Artificial Neural Networks:
On Time and Synthesis

Alexander G. Ororbia II

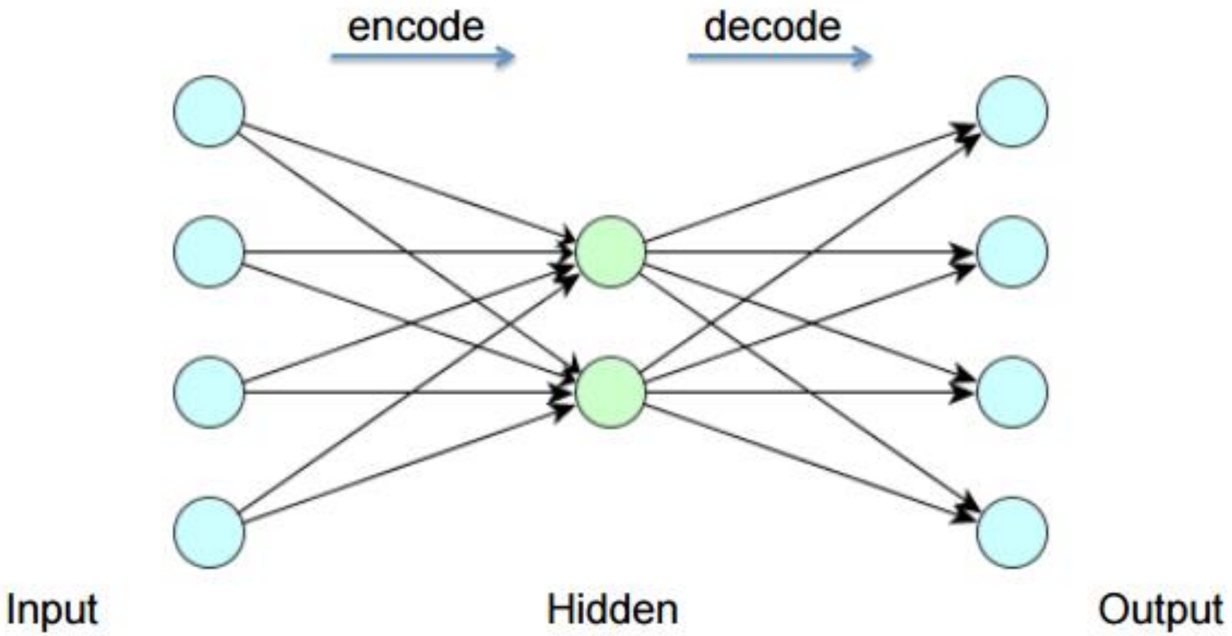
Introduction to Machine Learning

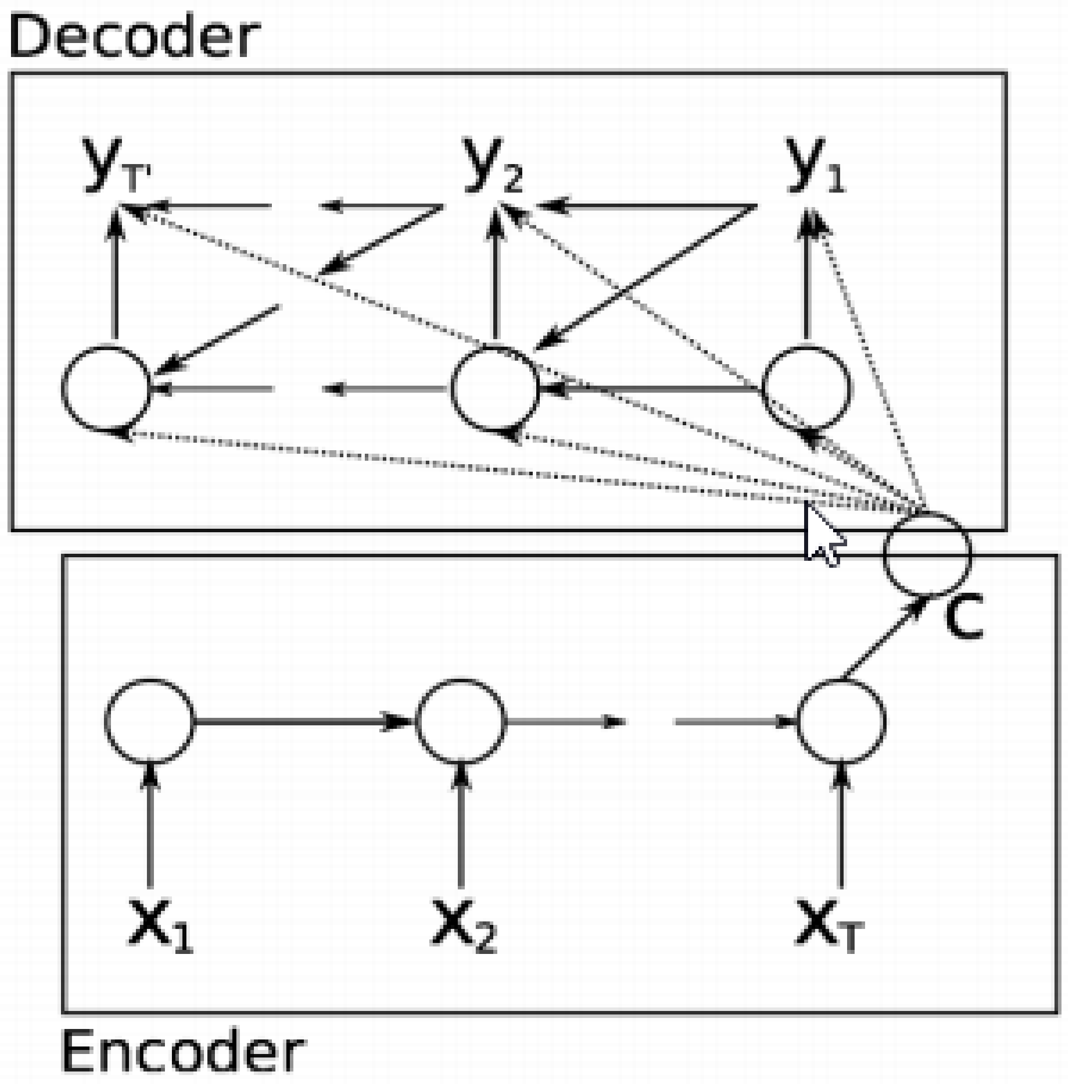
CSCI-635

12/6/2023

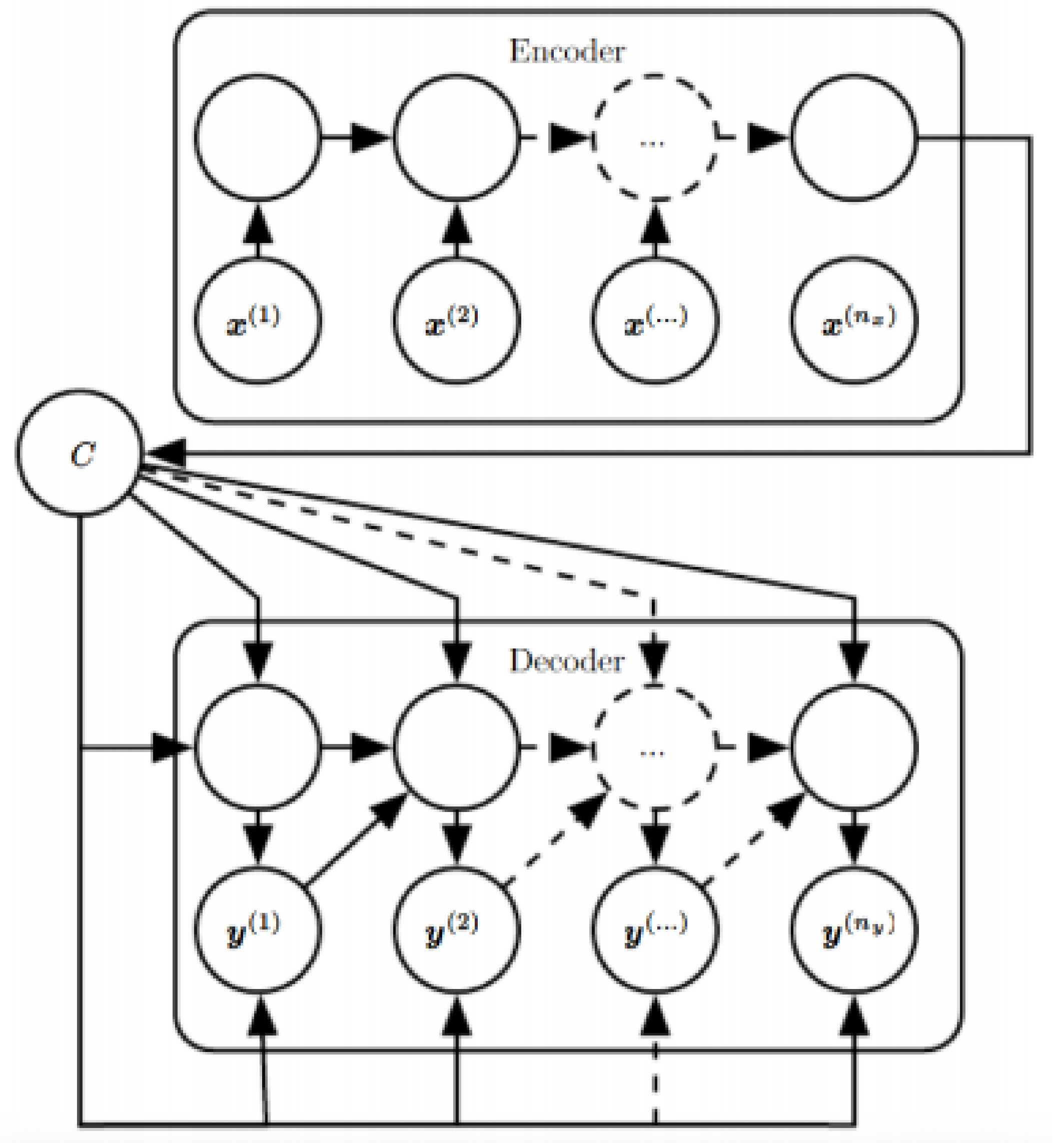
The Encoder-Decoder Framework

- Auto-association (auto-encoding)
  - Learn a compressed representation of the input (think of word2vec, except simpler)
  - Bottleneck layer = meaningful latent space
- Can de-couple encoder & decoder
  - Each can be complex, different functions





An Encoder-Decoder or Sequence-to-Sequence RNN



Learns to generate an output sequence

$(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n_y)})$

given an input sequence

$(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n_x)})$

It consists of an encoder RNN that reads an input sequence and a decoder RNN that generates the output sequence or computes the probability of a given output sequence)

The final hidden state of the encoder RNN Is used to compute a fixed size context C which represents a semantic summary of the input sequence and is given as input to the decoder

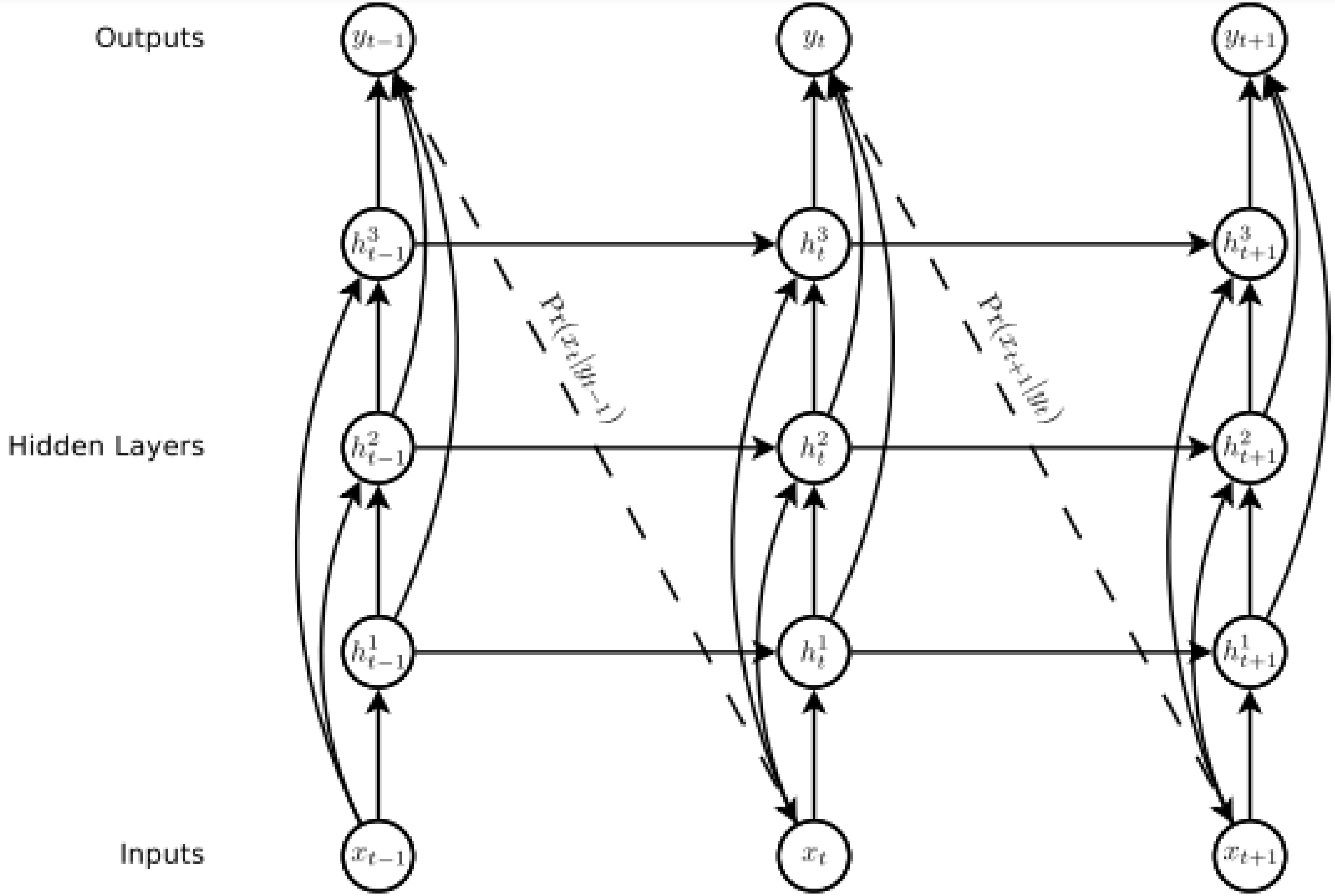
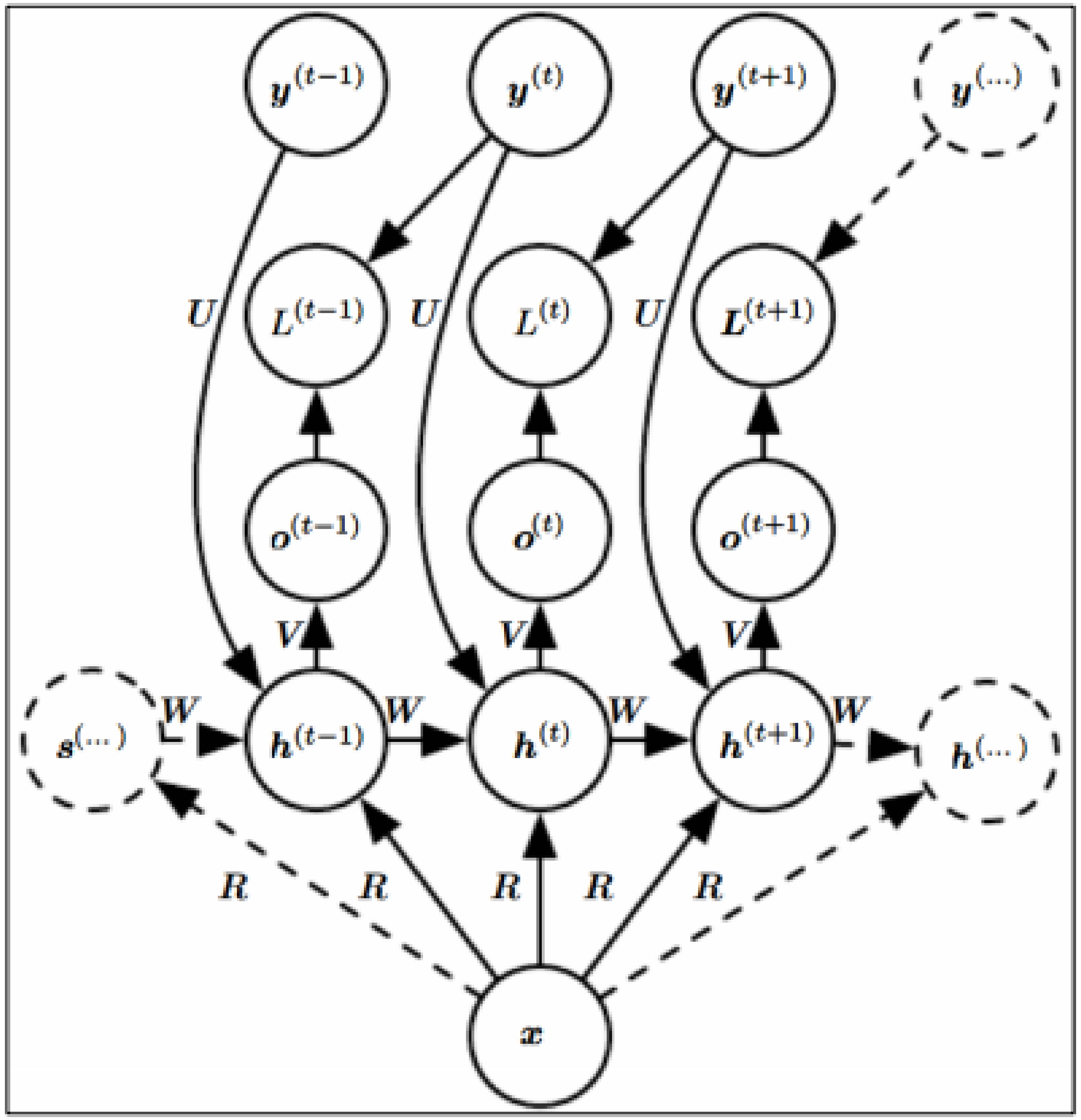


Figure 1: **Deep recurrent neural network prediction architecture.** The circles represent network layers, the solid lines represent weighted connections and the dashed lines represent predictions.

RNN to map a fixed length vector $\boldsymbol{x}$ over sequences $\mathbf{Y}$

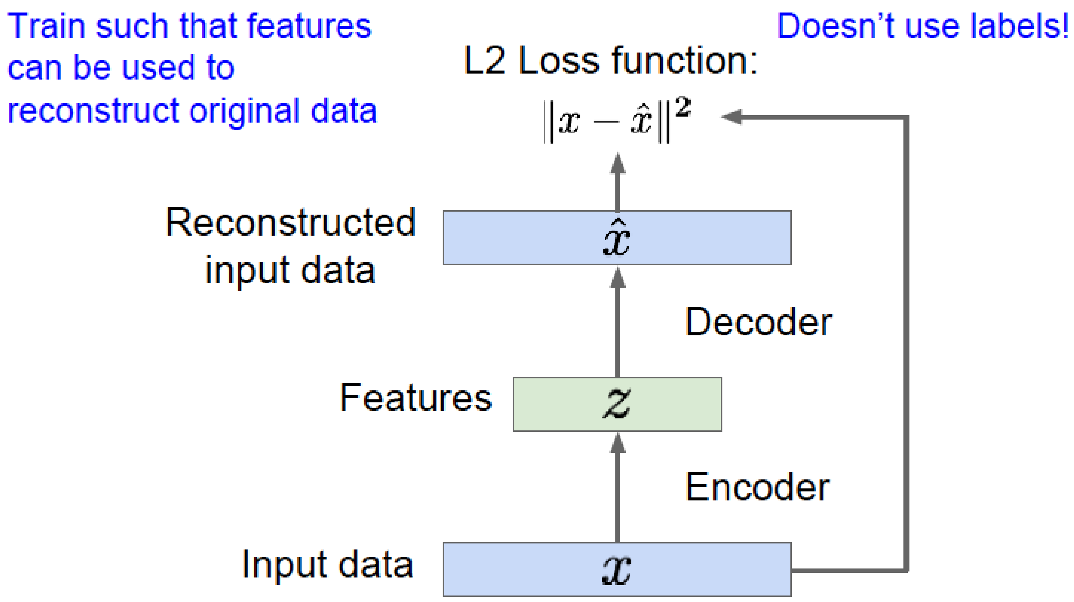


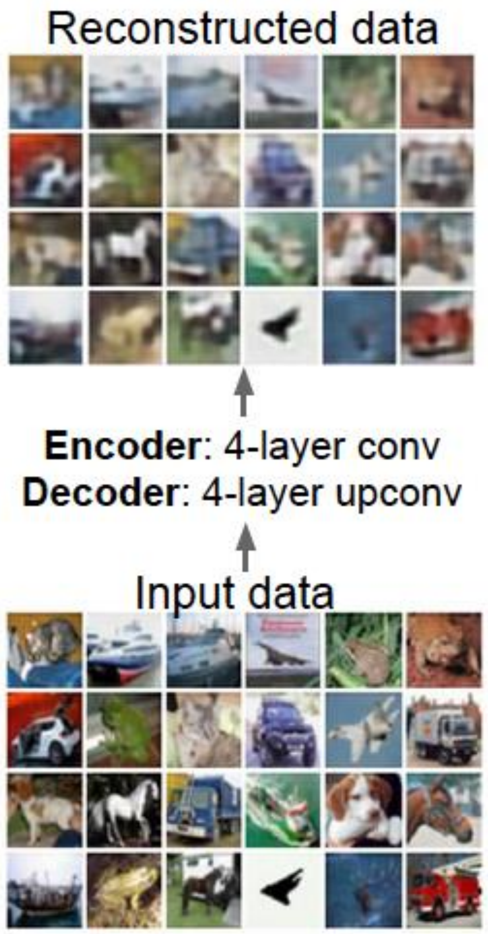
Appropriate for tasks such as image captioning where a single image is input which produces a sequence of words describing the image.

Each element of the observed output $\boldsymbol{y}^{(t)}$ of the observed output sequence serves both as input (for the current time step) and during training as target

Generative Models Revisited

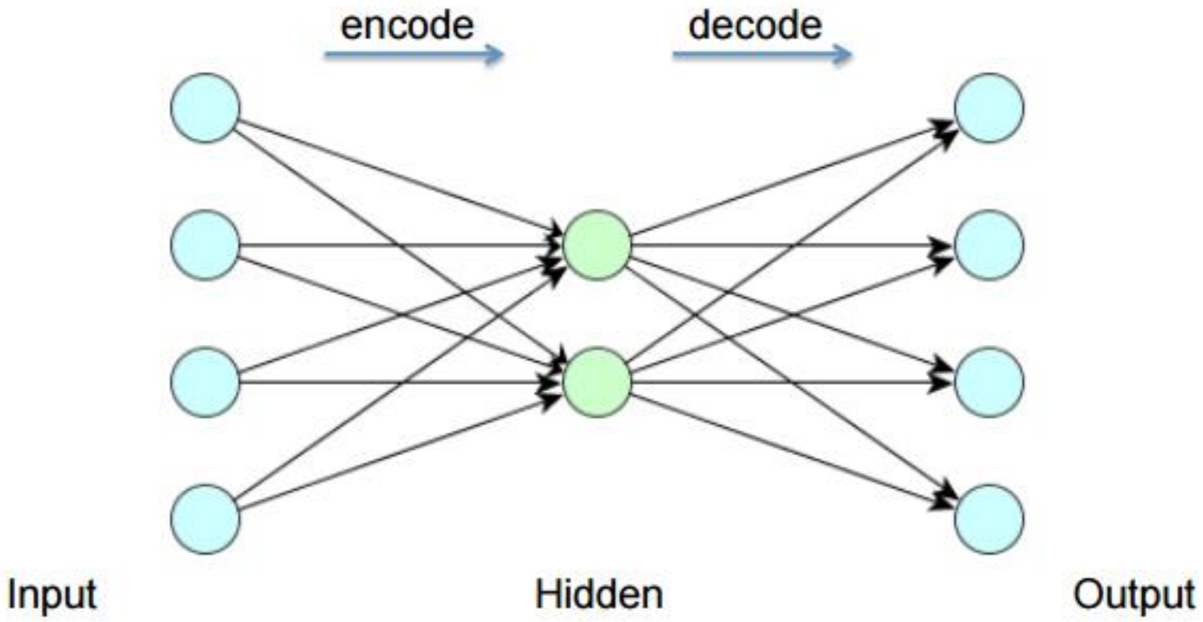
Auto-association

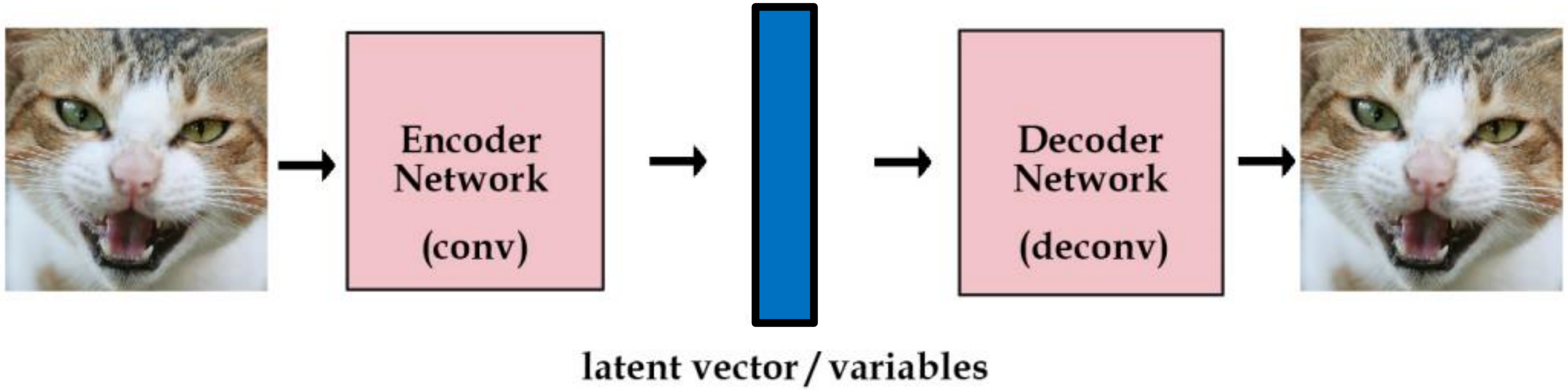




Autoencoding: The Encoder-Decoder Framework

- Auto-association (auto-encoding)
  - Learn a compressed representation of the input, i.e., word2vec
  - Bottleneck layer = meaningful latent space
- Can de-couple encoder & decoder
  - Each can be complex, different functions

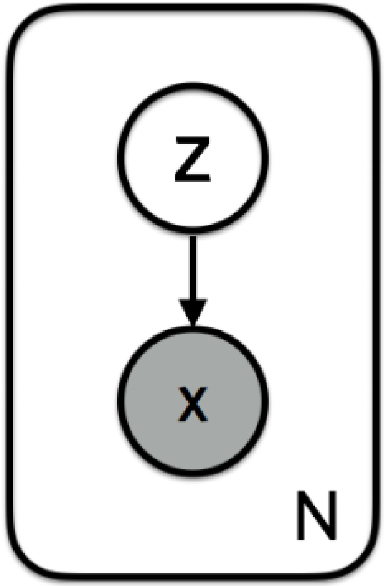




- Attempt to learn identity function
- Constrained in some way (e.g., small latent vector representation)
- Can generate new images by giving different latent vectors to trained network
- Variational: use probabilistic latent encoding

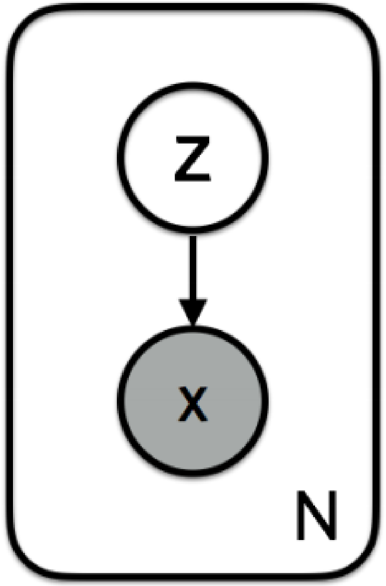
Probabilistic Model Perspective

- Data x and latent variables z
- Joint pdf of the model: $p(x, z) = p(x|z)p(z)$
- Decomposes into likelihood: $p(x|z)$, and prior: $p(z)$
- Generative process:
  Draw latent variables $z_i \sim p(z)$
  Draw datapoint $x_i \sim p(x|z)$
- Graphical model:



Probabilistic Model Perspective

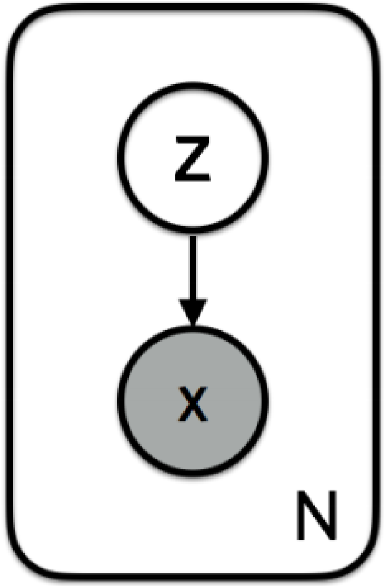
- Data x and latent variables z
- Joint pdf of the model: $p(x, z) = p(x|z)p(z)$
- Decomposes into likelihood: $p(x|z)$, and prior: $p(z)$
- Generative process:
  Draw latent variables $z_i \sim p(z)$
  Draw datapoint $x_i \sim p(x|z)$
- Graphical model:



To learn this model, we could appeal to Monte Carlo sampling or to the calculus of variations…

Probabilistic Model Perspective

- Data x and latent variables z
- Joint pdf of the model: $p(x, z) = p(x|z)p(z)$
- Decomposes into likelihood: $p(x|z)$, and prior: $p(z)$
- Generative process:
  Draw latent variables $z_i \sim p(z)$
  Draw datapoint $x_i \sim p(x|z)$
- Graphical model:



Not sure if a learnable generative model...

...or an intractable waste of time.

...so we 're going to develop a ***variational inference*** scheme using your neural building blocks!

QUESTIONS?