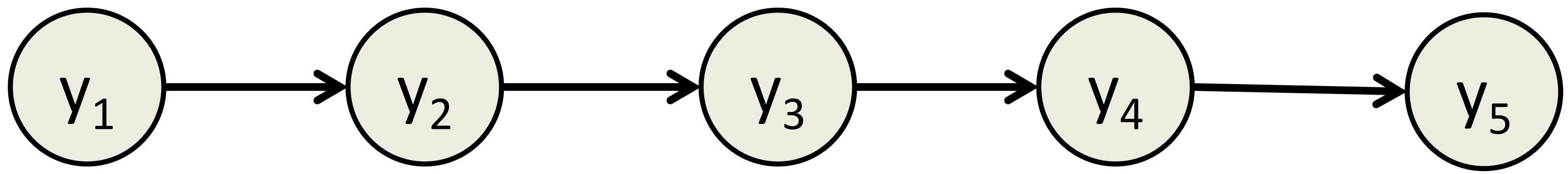
CPSC 540: Machine Learning

Recurrent Neural Networks

Winter 2020

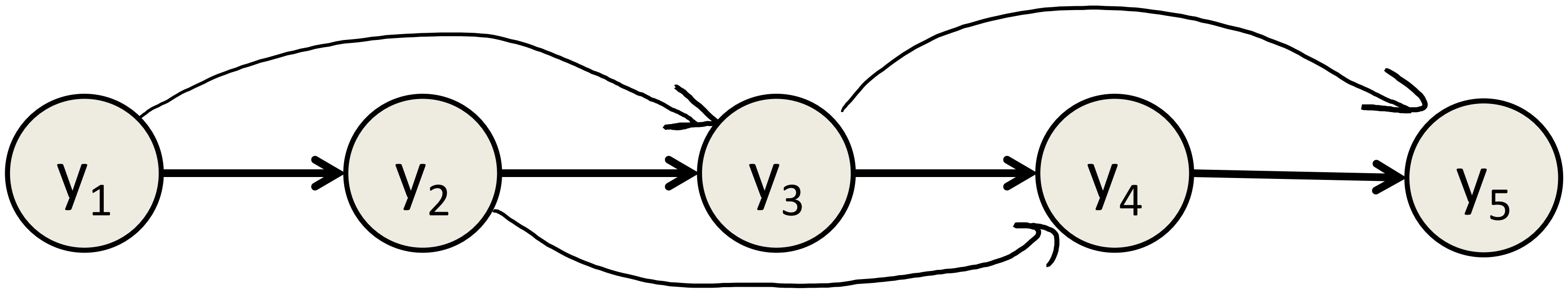
Motivation: Sequence Modeling

- We want to predict the next words in a sequence:
  - "I am studying to become a [?????????????????????????????]".
- Simple idea: supervised learning to predict the next word.
  - Applying it repeatedly to generate the sequence.
- Simple approaches:
  - Markov chain (doesn't work well, see "Garkov").



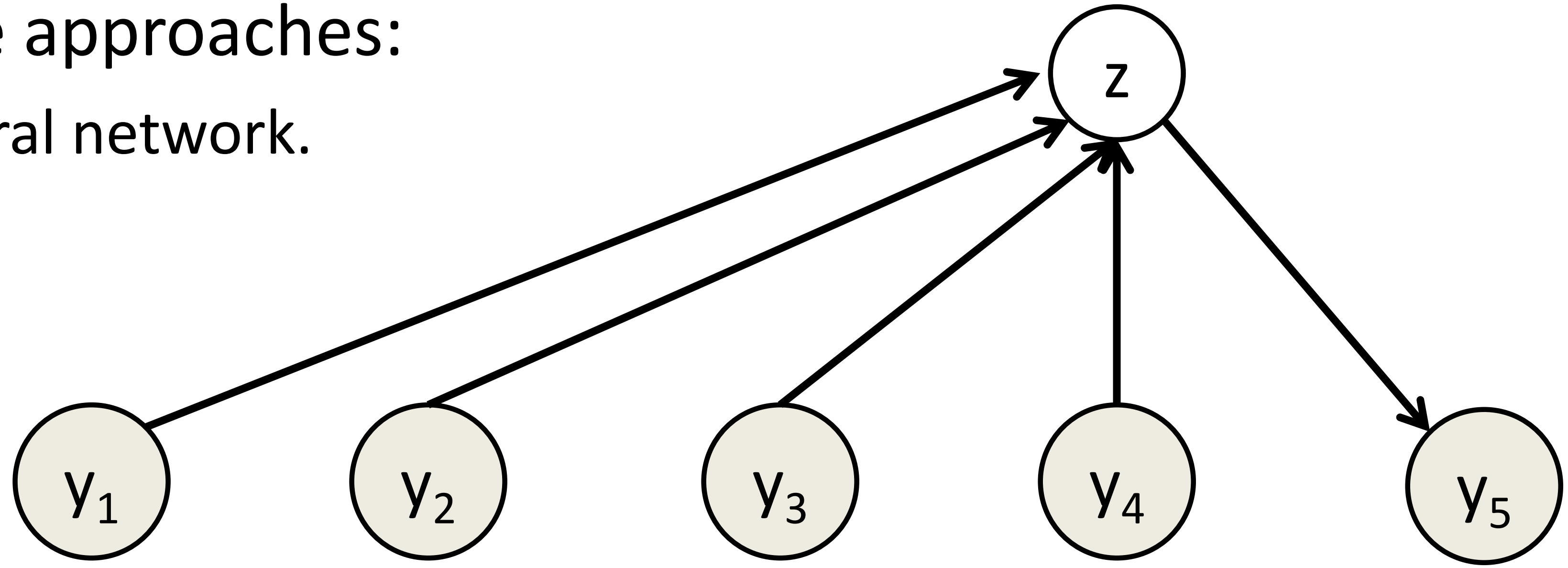
Motivation: Sequence Modeling

- We want to predict the next words in a sequence:
  - "I am studying to become a [???????????????????????????]".
- Simple idea: supervised learning to predict the next word.
  - Applying it repeatedly to generate the sequence.
- Simple approaches:
  - Higher-order Markov chain ("n-gram"):



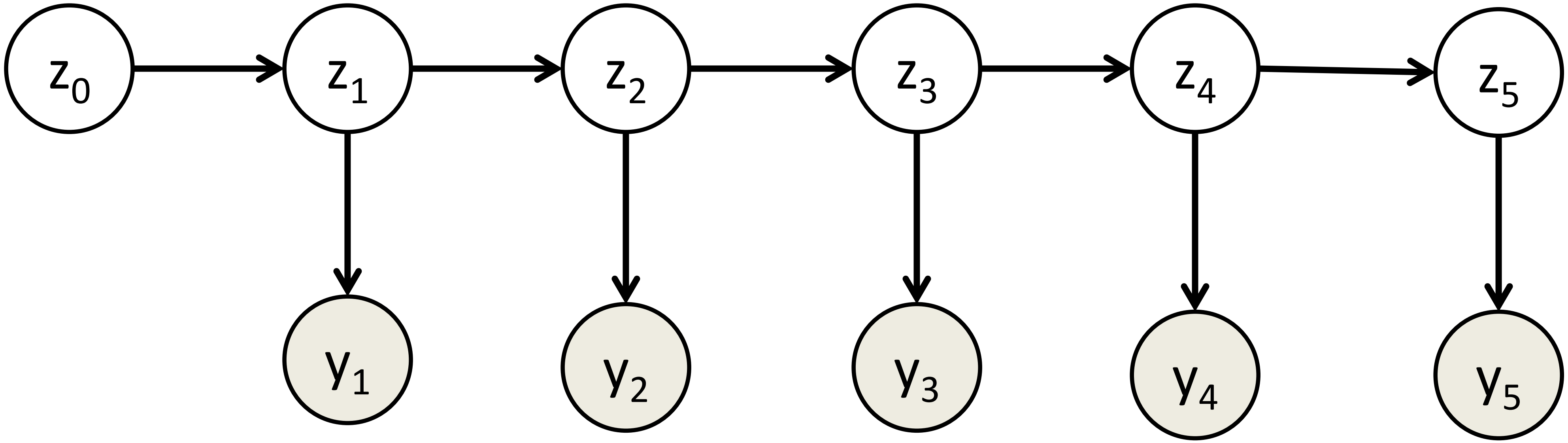
Motivation: Sequence Modeling

- We want to predict the next words in a sequence:
  - "I am studying to become a [?????????????????????????????]".
- Simple idea: supervised learning to predict the next word.
  - Applying it repeatedly to generate the sequence.
- Simple approaches:
  - Neural network.



State-Space Models

- Problem with simple approaches:
  - All information about previous decision must be summarized by x_t.
  - We 'forget' why we predicted x_t when we go to predict x_{t+1}.
- More complex dynamics possible with state-space models:
  - Add hidden states with their own latent dynamics (HMM-style)



Challenges of State-Space Models

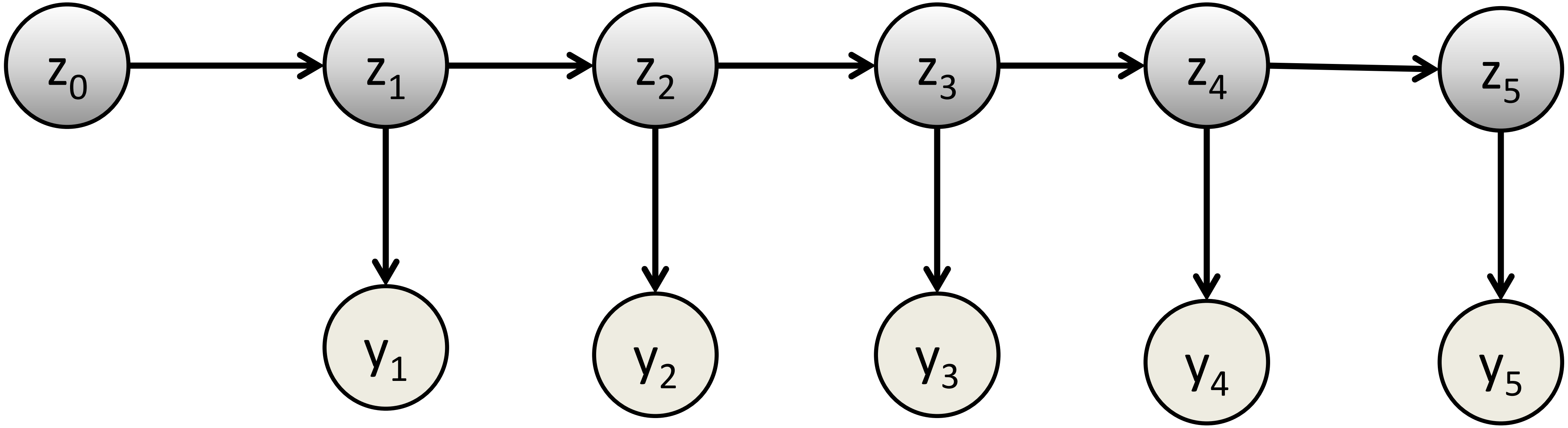
- Problem 1: inference only has closed-form in simple situations.
  - Only 2 cases: Gaussian z *and* y (Kalman filter) or discrete z (HMMs).
  - Otherwise, need to use approximate inference.

- Problem 2: memory is very limited.
  - You have to choose a z_t at time 't'.
    - But still need to compress information into a single hidden state.

- Obvious solution:
  - Have multiple hidden z_t at time 't', as we did before.
    - But now inference becomes hard.

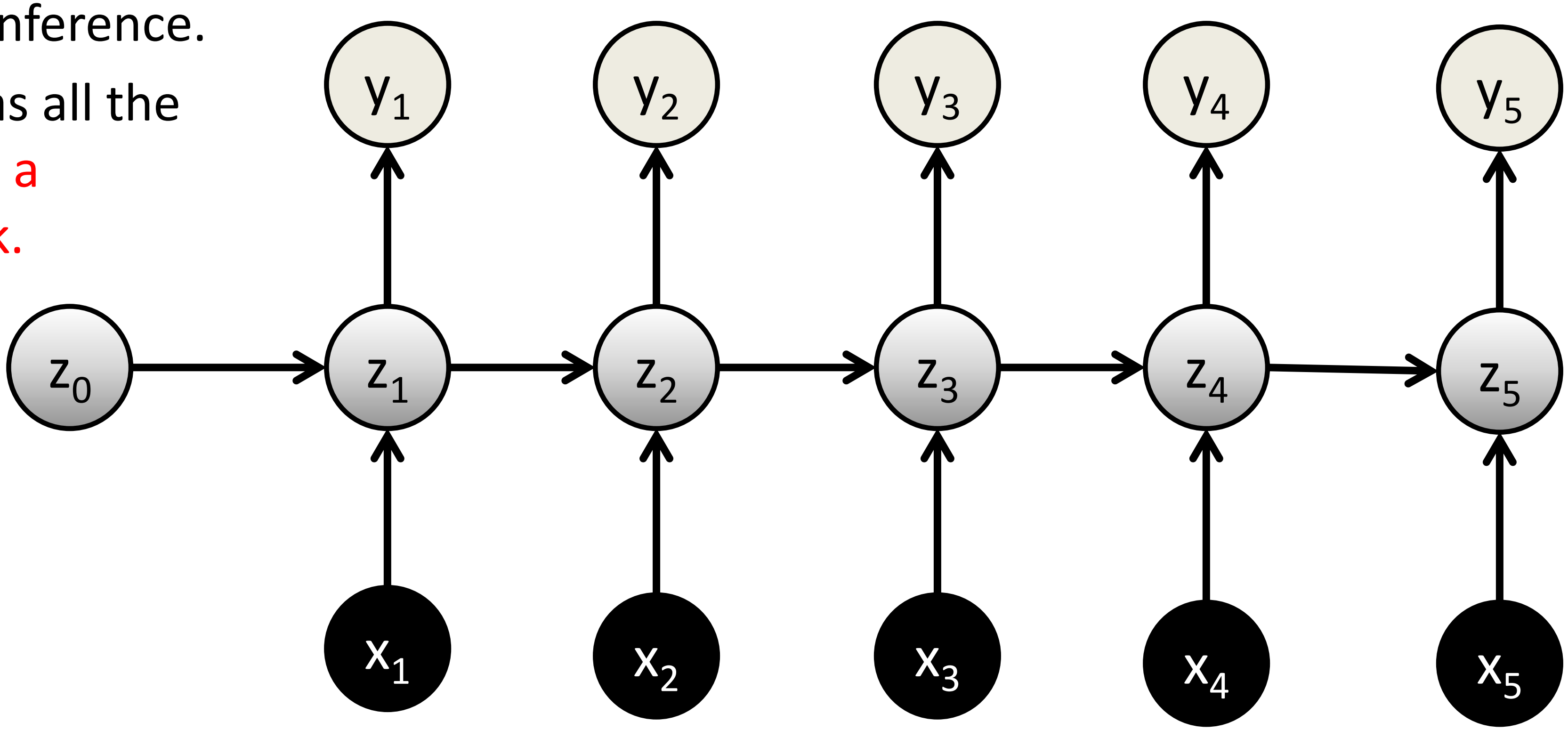
Recurrent Neural Networks

- Recurrent neural networks (RNNs) give solution to inference:
  - At time ‘t’, hidden units are deterministic transformations of time ‘t-1’.
  - Basically turns the problem into a big and structured neural network.



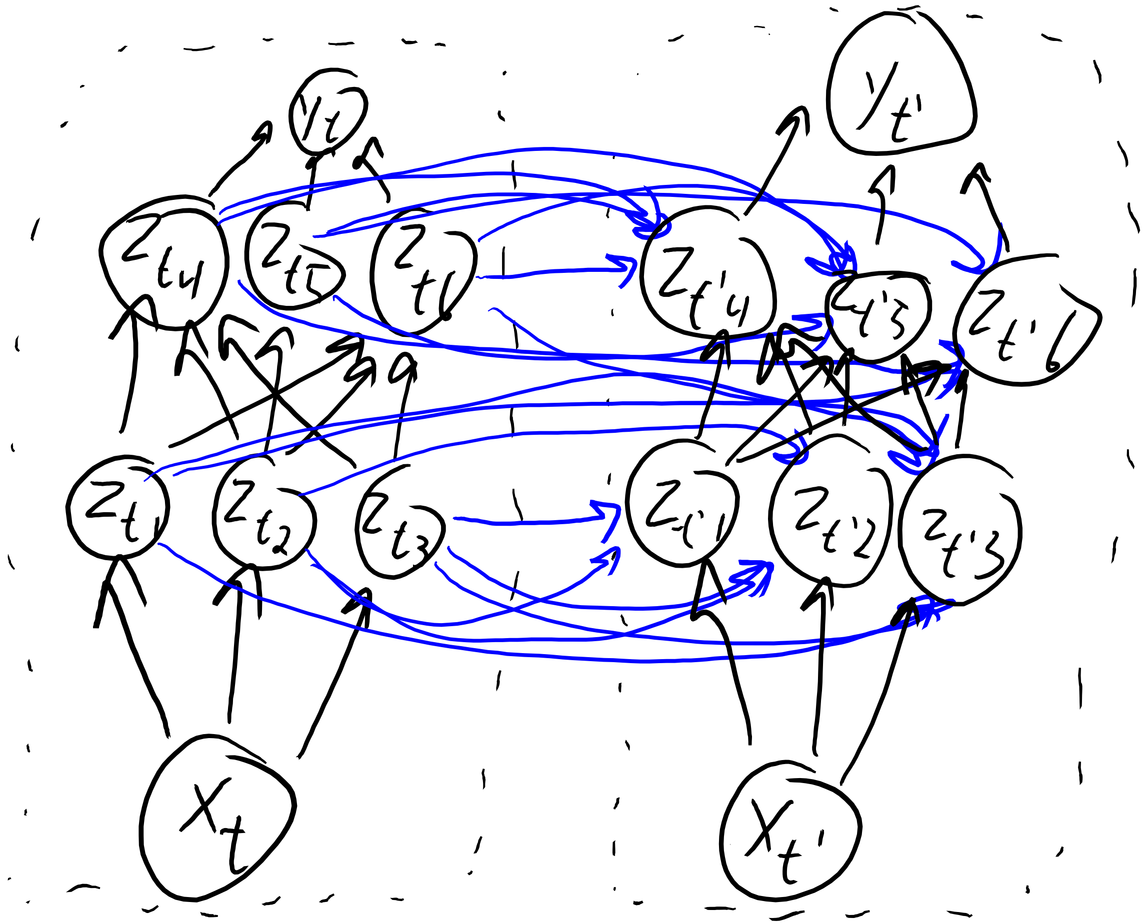
Recurrent Neural Networks

- RNNs can be used to translate input sequence to output sequence:
  - A neural network version of latent-dynamics models.
  - Deterministic transforms mean hidden ‘z’ can be really complicated.
    - But with easy inference.
    - I’m using “z_1” as all the hidden units in a neural network.



Recurrent Neural Networks

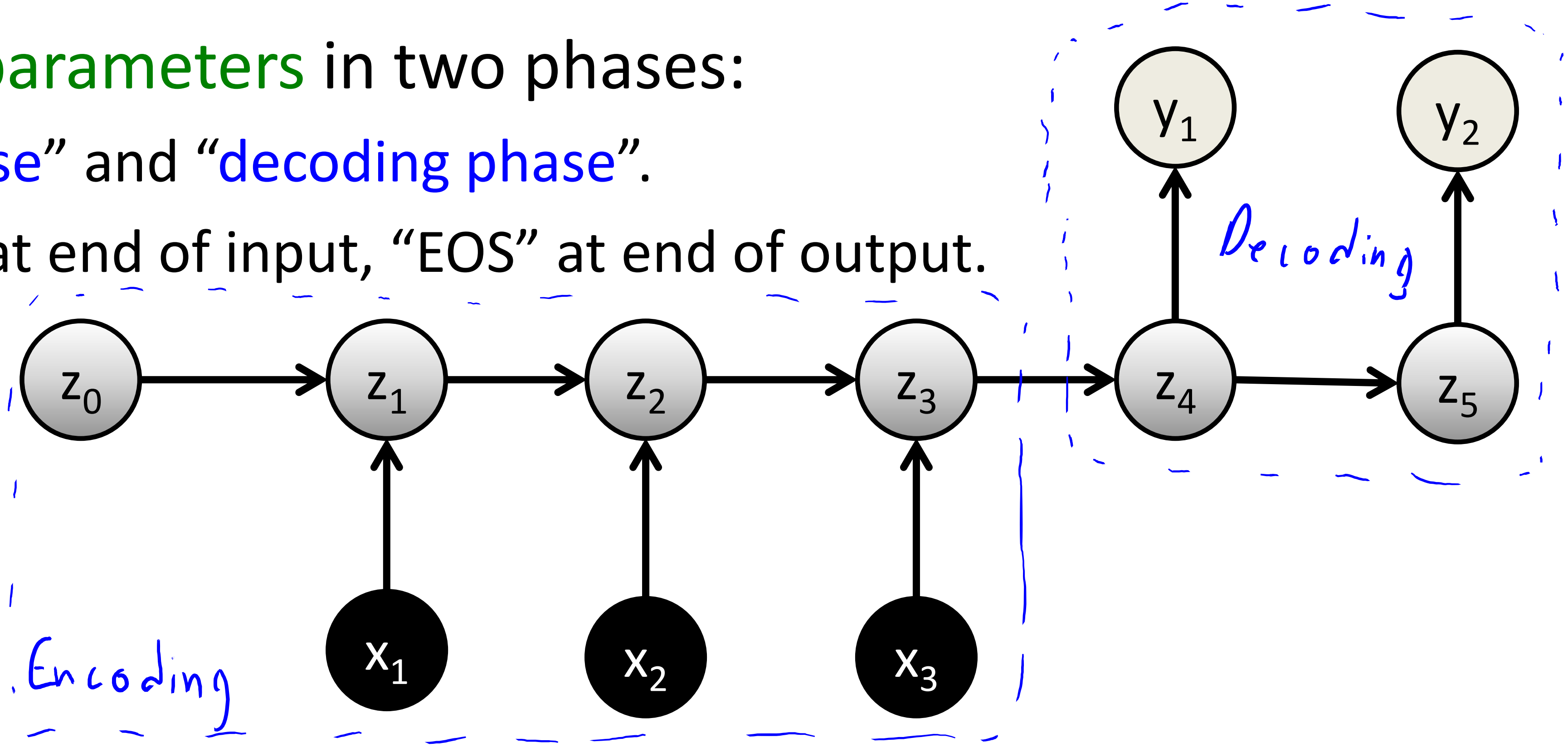
- Can think of each time as implementing the same neural network:
  - But with connections from hidden units at previous time.



Sequence-to-Sequence

- An interesting variation on this for sequences of different lengths:
  – Translate from French sentence ‘x’ to English sentence ‘y’.

- Usually we tie parameters in two phases:
  – “Encoding phase” and “decoding phase”.
  – Special “BOS” at end of input, “EOS” at end of output.



Training Recurrent Neural Networks

- Train using stochastic gradient: “backpropagation through time”.
- Similar challenges/heuristics to training deep neural networks:
    - “Exploding/vanishing gradient”, initialization is important, slow progress, etc.

- Exploding/vanishing gradient problem is now worse:
  – Parameters are tied across time:
    - Gradient gets magnified or shrunk exponentially at each step.

  – Common solutions:
    - “Gradient clipping”: limit gradient norm to some maximum value.
    - Long Short Term Memory (LSTM): make it easier for information to persist.

Summary

- Fully-convolutional networks:
  – Elegant way to apply convolutional networks for dense labeling problems.

- Recurrent neural networks:
  – Neural networks for model sequenctial inputs and/or sequential outputs.