \theta_\mu(FF2[\mathbf{h}_t^i, \mathbf{m}_t^i, 1]$$

Some Related Work

SNAIL: A Domain Adaptation Model



- Captures depedency on entire history of the sequence using
  - Dialated Causal Convolution
  - Self attention layers
- *O(log N)* convolution layers where N is length of the sequence.
- Self attention layers interleaved with conv layers.

Deepstate

- Based on State Space Models (SSM)
- Each timeseries has a local state space model
- A global RNN-based model is used to directly predict the parameters of the local model.

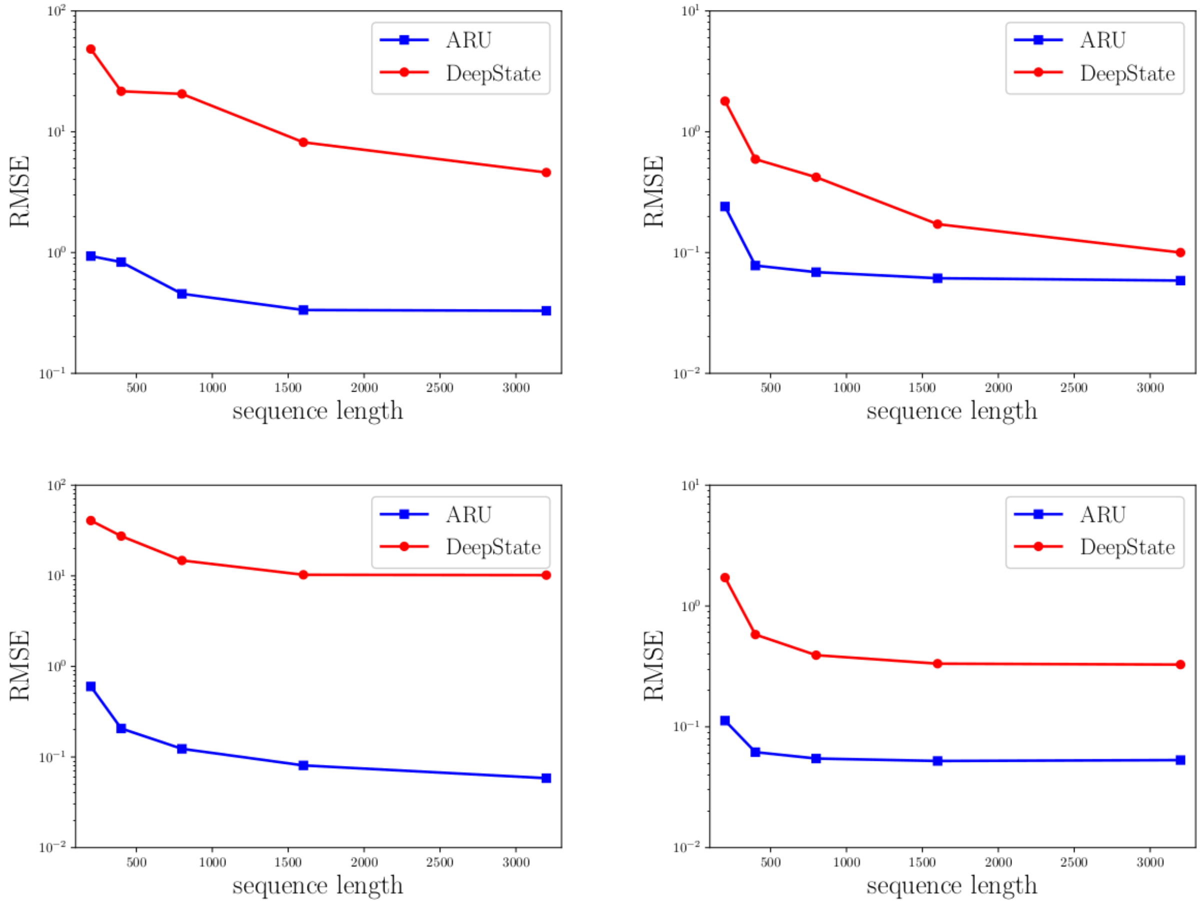
Synthetic Experiment

- Why is deepstate not a good model?
  - Similar to Deepstate, we use RNN to compute local weights of the ARU.

Weights ϵ [-20, 20]　　Weights ϵ [-1, 1]

with time-series id

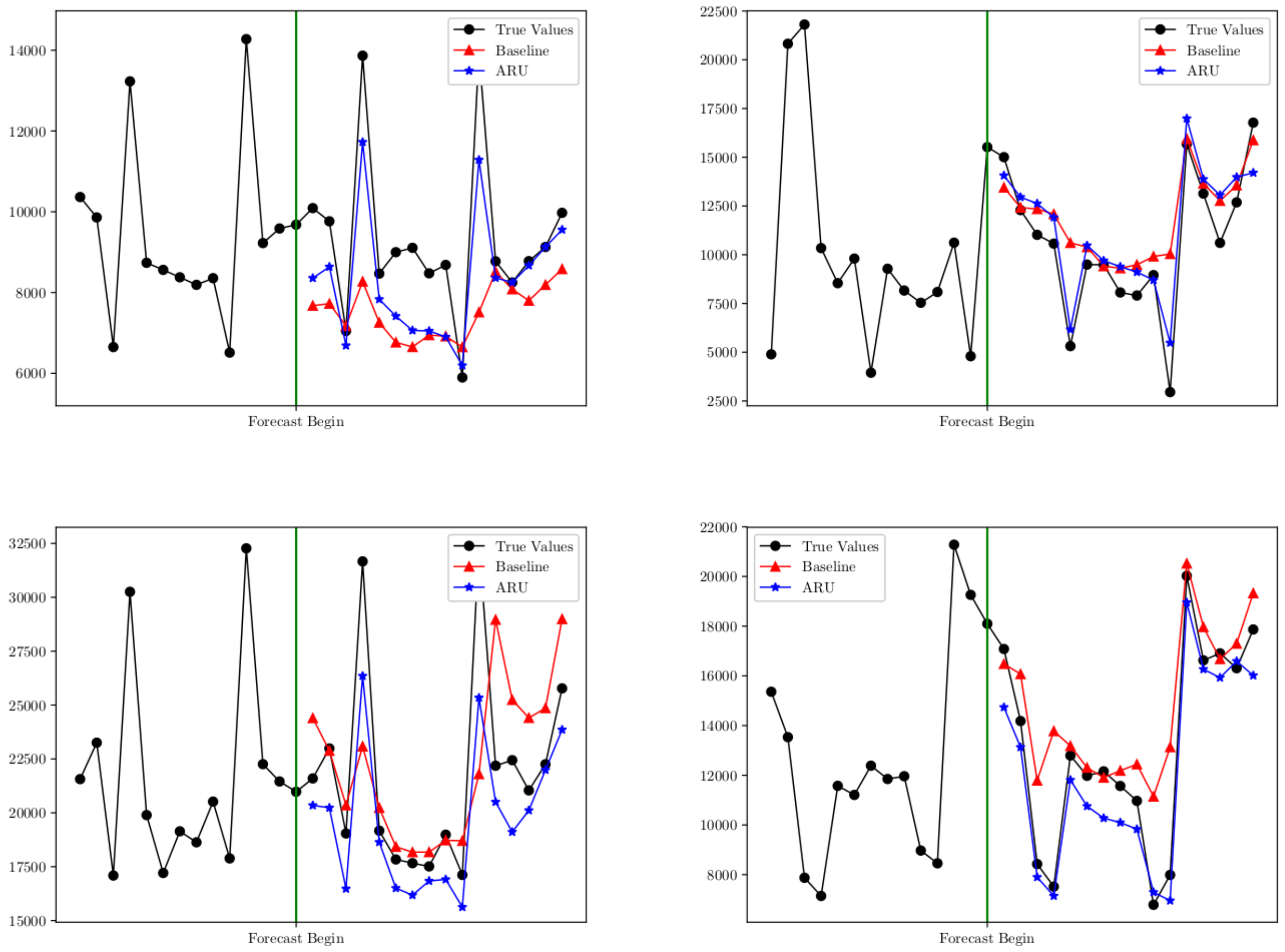
Without time-series id



Datasets

Dataset	No. of Timeseries	Length of each timeseries	Forecast Horizon	Encoder Length	No. of Features
Rossman	1115	1600	16	16	39
Walmart	3331	143	8	8	16
Electricity	370	44000	24	168	5
Traffic	963	2100	24	168	3
Parts	2246	52	8	8	1

Anecdotes on Rossman Dataset



Results on Datasets

Dataset	Method	Normalized Deviation (ND)	RMSE
Rossman	Baseline	0.094	983.2
	DeepAR	0.245	2389.0
	SNAIL	0.093	972.6
	ARU	**0.089**	**934.9**
Walmart	Baseline	0.137	4548.6
	DeepAR	0.233	6704.0
	SNAIL	0.144	4690.5
	ARU	**0.114**	**3938.6**
Electricity	Baseline	0.136	416.7
	DeepAR	0.172	544.0
	SNAIL	0.135	400.6
	ARU	**0.127**	**396.4**
Traffic	Baseline	0.170	0.0224
	DeepAR	**0.145**	**0.0216**
	SNAIL	0.165	0.0227
	ARU	0.161	0.0220

- ARU most effective on Rossman and Walmart datasets.
- Traffic dataset has little local information.

Inference Time

Dataset	Baseline	ARU	SNAIL
Electricity	1.0458	1.1788	3.0754
Traffic	2.0740	2.3383	5.7673
Walmart	0.7034	0.9434	1.2693
Rossman	0.4379	0.6717	2.2837

Table 7: Inference time (in seconds)

- SNAIL slower due to additional overhead of self-attention.

Summary

- ARU is a light-weight, parameter-less local model
- Can be easily coupled with the global model – Does not disturb dynamics of the global learning.
- Unlike existing local models which are memory-intensive, ARU only needs fixed-sized state.
- Found most effective in retail forecasting setting.

Traffic Congestion Prediction

Joint work with Avinash
Modi, M. Tech. 2, CSE.

Problem Setup

- Given a history of congestions at a location -

$$(t_1, d_1), (t_2, d_3), (t_3, d_3), \ldots\ldots,(t_N, d_N)$$

- Where (t_i, d_i) *denote*
  - *time of congestion occurrence and,*
  - *duration of congestion*

- Predict the time and duration of $(N+1)^{th}$ *to* $(N+k)^{th}$ congestion.
- OR predict all the congestions likely occur in the next day.

Challenges and Formulations

- An irregular timeseries – interval between consecutive observations not consistent.
- Timeseries Forecasting:
  - Unfold history into a “bitmap”. Each bit represents a congestion state – 1->congestion, 0-> no congestion.

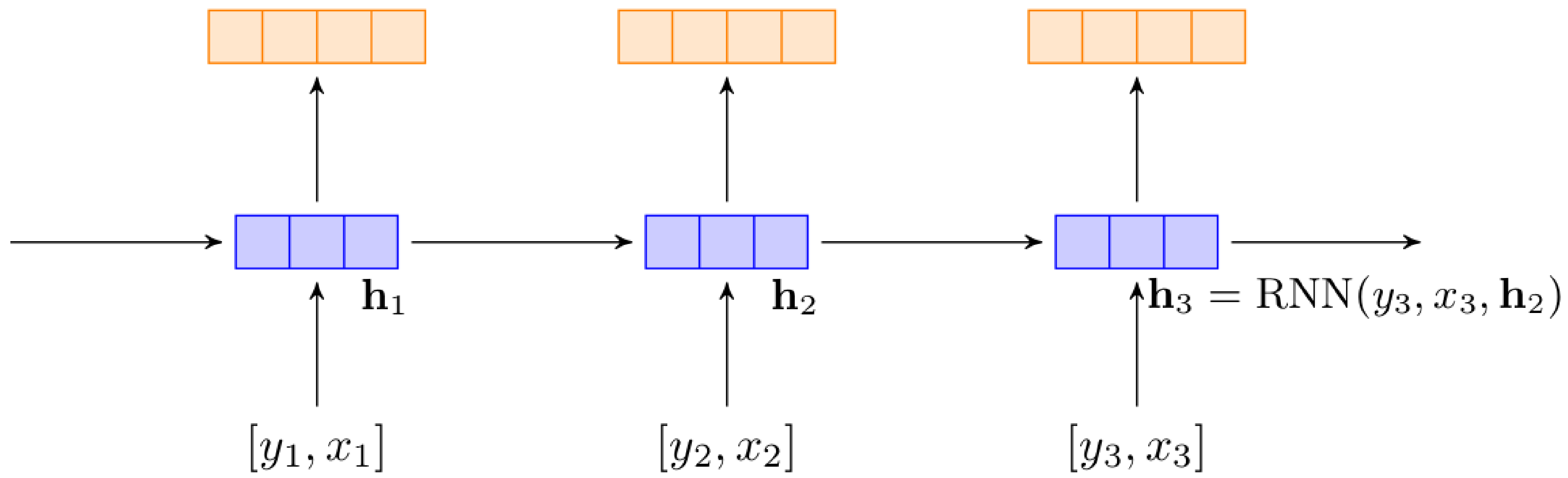
$$(t_1 = 10, d_1 = 4), (t_2 = 18, d_2 = 6), (t_3 = 35, d_3 = 3)$$

$$|0|0|0|0|0|0|0|0|1|1|1|1|0|0|0|0|1|1|1|1|1|1|0|0|0|0|0|0|0|0|0|0|0|1|1|1|\ldots$$

Challenges and Formulations

- Bitmap can be created with sutaible time granularity (e.g. *5 mins*)
- and used to train any recurrent model
- Skewed ratio of *1s* and *0s.*
- Solution: Undersampling of *0* label bits.

RNN based Model



- At each step, predicts next few bits.
- Number of bits to be predicted can be set based on the requirement.

Current Progress on RNN model

- Does not generalize well when number of bits to be predicted is large such as *288* (congestion states of entire next day).
- Continuity loss – Impose a constraint on consecutive predictions.
- Loss = $|(\hat{y}_t - \hat{y}_{t-1}) + (y_t - y_{t-1})|$
- Currently investigating better formulations of continuity loss

Thank You!