CPSC 540: Machine Learning

Mark Schmidt

University of British Columbia

Winter 2017

Admin

- Assignment 5:
  - Due Monday, 1 late day for Wednesday, 2 for the following Monday.
- Project description posted on Piazza.
- Final is here on April 25th at 3:30.
  - Final questions can be submitted up to April 17th.
- Bonus lecture on April 10th (same time/place) or this lecture will be extra long.

Outline

Stochastic Processes and Non-Parametric Bayes

- A stochastic process is an infinite collection of random variables $\{x^i\}$.
- Non-parametric Bayesian methods use priors defined on stochastic processes:
  - Allows extremely-flexible prior, and posterior complexity grows with data size.
  - Typically set up so that samples from posterior are finite-sized.
- The two most common priors are Gaussian processes and Dirichlet processes:
  - Gaussian processes define prior on space of functions (universal approximators).
  - Dirichlet processes define prior on space of probabilities (without fixing dimension).

Gaussian Processes

- Recall the partitioned form of a multivariate Gaussian

$$\mu = \left[\mu_x, \mu_y\right], \quad \Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix},$$

and in this case the marginal WRT x is a $\mathcal{N}(\mu_x, \Sigma_{xx})$ Gaussian.

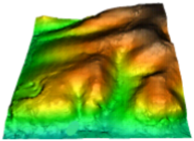
- Generalization of this to infinite set of variables is Gaussian processes (GPs):
  - Any finite set from collection follows a Gaussian distribution.

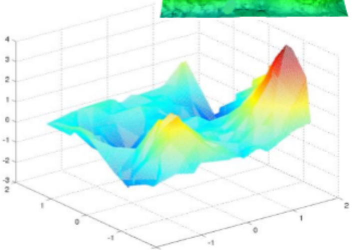
Gaussian Processes

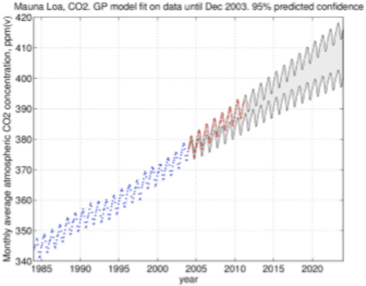
To date kriging has been used in a variety of disciplines, including the following:

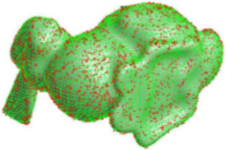
- Environmental science[5]
- Hydrogeology[6][7][8]
- Mining[9][10]
- Natural resources[11][12]
- Remote sensing[13]
- Real estate appraisal[14][15]

and many others.









Gaussian Processes

- GPs are specified by a mean function m and covariance function k,

$$m(x) = \mathbb{E}[f(x)], \quad k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))^T].$$

- We write that

$$f(x) \sim \mathsf{GP}(m(x), k(x, x')),$$

- As an example, we could have a zero-mean and linear covariance GP,

$$m(x) = 0, \quad k(x, x') = x^T x'.$$

Regression Models as Gaussian Processes

- For example, predictions made by linear regression with Gaussian prior

$$f(x) = \phi(x)^T w, \quad w \sim \mathcal{N}(0, \Sigma),$$

are a Gaussian process with mean function

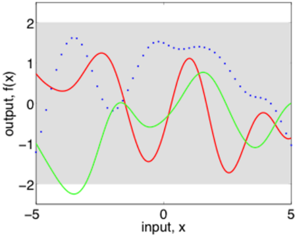
$$\mathbb{E}[f(x)] = \mathbb{E}[\phi(x)^T w] = \phi(x)^T \mathbb{E}[w] = 0.$$

and covariance function

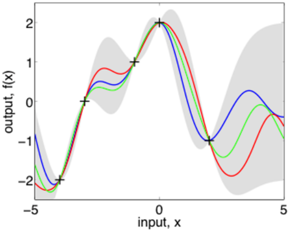
$$\mathbb{E}[f(x)f(x)^T] = \phi(x)^T \mathbb{E}[ww^T]\phi(x') = \phi(x)\Sigma\phi(x').$$

Gaussian Process Model Selection

- We can view a Gaussian process as a prior distribution over smooth functions.



(a), prior



(b), posterior

- Most common choice of covariance is RBF.
- Is this the same as using RBF kernels or the RBFs as the bases?
  - Yes, this is Bayesian linear regression plus the kernel trick.

Gaussian Process Model Selection

- So why do we care?
  - We can get estimate of uncertainty in the prediction.
  - We can use marginal likelihood to learn the kernel/covariance.
- Write kernel in terms of parameters, use empirical Bayes to learn kernel.
- Hierarchical approach: put a hyper-prior of types of kernels.
- Can be viewed as an automatic statistician:
  `http://www.automaticstatistician.com/examples`

Dirichlet Process

- Recall the finite mixture model:

$$p(x|\theta) = \sum_{c=1}^{k} \pi_c p(x|\theta_c).$$

- Non-parametric Bayesian methods allow us to consider infinite mixture model,

$$p(x|\theta) = \sum_{c=1}^{\infty} \pi_c p(x|\theta_c).$$

- Common choice for prior on π values is Dirichlet process:
  - Also called "Chinese restaurant process" and "stick-breaking process".
  - For finite datasets, only a fixed number of clusters have $\pi_c \neq 0$.
  - But don't need to pick number of clusters, grows with data size.

Dirichlet Process

- Gibbs sampling in Dirichlet process mixture model in action: `https://www.youtube.com/watch?v=0Vh7qZY9sPs`
- We could alternately put a prior on k:
  - "Reversible-jump" MCMC can be used to sample from models of different sizes.
    - AKA "trans-dimensional" MCMC.
- There a variety of interesting variations on Dirichlet processes
  - Beta process ("Indian buffet process").
  - Hierarchical Dirichlet process,.
  - Polya trees.
  - Infinite hidden Markov models.

Outline

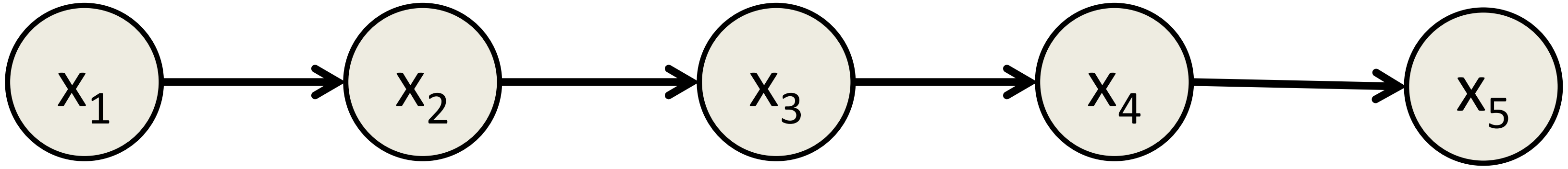
Outline

1. Non-Parametric Bayes
2. Recurrent Neural Networks
3. Generative Adversarial Networks
4. Reinforcement Learning

This section takes a lot from these sources:
http://www.cs.toronto.edu/~hinton/csc2535/notes/lec10new.pdf
https://ift6266h15.files.wordpress.com/2015/04/21_rnn.pdf

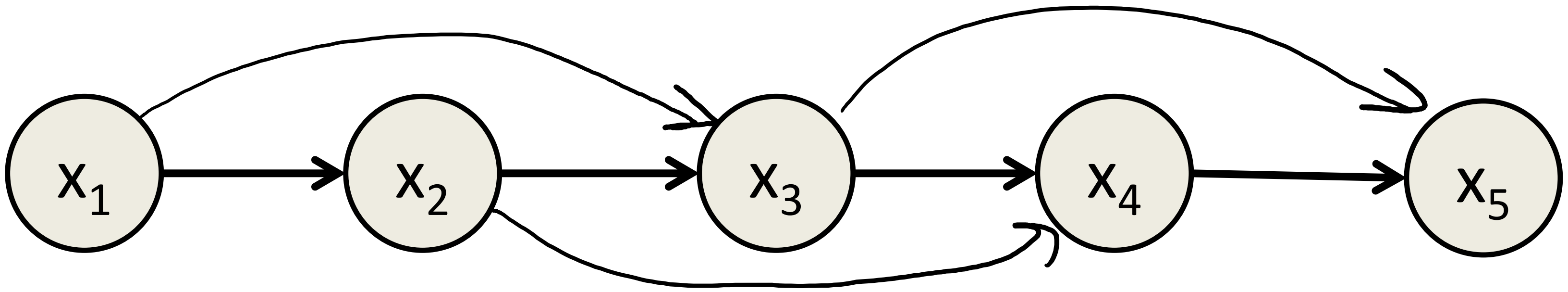
Motivation: Sequence Modeling

- 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").



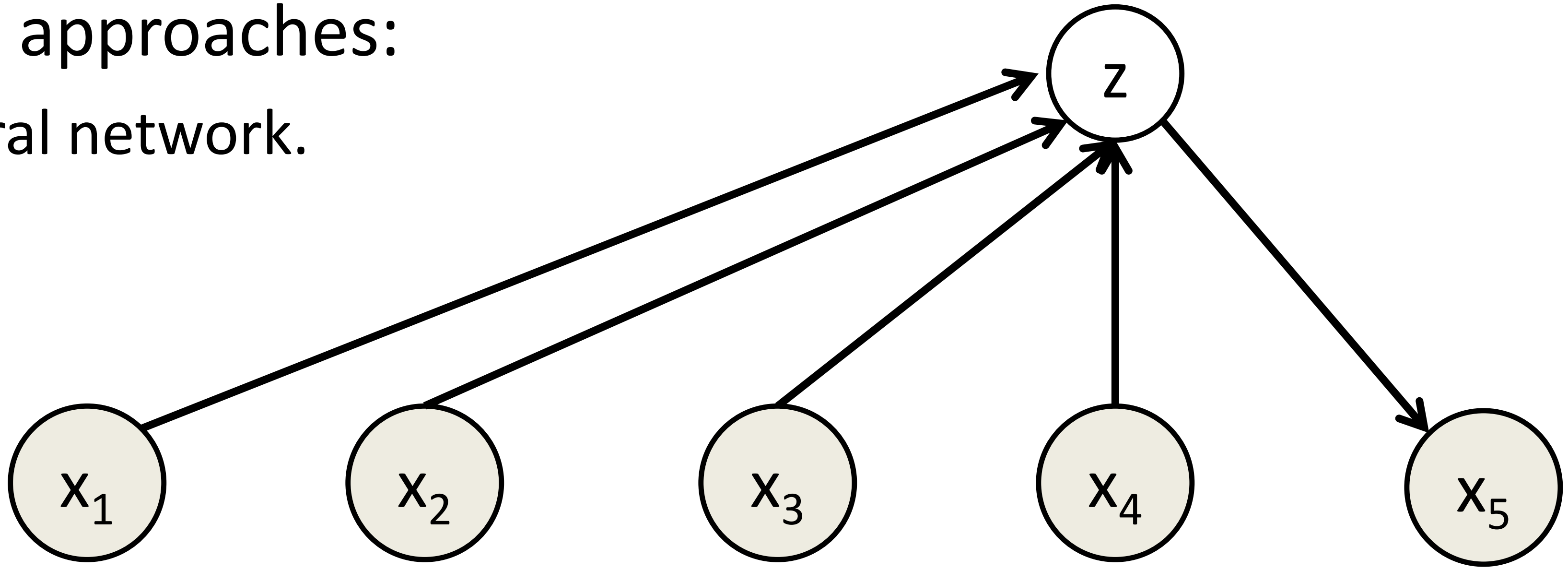
Motivation: Sequence Modeling

- 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:



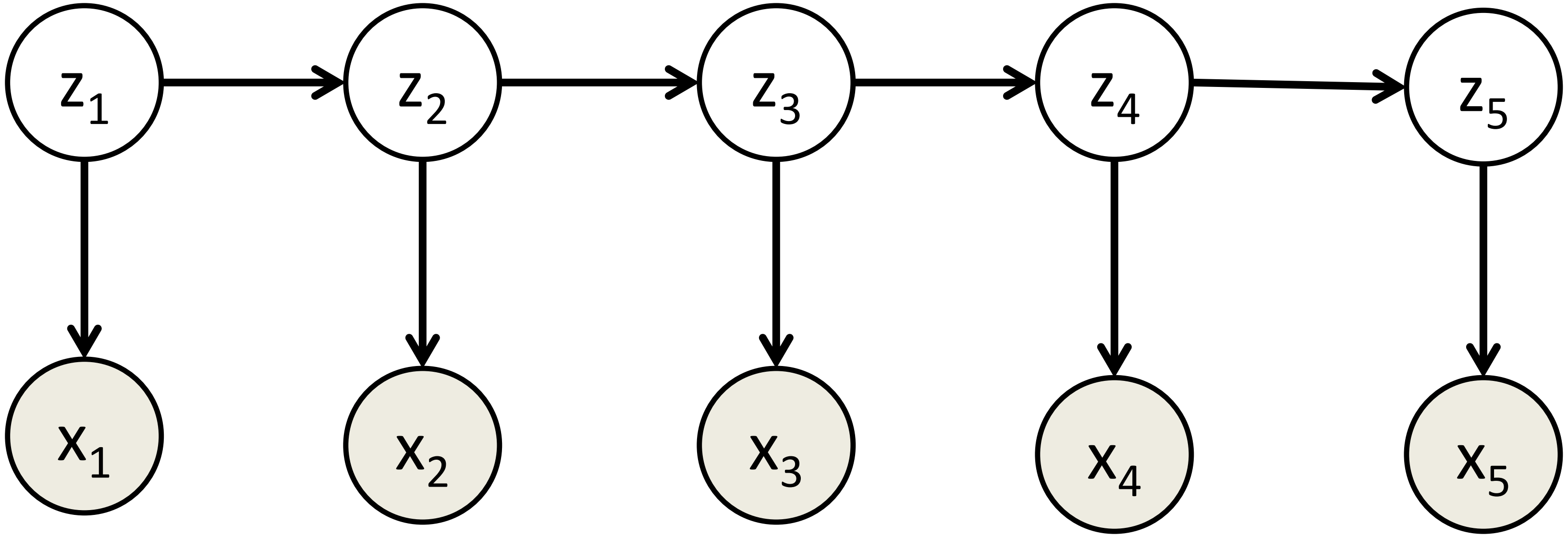
Motivation: Sequence Modeling

- 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.



Challenges of State-Space Models

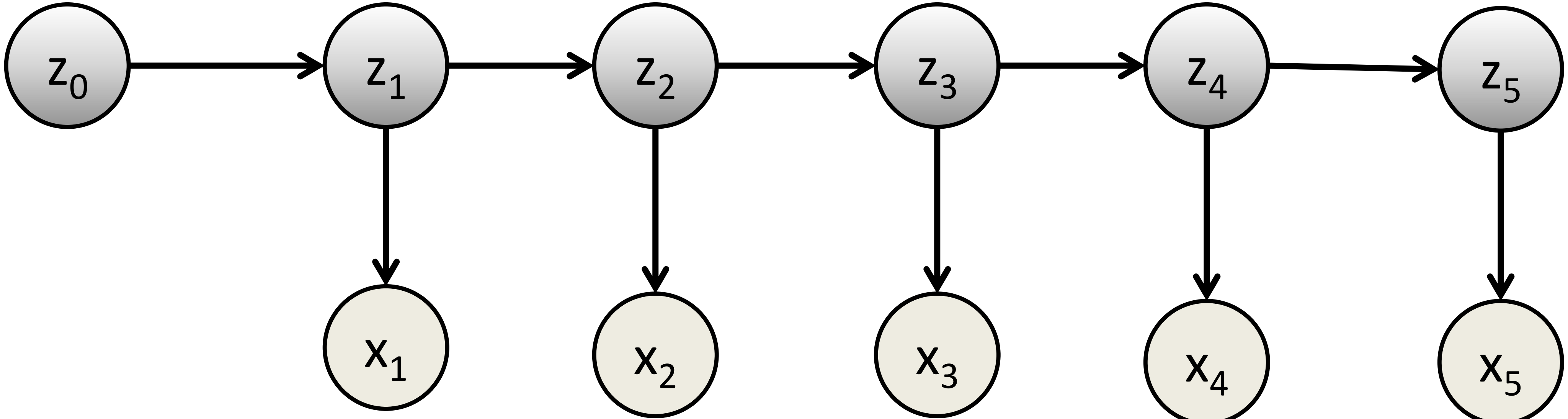
- Problem 1: inference only has closed-form in simple sitautions.
  - Markov blanket of each node must be conjugate to node.
  - Only 2 cases: Gaussian z *and* x (Kalman filter) or discrete z (HMMs).
  - Otherwise, need to use approximate inference.

- 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.

- Want (deep) hidden representation with combinatorial structure.

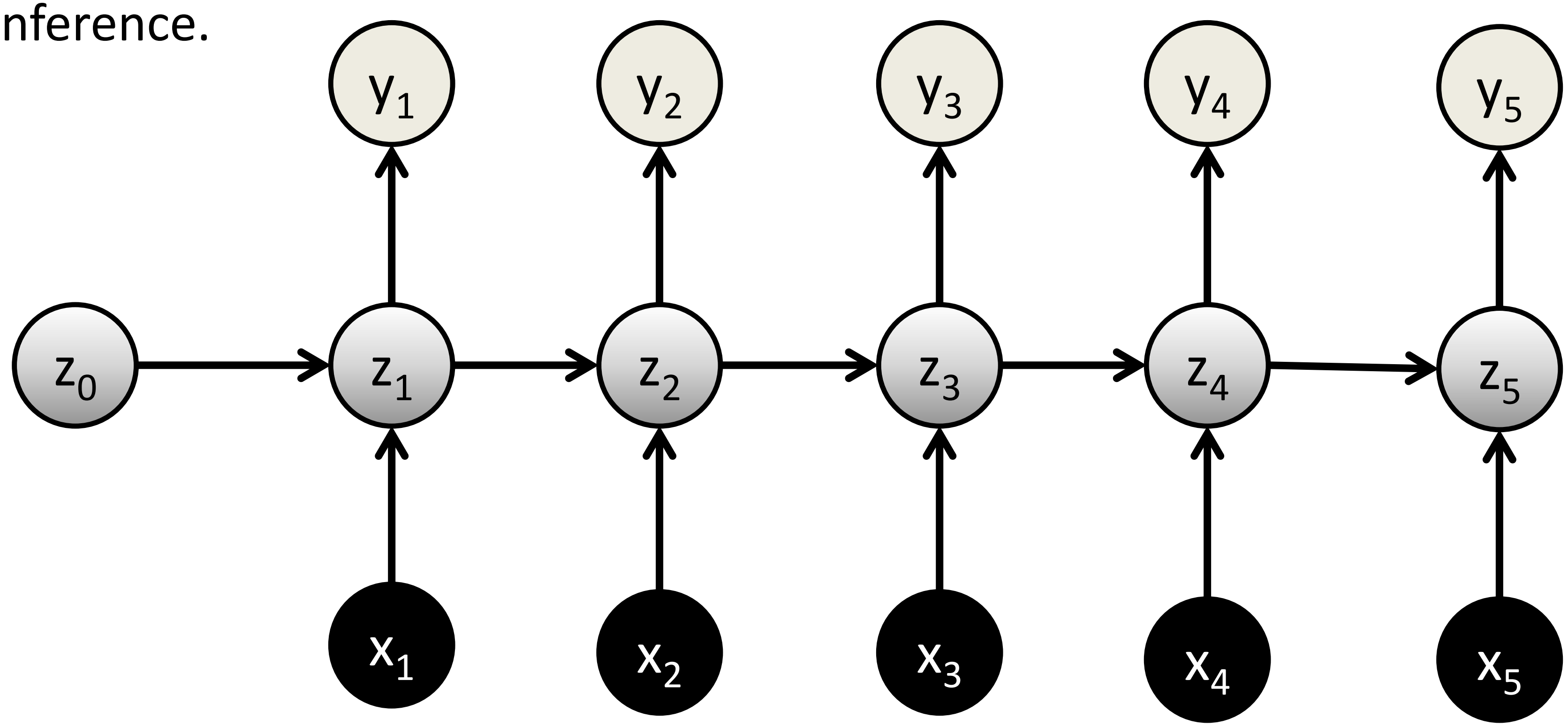
Recurrent Neural Networks

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

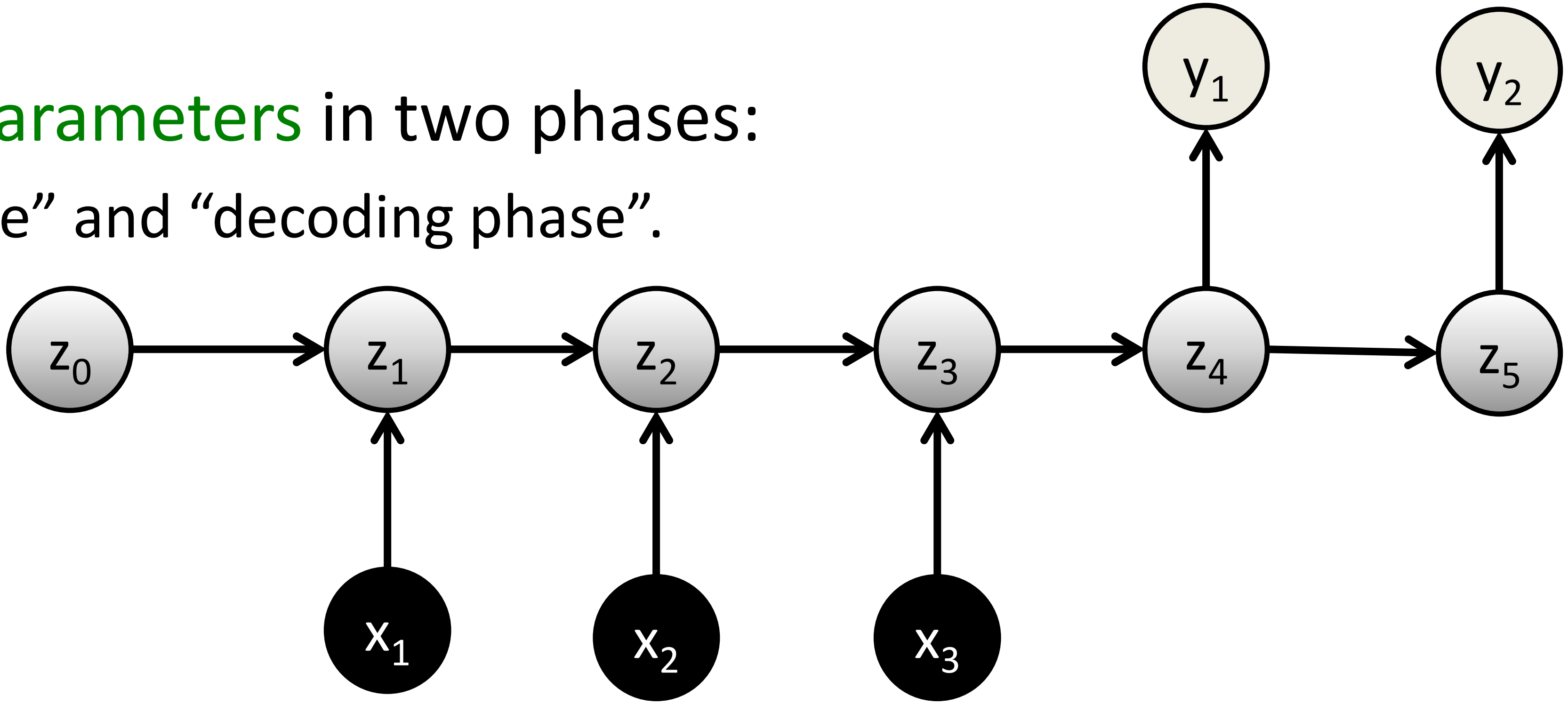
- RNNs can be used to translate input sequence to output sequence:
  - A neural network version of latent-dynamics models.
  - Deterministic transforms mean hidden 'z' can be really complicated.
    - But with easy inference.



Sequence-to-Sequence

- An interesting variation on this for sequences of different lengths:
  - Translate from French sentence ‘x’ to English sentence ‘y’.
  - Turn video frames into a sentence.

- Usually we tie parameters in two phases:
  - “Encoding phase” and “decoding phase”.



Discussion of 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.
- Interesting variations:
  - Skip connections: connections from older 'z_t' to current hidden state.
  - Bi-directional RNNs: feedforward from past and future.
  - Recursive neural networks: consider sequences through non-chain data.

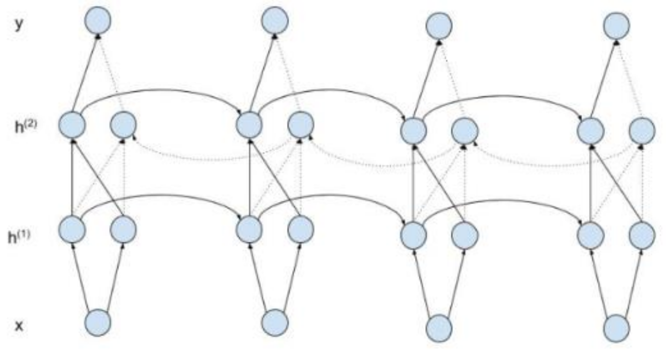
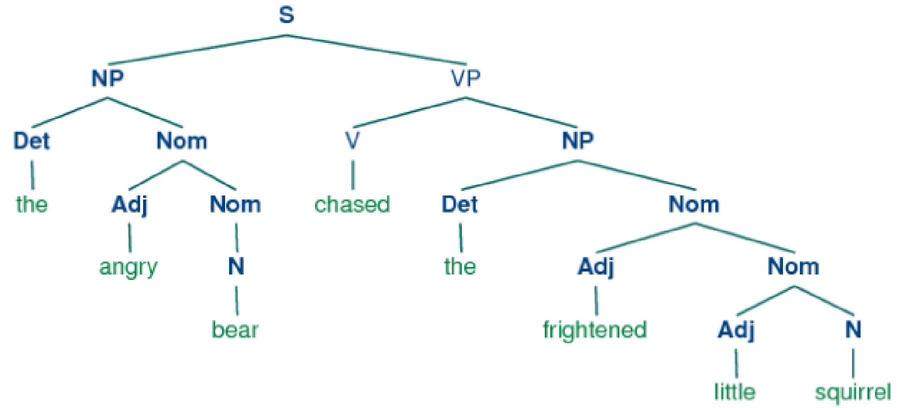


Figure 2: A deep bi-directional RNN with 2 stakced layers



Long Short Term Memory (LSTM)

- Long short term memory (LSTM) models are special case of RNNs:
  - Designed so that model can remember things for a long time.
- LSTMs are the analogy of convolutional neural networks for RNNs:
  - The trick that makes them work in applications.
- LSTMs are getting impressive performance in various settings:
  - Cursive handwriting recognition.
    - https://www.youtube.com/watch?v=mLxsbWAYIpw
  - Speech recognition.
  - Machine translation.
  - Image and video captioning.

LSTMs for Image Captioning

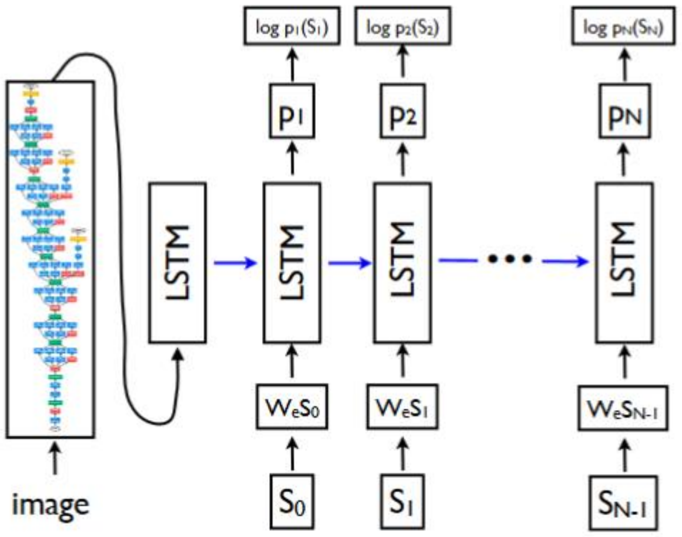
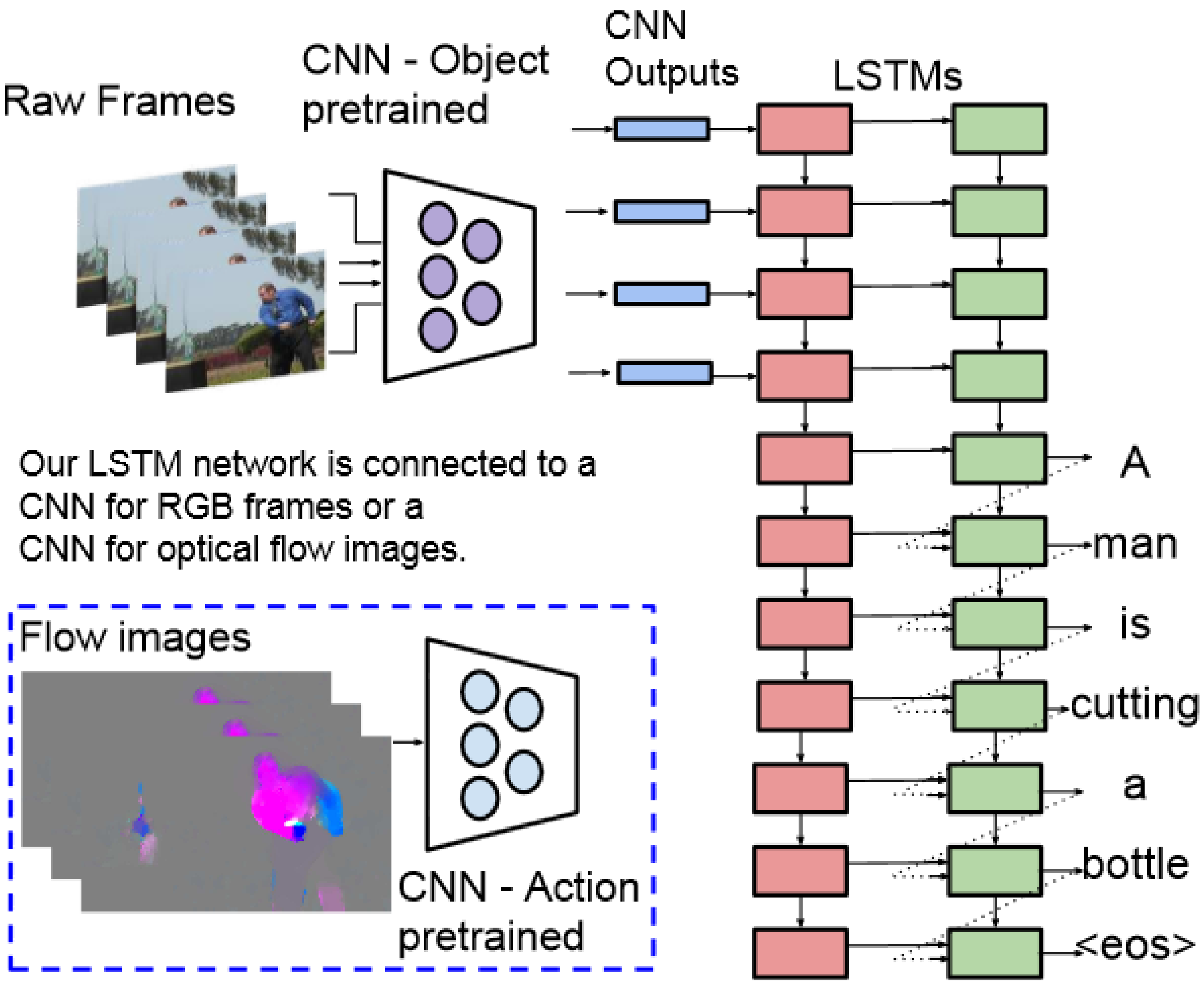


Figure 3. LSTM model combined with a CNN image embedder (as defined in [12]) and word embeddings. The unrolled connections between the LSTM memories are in blue and they correspond to the recurrent connections in Figure 2. All LSTMs share the same parameters.

Figure 5. A selection of evaluation results, grouped by human rating.

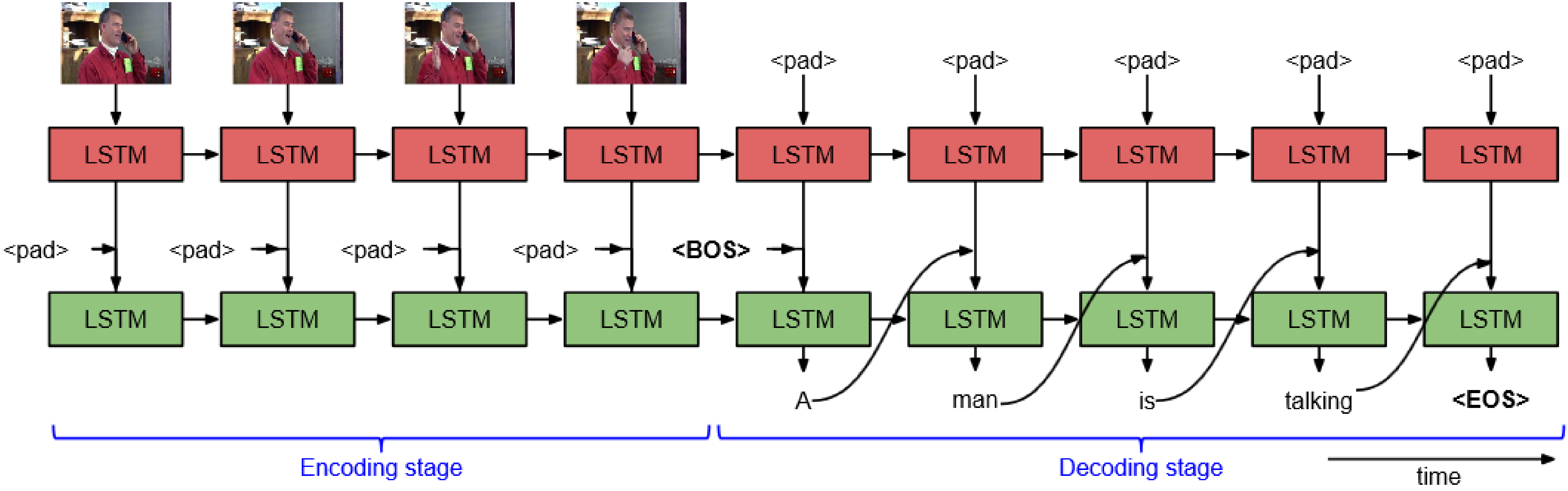
https://arxiv.org/pdf/1411.4555.pdf

LSTMs for Video Captioning



http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Venugopalan_Sequence_to_Sequence_ICCV_2015_paper.pdf

LSTMs for Video Captioning



http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Venugopalan_Sequence_to_Sequence_ICCV_2015_paper.pdf

LSTMs for Video Captioning

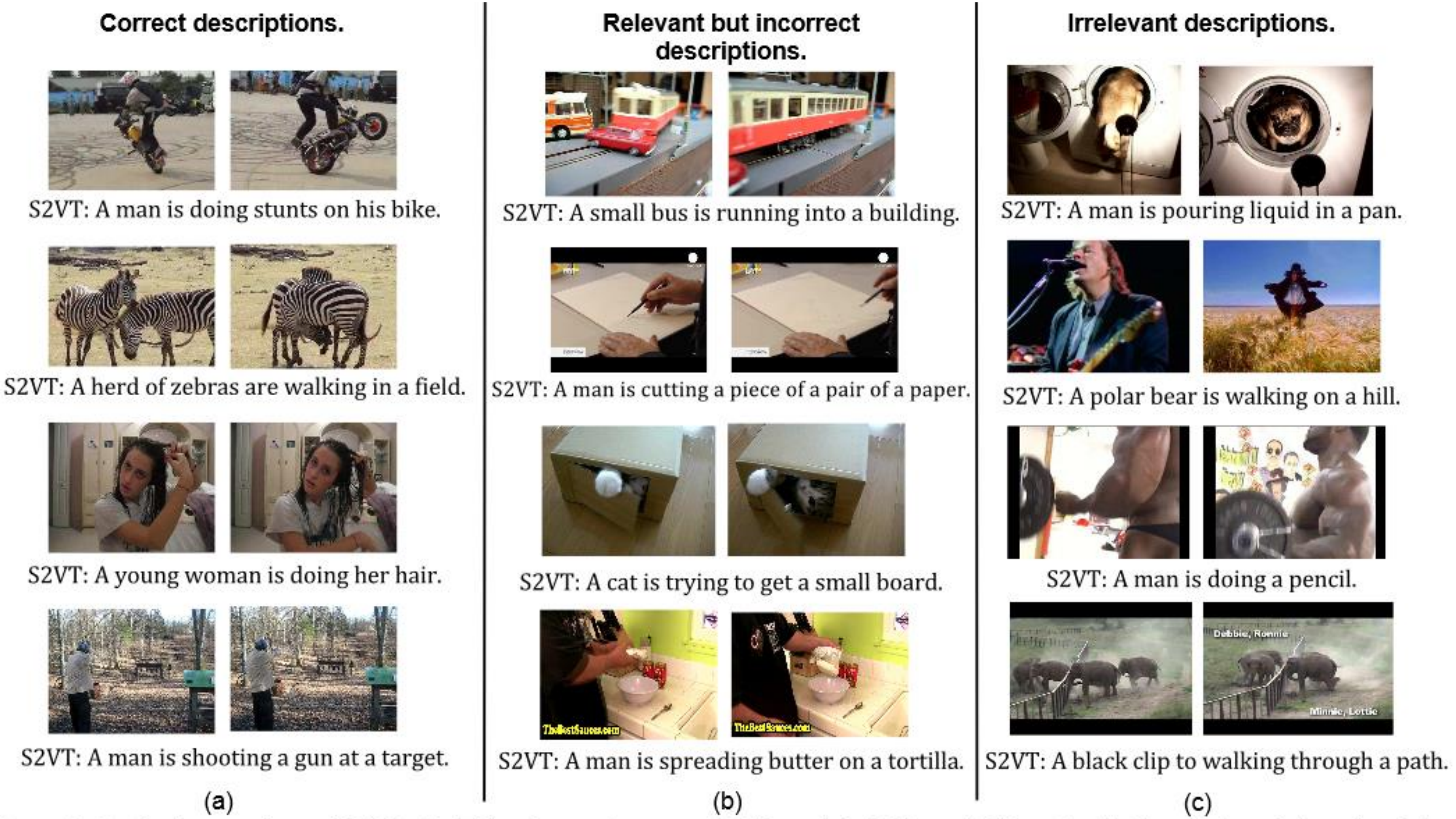


Figure 3. Qualitative results on MSVD YouTube dataset from our S2VT model (RGB on VGG net). (a) Correct descriptions involving different objects and actions for several videos. (b) Relevant but incorrect descriptions. (c) Descriptions that are irrelevant to the event in the video.

http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Venugopalan_Sequence_to_Sequence_ICCV_2015_paper.pdf

Long Short Term Memory

- In addition to usual hidden values ‘z’, LSTMs have memory cells ‘c’:
  – Purpose of memory cells is to remember things for a long time.
- Pieces of LSTM model:
  – Forget function: should we keep or forget value in a memory cell?
  – Candidate value: new value based on inputs.
  – Input function: should we take the new value?
  – Output function: should we output a value?
- Three of the above are “gate” functions:
  – Binary variables, which are approximated by sigmoids.

LSTM Structure

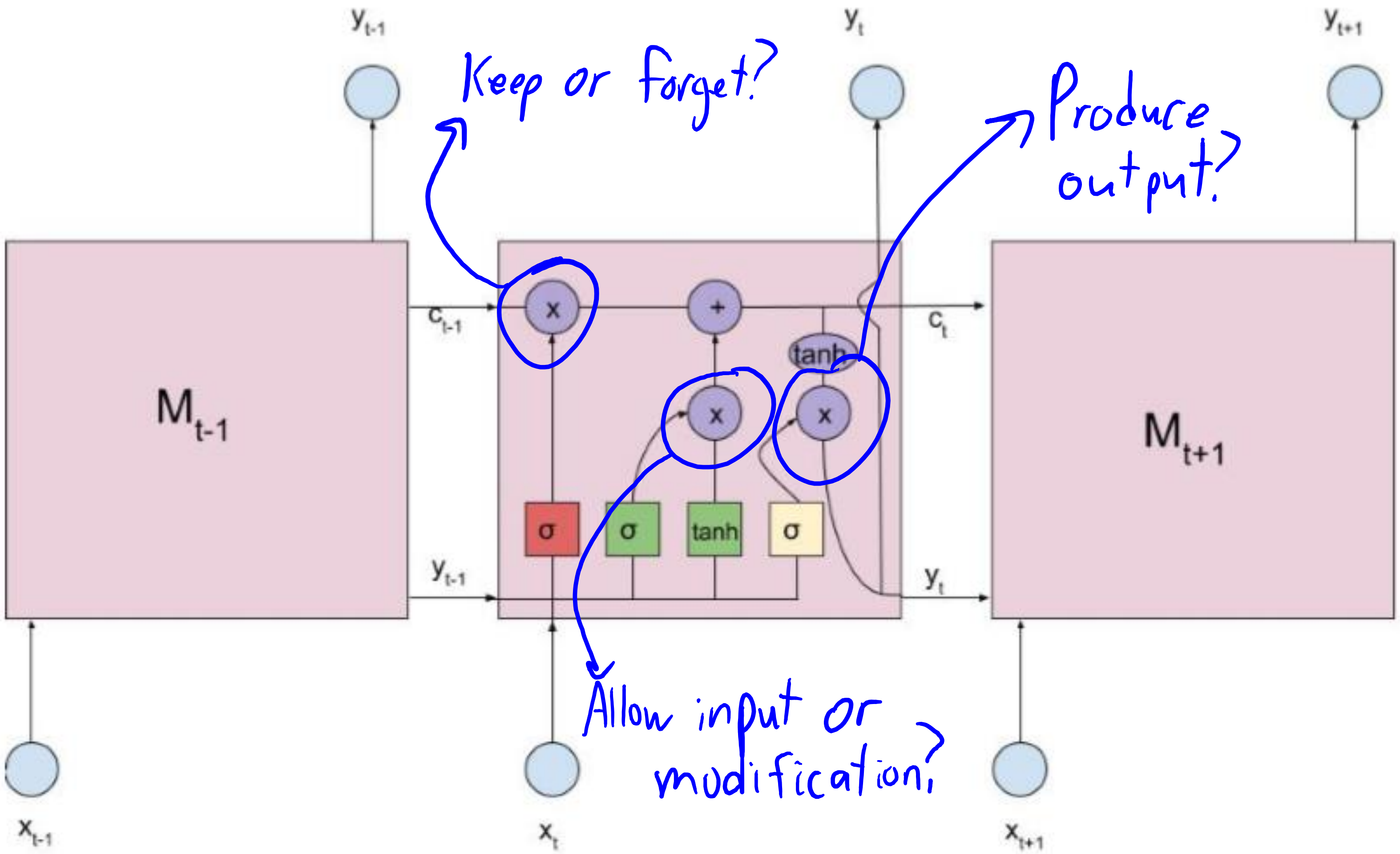


Figure 6: A close look at LSTM structure

Vanilla RNN vs. LSTM

Vanilla Recurrent Neural Network (RNN) has a recurrence of the form

$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

Previous layer, same time.

Same layer, previous time.

memory vector c_t^l. At each time step the LSTM can choose to read from, write to, or reset the cell using explicit gating mechanisms. The precise form of the update is as follows:

Forget times old memory.

Input

Forget

Output

Candidate

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

Cell $\longrightarrow$ $c_t^l = f \odot c_{t-1}^l + i \odot g$ Input times candidate.

Output $\longrightarrow$ $h_t^l = o \odot \tanh(c_t^l)$ Output times current memory

Here, the sigmoid function sigm and tanh are applied element-wise, and W^l is a $[4n \times 2n]$ matrix.

- More recent: gated recurrent unit (GRU):
  - Similar performance but a bit simpler.

More RNN Applications

- Generating text:
  - https://pjreddie.com/darknet/rnns-in-darknet
- PDF to LaTeX:

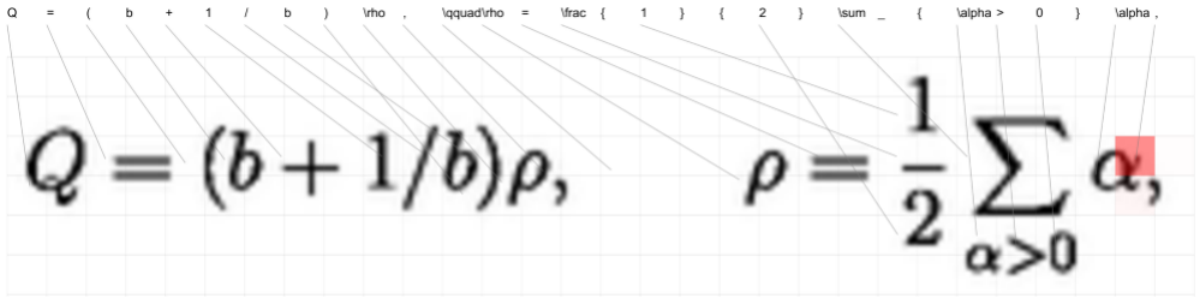


Figure 1: Example of the model generating mathematical markup. The model generates one LaTeX symbol y at a time based on the input image $\mathbf{x}$. The gray lines highlight $H' \times V'$ grid features after the CNN $\mathbf{V}$ and RNN Encoder $\tilde{\mathbf{V}}$. The dotted lines indicate the center of mass of α for each word (only non-structural words are shown). Red cells indicate the relative attention for the last token. See `http://lstm.seas.harvard.edu/latex/` for a complete interactive version of this visualization over the test set.

- Lip reading:
  - https://www.youtube.com/watch?v=5aogzAUPilE

https://arxiv.org/pdf/1609.04938v1.pdf

RNNs for Poetry

- Generating poetry:

And still I saw the Brooklyn stairs
With the shit, the ground, the golden haze
Of the frozen woods where the boat stood.
When I thought of shame and silence,
I was a broken skull;
I was the word which I called it,
And I saw the black sea still,
So long and dreary and true;
The way a square shook out my ground,
And the black things were worth a power,
To find the world in a world of reason,
And I saw how the mind saw me.

- Image-to-poetry:

A man is sitting on the edge of the waters.
I should see him begin to stand at the throat of the graveyard
and my love is like a stairway in his left arm and a piece of the stairs,
and there is a girl in the doorway and she and I am a good time.
I want to see her the best thing with the footprints in the woods
and the candle shifts back to the shrine and the last late sun
the sky and the candle and the noise of the snow.

Dropout 0.25, Loss 1.1465, 1:16:1, Railroad

A train traveling over a bridge over a river to the end of the street and the sea is a strange street with a cold sun on the street where the sun stands and the sun is still and the sun is still and the sun is gone. The sun is all around me. I am the same as the sun on the street with a strange contract.

A train traveling over a bridge over a river to the graveyard and the barn was a strange street of straw halls and the sun was always sinking in the sun.

I was the one who was still in the street when he was standing in the sun and the sun was still alive.

He was a big smile and I was a child who was a stranger.

- Movie script:
  - https://www.youtube.c

https://medium.com/artists-and-machine-intelligence/adventures-in-narrated-reality-6516ff395ba3

RNNs for Music and Dance

- Music generation:
  - https://www.youtube.com/watch?v=RaO4HpM07hE

- Text to speech and music waveform generation:
  - https://deepmind.com/blog/wavenet-generative-model-raw-audio

- Dance choreography:
  - http://theluluartgroup.com/work/generative-choreography-using-deep-learning

Beyond LSTMs

- Google’s neural machine translation incorporates attention.

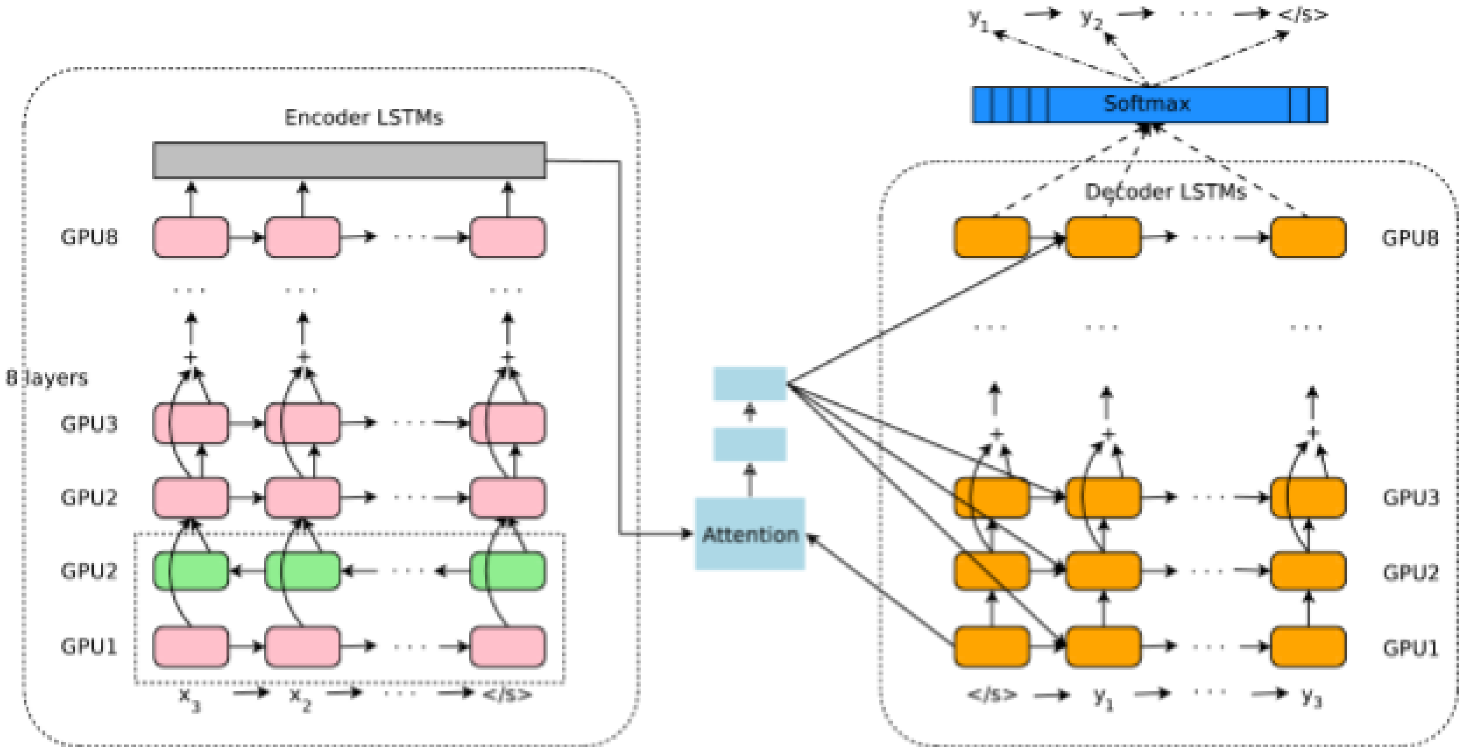


Figure 1: The model architecture of GNMT, Google’s Neural Machine Translation system. On the left is the encoder network, on the right is the decoder network, in the middle is the attention module. The bottom encoder layer is bi-directional: the pink nodes gather information from left to right while the green nodes gather information from right to left. The other layers of the encoder are uni-directional. Residual connections start from the layer third from the bottom in the encoder and decoder. The model is partitioned into multiple GPUs to speed up training. In our setup, we have 8 encoder LSTM layers (1 bi-directional layer

https://arxiv.org/pdf/1609.08144.pdf

Beyond LSTMs

- Many interesting recent variations on readable/writeable memory:
  - Memory networks and neural Turing machines.

Here is an example of what the system can do. After having been trained, it was fed the following short story containing key events in JRR Tolkien's Lord of the Rings:

Bilbo travelled to the cave.
Gollum dropped the ring there.
Bilbo took the ring.
Bilbo went back to the Shire.
Bilbo left the ring there.
Frodo got the ring.
Frodo journeyed to Mount-Doom.
Frodo dropped the ring there.
Sauron died.
Frodo went back to the Shire.
Bilbo travelled to the Grey-havens.
The End.

After seeing this text, the system was asked a few questions, to which it provided the following answers:

Q: Where is the ring?
A: Mount-Doom
Q: Where is Bilbo now?
A: Grey-havens
Q: Where is Frodo now?
A: Shire

It's probably one of the few technical papers that cite "Lord of the Rings".

https://www.facebook.com/FBAIResearch/posts/362517620591864

Outline

1. Non-Parametric Bayes
2. Recurrent Neural Networks
3. Generative Adversarial Networks
4. Reinforcement Learning

This section takes a lot from this source:
https://arxiv.org/pdf/1701.00160.pdf

Density Estimation Strikes Back

- The hottest topic at NIPS in December: density estimation?
  - In particular, deep learning for density estimation.
- Very fast-moving, but two most-popular methods are:
  - Variational autoencoders (VAEs).
  - Generative adversarial networks (GANs).
- We're getting closer to generating realistic images (not just digits):

Generative Adversarial Networks

- These models are showing promising results going beyond digits:

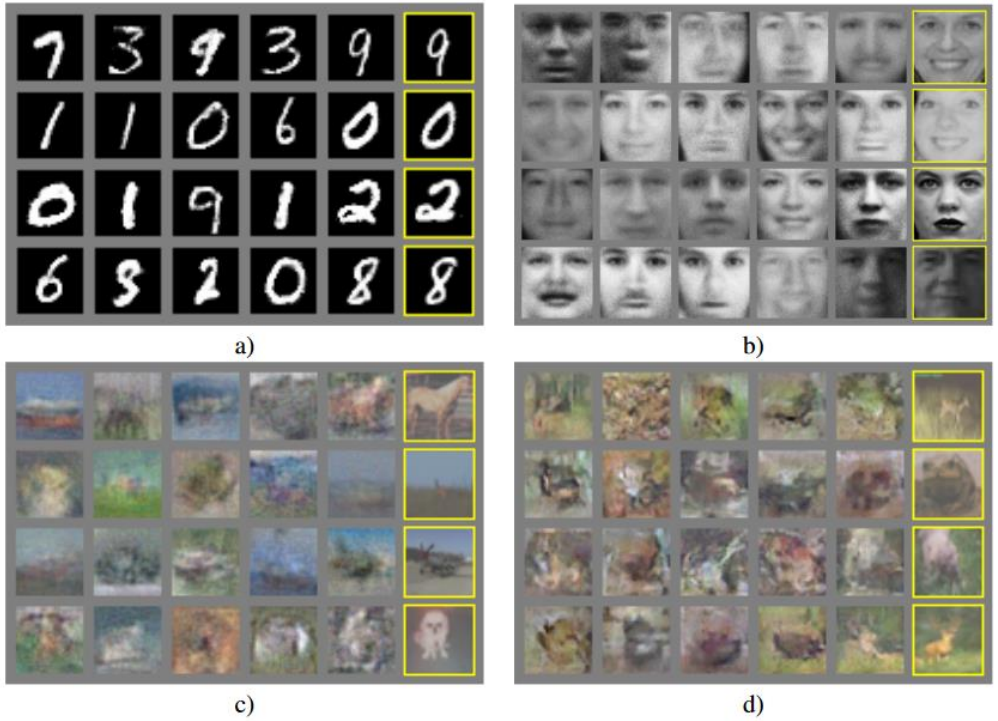


Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

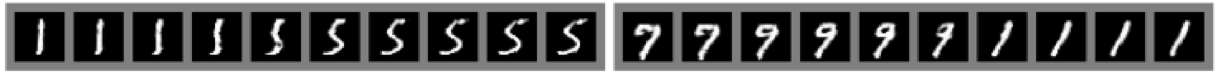
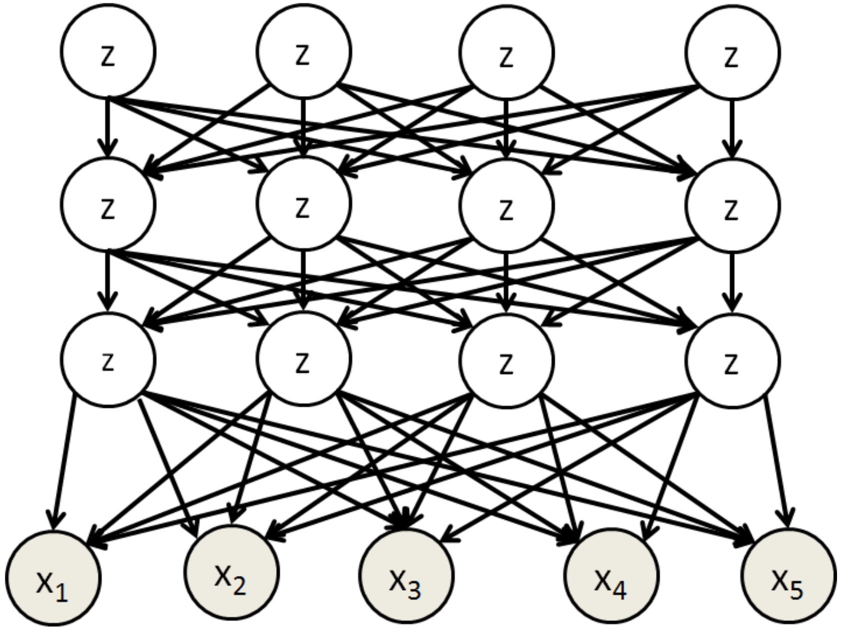


Figure 3: Digits obtained by linearly interpolating between coordinates in z space of the full model.

https://arxiv.org/pdf/1406.2661.pdf

Neural Network Generative Model

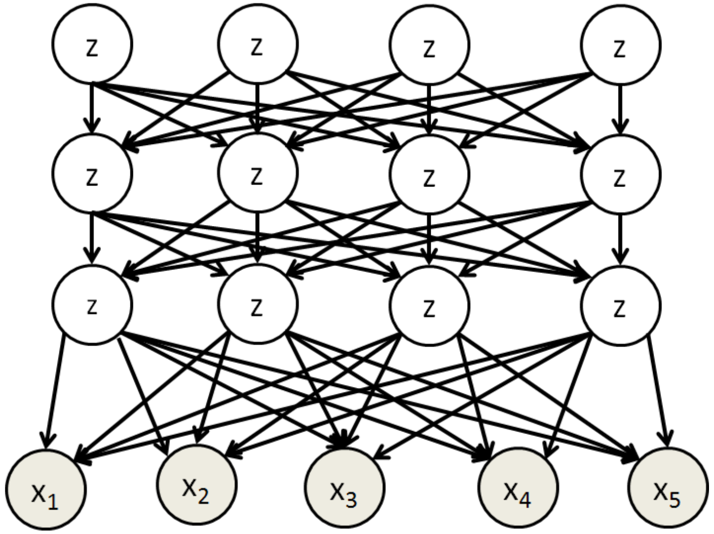
- Recall the structure of a deep belief network:



- Notice that the edges are backwards compared to neural networks.
  - We "generate" the features based on the latent 'z' variables.
- Inference is a nightmare: observing 'x' makes everything dependent.

Neural Network Generative Model

- Inference is easier if we make everything deterministic.
  - But we need randomization since otherwise you generate same ‘x’.



- We usually assume top layer comes from multivariate Gaussian.
  - So you sample a Gaussian, and neural network tries to convert to image.

Generative Adversarial Network

- Inference is still hard under the “convert Gaussian to sample”.
  - We can’t compute the likelihood needed for training.

- Key ideas of generative adversarial networks (GANs):
  - Sampling in this “generator” network is easy.
  - Use a second “discriminator” network to decide if samples look real.

- Discriminator “teaches” generator to make real-looking samples.

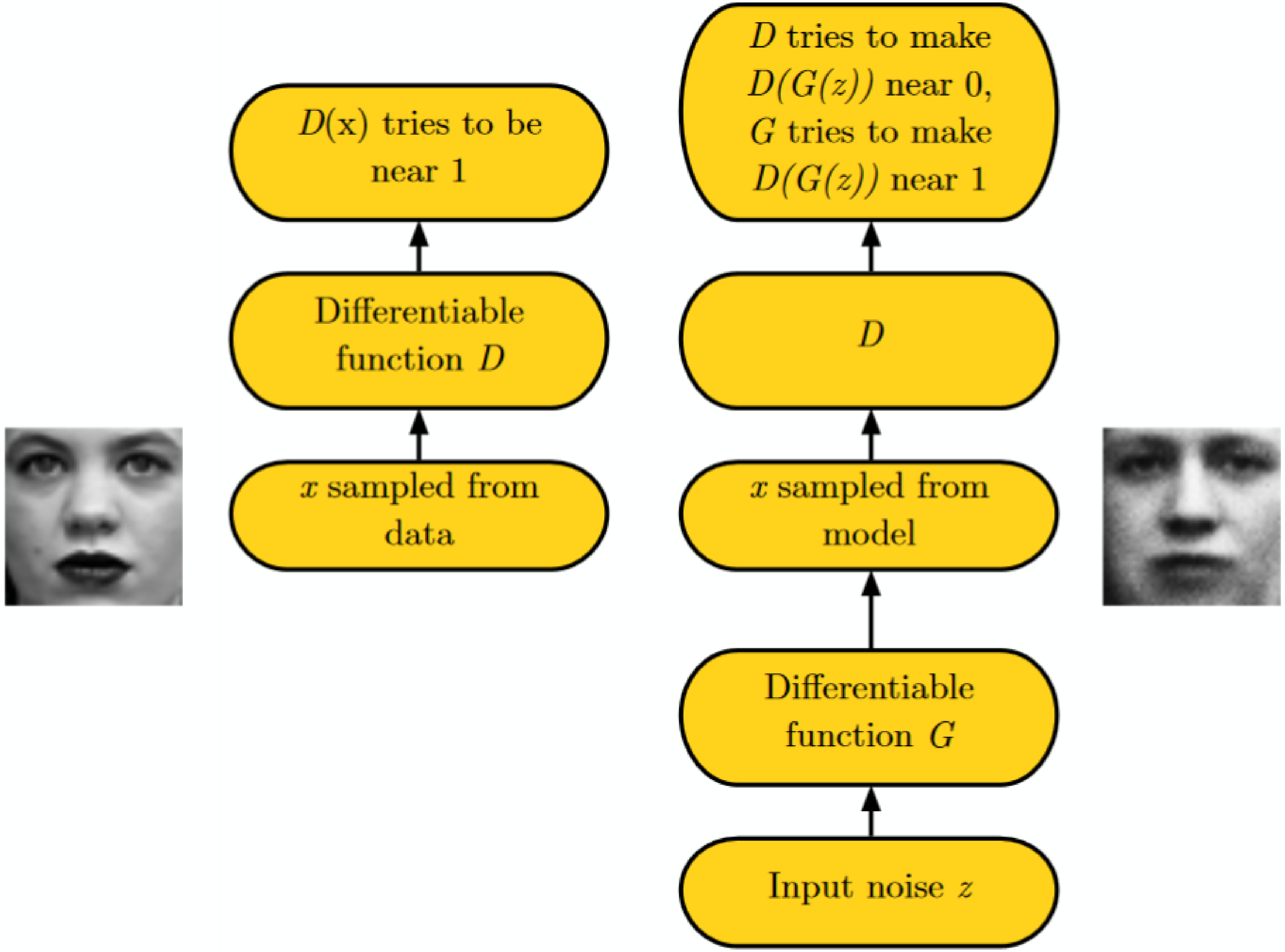
Generative Adversarial Networks

- The generator and discriminator networks compete:
  - Discriminator network trains to classify real vs. generated images.
    - Tries to maximize probability of real images, minimize probability of sampled images.
    - A standard supervised learning problem.
  - Generator network adjust parameters so samples fool the discriminator.
    - It never sees real data.
    - Trains using the gradient of the discriminator network.
      - Backpropagated through the network so samples look more like real images.
- Can be written as a saddle-point problem:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

https://arxiv.org/pdf/1406.2661.pdf

Generative Adversarial Networks

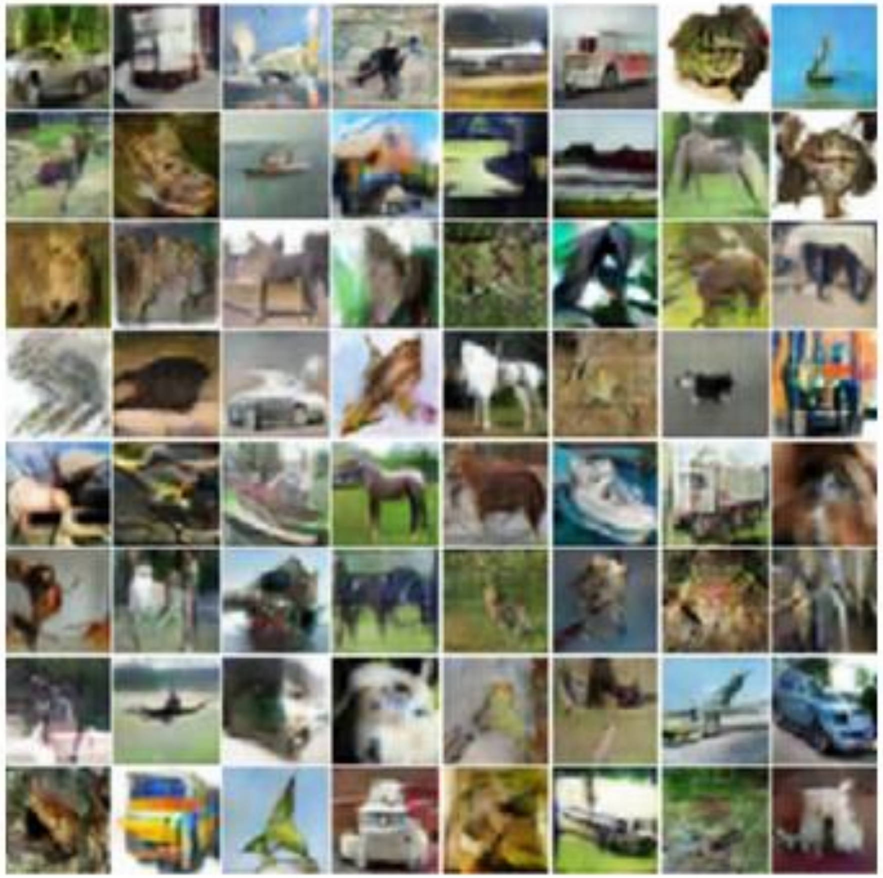


https://arxiv.org/pdf/1701.00160.pdf

Beyond Initial GAN Model

- Improving GANs is an active research area…

Real images (CIFAR-10)



Generated images

https://blog.openai.com/generative-models

GANs for Other Problems

- GANs for text-to-image translation:

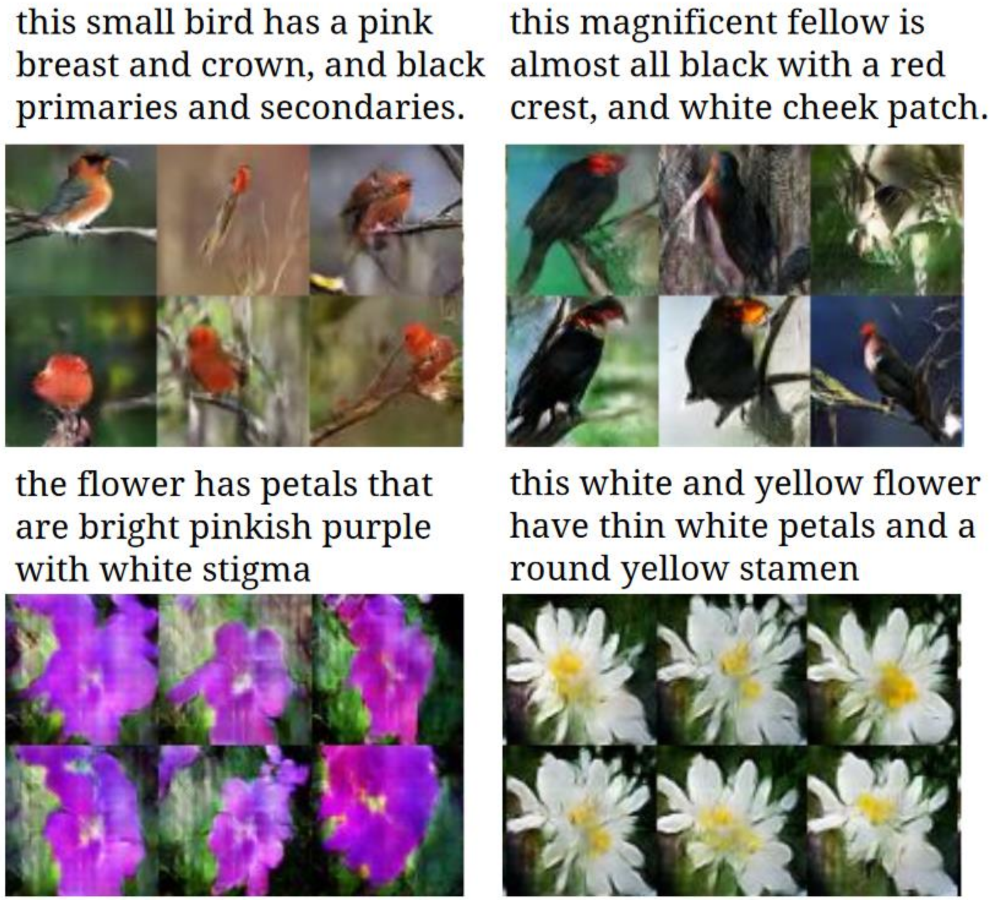


Figure 23: Text-to-image synthesis with GANs. Image reproduced from Reed *et al.* (2016b).

https://arxiv.org/pdf/1701.00160.pdf

GANs for Other Problems

- GANs for text-to-image translation:

Figure 25: StackGANs are able to achieve higher output diversity than other GAN-based text-to-image models. Image reproduced from Zhang *et al.* (2016).

https://arxiv.org/pdf/1701.00160.pdf

GANs for Other Problems

- GANs for super-resolution:

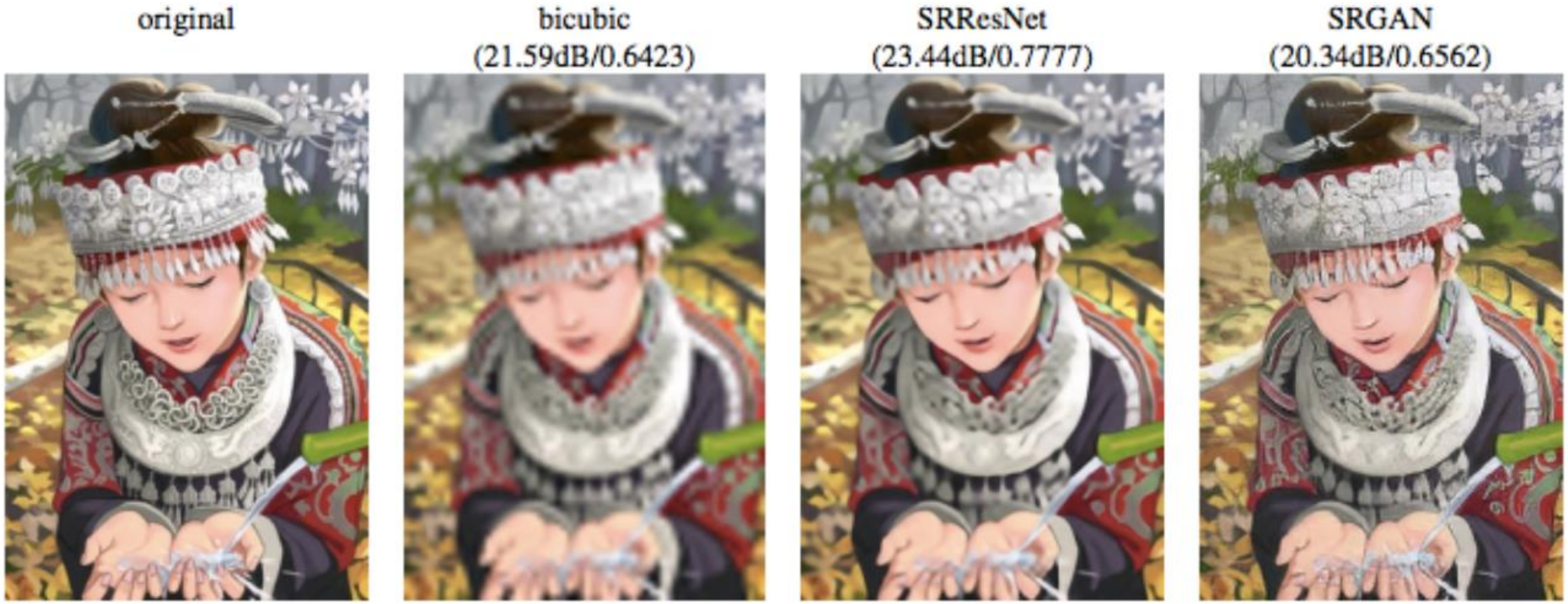


Figure 4: Ledig *et al.* (2016) demonstrate excellent single-image superresolution results that show the benefit of using a generative model trained to generate realistic samples from a multimodal distribution. The leftmost image is an original high-resolution

https://arxiv.org/pdf/1701.00160.pdf

GANs for Other Problems

- GANs for image manipulation:
  - https://www.youtube.com/watch?v=9c4z6YsBGQ0
  - https://www.youtube.com/watch?v=FDELBFSeqQs

GANs for Other Problems

- GANs for image-to-image translation:
  - https://affinelayer.com/pixsrv

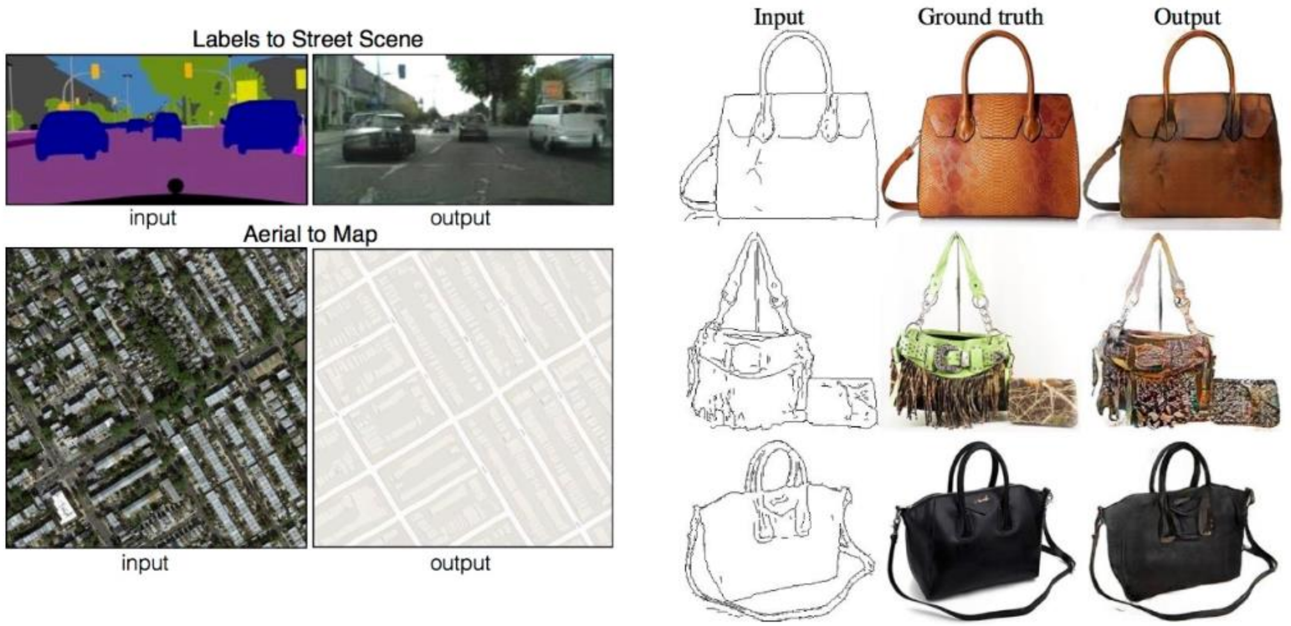


Figure 7: Isola *et al.* (2016) created a concept they called image to image translation, encompassing many kinds of transformations of an image: converting a satellite photo into a map, coverting a sketch into a photorealistic image, etc. Because many of these conversion processes have multiple correct outputs for each input, it is necessary to use generative modeling to train the model correctly. In particular, Isola *et al.* (2016) use a GAN. Image to image translation provides many examples of how a creative algorithm designer can find several unanticipated uses for generative models. In the future, presumably many more such creative uses will be found.

https://arxiv.org/pdf/1701.00160.pdf

In Progress…

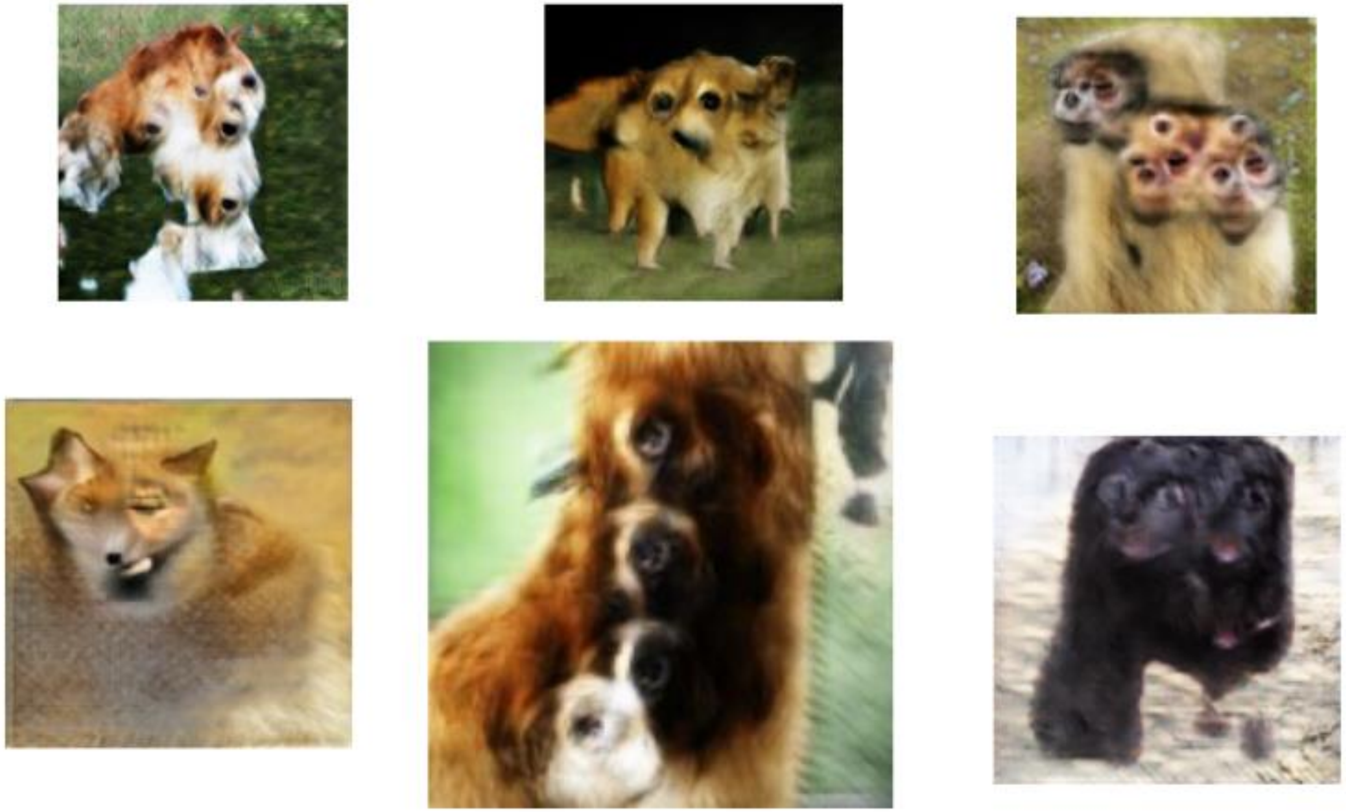


Figure 29: GANs on 128 × 128 ImageNet seem to have trouble with counting, often generating animals with the wrong number of body parts.

https://twitter.com/search?q=%23edges2cats&lang=en

https://arxiv.org/pdf/1701.00160.pdf

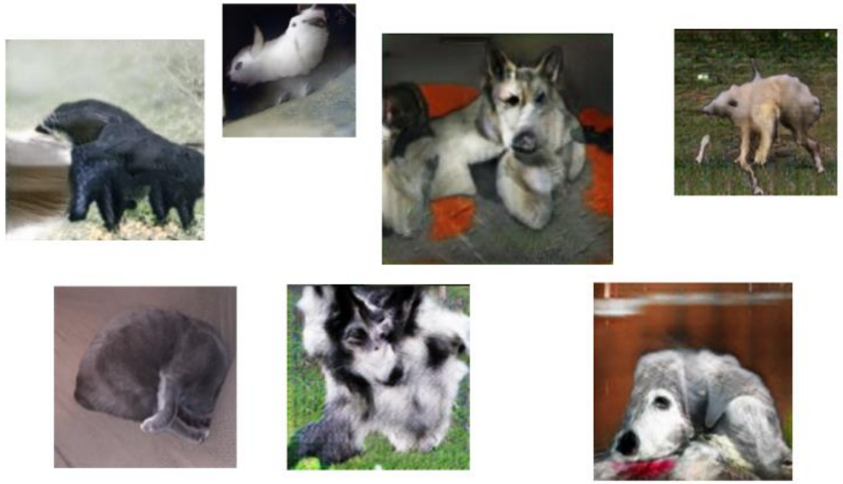


Figure 30: GANs on 128 × 128 ImageNet seem to have trouble with the idea of three-dimensional perspective, often generating images of objects that are too flat or highly axis-aligned. As a test of the reader's discriminator network, one of these images is actually real.

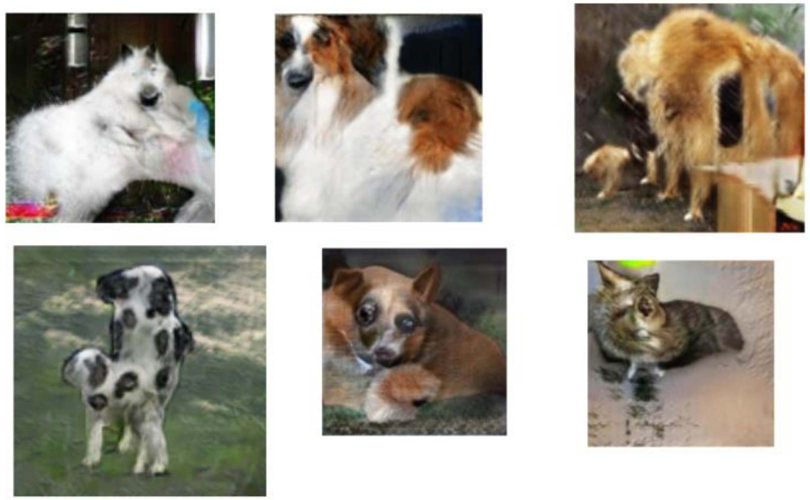


Figure 31: GANs on 128 × 128 ImageNet seem to have trouble coordinating global structure, for example, drawing "Fallout Cow," an animal that has both quadrupedal and bipedal structure.

Plug and Play Generative Networks

- New generative models are appearing at a very-fast rate:

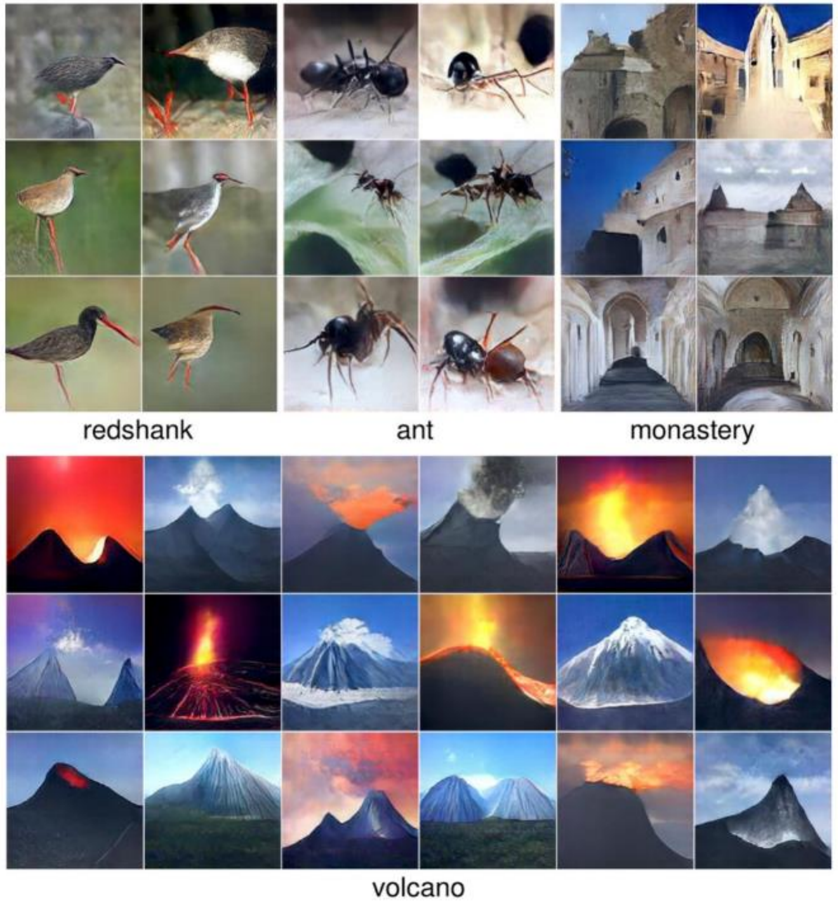


Figure 33: PPGNs are able to generate diverse, high resolution images from ImageNet classes. Image reproduced from Nguyen *et al.* (2016).

Outline

1. Non-Parametric Bayes
2. Recurrent Neural Networks
3. Generative Adversarial Networks
4. Reinforcement Learning

Why Reinforcement Learning?

https://www.youtube.com/watch?v=Ih8EfvOzBOY
https://www.youtube.com/watch?v=SH3bADiB7uQ
https://www.youtube.com/watch?v=nUQsRPJ1dYw

Building up to Reinforcement Learning

- Reinforcement learning (RL) is very general/difficult:
  - It includes many other machine learning problems as special cases.

- Other names for reinforcement learning:
  - Approximate dynamic programming.
  - Neurodynamic programming.

- To build up to RL, let's start with supervised learning:
  - Introduce notation, and discuss ways RL is harder.

Supervised Learning

- Supervised learning notation:
  - We have input features x^t.
  - There are possible outputs y^t.
  - We have a loss function $L(x^t,y^t)$.
    - E.g., loss of 0 if you classify correctly and loss of 1 is you classify incorrectly.

- Reinforcement learning notation:
  - The features are referred to as states s^t.
  - The outputs are referred to as actions a^t.
  - The (negative) loss function is called the reward $r(s^t,a^t)$.
    - E.g., reward of 0 if you classify correctly and reward of -1 if you classify incorrectly.

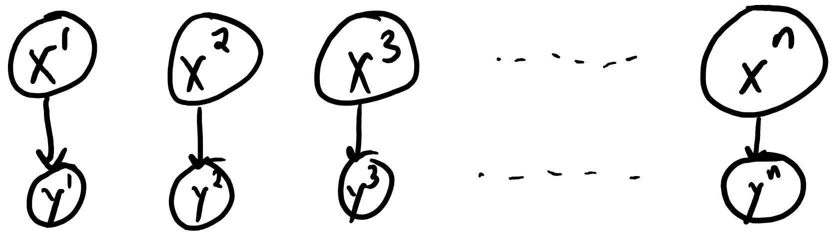
Supervised Learning

- Supervised learning training phase:
  - We have ‘n’ training examples, we can do whatever we want with them.
  - The output of training is a classifier: maps from x^t to y^t.
  - This is called a policy in RL: policies map from s^t to a^t.
- Goal: classifier minimizes loss ⇔ policy maximizes reward

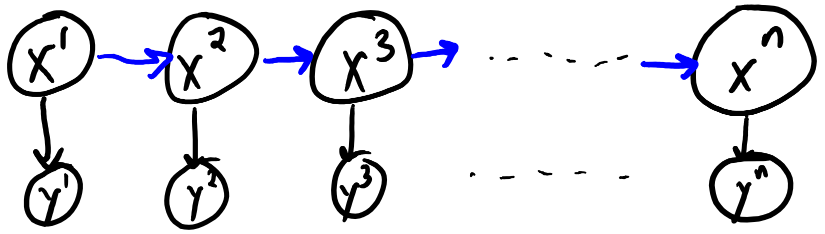
- Some models give score for each label:
  - For example, softmax gives probability of each y^t given x^t.
  - This is a Q function: $Q(s^t,a^t)$ is “value” of action a^t in state s^t.
  - Given a policy, we can define the value function $V(s^t)$ as “value” given policy that chooses a^t (which may be deterministic or stochastic).

State-Space Models

- In standard supervised learning setup, the x^t are IID samples:



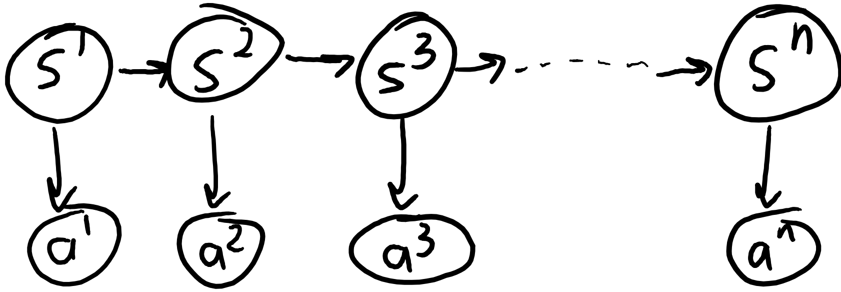
- In state-space models, the x^t come from a Markov chain:



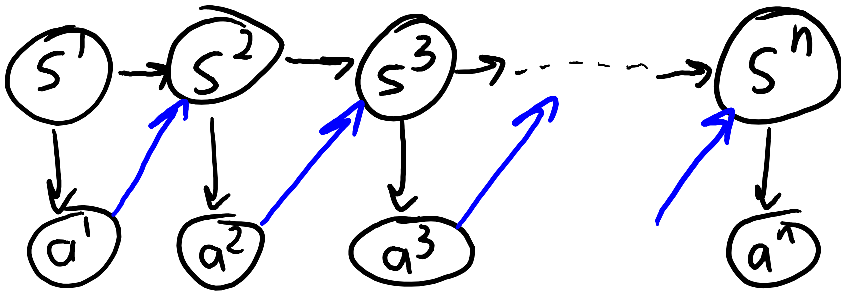
  - Value of x^t depends on the value of x^{t-1}.
  - We obtain IID samples in the special case of no dependencies.
  - Learning if we observe the x^t full-observed DAG is pretty similar.

Markov Decision Processes

- State-space model in RL notation



- In Markov decision processes (MDPs), s^t also depends on a^{t-1}.
  - The action affects the value of the next state.
    - Here we need planning:
      - Choose actions that will lead to future states with high reward.
  - In MDPs we assume we have the "model":
    - Know all rewards $r(s^t,a^t)$ and transition probabilities $p(s^t \mid s^{t-1}, a^{t-1})$.



  - Given "model", we can find optimal values/policy by dynamic programming:
    - [Value iteration and policy iteration]

Reinforcement Learning

- Reinforcement learning is MDPs when we don't know the "model".
  - All we can do is take actions and observe states/rewards that result.

- We need to simultaneously solve three problems:
  - We need to solve a supervised learning problem, $r(s^t,a^t)$.
  - We need to discover dynamics of a state-space model, $p(s^t \mid s^{t-1}, a^{t-1})$.
  - We need to plan an MDP policy maximizing long-term reward, $s^t \rightarrow a^t$.
- All while working with simulations.

- Unfortunately, this combination gives a few more challenges…

Active Learning

- Let's go back to the basic supervised learning setting:
  - Features s^t are just IID samples.

- Active learning considers the following variation:
  - The training examples are unlabeled.
  - The learner can query the user to label a training example s^t.
  - Goal is to do well with a limited budget of queries.

- The limited budget means we can't visit all features/states.
  - Here we need exploration: which states do we visit to learn the most?

Online Learning and Bandit Feedback

- In online learning there is no separate training/testing phase:
  - We receive a sequence of features/states s^t.
  - We have to choose prediction/action a^t on each example as it arrives.
  - Our "score" is the average loss/reward over time.
  - Here we need to predict well as we go (not at the end).
    - You pay a penalty for trying bad actions as you are learning.

- A common variation is with bandit feedback:
  - We only observe the reward function $r(s^t,a^t)$ for actions a^t that we choose.
  - Here we have an exploration vs. exploitation trade-off:
    - Should we explore by picking an a^t we don't know much about?
    - Should we exploit by picking an a^t that gives high reward?

Causal Learning

- Causal learning:
  - Observational prediction:
    - Do people who take Cold-FX have shorter colds?
  - Causal prediction:
    - Does taking Cold-FX cause you to have shorter colds?
  - Counter-factual prediction:
    - You didn't take Cold-FX and had long cold, would taking it have made it shorter?
- Here we need to learn effects of actions.
  - Including predicting effects of new actions.
- We may not control the actions: off-policy learning.
  - Actions are often randomized, but still want to find best actions.

Reinforcement Learning

- Reinforcement needs to consider:
  - Modeling how (s^t,a^t) combinations affects reward (supervised learning)
  - Learning how (s^t,a^t) affects s^{t+1} (state-space models, causality).
  - Planning for long-term reward (MDPs).
  - Exploring space of states and actions (active learning, bandit feedback).

- Two common frameworks:
  - Monte Carlo methods collects a lot of simulations to turn it into an MDP.
  - Temporal-difference learning considers online prediction as you go.
    - Need to consider exploration vs. exploitation, penalties for trying bad actions.

Outline

1. Non-Parametric Bayes
2. Recurrent Neural Networks
3. Generative Adversarial Networks
4. Reinforcement Learning
5. What’s next?

My Original Plan

- CPSC 340:

1. Data representation/summarization.
2. Supervised learning (counting/distances)
3. Unsupervised learning (counting/distances)
4. Supervised learning (linear models).
5. Unsupervised learning (latent-factor).
6. Deep Learning.
7. Random walks.

- CPSC 540:

1. Large-Scale Learning.
2. Density Estimation.
3. Graphical Models.
4. More Deep Learning.
5. Bayesian Methods.
6. Causal, active, and online learning.
7. Reinforcement learning.

Hopefully next year we'll have 3 courses (not clear if it will be 240, 440, or 550).

Remaining Topics

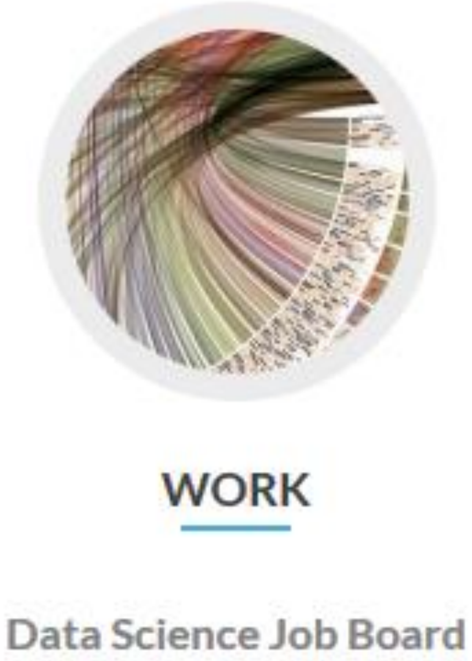
- For online learning, active learning, and causality:
  - We'll be covering these in the MLRG this summer:
    - http://www.cs.ubc.ca/labs/lci/mlrg

- To learn about reinforcement learning:
  - Read Sutton ad Barto's "Introduction to Reinforcement Learning".
  - You can also take EECE 592.

- Other major topics we didn't cover:
  - Learning theory (VC dimension).
  - Probabilistic context-free grammars (recursive version of Markov chains).
  - Relational models (Markov logic networks).
  - Sub-modularity (discrete version of convexity).
  - Spectral methods (consistent HMMs).

Data Science Job Board

- Many local companies are looking for people with CPSC 540 skills.
- If you are looking for local jobs, go here and make a profile.
  - http://makedatasense.ca/jobs



- Thank you for your patience, I'm still learning to teach!