Recurrent Neural Networks

CPSC 440/550: Advanced Machine Learning

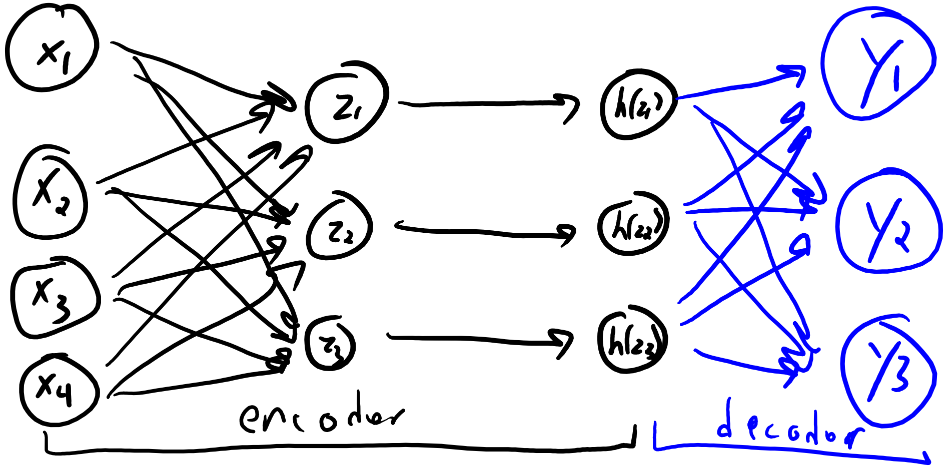
`cs.ubc.ca/~dsuth/440/23w2`

University of British Columbia, on unceded Musqueam land

2023-24 Winter Term 2 (Jan–Apr 2024)

Last Time: Multi-Class Neural Networks

- We discussed multi-class classification with neural networks:



$$\hat{y}_1 = v_1^T h(Wx)$$
$$\hat{y}_2 = v_2^T h(Wx)$$
$$\hat{y}_3 = v_3^T h(Wx)$$

- We use the softmax function to convert the $\hat{y}_c$ to probabilities:
  - We use this for inference.
  - Likelihood is softmax for true label.
  - Last layer is all that changes.

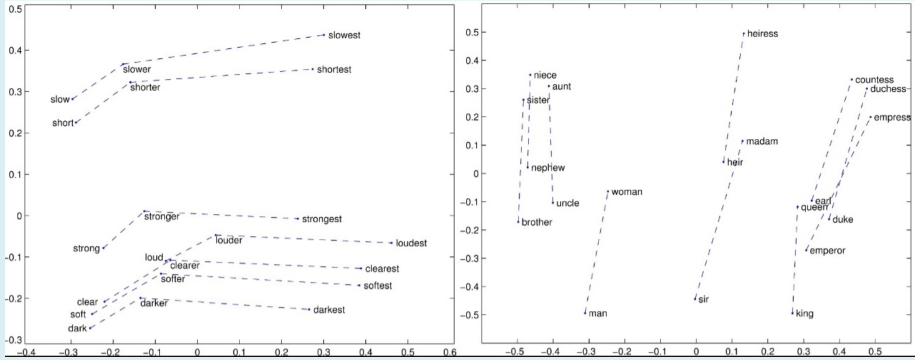
$$p(y = c \mid x, W, v) = \frac{\exp(\hat{y}_c)}{\sum_{c'=1}^{k} \exp(\hat{y}_{c'})}$$

- We train by minimizing the sum of negative log-likelihoods over *i*.
  - We can add multiple layers, convolution layers, max pooling, ReLu, and so on.

review

Review: Word Representations

- How do we represent words with features?
- Lexical features:
  - Represent words using a "1 of k" encoding
    - Where *k* is the number of words in training data
      - Or "words that appear at least 5 times in the training data"
      - Set all these features to 0 for other words
  - Or: sample a random high-dim vector per word
    - If *d* really big but << *k*, still approximately orthogonal

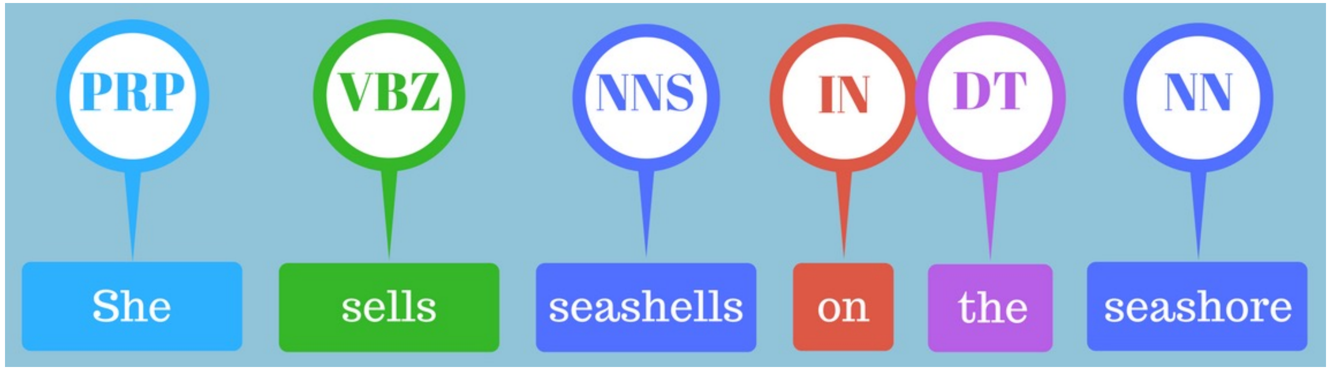


- Latent-factor models like word2vec, GloVe, fasttext:
  - Unsupervised learning of continuous features for each word
    - Distances in this space may approximate semantic meaning
    - May do sensible things for words not seen during training

http://sebastianruder.com/secret-word2vec

Motivation: Part of Speech (POS) Tagging

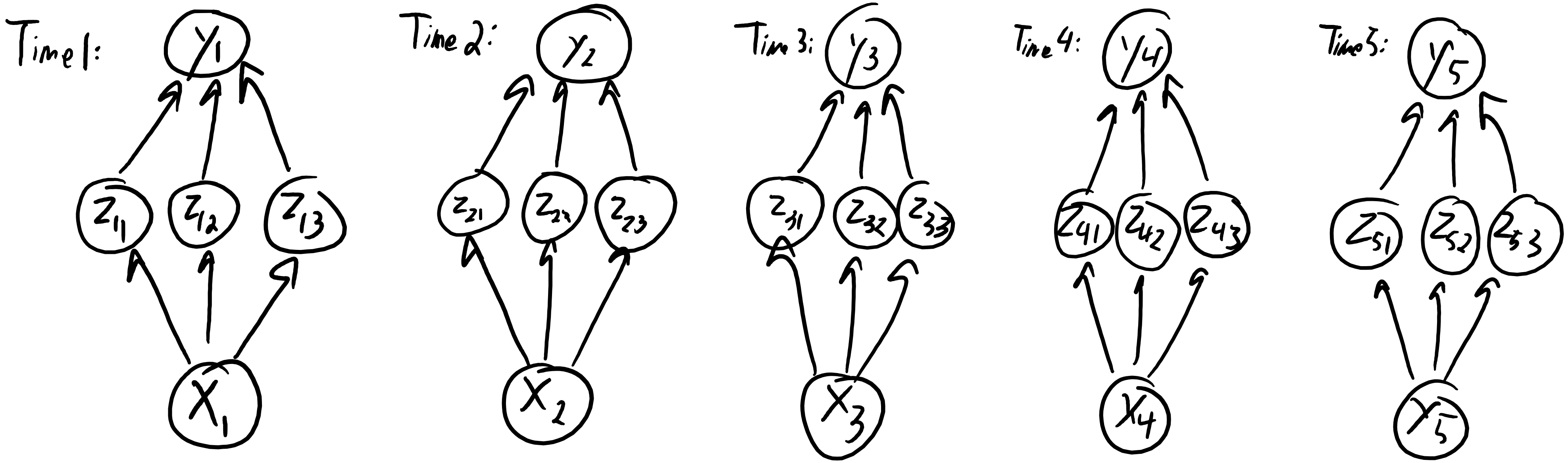
- Consider predicting part of speech for each word in a sentence:



- Input is a sequence of words
  - Could be represented as “1 of k” or using continuous vectors like word2vec
- Output is a categorical label for each word
  - In English there are ~40 reasonable categories
    - And there are some dependencies in labels (like “only 1 verb in the clause”)
- General problem: sequence labeling
  - Biological sequences, various language tasks, sound processing

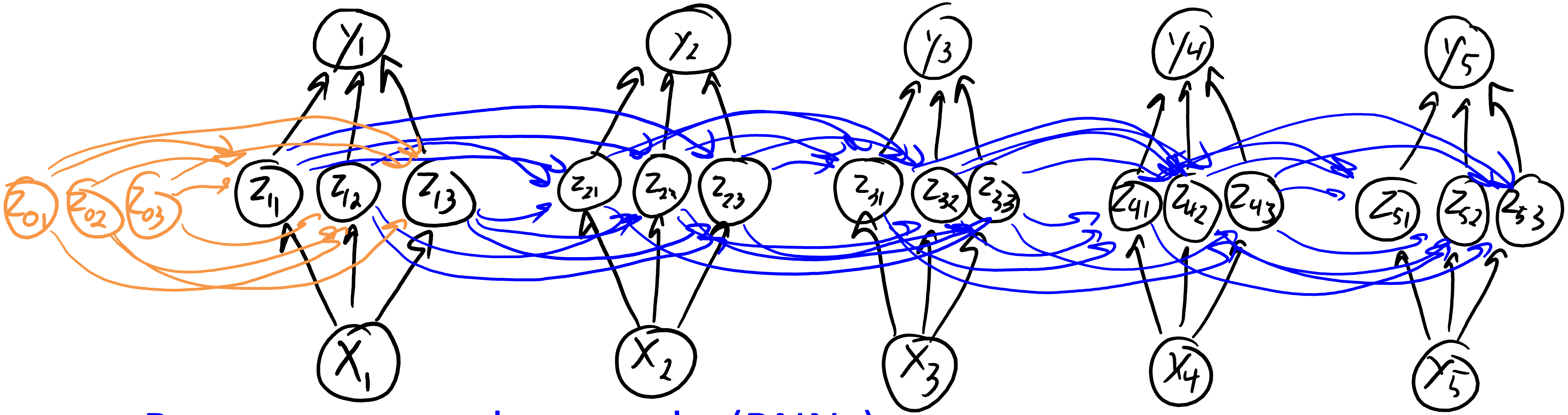
https://medium.com/analytics-vidhya/pos-tagging-using-conditional-random-fields-92077e5eaa31

Individual-Word Neural Network Classifier



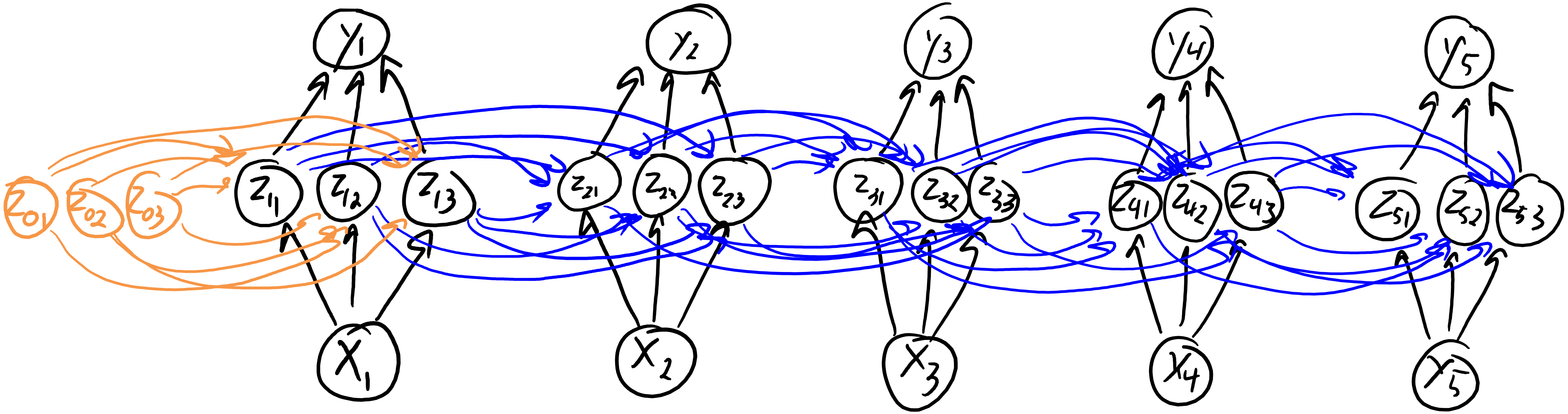
- We could train a neural network to predict label of a given word
  - Above, show have 1 input feature for each word; usually have *d* > 1
  - Also not showing the non-linear transform or bias variables.
- But this type of model would not capture dependencies
  - Information from earlier in sentence does influence prediction
    - “Don’t desert me in the desert!”

Recurrent Neural Network for Sequence Labeling



- Recurrent neural networks (RNNs):
  - Add connections between adjacent different times to model dependencies
  - Add an initial hidden state
  - Use the same parameters across time
- Repeating parameters in different places is called parameter tying
  - Convolutions use parameter tying across space
  - By tying parameters across time, RNNs can label sequences of different lengths

Recurrent Neural Network for Sequence Labeling



$$\hat{y}_t = \underbrace{V}_{}h(z_t) \qquad z_t = Wx_t + \underbrace{U}_{}\underbrace{h(z_{t-1})}_{}$$

Parameters: W, V, U
(and possibly z_0)

We have a matrix 'V' because we are doing multi-class

Use $\hat{y}_t$ vector in softmax at each time

weights on temporal connections

hidden units at previous time

(Notice that we use the <u>same</u> "tied" matrices $\{W, V, U\}$ for all times 't'."

Recurrent Neural Network Inference

$$\underbrace{\hat{y}_t}_{k\times 1} = \underbrace{V}_{k\times m} h(\underbrace{z_t}_{m\times 1}) \qquad \underbrace{z_t}_{m\times 1} = \underbrace{W}_{m\times d}\underbrace{x_t}_{d\times 1} + \underbrace{U}_{m\times m} h(\underbrace{z_{t-1}}_{m\times 1})$$

- Assume we have:
  - *k* different classes that each $\hat{y}_t$ can take
  - *m* hidden units at each time
  - *T* times (length of sequence)
- Cost to compute all $\hat{y}_t$ if each time has *m* units and we have *T* times:
  - We need to do an O(*md*) operations *T* times to compute Wx_t for all *t*
  - We need to do an O(*km*) operation *T* times to compute $\hat{y}_t$ for all *t*
  - We need to do a O(m^2) operation *T* times to compute each z_t
  - Total cost: O($Tmd + Tkm + Tm^2$)
- For the likelihood, we could use an independent softmax for each time
  - $p(y_{1:T} \mid x_{1:T}, W, V, U) = p(y_1 \mid x_1, W, V, U)\, p(y_2 \mid x_{1:2}, W, V, U) \cdots p(y_T \mid x_{1:T}, W, V, U)$
    - Each p(yt | x1:T, W, V, U) is given by softmax over $\hat{y}_t$ values
    - Conditioned on features and parameters, this assumes a "product of categoricals" model

RNN Learning

- The objective function we use to train RNNs is the NLL:

$$f(W,V,U) = -\sum_{i=1}^{n}\sum_{t=1}^{T^i} \log p(y_t^i \mid x_{1:T^i}^i, W, V, U)$$

  - Sequence *i* has length $T^{(i)}$ (might vary)
- Computing gradient is called "backpropagation through time" (BTT)
  - Equations are the same as usual backpropagation/chain-rule
    - If you do it by hand, make sure to add all terms for tied parameters
    - Automatic differentiation will handle this automatically
- Usually trained with SGD
  - Sample an example *i* on each iteration, do BTT, update all parameters
  - This has the usual challenges

RNN Learning – Extra Challenges

- Computing gradient requires a lot of memory for long sequences
  - There are a lot of intermediate calculations
- Parameter tying often leads to vanishing/exploding gradient problems
  - For a linear RNN, if all the input features are zero:
    - $z_T = U\,U\,U \cdots U\,z_0 = \mathrm{U}^T z_0$
    - Usually z_T either diverges exponentially or converges to zero exponentially
      - If largest singular value of U is > 1, $||z_T||$ increases exponentially with T
      - If largest singular value of U is < 1, $||z_T||$ converges to zero exponentially with T
- Usual SGD methods tend not to work well
  - Often need to use optimizers like Adam or use gradient clipping:

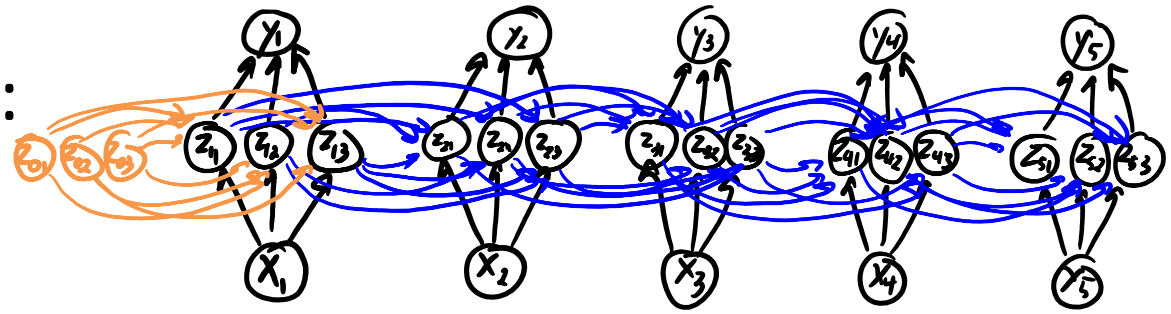
$$\text{if } \|g\| > u$$
$$g \leftarrow \frac{gu}{\|g\|}$$

    - If norm of gradient is larger than some threshold, “shrink” norm to threshold:
  - Special initialization / keeping ‘U’ orthogonal might help
    - Makes all singular values 1 – some positive, some negative results on this

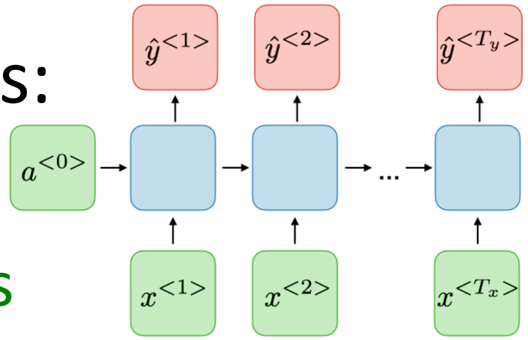
https://towardsdatascience.com/neural-network-optimization-7ca72d4db3e0

Deep RNNs

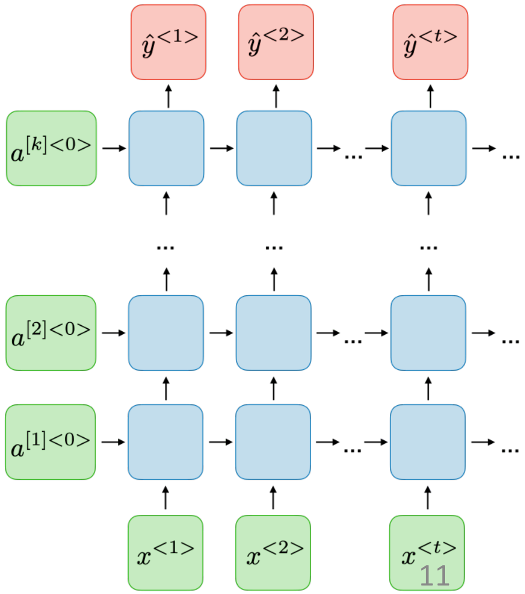
- Instead of drawing this:



- We often use diagrams like this:
  - Up to some notation changes
  - We connect everything in blocks connected by arrows

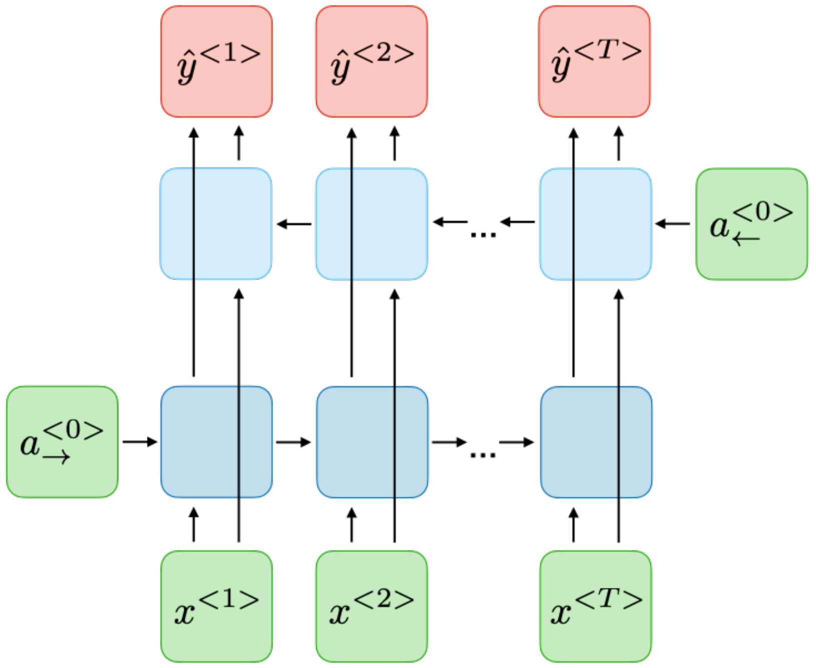


- Deep RNNs add multiple hidden layers at each time:



Bi-Directional RNNs

- Sometimes later information later changes meaning:
  - "I've had a perfectly wonderful evening, but this wasn't it.” (“paraprosdokian”)
  - “The old man the boat.” (“garden path sentence”)
- Bi-directional RNNs have hidden layers running in both directions:
  - Different parameters for the forward and backward directions



https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Next Topic: Sequence to Sequence RNNs (seq2seq)

Motivating Problem: Machine Translation

- Consider the problem of machine translation:
  - Input is text from one language
  - Output is text from another language with the same meaning

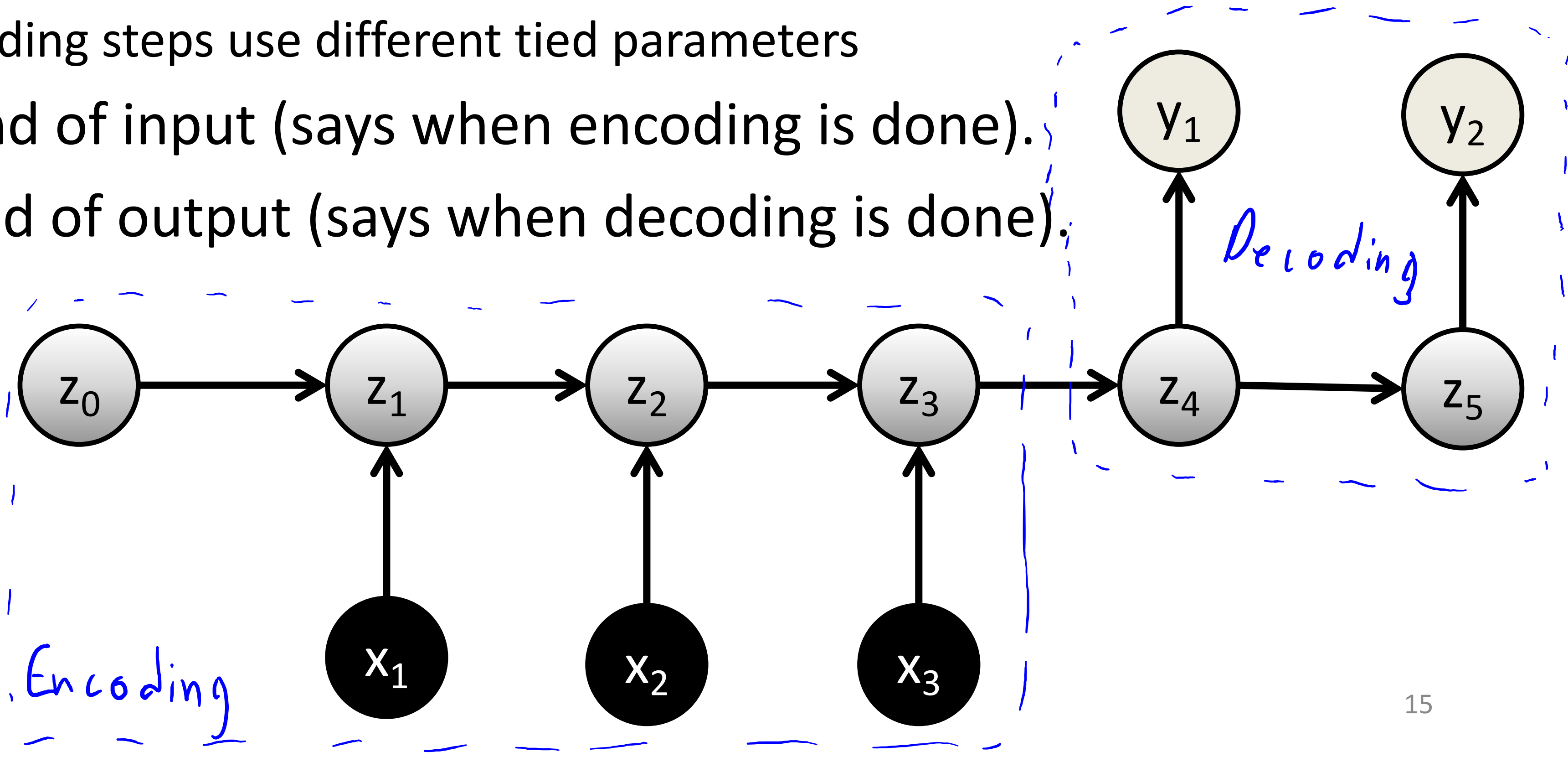
This course is intended as a second or third university-level course on machine learning, a field that focuses on using automated data analysis for tasks like pattern recognition and prediction.

Ce cours est conçu comme un cours de deuxième ou troisième niveau universitaire sur l'apprentissage automatique, un domaine qui se concentre sur l'utilisation de l'analyse de données automatisée pour des tâches telles que la reconnaissance de formes et la prédiction.

- A key difference with pixel labeling:
  - Input and output sequences may have different lengths and "orders"
    - We do not just "find the French word corresponding to the English word"
  - We probably don't know the output length

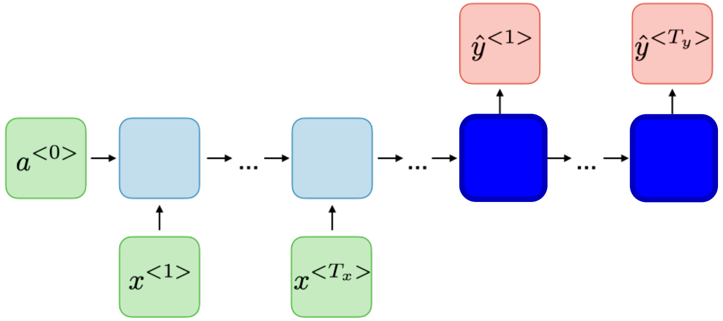
Sequence-to-Sequence RNNs

- Sequence-to-sequence RNNs encode and decode sequences:
  - Each encoding step has one word as input, and no output
  - Each decoding step outputs one word, with no input
    - Encoding and decoding steps use different tied parameters
  - Special "BOS" at end of input (says when encoding is done).
  - Speical "EOS" at end of output (says when decoding is done).



Discussion: Sequence-to-Sequence Models

- Representing input and outputs:
  - Could use lexicographic or word2vec representations
  - Could just have a single character at each time
    - Could make more sense for some languages
    - May be able to better handle slang or typos
  - These days, usually an in-between of tokens (more Monday)



- Loss function assuming independent labels given hidden states:

$$f(\Omega) = -\sum_{i=1}^{n}\sum_{j=1}^{|y^i|} \log p(y_j^i \mid x_{1:t}^i, \Omega)$$

Ω → $\{W_e, V_e, U_e, W_d, V_d, U_e\}$ "all parameters"

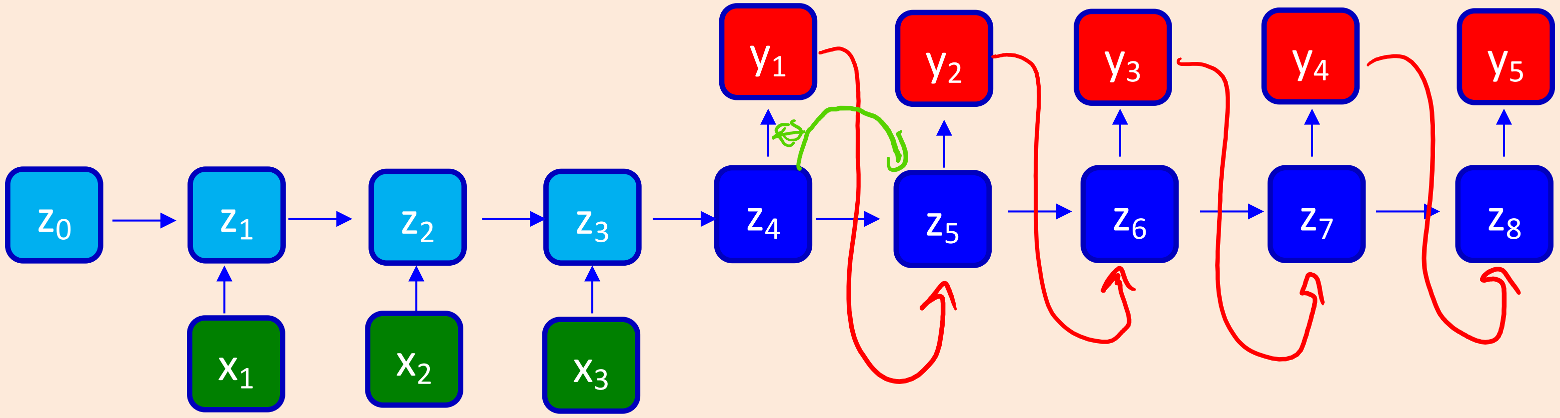
$p(y_j^i \mid x_{1:t}^i, \Omega)$ → softmax value for word at position 'j' in training data.

  - This is just trying to get the label right at each "time"
    - *Not* "trying to get the full sequence right"

bonus!

Digression/Preview: Dependent Predictions

- Standard RNNs assume conditional independence of $\hat{y}_t$ values
  - We assume they are independent given the z_t values (make inference easy)
  - This makes inference easy, but $\hat{y}_t$ "forgets" what was used for $\hat{y}_{t-1}$
- In many applications, you want to model dependencies in the $\hat{y}_t$
  - A common way to do this is to add edges like this:



  - Fine in training (where we know the y_t values)
  - But it makes inference and decoding challenging since the y_t are dependent
    - We'll discuss variants like this after discussing Markov chains

Next Topic: LSTMs

Exponential "Forgetting" in RNNs

- Sequence-to-sequence RNNs:
  - Elegant way to handle inputs/outputs of different/unknown sizes
  - Final "encoding" is the hidden states once the last input has been entered
    - We hope this captures the semantics of the sentence
  - The "decoding" steps try use the hidden states to output translation, and also updates the hidden states

- Using tied parameters allows using the model for any sequence lengths
- But with tied parameters, we "forget" information exponentially fast
  - If you want to "remember" something about x_1, it has to go through $U * U * U * \cdots$.
    - "Initial conditions" for before the multiplication are forgotten at an exponential speed

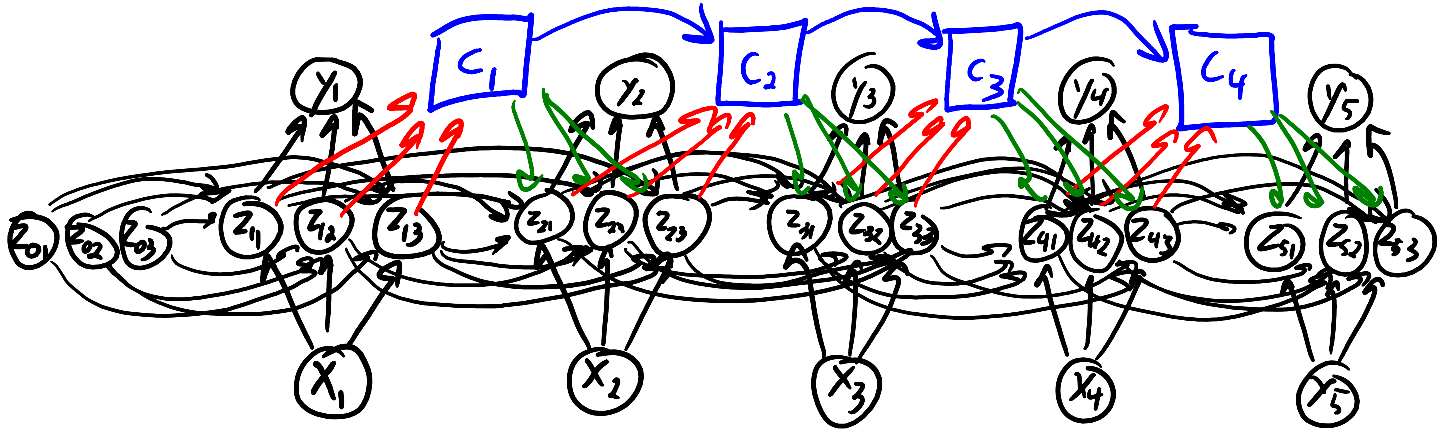
https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Adding a "Memory"

- One possible way to help RNNs remember is with skip connections:

$$\hat{y}_t = Vh(z_t) \qquad z_t = Wx_t + U_1 h(z_{t-1}) + U_2 h(z_{t-2})$$

  - We will come back to several variations on this idea later
- Another idea is to add a memory where you can "save" and "load":



- Relevant information can be saved to the memory, then accessed at a much later time

Long Short Term Memory (LSTM)

- Long short term memory (LSTM) models are variant of RNNs:
  - Modification to try to remember short-term and long-term dependencies
- In addition to usual hidden values *z*, LSTMs have memory cells *c*:
  - Purpose of memory cells is to remember things for a long time
- LSTMs are maybe analogous to convolutions for RNNs:
  - “The first trick that made them work in many applications”
- LSTMs have been used in a huge variety of settings:
  - Cursive handwriting recognition https://www.youtube.com/watch?v=mLxsbWAYIpw
  - Speech recognition and text-to-speech (Google, Apple, Amazon c. 2015-17)
  - Machine translation (Google, Facebook c. 2016)
  - iPhone autocorrect (c. 2016)
  - AIs for Dota 2 (OpenAI 2018), Starcraft 2 (DeepMind 2019), …

Long Short Term Memory – Ugly Equations

- Computing activations at time *t* in an RNN:

$$a_t = h(z_t)$$

With $z_t = Wx_t + U\underbrace{a_{t-1}}_{h(z_{t-1})}$

- Computing activations at time *t* in an LSTM:

$$a_t = \underbrace{o_t}_{\text{"output gate"}} \circ h_o(c_t)$$

$$c_t = f_t \circ \underbrace{c_{t-1}}_{\text{"memory"}} + i_t \circ g_t$$

"forget gate"

"input gate"

"value"

With:

$$f_t = h(W_f x_t + U_f a_{t-1})$$

$$i_t = h(W_i x_t + U_i a_{t-1})$$

$$g_t = h_o(W_g x_t + U_g a_{t-1})$$

$$o_t = h(W_o x_t + U_o a_{t-1})$$

Two activation functions:

"gate" function 'h':

→ usually sigmoid

"value" function 'h_o':

→ usually tanh

"$\circ$": element-wise multiplication of vectors.

Long Short Term Memory – Equation Intuition

- Conceptually, we think of LSTMs as having a "memory" c_t:
- We update and access this memory with a set of "gates":
  - Gates take weighted combination of input and previous activation, and output a value between 0 and 1 (differentiable approximation to binary values)
    - In a computer these gates would be exactly 0 or 1, but we use sigmoids so "gate" can have values like 0.7
- "Forget gate" f_t:
  - If element 'j' of f_t is 0, then we clear element c_{tj} from the memory (set it to 0)
    - If it is 1, then we keep the old value
  - "Given the input and previous activation, are the elements in memory still relevant?"
- "Input gate" i_t:
  - If element 'j' of i_t is 0, then we do not add any new information to c_{tj} (no input)
    - If it is 1, then we "value" to the memory (where "value" is also a function of input and previous a_t)
  - "Given the input and previous activation, should I write something new to memory?"
- "Output gate" o_t:
  - If element 'j' of o_t is 0, then we do not read value c_{tj} from the memory (no output)
    - If it is 1, then we load from the memory
  - "Given the input and previous activation, should I read what is in memory?"

c_t
0.3
-3.5
-0.2
0
0.4
0.3
-0.2

LSTM Equations (same slide as 2 slides ago)

- Computing activations at time 't' in an RNN:

$$a_t = h(z_t)$$

With $z_t = Wx_t + U\underbrace{a_{t-1}}_{h(z_{t-1})}$

- Computing activations at time 't' in an LSTM:

$$a_t = o_t \circ h_o(c_t)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ g_t$$

"memory"

"output gate"

"forget gate"

"input gate"

"value"

With:

$$f_t = h(W_f x_t + U_f a_{t-1})$$

$$i_t = h(W_i x_t + U_i a_{t-1})$$

$$g_t = h_o(W_g x_t + U_g a_{t-1})$$

$$o_t = h(W_o x_t + U_o a_{t-1})$$

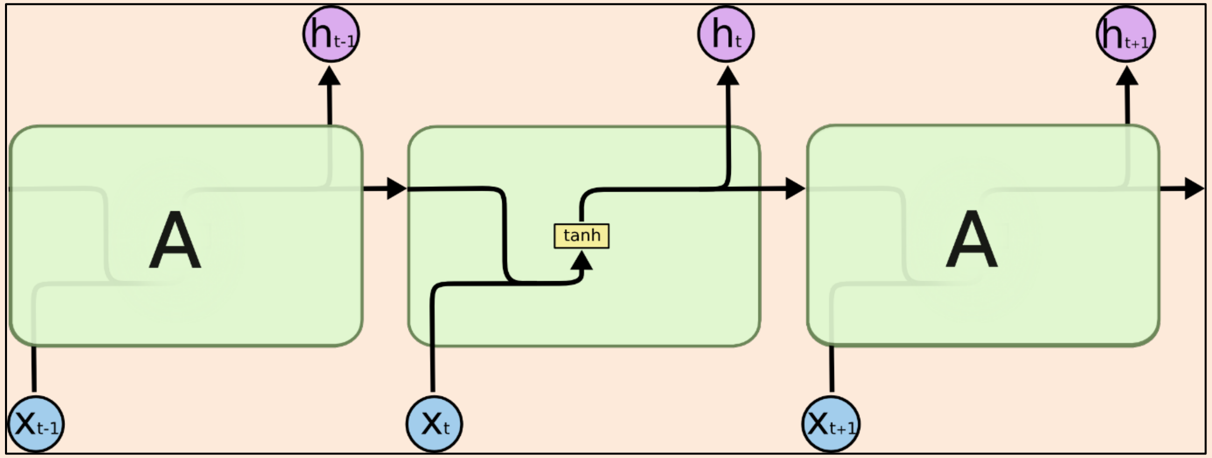
Two activation functions:
"gate" function 'h':
→ usually sigmoid
"value" function 'h_o':
→ usually tanh

"$\circ$": element-wise multiplication of vectors.

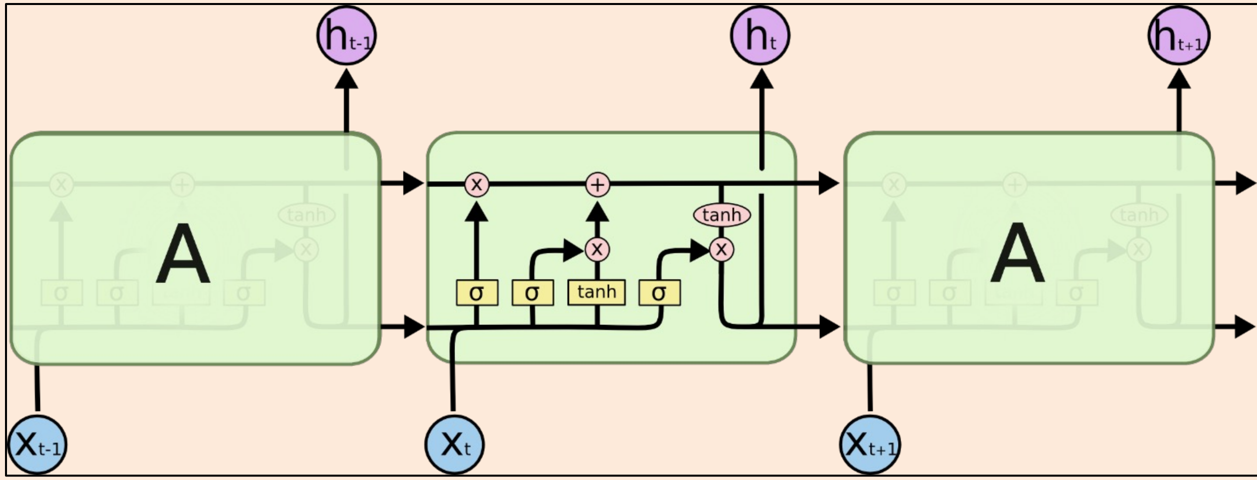
LSTM Activation Calculation as a Picture

- We often see pictures like this to represent the different operations:

RNN



LSTM



- I find these pictures confusing unless you have gone through equations.
  - For example, where are the weights?

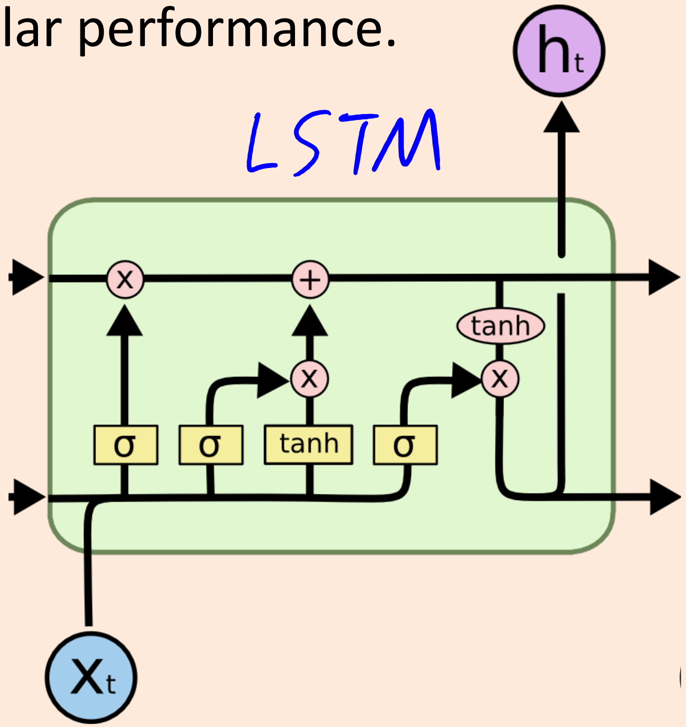
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

bonus!

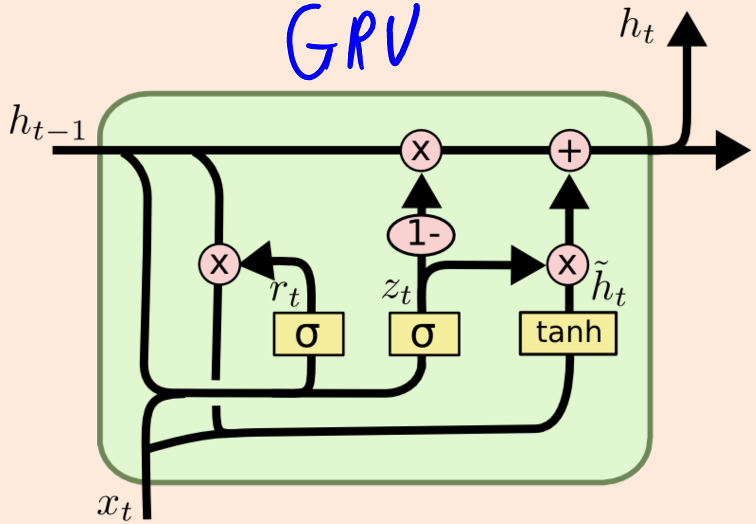
Gated Recurrent Units (GRUs)

- Many variations on LSTMs exist.
  - A popular one is gated recurrent units (GRUs).
    - A bit simpler (merges “forget”+”input”, and “activation”+”memory”).
    - Similar performance.

LSTM



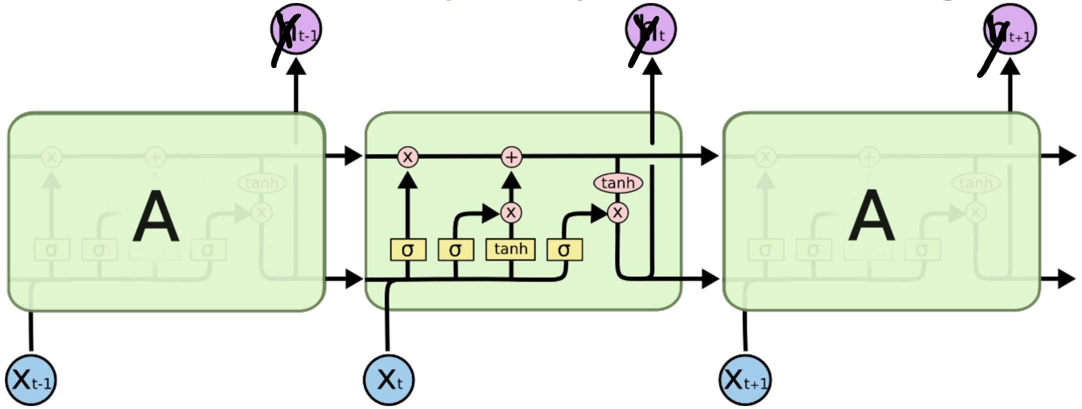
GRU



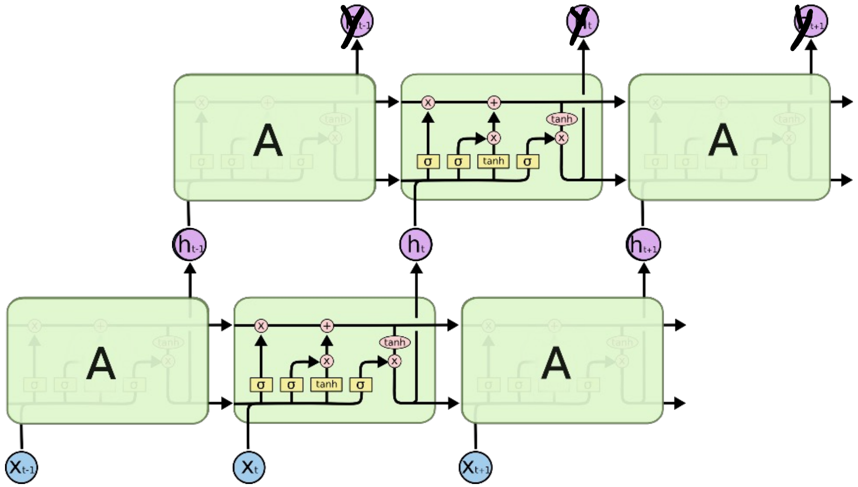
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Deep LSTM Models

- LSTM model with one hidden layer (pixel labeling version):



- LSTM model with two hidden layers:
  - As with regular RNNs, activations feed into next layer and next time
  - Each layer has own memory
    - Parameter tying only within layers
  - Might have residual connections

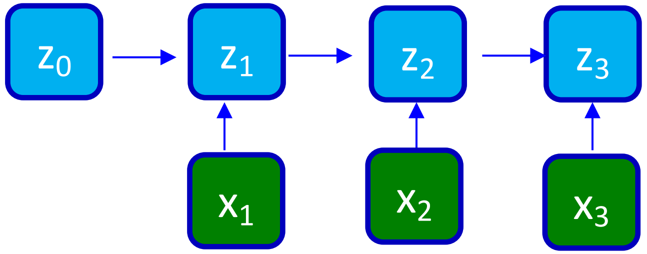


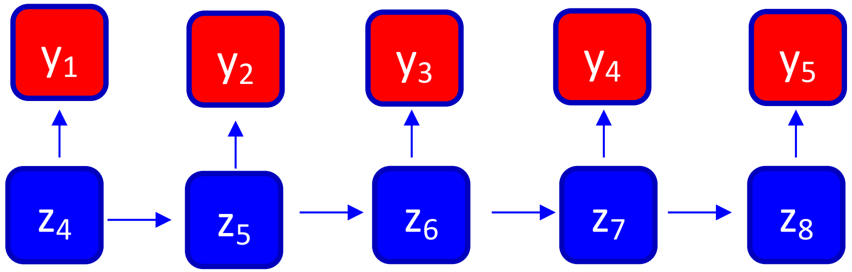
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Next Topic: Multi-Modal Models

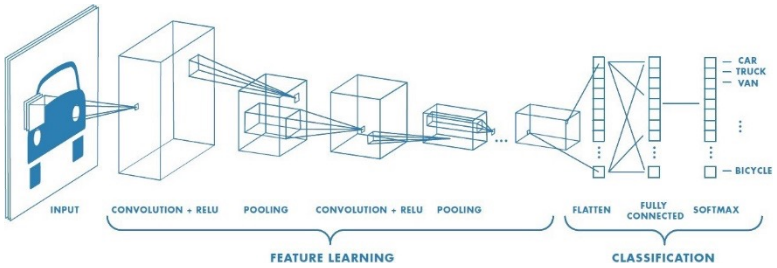
Encoding-Decoding For Different Data Types

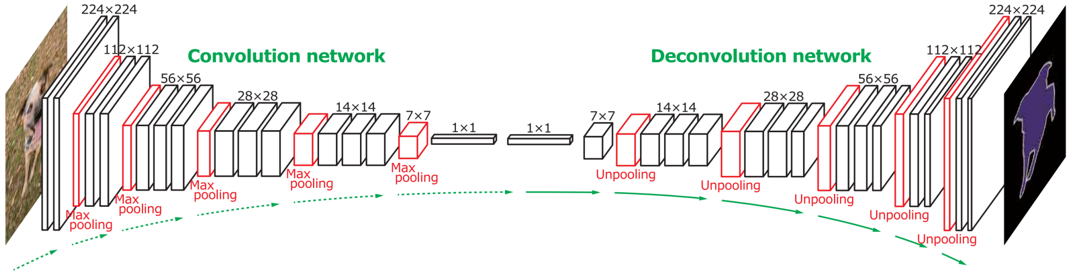
- Consider the encoding and decoding phase as separate "models":





  - Encoder takes a sequence and returns a set of numbers
  - Decoding takes a set of numbers and outputs a sequence
- We have also seen encoding and decoding of images:





  - Encoder takes an image and returns a set of numbers
  - Decoder takes a set of numbers and outputs an image (or a class or set of labels)

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Noh_Learning_Deconvolution_Network_ICCV_2015_paper.pdf

LSTMs for Image Captioning

- Use a CNN to do the encoding and an RNN to do the decoding

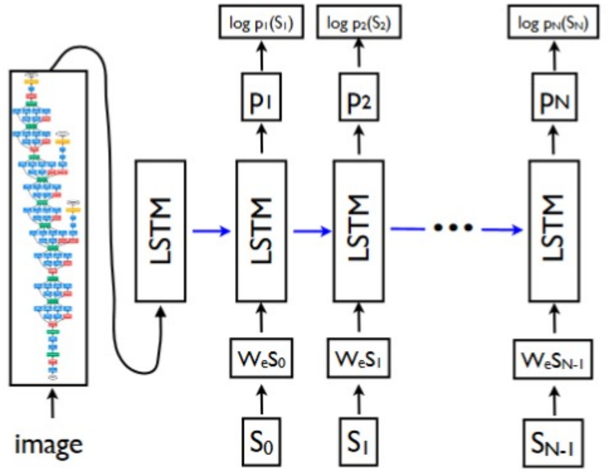


Figure 3. LSTM model combined with a CNN image embedder (as defined in [12]) and word embeddings. The unrolled connections between the LSTM memories are in blue and they correspond to the recurrent connections in Figure 2. All LSTMs share the same parameters.

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

- To train this model, we need images and corresponding captions
  - So the image encoder and sequence decoder are trained together

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

Image Captioning Application: PDF to LaTeX

- Use CNN to encode an image, use RNN to decode LaTeX

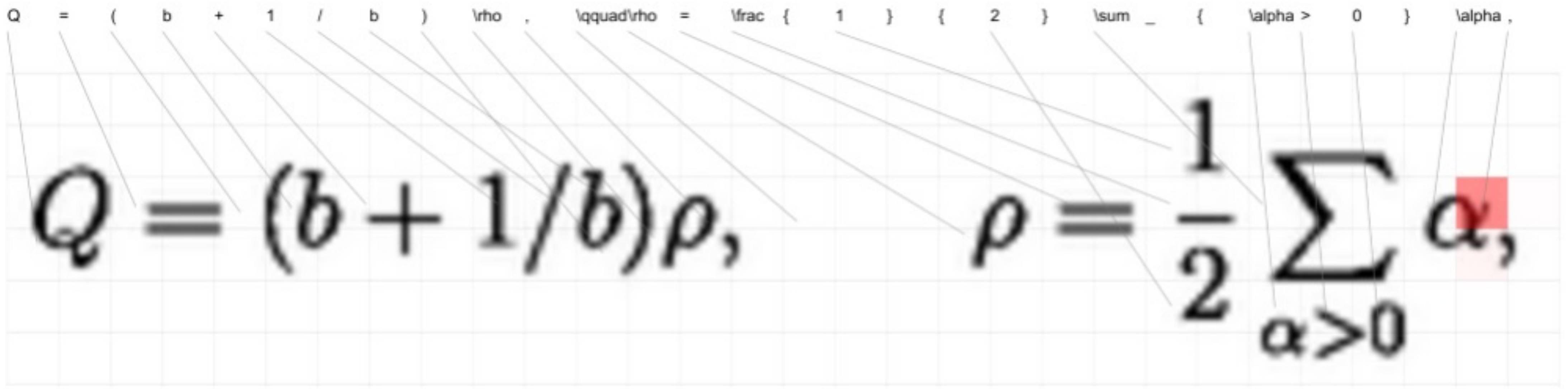
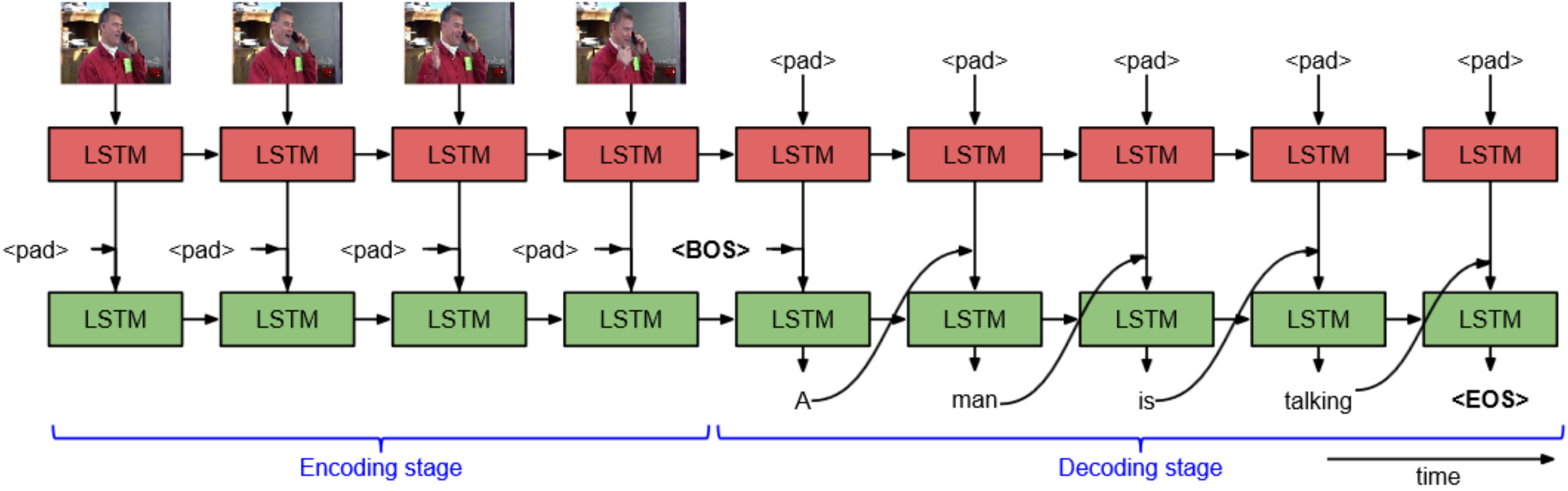


Figure 1: Example of the model generating mathematical markup. The model generates one LaTeX symbol y at a time based on the input image $\mathbf{x}$. The gray lines highlight $H' \times V'$ grid features after the CNN $\mathbf{V}$ and RNN Encoder $\tilde{\mathbf{V}}$. The dotted lines indicate the center 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.

- Unlike generic image captioning, there is a "correct" label
  – Although not necessarily unique

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

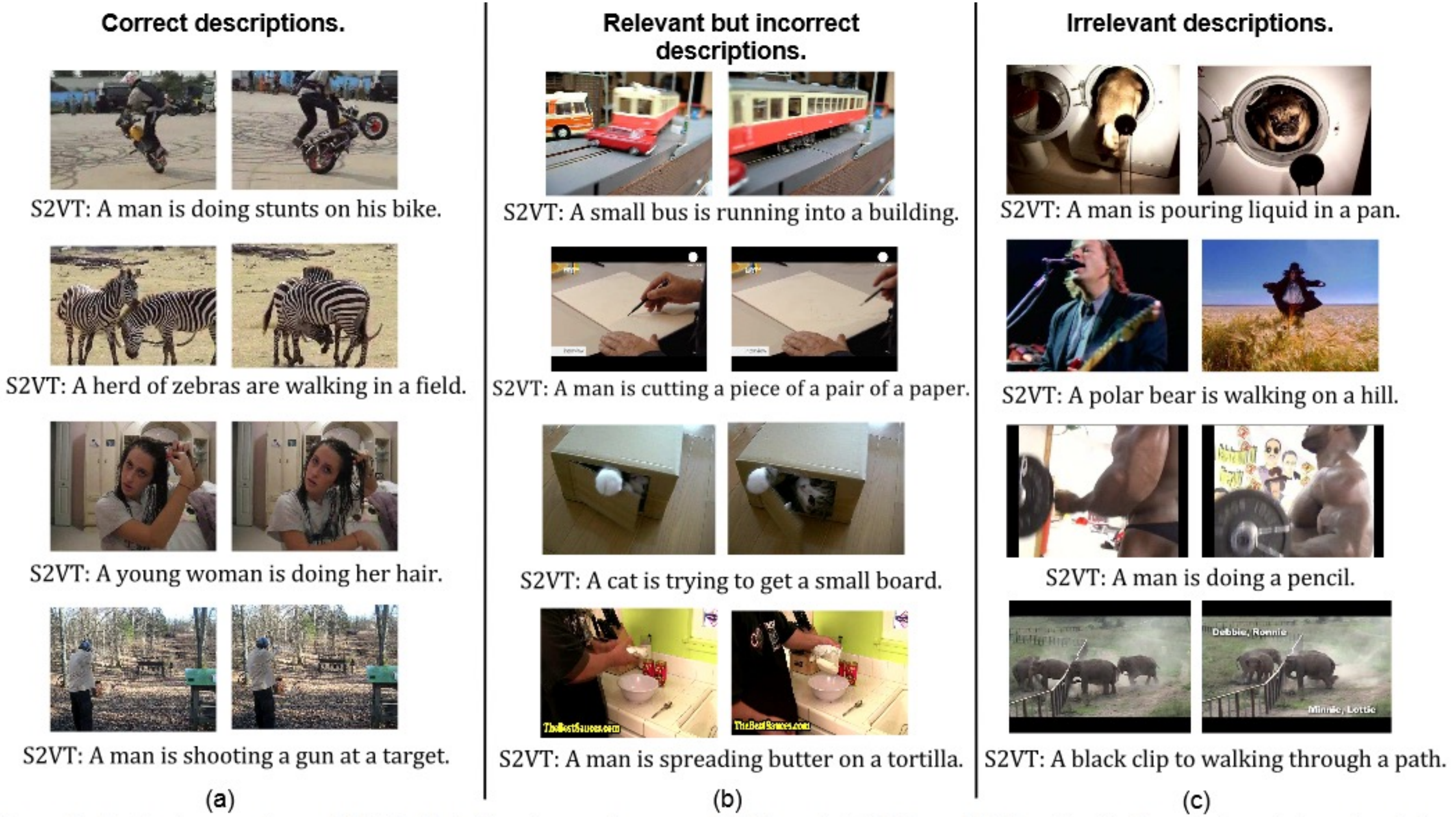


Figure 3. Qualitative results on MSVD YouTube dataset from our S2VT model (RGB on VGG net). (a) Correct descriptions involving different objects and actions for several videos. (b) Relevant but incorrect descriptions. (c) Descriptions that are irrelevant to the event in the video.

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

Video Captioning Application: Lip Reading

- Unlike generic video captioning, there is a “correct” label

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

bonus!

RNNs/CNNs 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

bonus!

State-space models

- Model with a “latent state” that evolves over time (like RNNs)
  - Continuous-time SSMs use differential equations (limit of small steps)
    - Usually linear evolution of underlying state
  - Can integrate as a layer of a deep network

Hungry Hungry Hippos: Towards Language Modeling with State Space Models

Daniel Y. Fu[*†], Tri Dao[*†], Khaled K. Saab[‡], Armin W. Thomas[††], Atri Rudra[‡‡], and Christopher Ré[†]

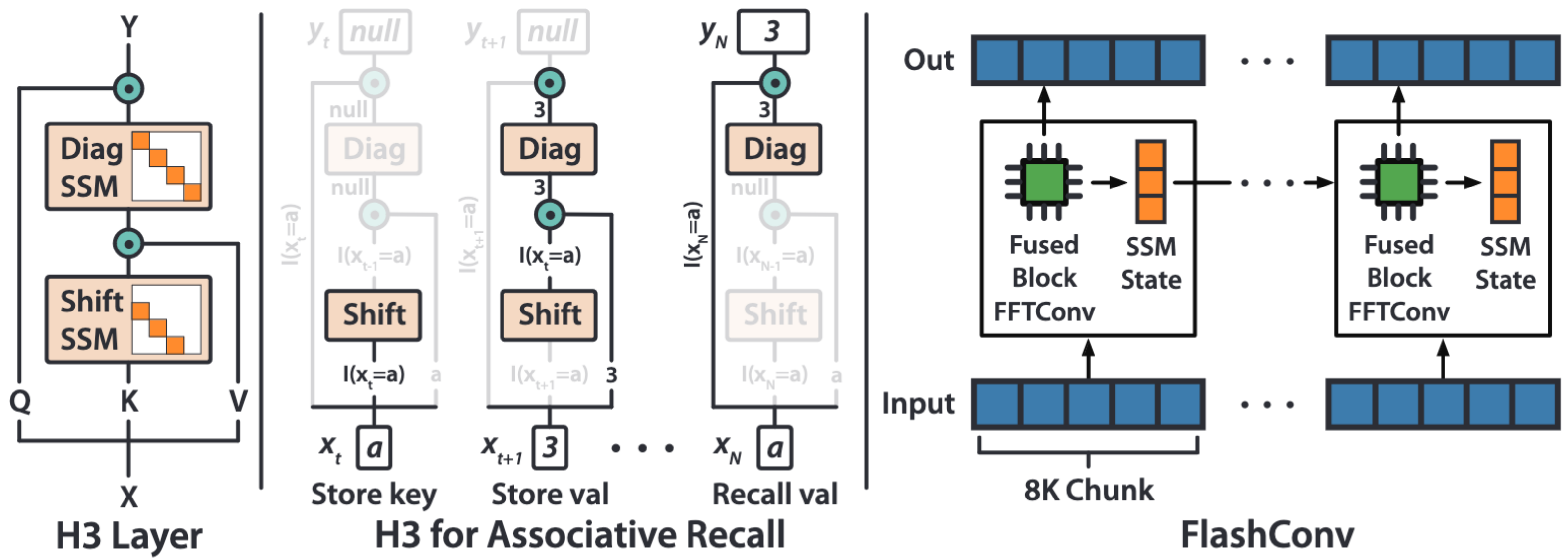


Figure 1: Left: H3 stacks two discrete SSMs with shift and diagonal matrices and uses multiplicative interactions between input projections and their outputs to model comparisons between points in a sequence. Middle: H3 can perform associative recall—which is easy for attention, but not existing SSMs. Right: FlashConv uses a new state-passing algorithm over fused block FFTConv to increase hardware efficiency of SSMs, allowing H3 to scale to billion-parameter models.

https://arxiv.org/abs/2212.14052

Summary

- Recurrent neural networks (RNNs):
  - Neural networks for sequence predictio.
  - Have connections between hidden units at adjacent times
  - Use parameter tying across time.
    - Allows sequences of different lengths
    - Leads to vanishing and exploding gradients
- Sequence-to-Sequence RNNs:
  - Encoding phase takes in one input at a time until we reach “BOS”
  - Decoding phase outputs one output at a time until we output “EOS”
  - Allows input and output sequences whose lengths differ
  - But: standard RNNs lead to exponential forgetting of information
- Long short term memory:
  - The trick that made RNNs start working
  - Gating functions which update “memory cells” for long-range interactions
- Multi-modal learning:
  - Encoder and decoder may work with different types of data
  - For example, CNN as encoder and RNN as decoder for image-to-text
- Next time: ChatGPT