ECBM E4040
Neural Networks and Deep Learning

Recurrent Neural Nets (RNNs)
Applications

Zoran Kostić
Columbia University
Electrical Engineering Department

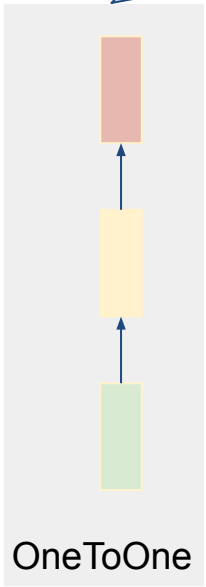
COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

References and Acknowledgments

- **NVIDIA GPU Teaching Kit (NVIDIA and New York University)**
- **Andrej Karpathy blog: The Unreasonable Effectiveness of Recurrent Neural Networks**
- **Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016.**

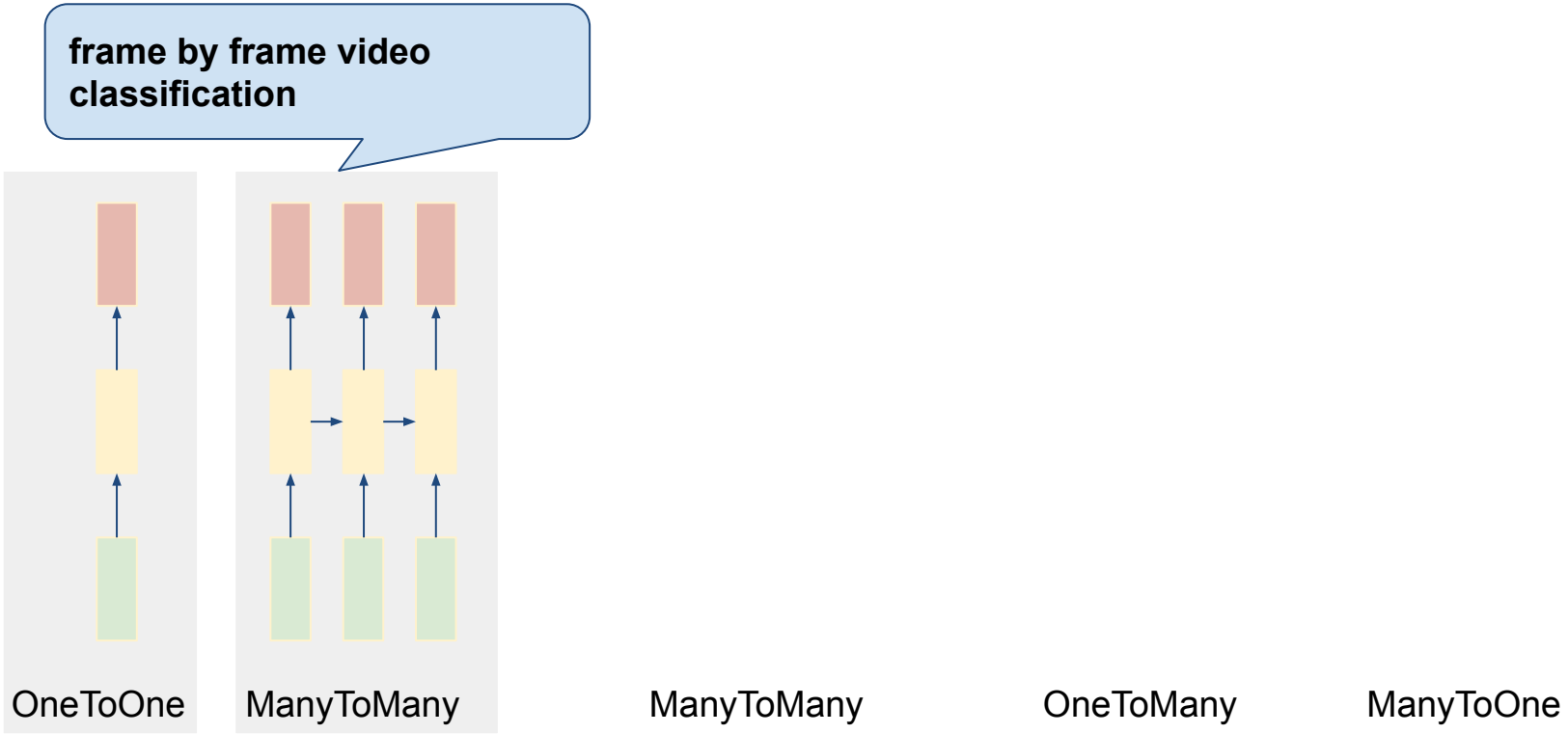
RNNs Architectures

simple neural network



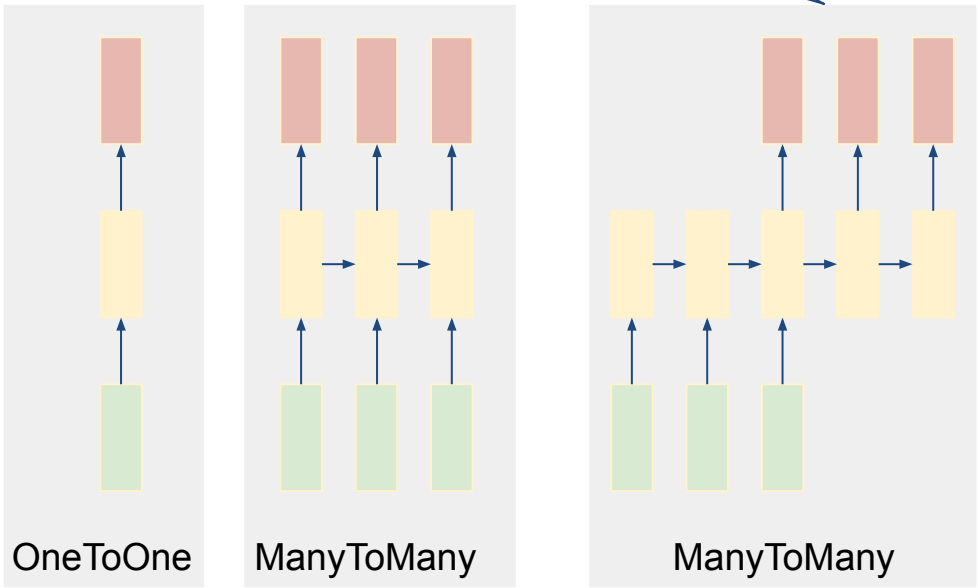
OneToOne ManyToMany ManyToMany OneToMany ManyToOne

RNNs Architectures

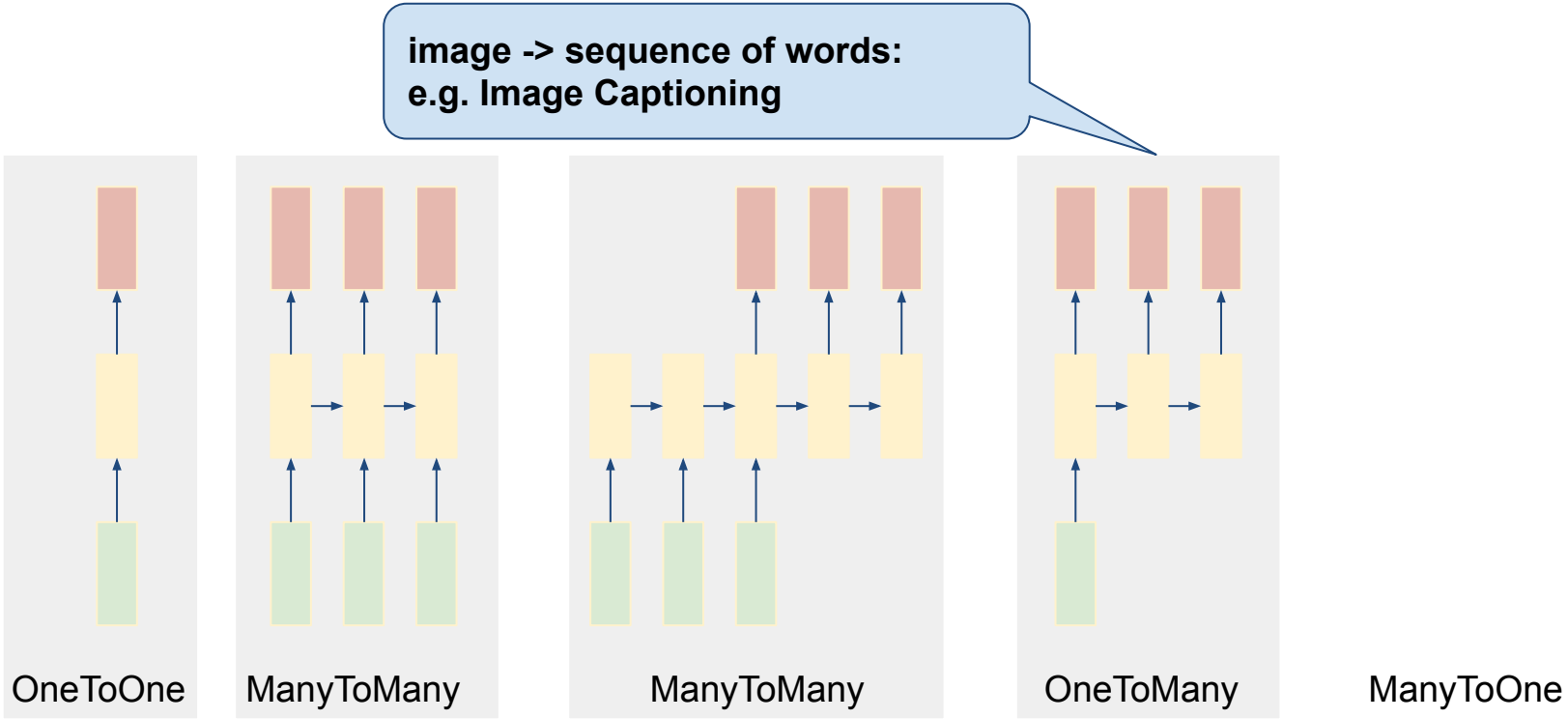


RNNs Architectures

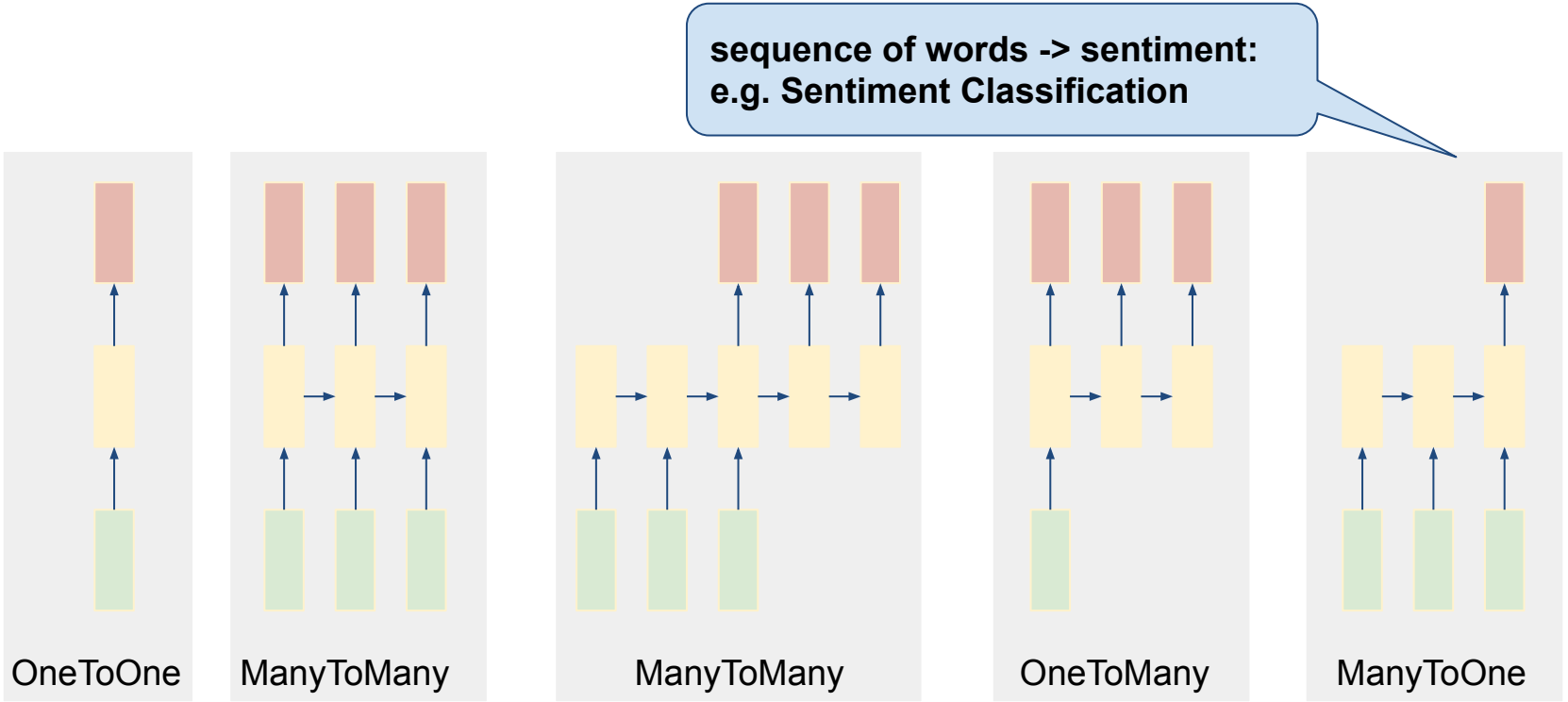
word sequence -> word sequence:
machine translation



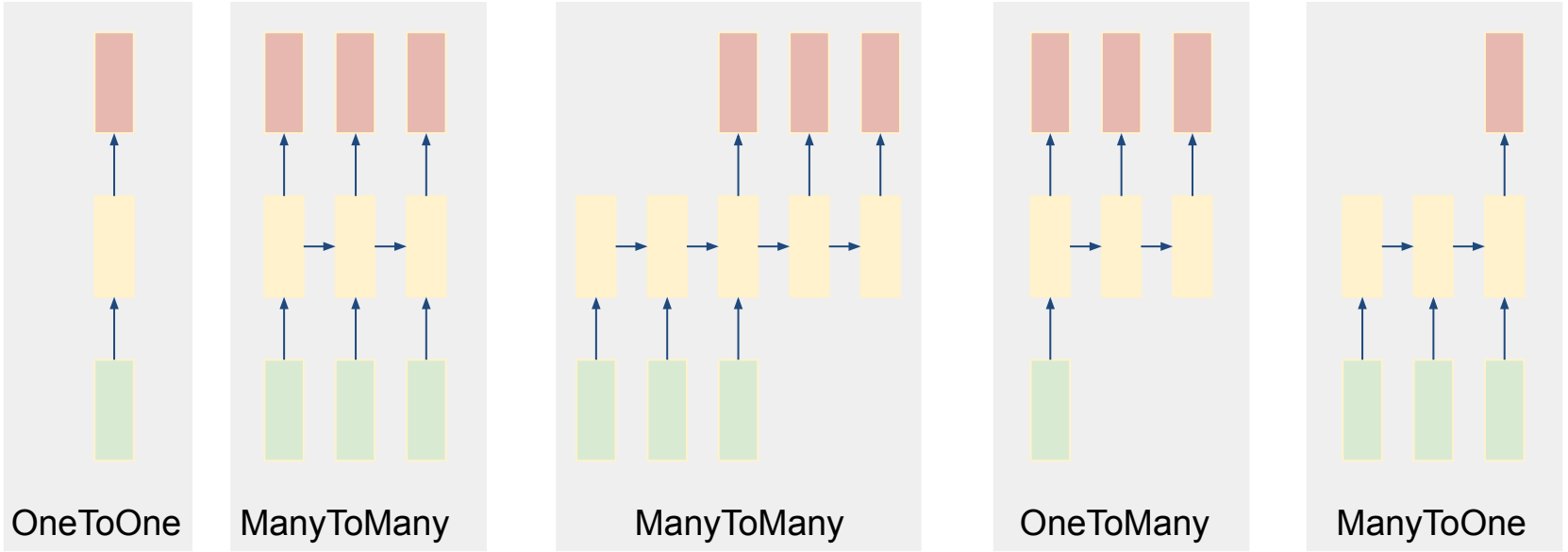
RNNs Architectures



RNNs Architectures



RNNs Architectures



RNN Computation

```
rnn = RNN()
  y = rnn.step(x) # x is an input vector, y is the RNN's output vector


class RNN:
    # ...
    def step(self, x):
      # update the hidden state
      self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
      # compute the output vector
      y = np.dot(self.W_hy, self.h)
      return y
```

RNN Computation

The hidden state self.h is initialized with the zero vector.

The np.tanh function implements a non-linearity that squashes the activations to the range [-1, 1].

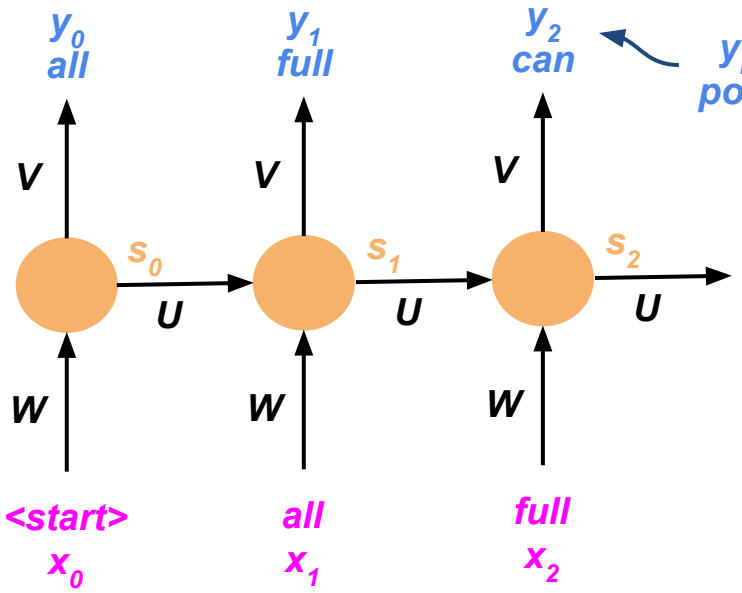
$$ht=\tanh(W_{hh}h_t-1+W_{xh}x_t)$$

A 2-layer recurrent network:

y_1 = rnn1.step(x)

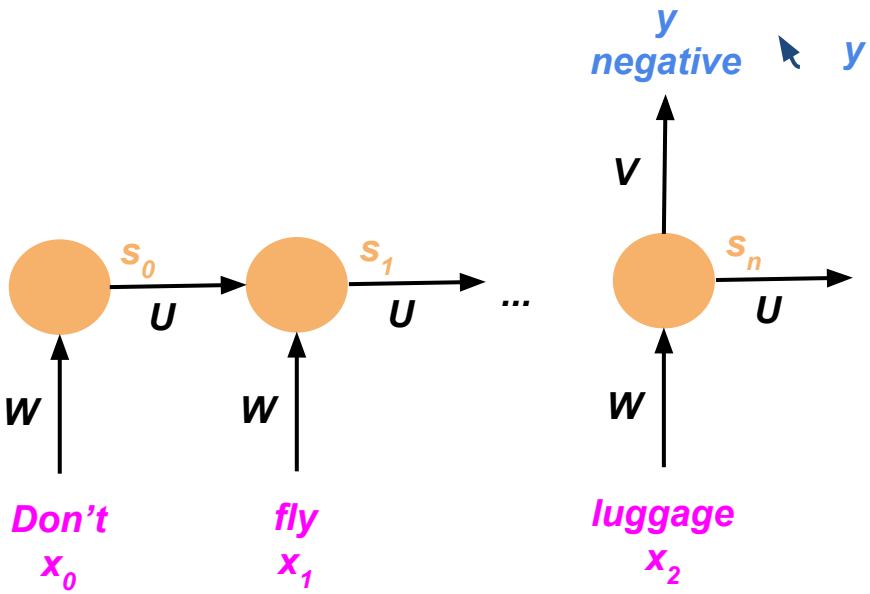
y = rnn2.step(y_1)

RNNs for Language Modeling: From Words



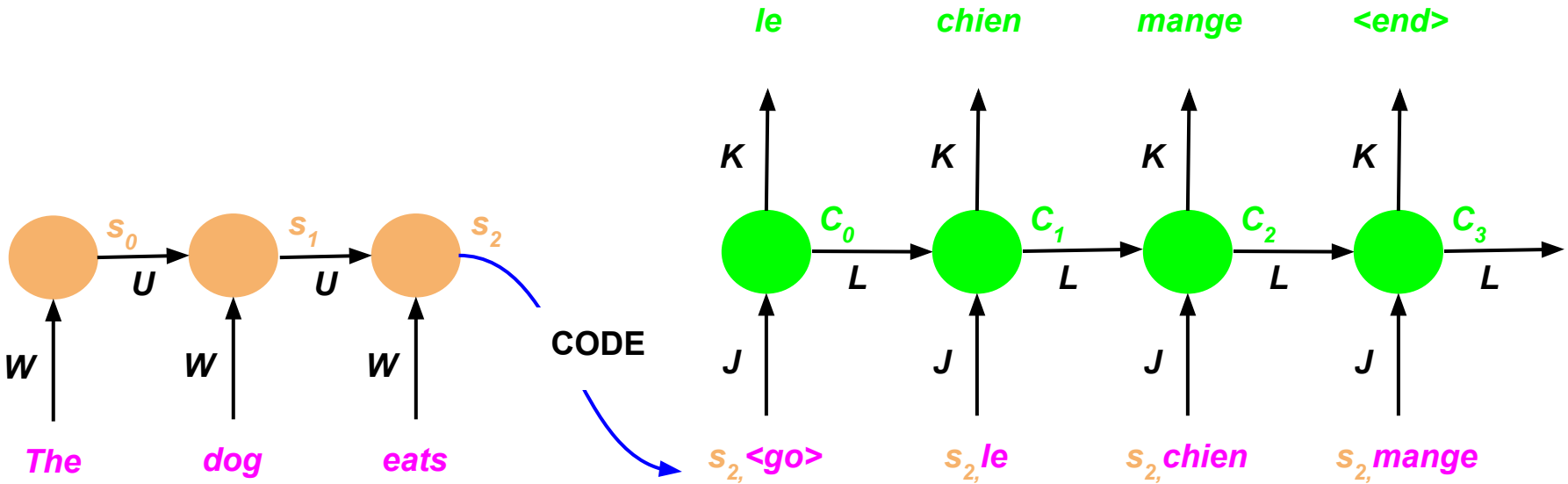
RNNs for Classification (i.e. sentiment)

RNNs for Classification (i.e. sentiment)



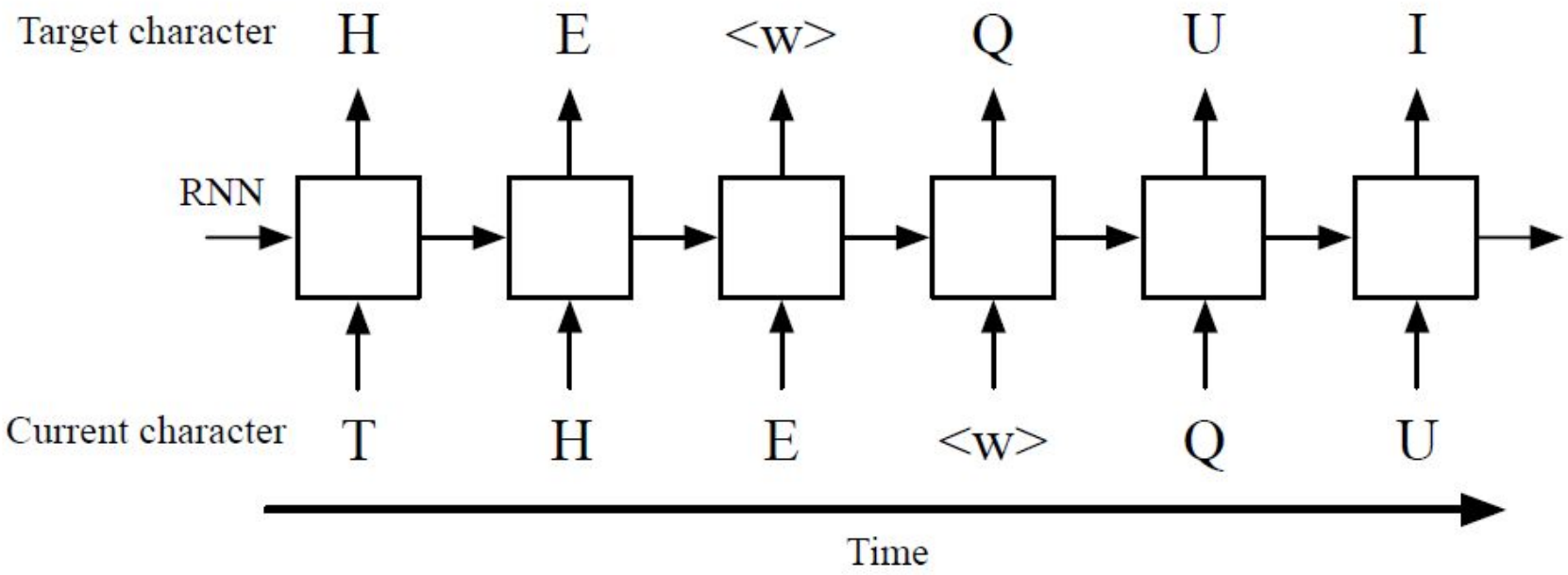
y is a probability distribution over possible classes (like positive, negative, neutral), aka softmax

RNNs for Machine Translation (Encoding/Decoding)



The last cell state represents the whole input

RNNs for Character-level Language Modeling



Character-level Language Modeling with Hierarchichal RNNs https://arxiv.org/pdf/1609.03777.pdf

RNN Example: Character-Level Language Models

Give the RNN a huge chunk of text and ask it to model the probability distribution of the next character in the sequence given a sequence of previous characters. This will then allow it to generate new text one character at a time.

Working example:

- **Suppose a vocabulary of four possible letters {h,e,l,o}, and goal to train an RNN on the training sequence "hello".**
- **This training sequence is in fact a source of 4 separate training examples: 1. The probability of "e" should be likely given the context of "h", 2. "l" should be likely in the context of "he", 3. "l" should also be likely given the context of "hel", and finally 4. "o" should be likely given the context of "hell"**
- **http://karpathy.github.io/2015/05/21/rnn-effectiveness/**

RNN Example: Character-Level Language Models Process

Encode each character into a vector using 1-of-k encoding (i.e. all zero except for a single one at the index of the character in the vocabulary)

Feed vectors into the RNN one at a time with the step function.

Observe a sequence of 4-dimensional output vectors (one dimension per character),

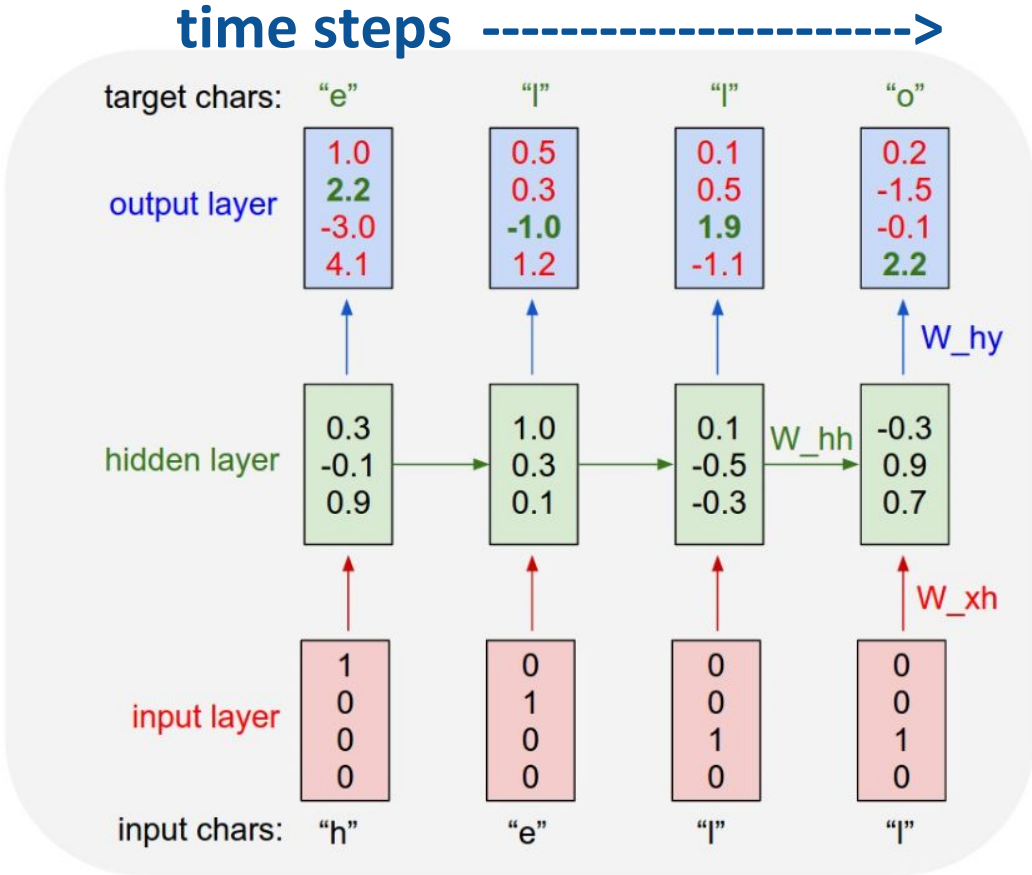
- **which can be interpreted as the confidence the RNN currently assigns to each character coming next in the sequence.**

RNN Example: Character-Level Language Models Process

An example RNN with 4-dimensional input and output layers, and a hidden layer of 3 units (neurons).

This diagram shows the activations in the forward pass when the RNN is fed the characters "hell" as input.

The output layer contains confidences the RNN assigns for the next character (vocabulary is "h,e,l,o"); We want the green numbers to be high and red numbers to be low.



RNN Example: Character-Level Language Models Process

time steps ---------->

Step 1: would like to increase the confidence (green) of e (from 2.2) and decrease the confidence of all other letters (red).

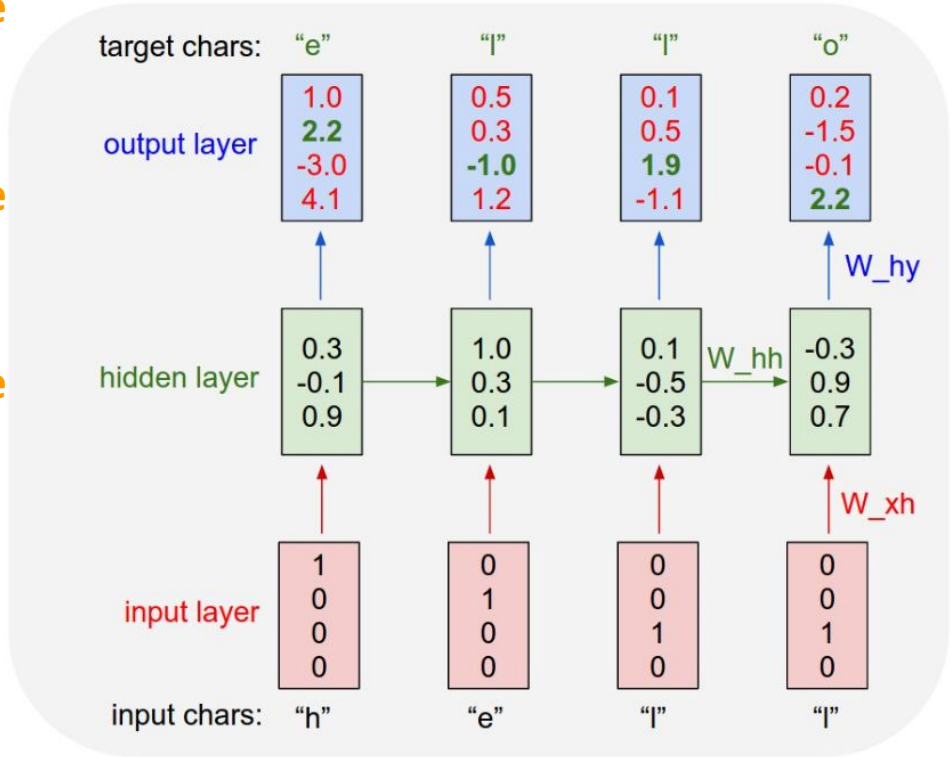
Step 2: would like to increase the confidence (green) of l (from -1.0) and decrease the confidence of all other letters (red).

Step 3: would like to increase the confidence (green) of l (from 1.9) and decrease the confidence of all other letters (red).

BY TRAINING!

(backpropagation works here)

NOTICE how l has "different needs "



RNN Example: Character-Level Language Model

Training:

- **Using the standard Softmax classifier (also commonly referred to as the cross-entropy loss) on every output vector simultaneously.**
- **The RNN is trained with mini-batch Stochastic Gradient Descent RMSProp or Adam (per-parameter adaptive learning rate methods) to stablilize the updates.**
- **Notice also that the first time the character "l" is input, the target is "l", but the second time the target is "o". The RNN therefore cannot rely on the input alone and must use its recurrent connection to keep track of the context to achieve this task.**
- **Train with many inputs, obviously outputs are known.**

RNN Example: Character-Level Language Model

Test:

- **Feed a character into the RNN and get a distribution over what characters are likely to come next. We sample from this distribution, and feed it right back in to get the next letter.**
  - **one can experiment/change the softmax parameter (0.5, 1, …)**
- **Repeat this process and you're sampling text!**
- **One can train an RNN on different datasets and see what happens.**
- **A minimal character-level RNN language model in Python/numpy: (karpathy/min-char-rnn.py)**

RNN Example: Character-Level Language Model

Python code by Karpathy: [min-char-rnn.py]

```
# forward pass
for t in xrange(len(inputs)):
        xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation
        xs[t][inputs[t]] = 1
        hs[t] = np.tanh(np.dot(Wxh, xs[t]) + np.dot(Whh, hs[t-1]) + bh) # hidden state
        ys[t] = np.dot(Why, hs[t]) + by # unnormalized log probabilities for next chars
        ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars
```

RNN Example: Character-Level Language Model

Example: RNN-generated Shakespeare Dialogue

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.

RNN Example: Character-Level Language Model

Example: RNN-generated Wikipedia Page

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict. Copyright was the succession of independence in the slop of Syrian influence that was a famous German movement based on a more popular servicious, non-doctrinal and sexual power post. Many governments recognize the military housing of the [[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]], that is sympathetic to be to the [[Punjab Resolution]] (PJS)[http://www.humah.yahoo.com/guardian. cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery was swear to advance to the resources for those Socialism's rule, was starting to signing a major tripad of aid exile.]]

Notes:

- **Trained on 100MB of raw wikipedia**
- **URL does not exist**
- **Opens and closes parenthesis**
- **Timestamps are made up**

RNN Example: Character-Level Language Model Example: RNN-generated Code Sample

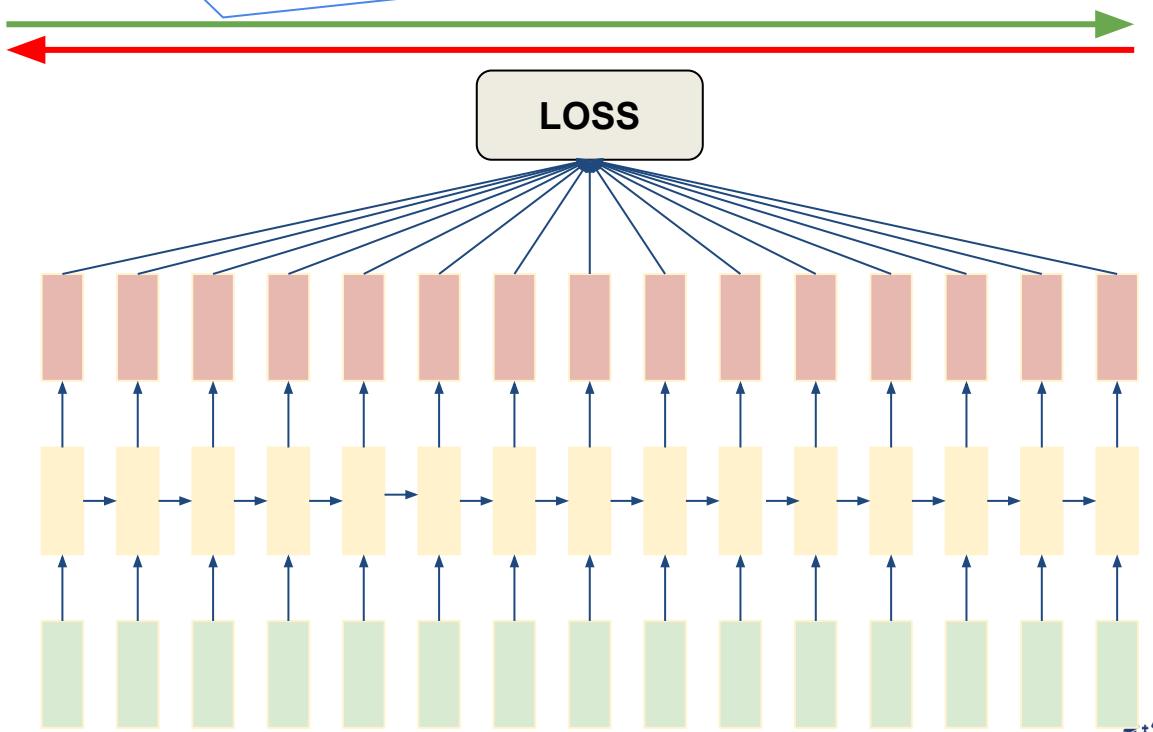
```
/*
 * Increment the size file of the new incorrect UI_FILTER
group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
```

Notes:

- The model compares tty == tty, which is vacuously true. On the other hand, at least the variable tty exists in the scope this time!
- In the last function, the code does not return anything, which happens to be correct since the function signature is void.
- However, the first two functions were also declared void and did return values. This is again a form of a common mistake due to long-term interactions.

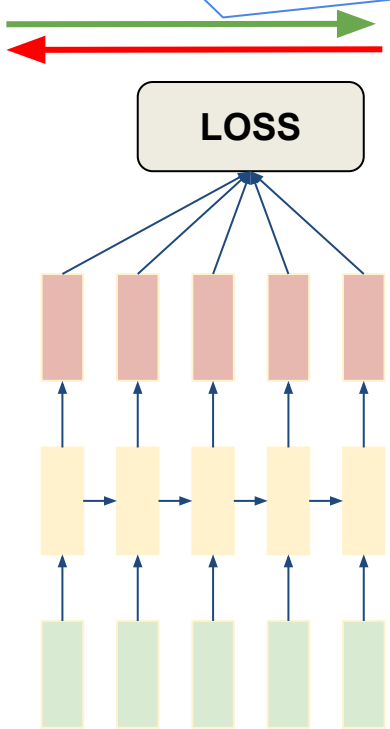
RNNs: Forward - Backpropagation processes

Forward through the whole sequence to compute loss, then backward through the whole sequence to compute gradient



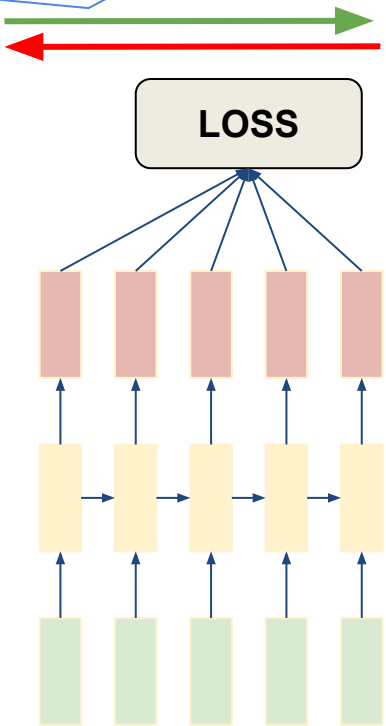
RNNs: Forward - Backpropagation processes

Forward through a chunk of a sequence to compute loss, then backward through the same chunk to compute gradient



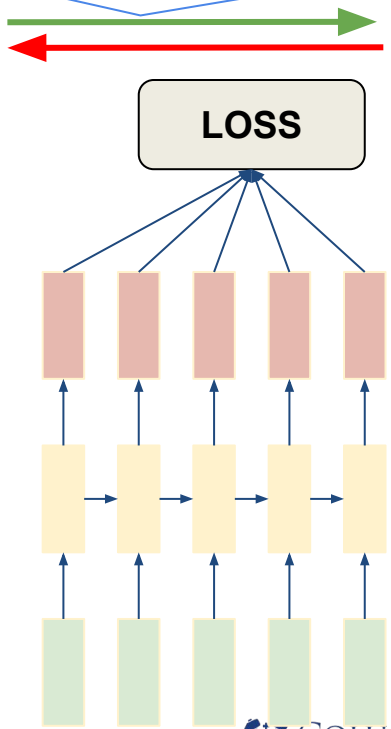
RNNs: Forward - Backpropagation processes

Forward through a chunk of a sequence to compute loss, then backward through the same chunk to compute gradient



RNNs: Forward - Backpropagation processes

Forward through a chunk of a sequence to compute loss, then backward through the same chunk to compute gradient



Visualizing the predictions and the "neuron" firings in the RNN

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

A fun visualization is to look at the predicted distributions over characters.

In the visualizations below we feed a Wikipedia RNN model character data from the validation set (shown along the blue/green rows) and

under every character we visualize (in red) the top 5 guesses that the model assigns for the next character.

The guesses are colored by their probability (so dark red = judged as very likely, white = not very likely).

For example, notice that there are stretches of characters where the model is extremely confident about the next letter (e.g., the model is very confident about characters during the http://www. sequence).

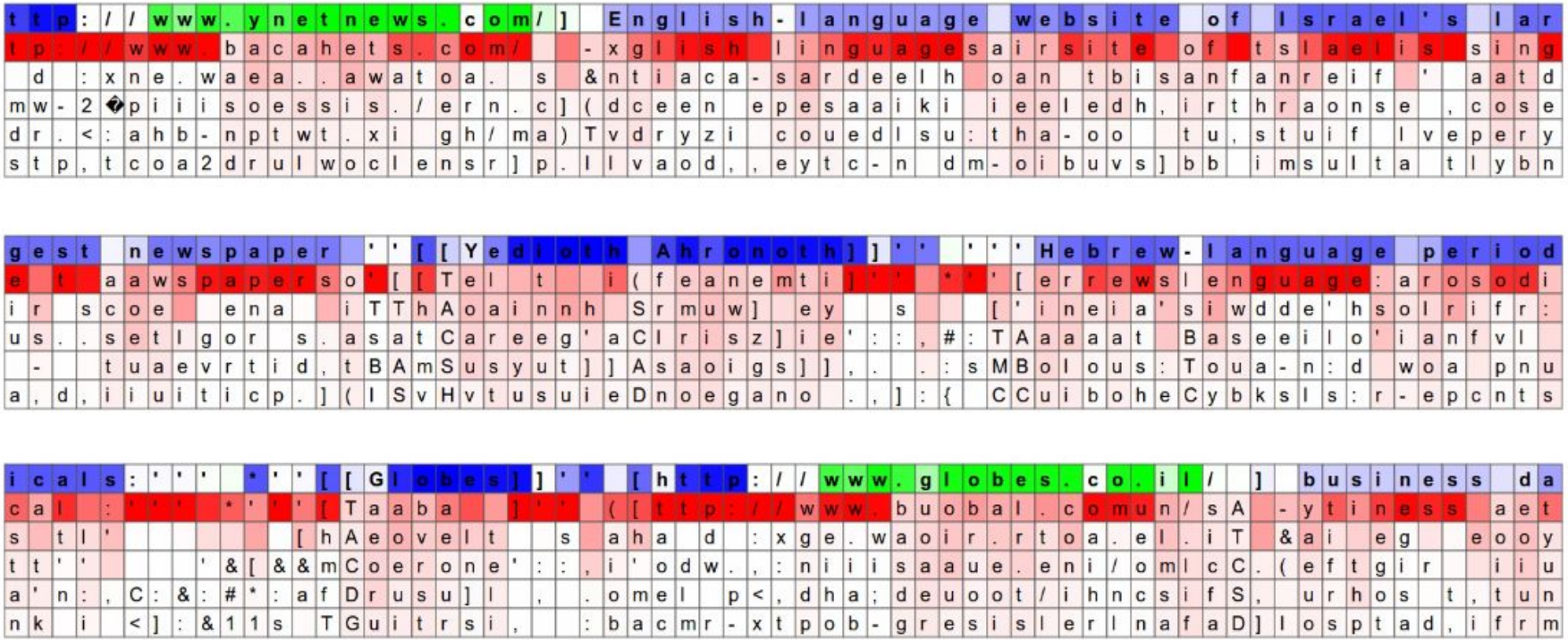
The input character sequence (blue/green) is colored based on the *firing of a randomly chosen neuron in the hidden representation of the RNN*.

Think about it as green = very excited and blue = not very excited (for those familiar with details of LSTMs, these are values between [-1,1] in the hidden state vector, which is just the gated and tanh'd LSTM cell state).

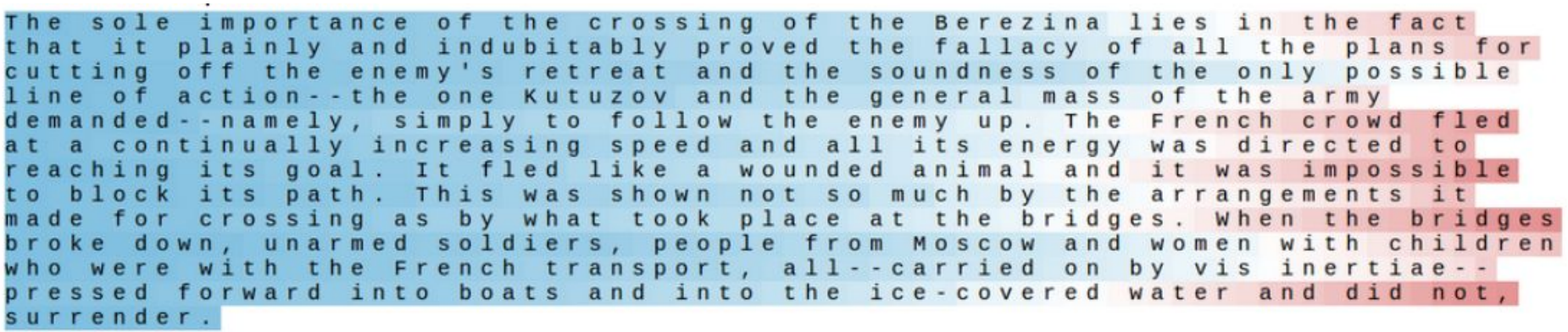
Intuitively, this is visualizing the firing rate of some neuron in the "brain" of the RNN while it reads the input sequence.

Different neurons might be looking for different patterns; Below we'll look at 4 different ones that I found and thought were interesting or interpretable (many also aren't):

Visualizing the predictions and the “neuron” firings in the RNN



RNNs: Sensitivity of Individual Neurons



Cell sensitive to position in line

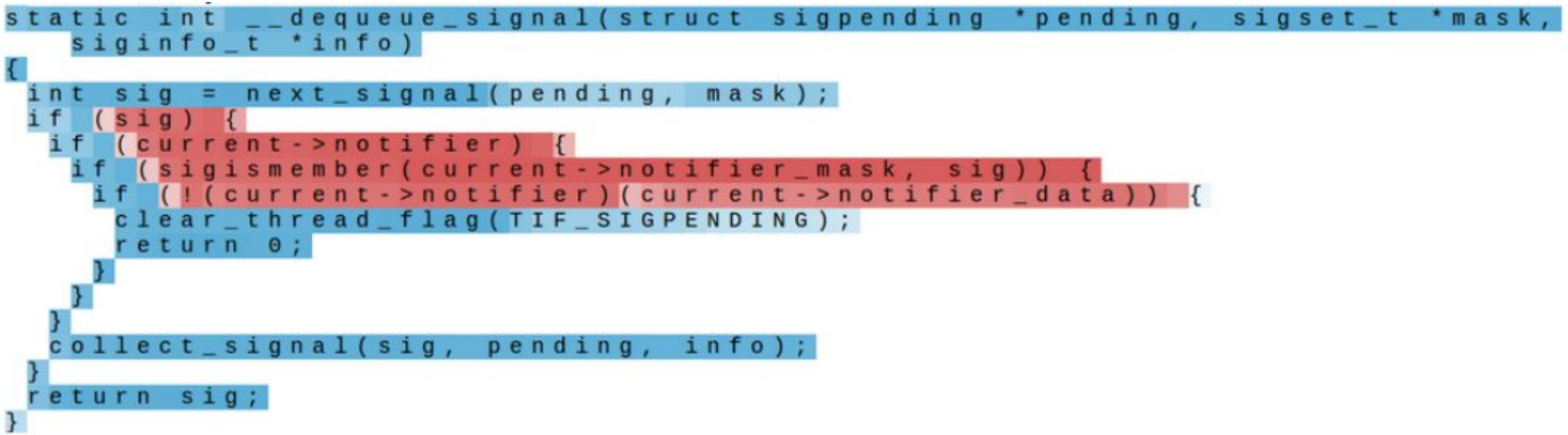
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs: Sensitivity of Individual Neurons

Cell that turns on inside quotes

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs: Sensitivity of Individual Neurons



```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

Cell that robustly activates inside if statements

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

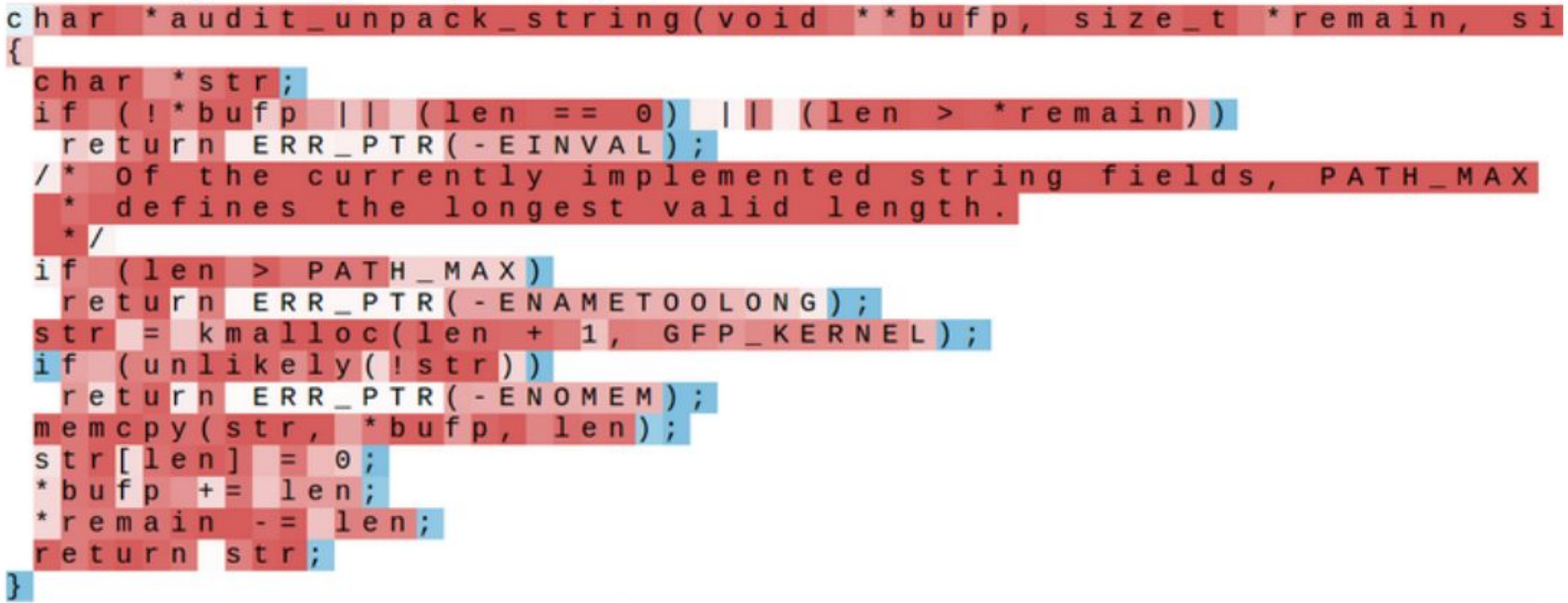
RNNs: Sensitivity of Individual Neurons

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

Cell that is sensitive to the depth of an expression

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs: Sensitivity of Individual Neurons

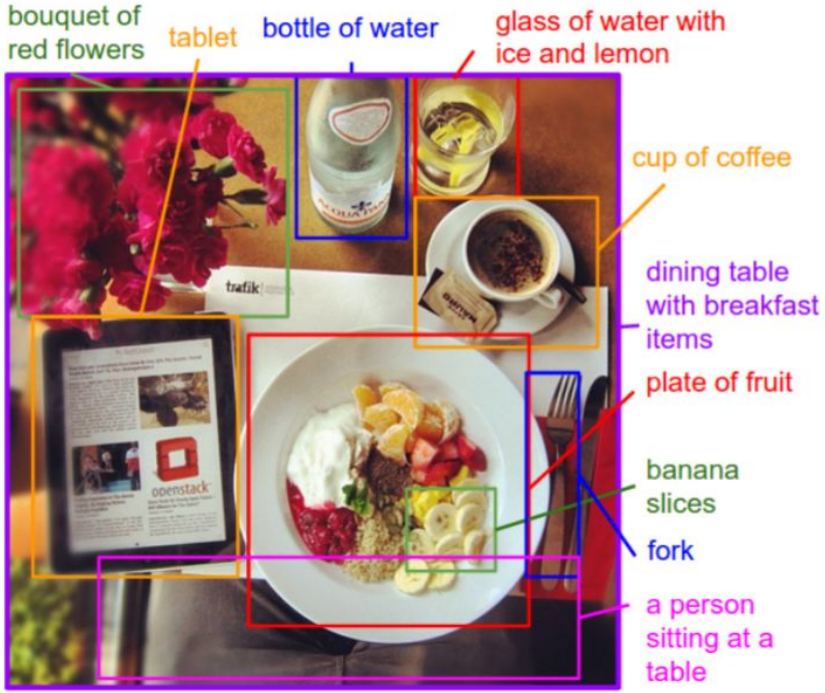


```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
  if (len > PATH_MAX)
    return ERR_PTR(-ENAMETOOLONG);
  str = kmalloc(len + 1, GFP_KERNEL);
  if (unlikely(!str))
    return ERR_PTR(-ENOMEM);
  memcpy(str, *bufp, len);
  str[len] = 0;
  *bufp += len;
  *remain -= len;
  return str;
}
```

Cell helpful for predicting new line - turns on for some “)”

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

CNN + RNN -> Image Captioning



A model that generates natural language descriptions of images and their regions.

Combination of

- **Convolutional Neural Networks over image regions**
- **Bidirectional Recurrent Neural Networks over sentences, and**
- **A structured objective that aligns the two modalities through a multimodal embedding**

Karpathy et a, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015;

CNN + RNN -> Image Captioning

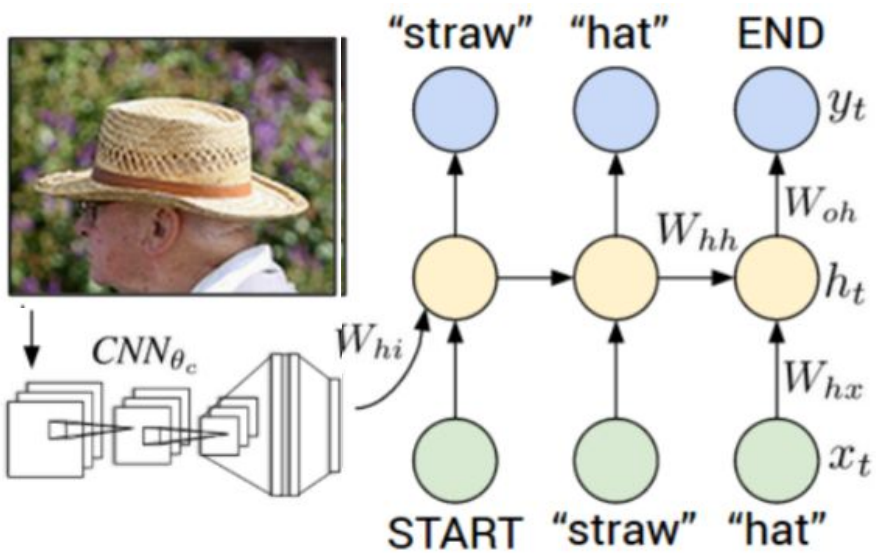


Diagram of the multimodal Recurrent Neural Network generative model.

The RNN takes a word,

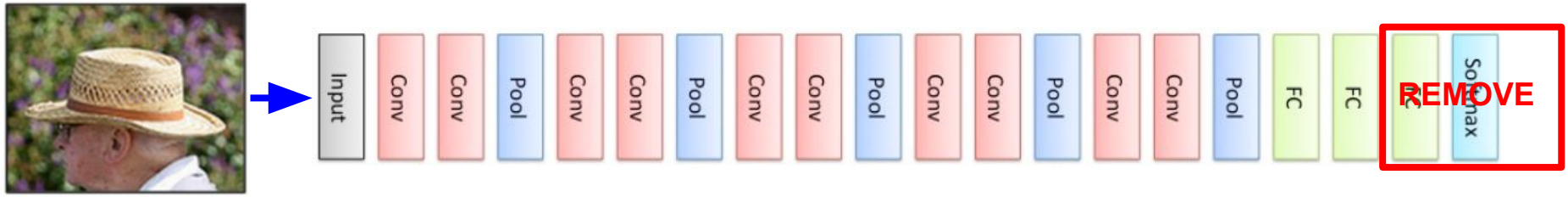
- **the context from previous time steps,**
- **and defines a distribution over the next word in the sentence.**

The RNN is conditioned on the image information at the first time step.

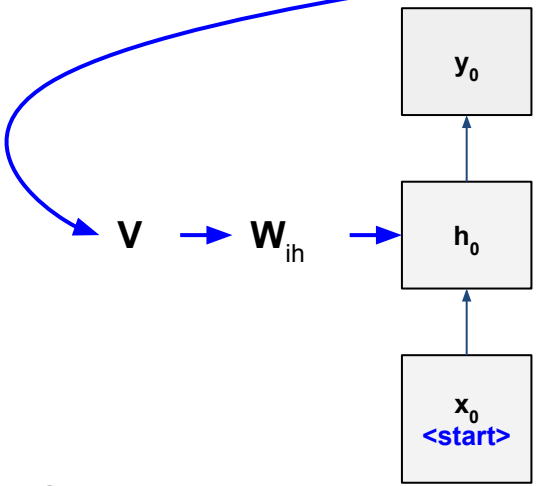
START and END are special tokens.

Karpathy et a, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015;

CNN + RNN -> Image Captioning: Flow



CNN + RNN -> Image Captioning: Flow



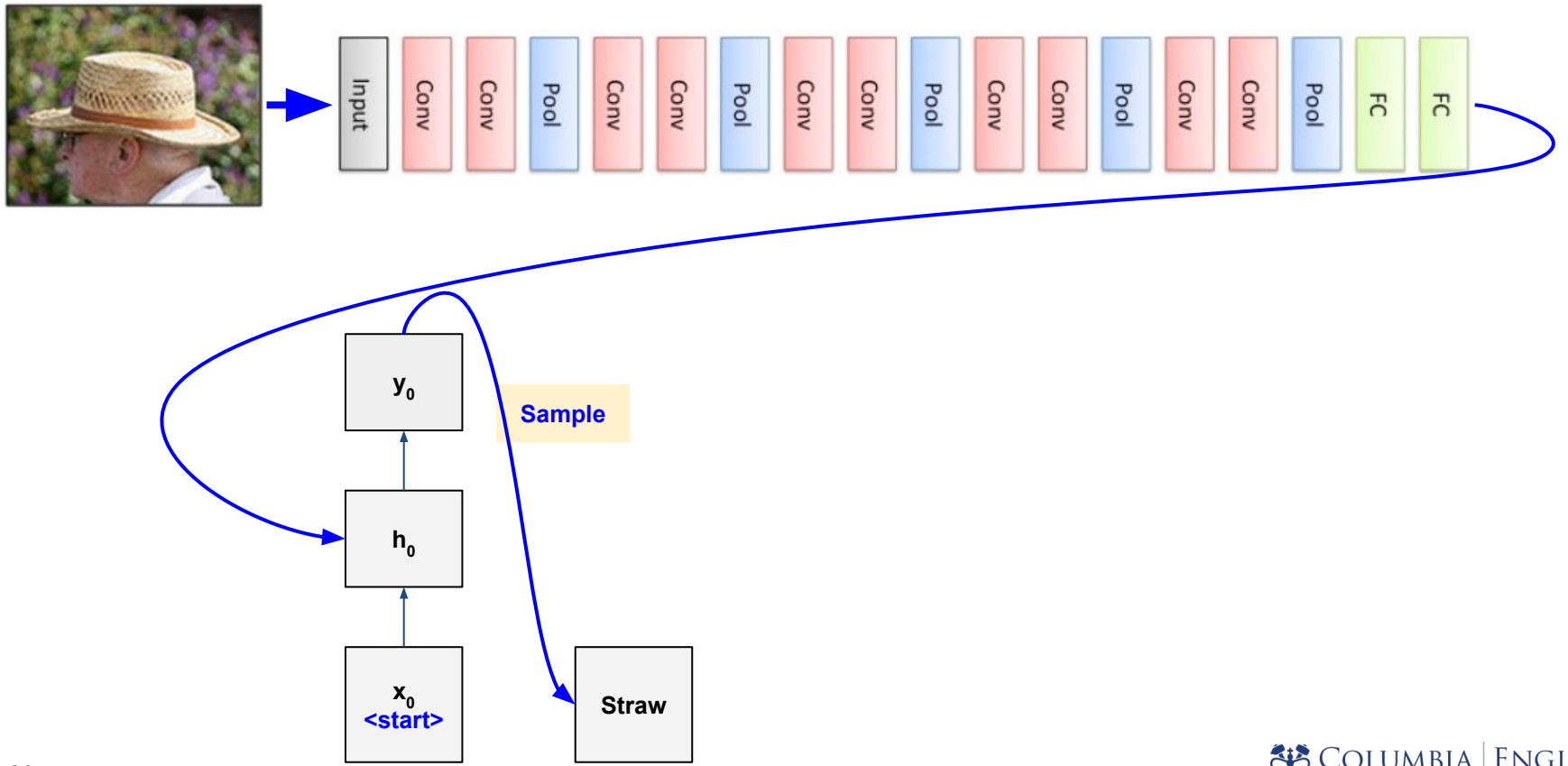
Usual computation:

h = tanh(Wxh * x + Whh * h)

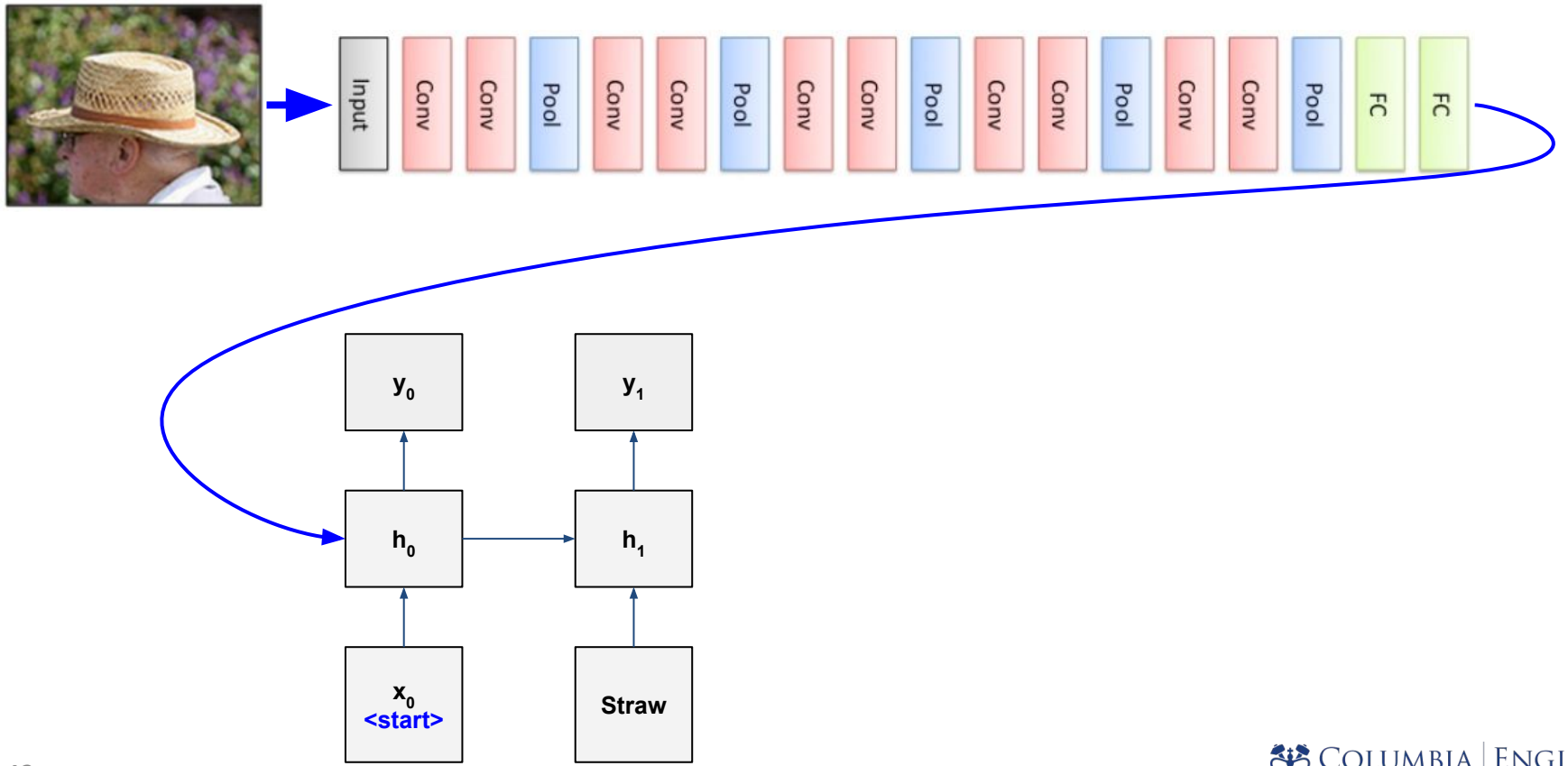
Computation now:

h = tanh(Wxh * x + Whh * h + Wih * v)

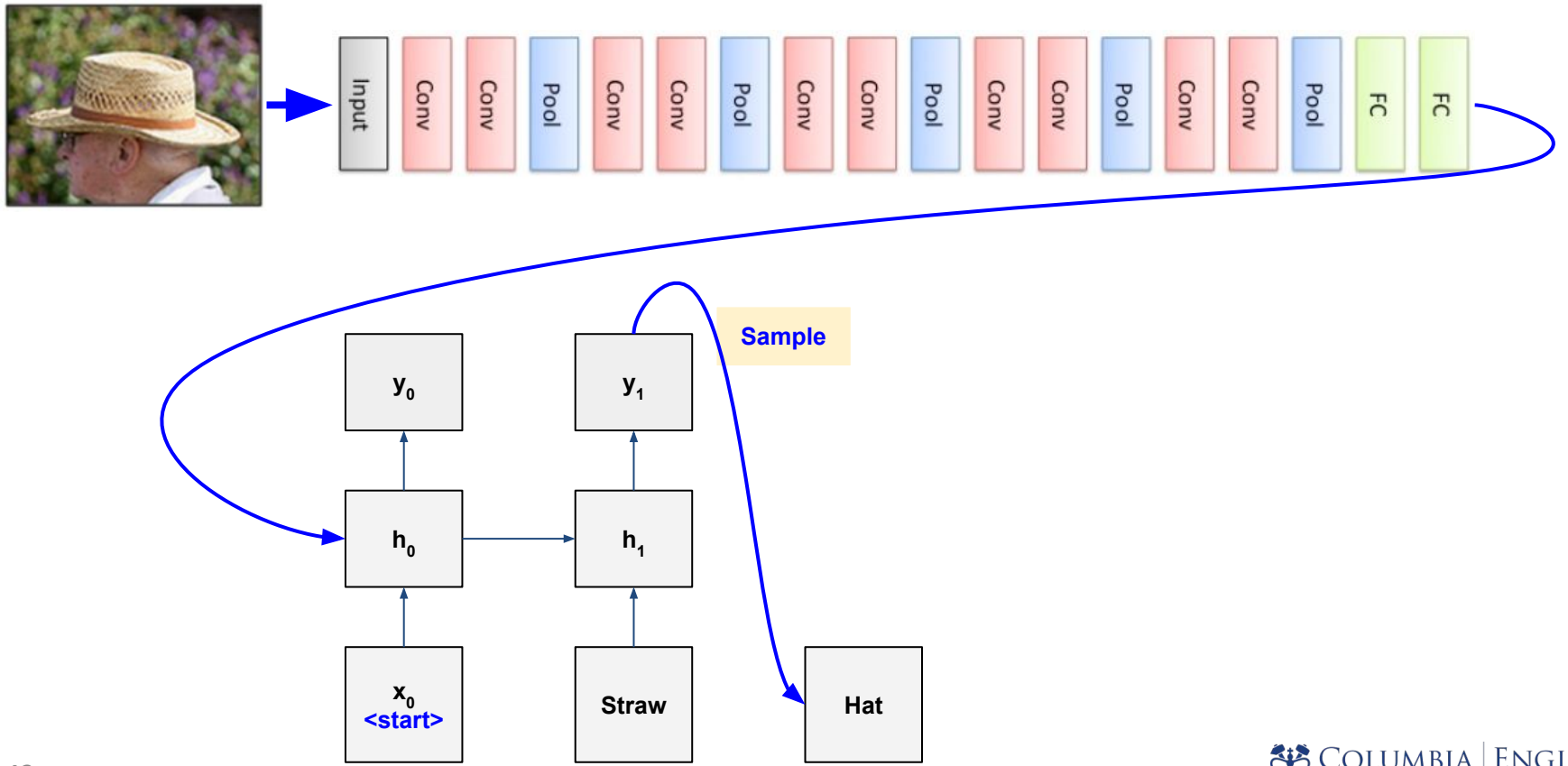
CNN + RNN -> Image Captioning: Flow



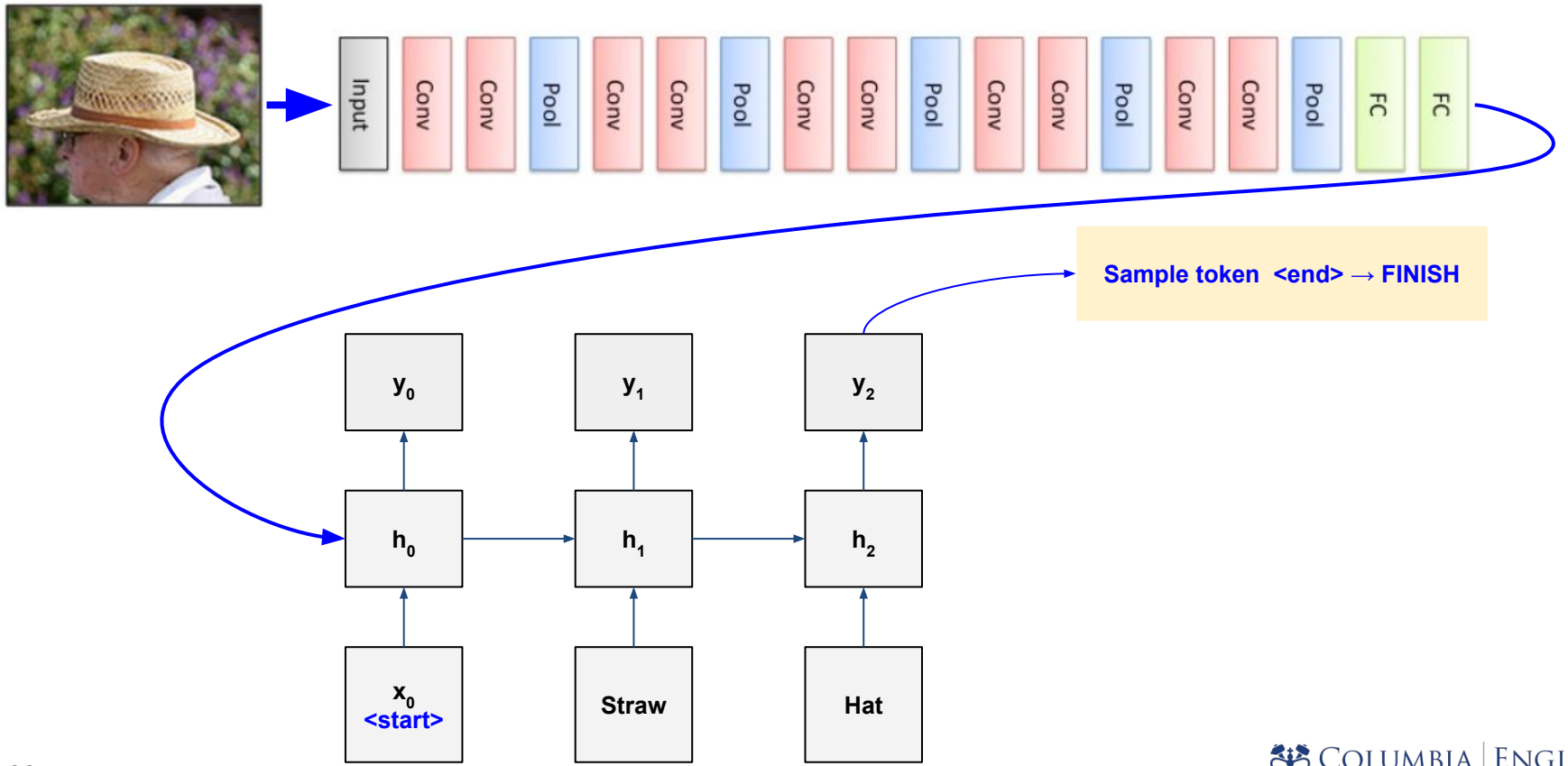
CNN + RNN -> Image Captioning: Flow



CNN + RNN -> Image Captioning: Flow



CNN + RNN -> Image Captioning: Flow



CNN + RNN -> Image Captioning

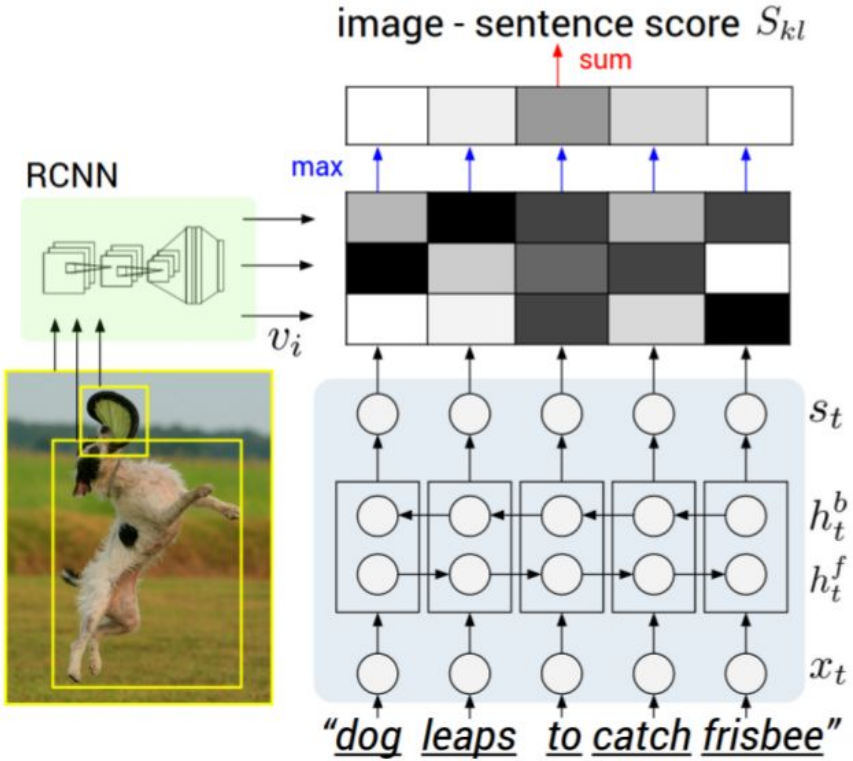


Diagram for evaluating the image-sentence score S_{kl}.

Object regions are embedded with a CNN (left).

Words (enriched by their context) are embedded in the same multimodal space with a BRNN (right).

Pairwise similarities are computed with inner products (magnitudes shown in grayscale) and finally reduced to image-sentence score with Equation...

Karpathy et a, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015;

CNN + RNN -> Image Captioning

Intuitively, a sentence-image pair should have a high matching score if its words have a confident support in the image.

The model of Karpathy et al. interprets the dot product $v^T_i s_t$ between the i-th region and t-th word as a measure of similarity and use it to define the score between image k and sentence l as:

$$S_{kl} = \sum_{t \in g_l} \sum_{i \in g_k} max(0, v_i^T s_t)$$

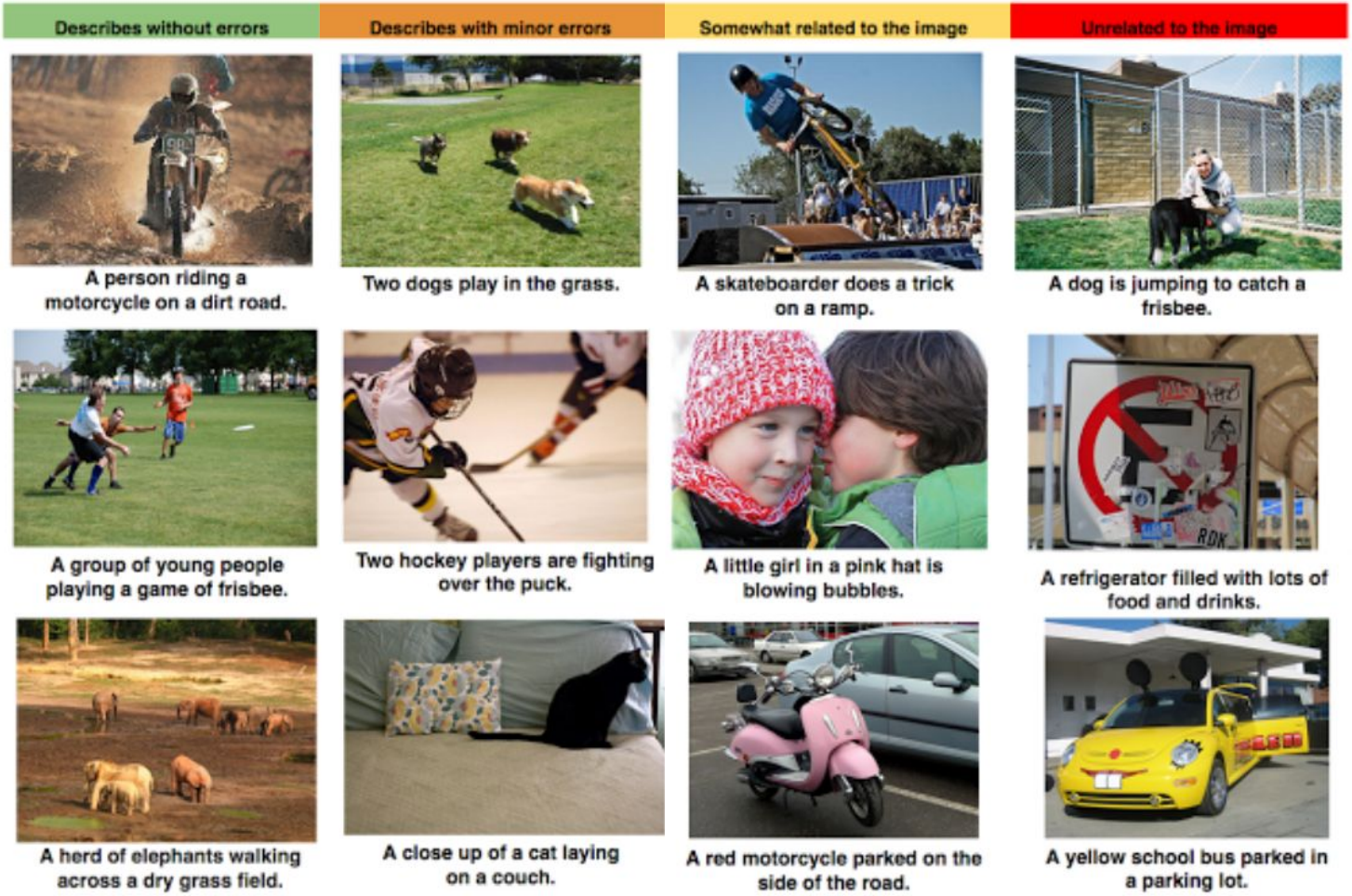
Here, g_k is the set of image fragments in image k and g_l is the set of sentence fragments in sentence l.

The indices k, l range over the images and sentences in the training set.

Together with their additional Multiple Instance Learning objective, this score carries the interpretation that a sentence fragment aligns to a subset of the image regions whenever the dot product is positive.

$$S_{kl} = \sum_{t \in g_l} max_{i \in g_k} v_i^T s_t$$

Image Caption Results



https://research.googleblog.com/2014/11/a-picture-is-worth-thousand-coherent.html

Image Captioning Results

Two pizzas sitting on top of a stove oven

https://research.googleblog.com/2014/11/a-picture-is-worth-thousand-coherent.html

Applications of RNN - A Curated List (4/14/2017)

RNN Applications (click to drill into)

- Natural Language Processing
  - Language Modeling
  - Speech Recognition
  - Machine Translation
  - Conversation Modeling
  - Question Answering
- Computer Vision
  - Object Recognition
  - Image Generation
  - Video Analysis
- Multimodal (CV+NLP)
  - Image Captioning
  - Video Captioning
  - Visual Question Answering
- Turing Machines
- Robotics
- Other

https://github.com/kjw0612/awesome-rnn

RNNs - Summary

- **RNNs can be deployed in a variety of architectures**
- **For working models, one needs to look beyond simple RNNs**
- **LSTM models are powerful**
  - **their additive interactions improve gradient flow**
- **RNNs are subject to vanishing or exploding gradients**
  - **Exploding is controlled with gradient clipping.**
  - **Vanishing is controlled with additive interactions (LSTM)**
- **RNNs need better understanding, particularly theoretical**

Backup Slides

Various