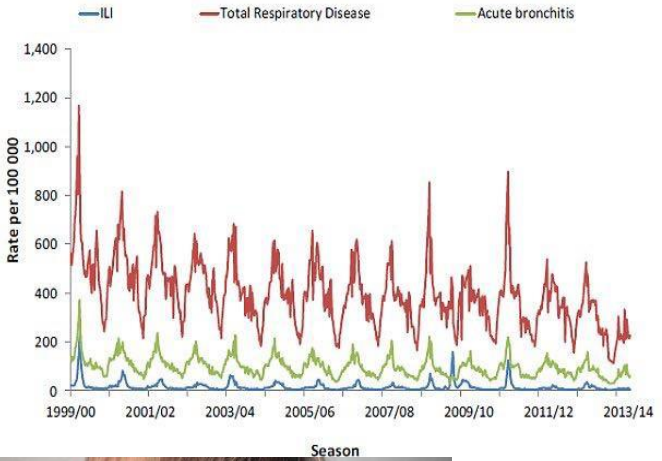
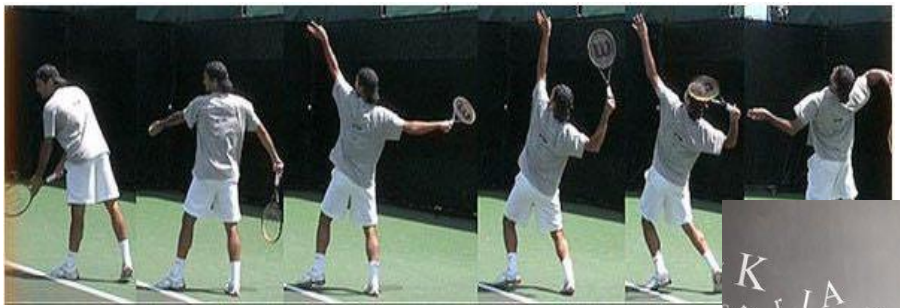
Learning With Memory - I
Recurrent Neural Networks

Motivation

Data is often sequential in nature





Not logged in Talk Co

Article Talk Read Edit View history Se

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

Deep learning

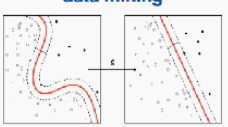
From Wikipedia, the free encyclopedia

For deep versus shallow learning in educational psychology, see Student approaches to learning.

Deep learning (also known as **deep structured learning**, **hierarchical learning** or **deep machine learning**) is a class of machine learning algorithms that:[1](pp199–200)

- use a cascade of many layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input. The algorithms may be supervised or unsupervised and applications include pattern analysis (unsupervised) and classification (supervised).
- are based on the (unsupervised) learning of multiple levels of features or representations of the data. Higher level features are derived from lower level features to form a hierarchical representation.
- are part of the broader machine learning field of learning representations

Machine learning and data mining



Problems [show]
Supervised learning (classification · regression) [show]
Clustering [show]
Dimensionality reduction [show]

Fundamental problems when dealing with sequences

- Is the sequential structure important for the prediction task?
- How to leverage structure at the input?
- How to deal with variable length inputs/outputs? How to align sequences?

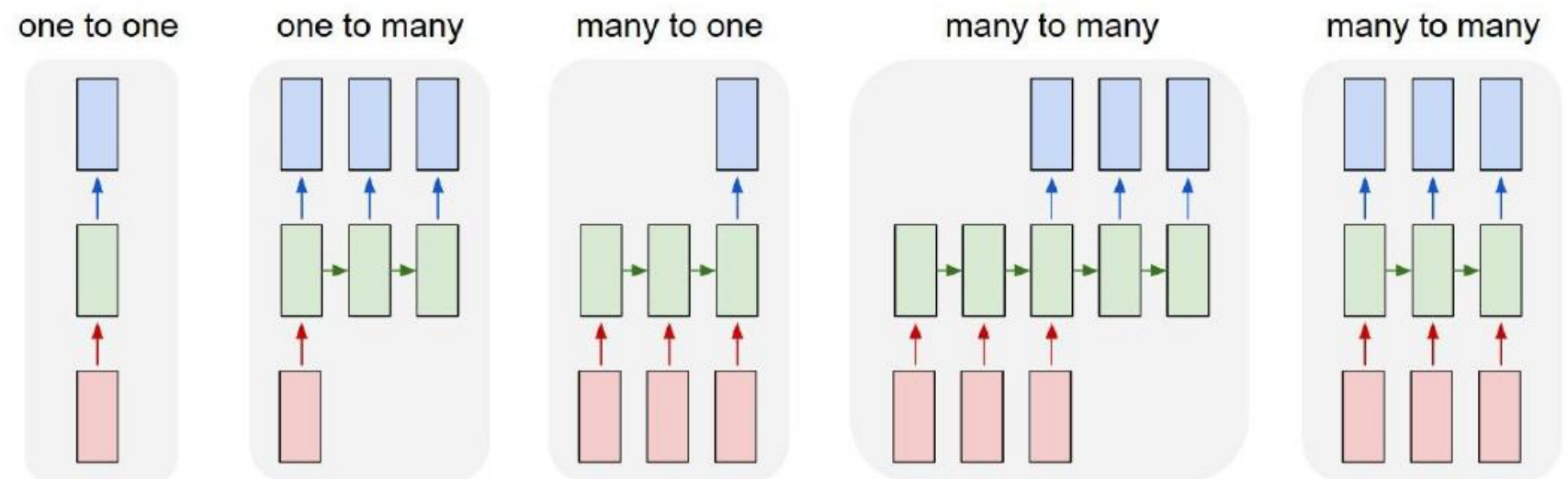
Fundamental problems when dealing with sequences

– How to deal with large output spaces? how to predict and what loss function to use?

Image source: Ali Mousavi, Embedding-based classifiers for large output spaces - Kirkland ML Summit '19

Learning Scenarios

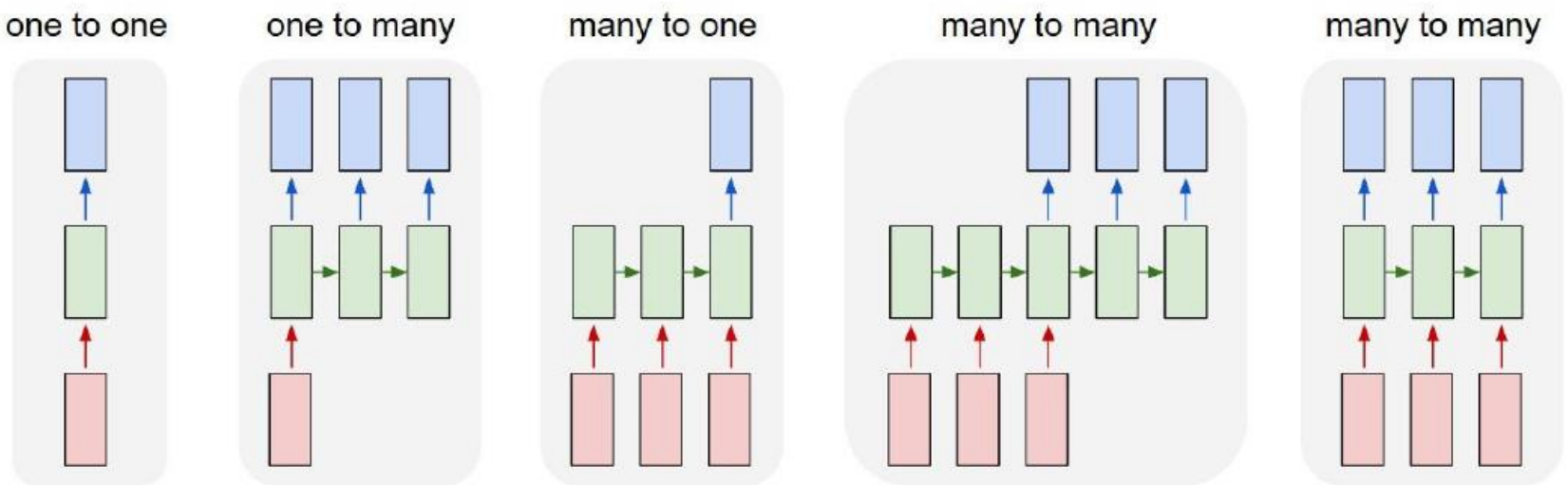
Single input -> Sequence



e.g. **Image Captioning**
image -> sequence of words

Learning Scenarios

Sequence -> Single label

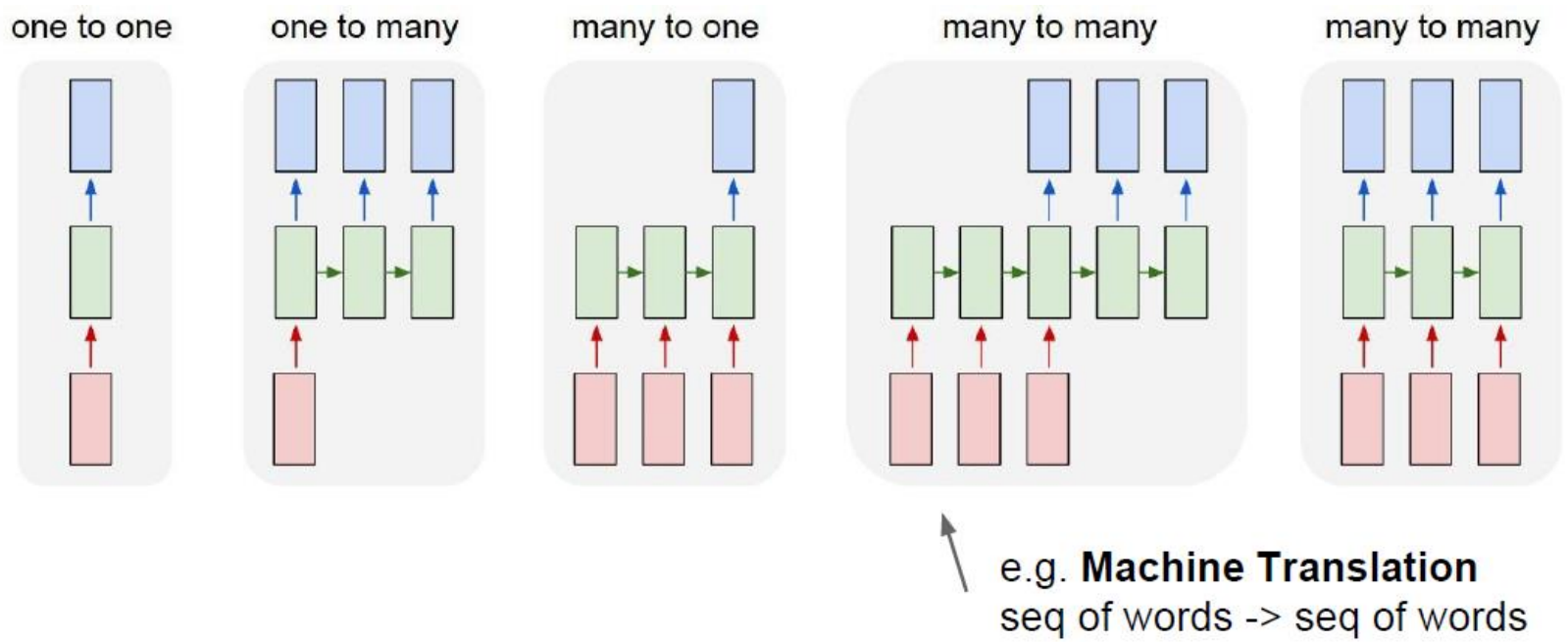


e.g. **Sentiment Classification**
sequence of words -> sentiment

Other Examples: Text classification, Language modeling, Action recognition, Music genre classification

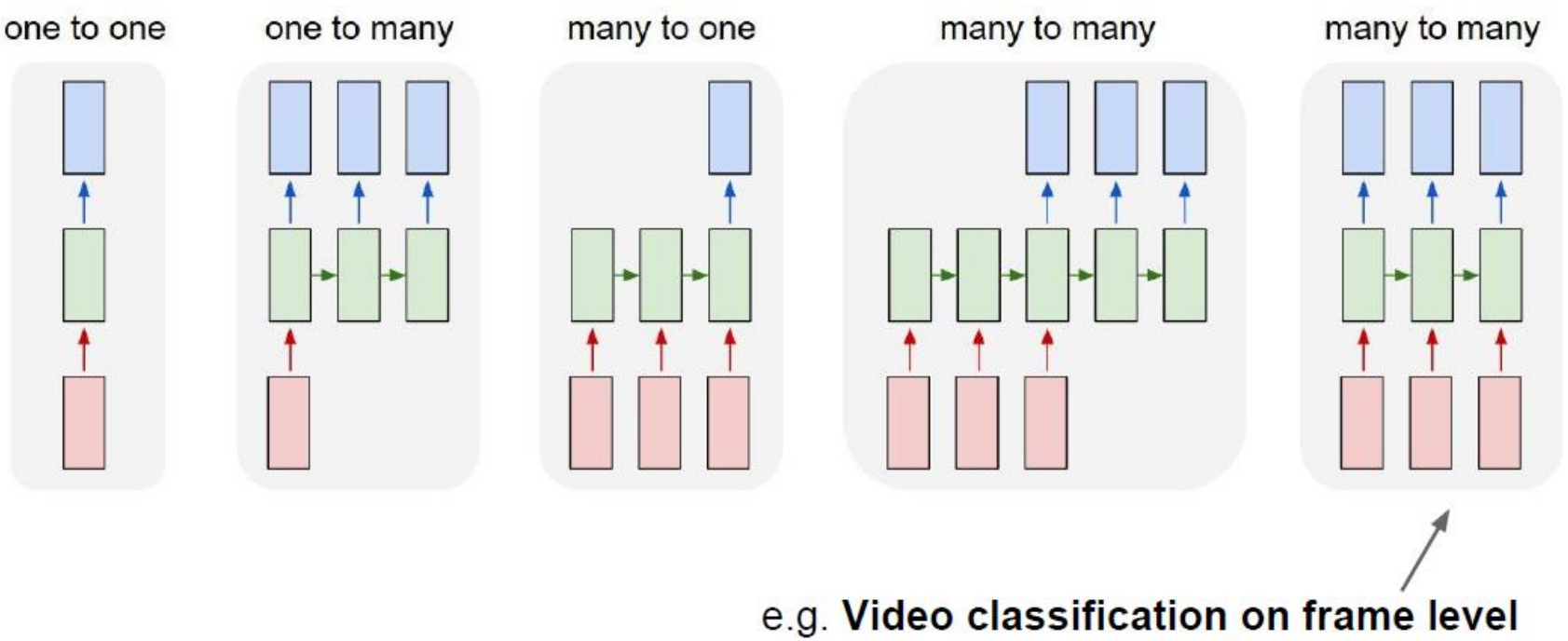
Learning Scenarios

Sequence -> Sequence

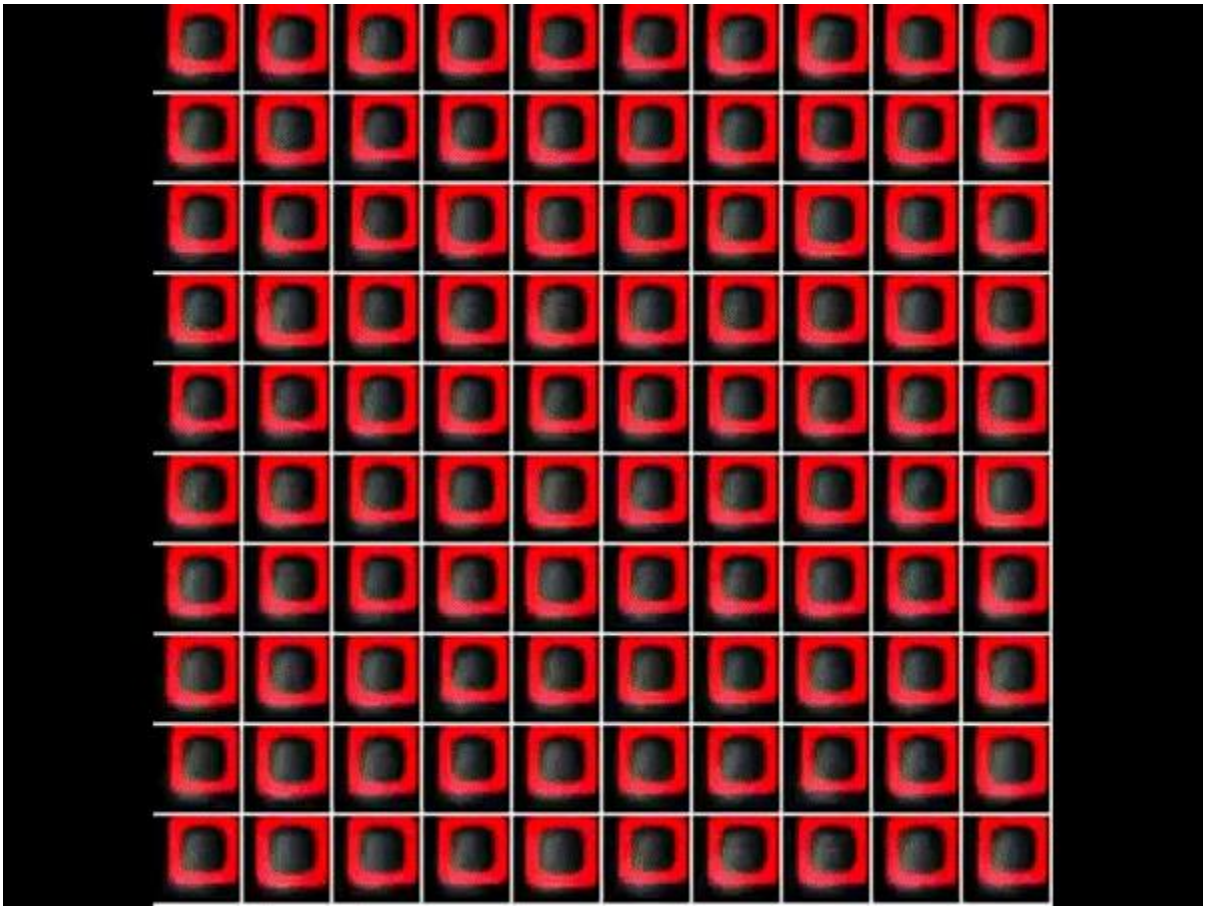


Other Examples: Summarization, Speech recognition, OCR, Video frame prediction

Learning Scenarios



RNN for Image Generation



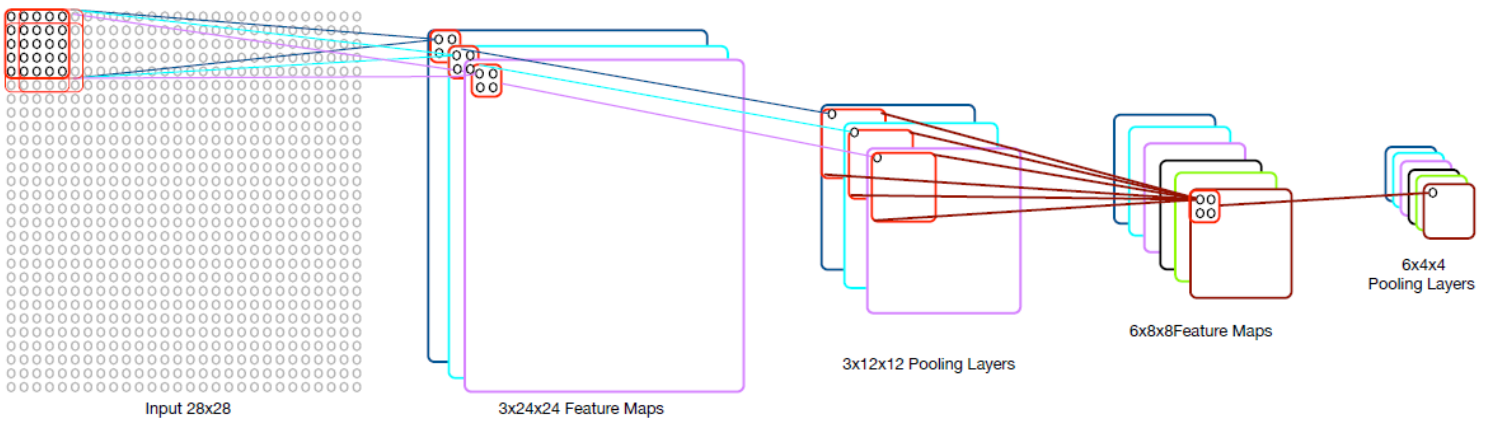
DRAW model improving its output by iterating over the canvas rather than producing the image in one shot

Gregor, Karol, et al. "DRAW: A recurrent neural network for image generation." *arXiv preprint arXiv:1502.04623* (2015).

TDNN and RNN

– Convolutional networks model invariances across space

Recap: Space invariance



- Local connectivity
- Weight sharing

Source: https://www.inf.ed.ac.uk/teaching/courses/mlp/2017-18/mlp09-rnn.pdf

TDNN and RNN

– Convolutional networks model invariances across space

- Can we do something similar across time?
  Yes - time-delay neural networks (TDNN)

- Can we use units to act as memories?
  Yes - recurrent neural networks (RNN)

Source: https://www.inf.ed.ac.uk/teaching/courses/mlp/2017-18/mlp09-rnn.pdf

TDNN and RNN – Parameter Sharing

- Parameter sharing makes it possible to extend and apply the model to examples of different forms (different lengths) and generalize across them.

- If we had separate parameters for each value of the time index, we could not generalize to sequence lengths not seen during training, nor share statistical strength across different sequence lengths and across different positions in time.

- Such sharing is particularly important when a specific piece of information can occur at multiple positions within the sequence.

TDNN and RNN – Parameter Sharing

- For example, consider the two sentences
  - "I went to Nepal in 2009"
  - "In 2009, I went to Nepal."

- If we ask a machine learning model to read each sentence and extract the year in which the narrator went to Nepal, we would like it to recognize the year 2009 as the relevant piece of information, whether it appears in the sixth word or the second word of the sentence

TDNN and RNN – Parameter Sharing

- For example, consider the two sentences
  - "I went to Nepal in 2009"
  - "In 2009, I went to Nepal."

- A traditional fully connected feedforward network would have separate parameters for each input feature, so it would need to learn all of the rules of the language separately at each position in the sentence.

- By comparison, a recurrent neural network shares the same weights across several time steps.

Time-delay Neural Network (TDNN)

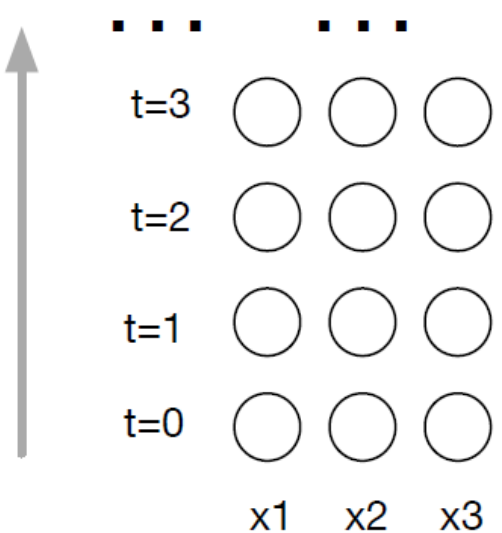
- TDNN is a multilayer artificial neural network architecture whose purpose is to classify patterns **shift-invariantly**.

- The classifier does not require explicit segmentation prior to classification.

- For the classification of a temporal pattern (such as speech), the TDNN thus avoids having to determine the beginning and end points of sounds before classifying them.

Time-delay Neural Network (TDNN)

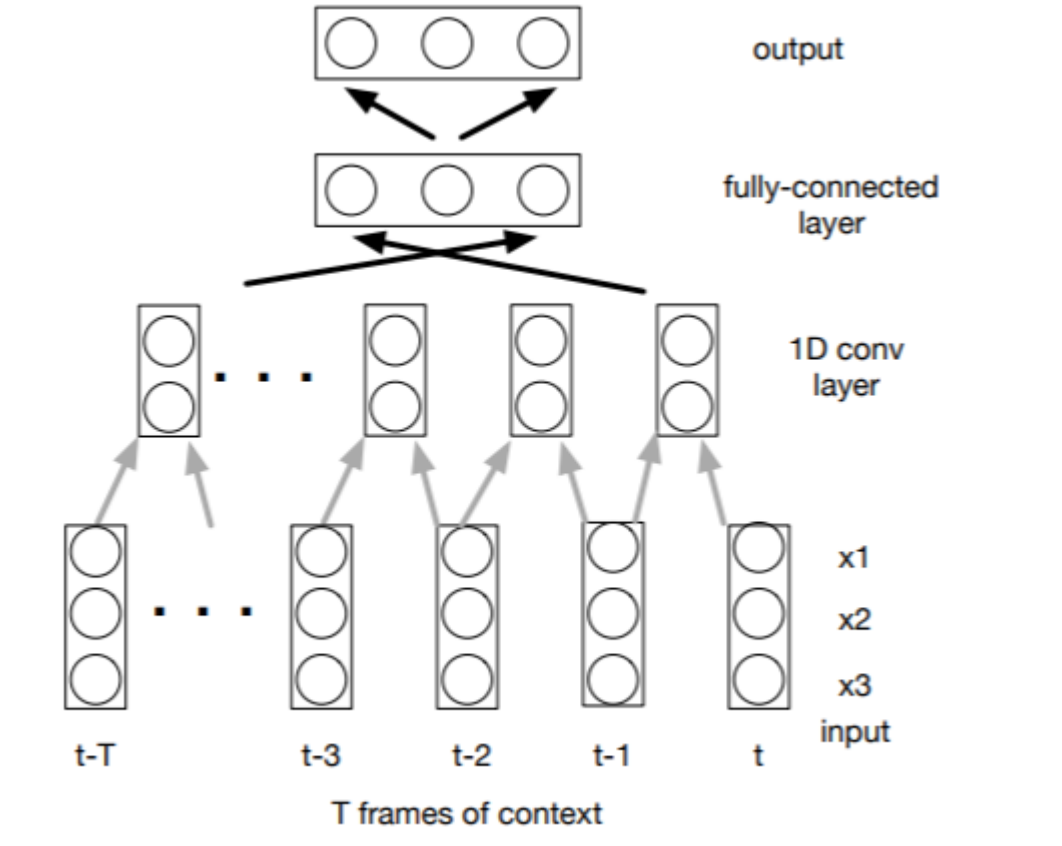
- Imagine modelling a time sequence of 3D vectors



Source: https://www.inf.ed.ac.uk/teaching/courses/mlp/2017-18/mlp09-rnn.pdf

Time-delay Neural Network (TDNN)

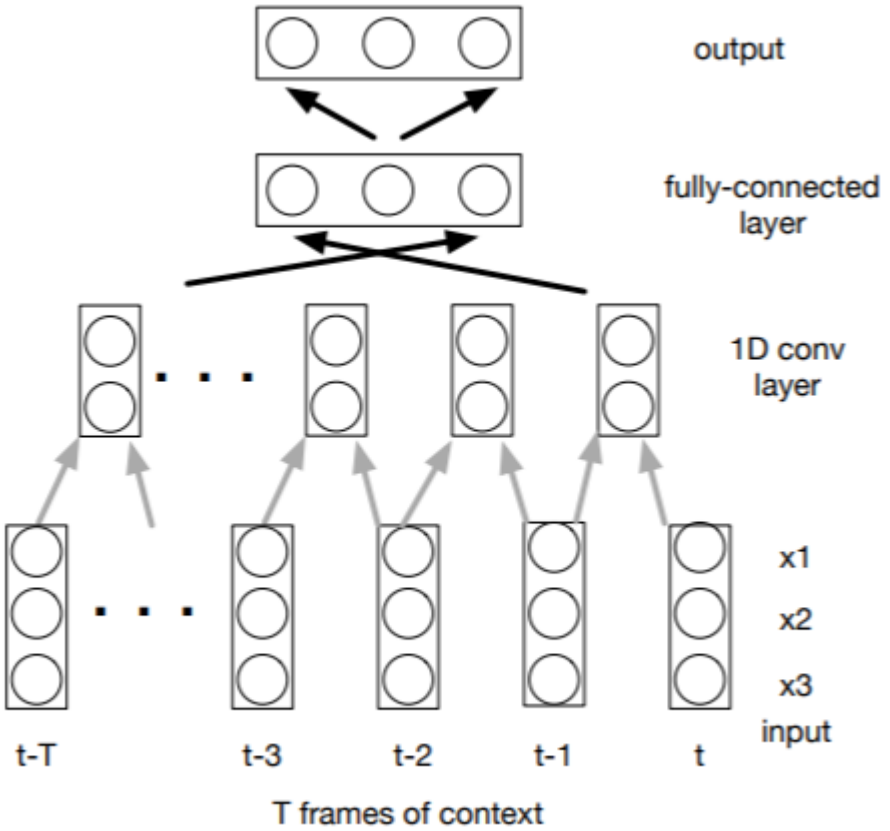
- Imagine modelling a time sequence of 3D vectors
- Can model fixed context with a feed-forward network with previous time input vectors added to the network input
- Model using 1D convolutions in time



Source: https://www.inf.ed.ac.uk/teaching/courses/mlp/2017-18/mlp09-rnn.pdf

Time-delay Neural Network (TDNN)

- The convolution operation allows a network to share parameters across time, but is shallow
- For two-dimensional signals (e.g. time-frequency patterns or images), a 2-D context window is observed at each layer. Higher layers have inputs from widening context windows than lower layers and thus generally model coarser levels of abstraction.



Source: https://www.inf.ed.ac.uk/teaching/courses/mlp/2017-18/mlp09-rnn.pdf

Time-delay Neural Network (TDNN)

- Shift-invariance is achieved by explicitly removing position dependence during backpropagation training. This is done by making time-shifted copies of a network across the dimension of invariance (here: time).

- The error gradient is then computed by backpropagation through all these networks from an overall target vector, but before performing the weight update, the error gradients associated with shifted copies are averaged and thus shared and constraint to be equal.

- Thus, all position dependence from backpropagation training through the shifted copies is removed and the copied networks learn the most salient hidden features shift-invariantly, i.e. independent of their precise position in the input data.

Source: https://www.inf.ed.ac.uk/teaching/courses/mlp/2017-18/mlp09-rnn.pdf

Time-delay Neural Network (TDNN)

- The TDNN was first proposed to classify phonemes in speech signals for automatic speech recognition, where the automatic determination of precise segments or feature boundaries is difficult or impossible.

- Because the TDNN recognizes phonemes and their underlying acoustic/phonetic features independent of position in time, it improved performance over static classification.

- It was also applied to two-dimensional signals: time-frequency patterns in speech and coordinate space pattern in OCR

Time-delay Neural Network (TDNN)

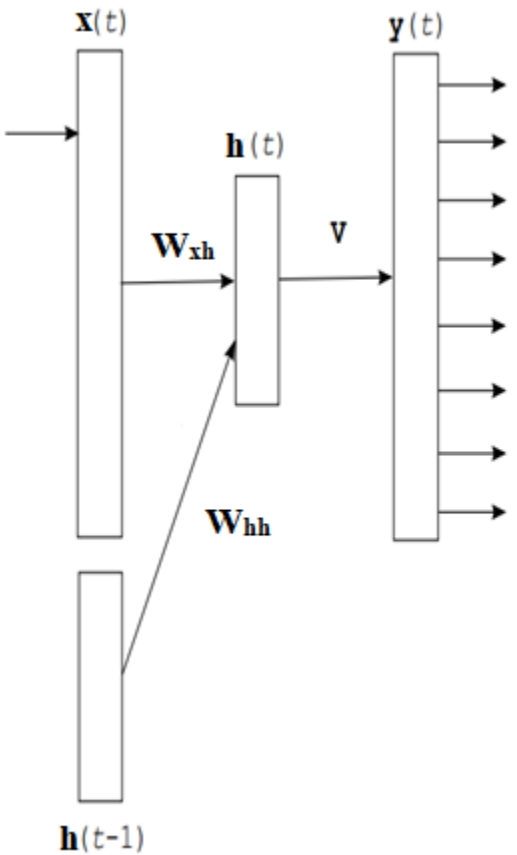
- The convolution operation allows a network to share parameters across time, but is shallow.
- The output of convolution is a sequence where each member of the output is a function of a small number of neighbouring members of the input.
- The idea of parameter sharing manifests in the application of the same convolution kernel at each time step.

Recurrent Neural Networks (RNN)

- Recurrent networks share parameters in a different way: Each member of the output is a function of the previous members of the output.
- Each member of the output is produced using the same update rule applied to the previous outputs.
- This recurrent formulation results in the sharing of parameters through a very deep computational graph.

Architecture: simple RNN

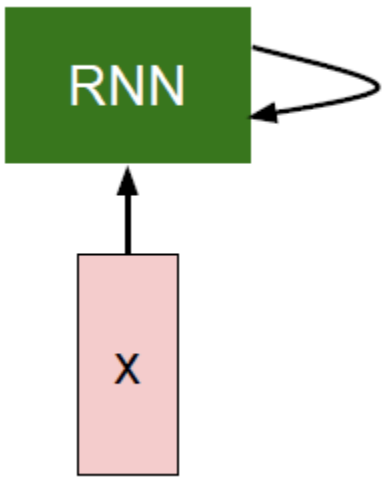
– Input layer
– Hidden layer with recurrent connections
– Output layer



In theory, the hidden layer can learn to represent unlimited memory

Also called Elman network
(*Finding structure in time*, Elman 1990)

Architecture: simple RNN



- RNN has a state and it receives input vectors through time
- It updates that state as a function of what it receives at every time step
- So the weights inside the RNN are tuned and will have different behaviour in terms of how its state evolves as it receives these inputs

Architecture: simple RNN

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

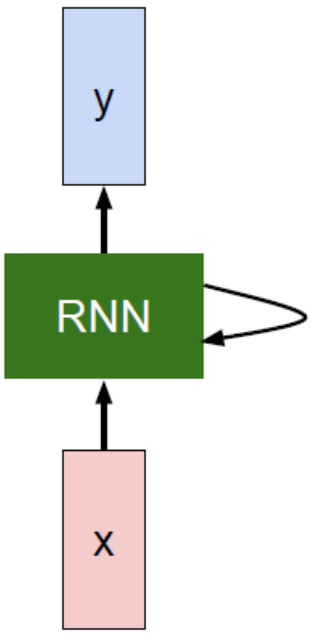
$$h_t = f_W(h_{t-1}, x_t)$$

new state

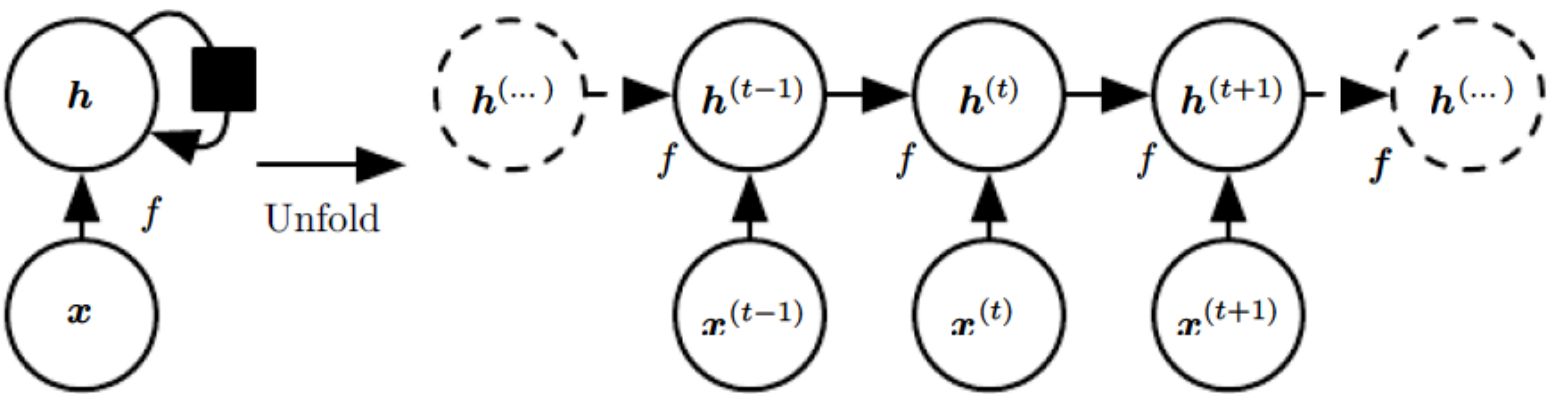
some function with parameters W

old state

input vector at some time step



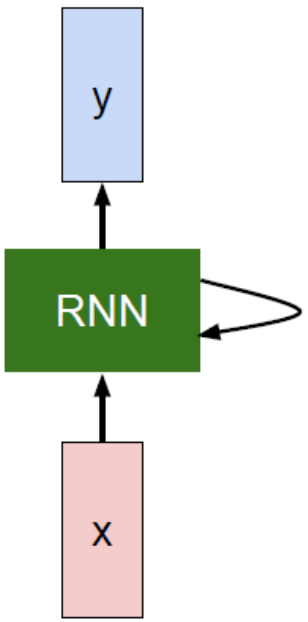
Architecture: simple RNN



Architecture: simple RNN

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

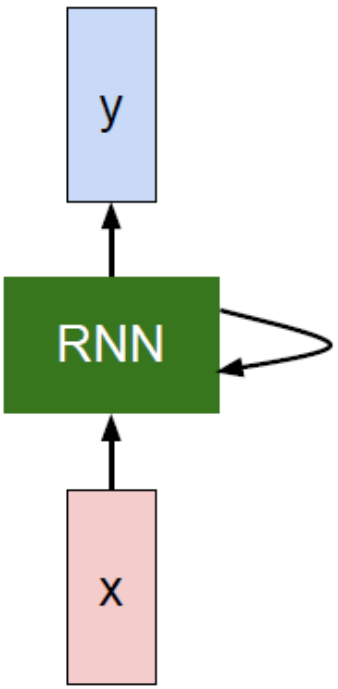
$$h_t = f_W(h_{t-1}, x_t)$$



Notice: the same function and the same set of parameters are used at every time step.

Architecture: simple RNN

The state consists of a single *"hidden"* vector **h**:

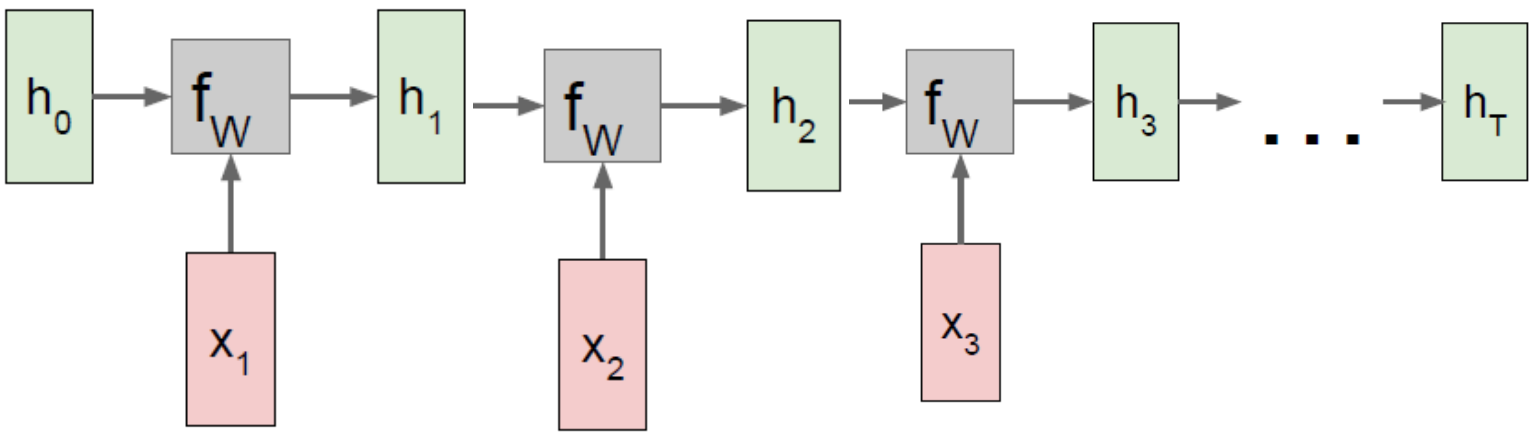


$$h_t = f_W(h_{t-1}, x_t)$$

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

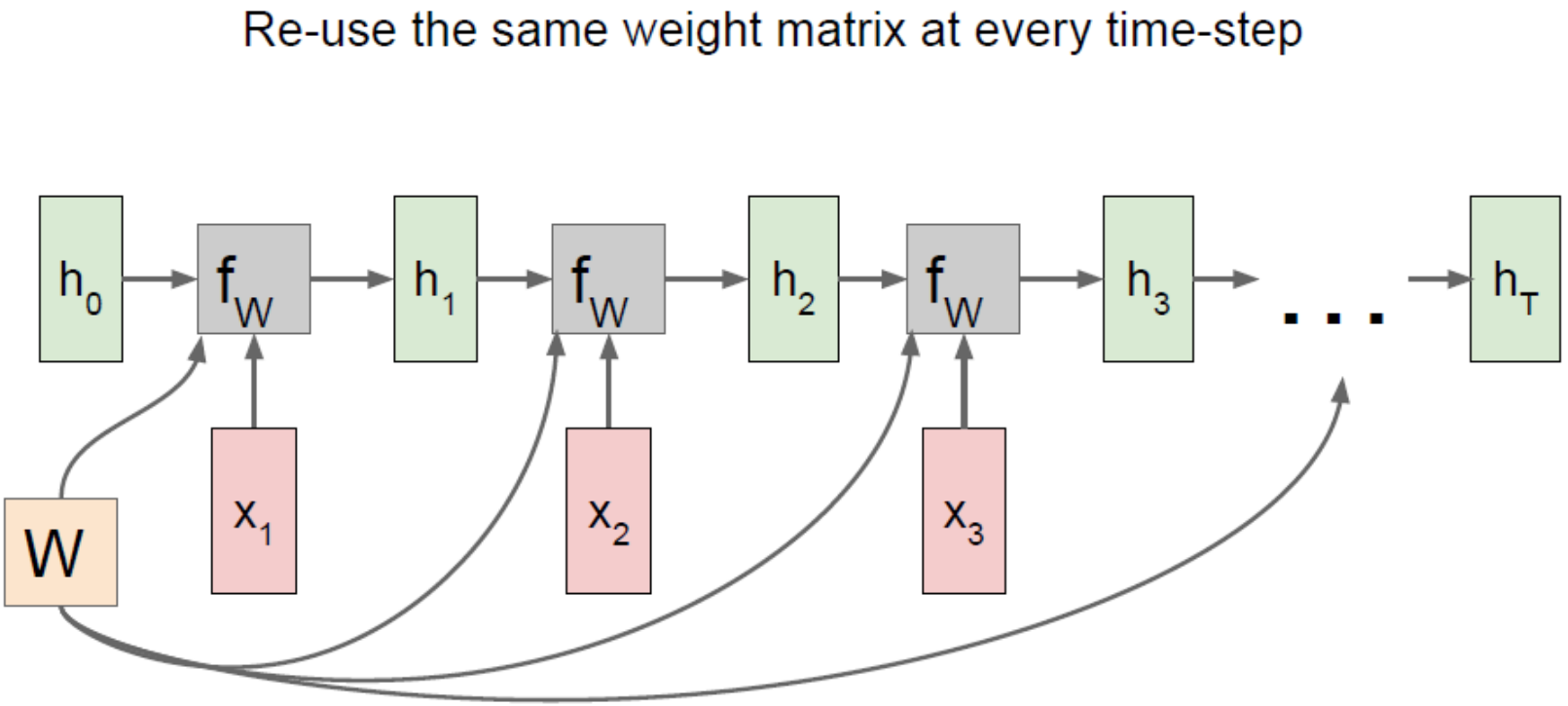
$$y_t = W_{hy}h_t$$

Architecture: simple RNN



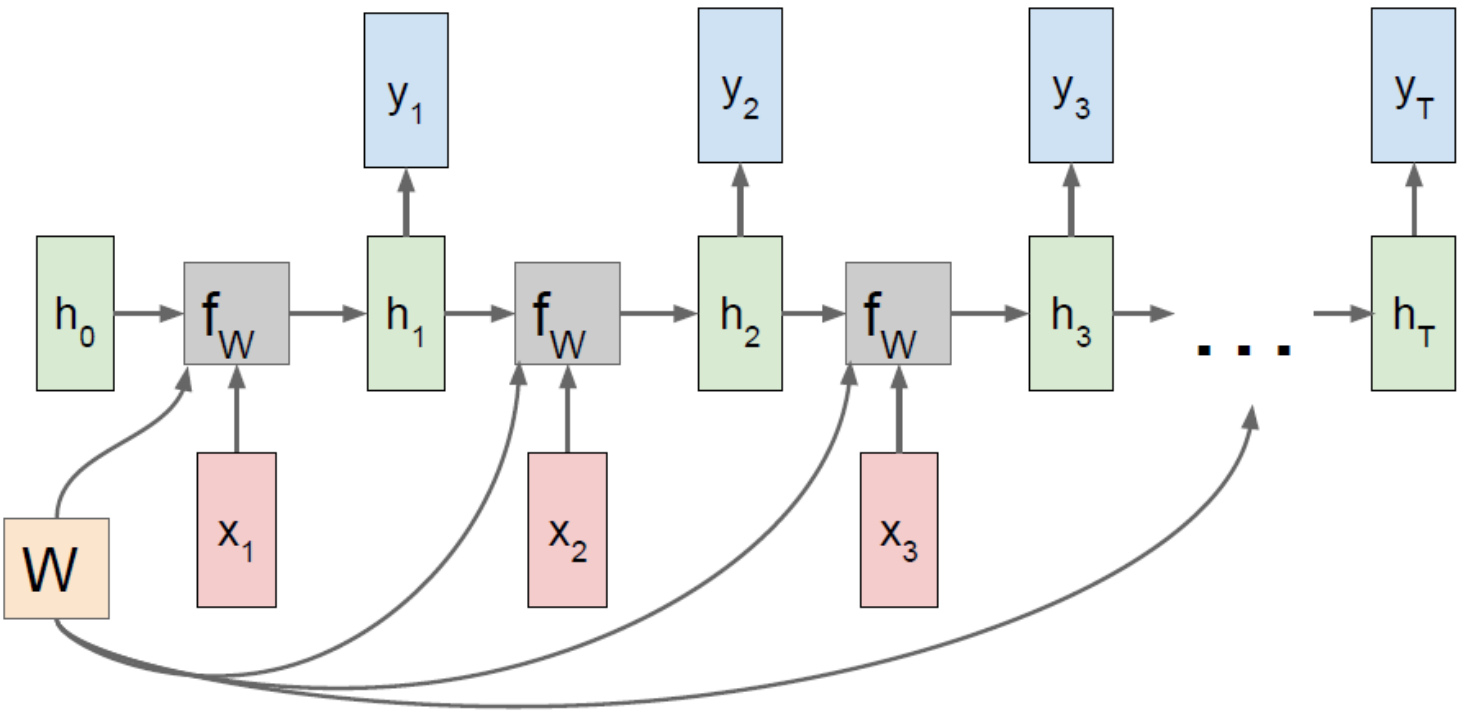
We can unroll the computational graph for visualization

Architecture: simple RNN



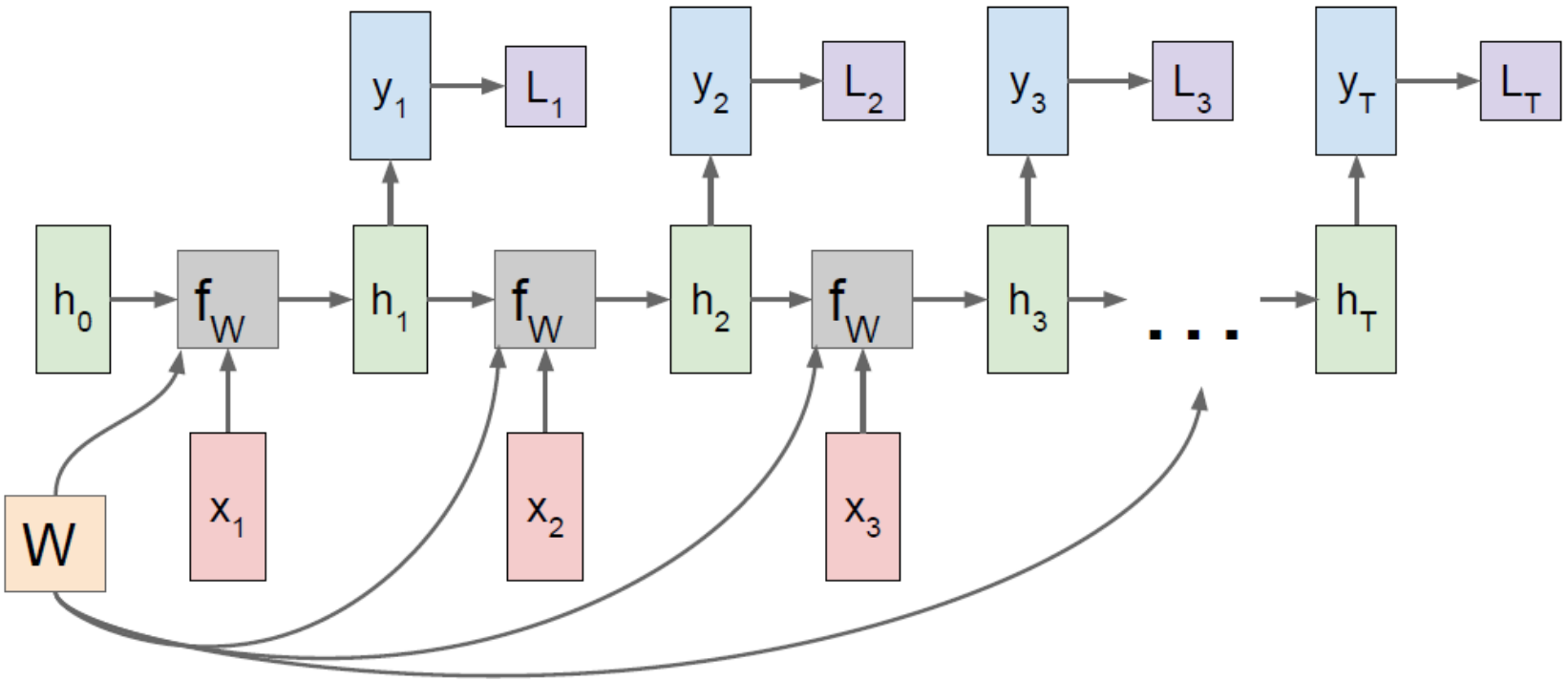
- We can make this process more explicit by writing W in this computational graph
- At each time step f_w blocks are receiving a unique h and unique x but the same W
- Backpropagation: a separate gradient will be flowing for W at each time step.
- Final gradient for the W will be the sum of these individual per-timestep gradients

Architecture: simple RNN



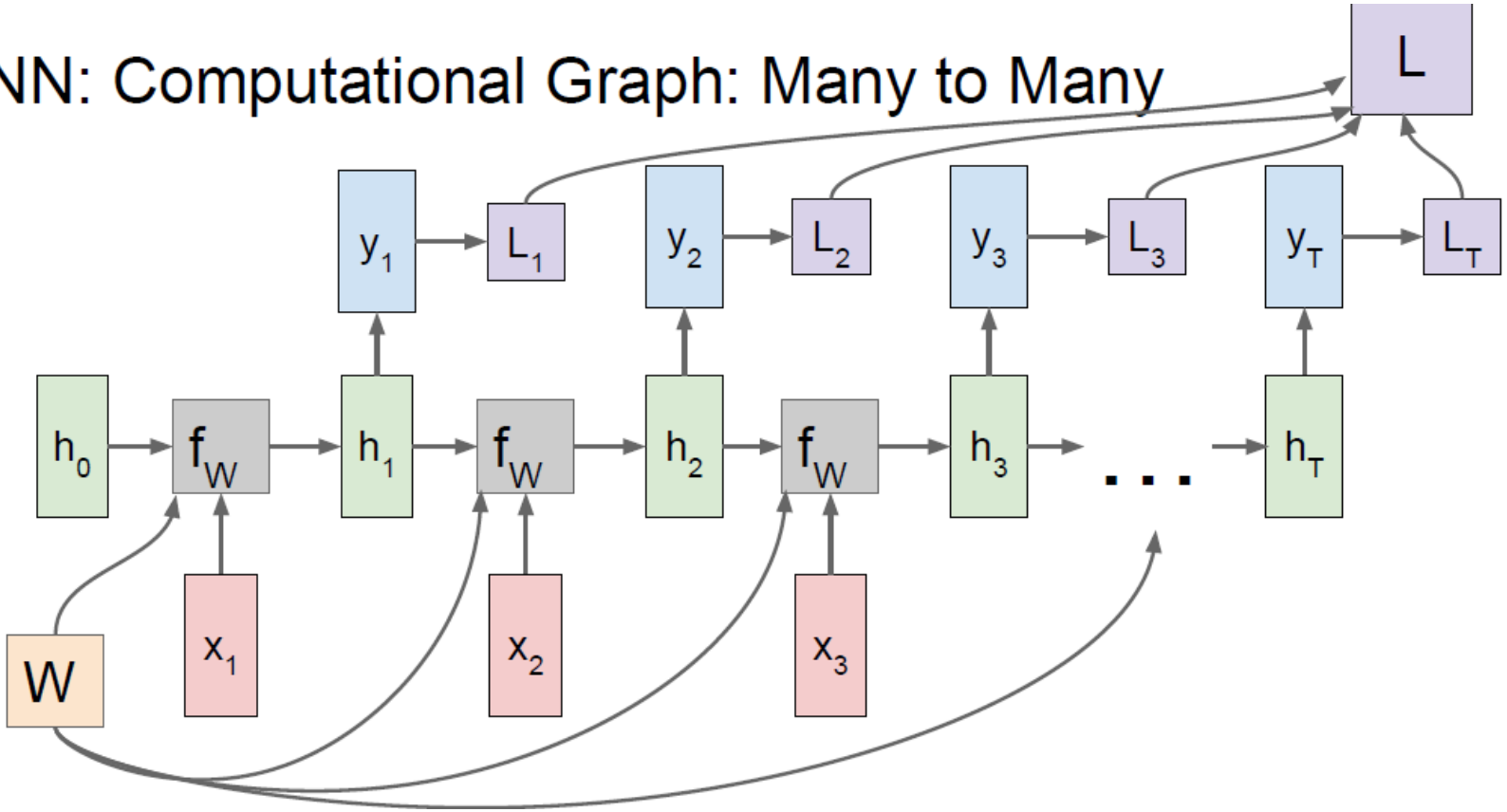
- We can explicitly write y_t into this graph so at every time step we can have an output such as class scores

Architecture: simple RNN



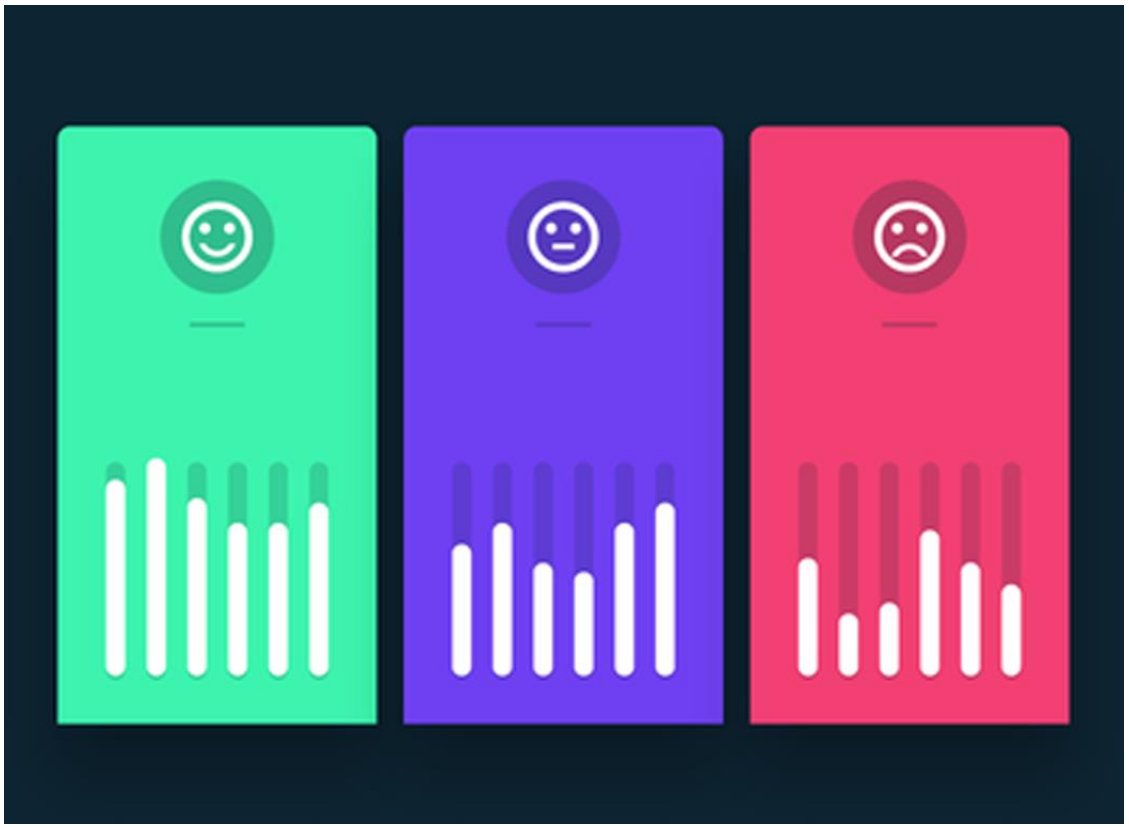
- We can also make the loss more explicit
- If we have a ground truth label at every time step, we can compute individual loss at every time step

RNN: Computational Graph: Many to Many



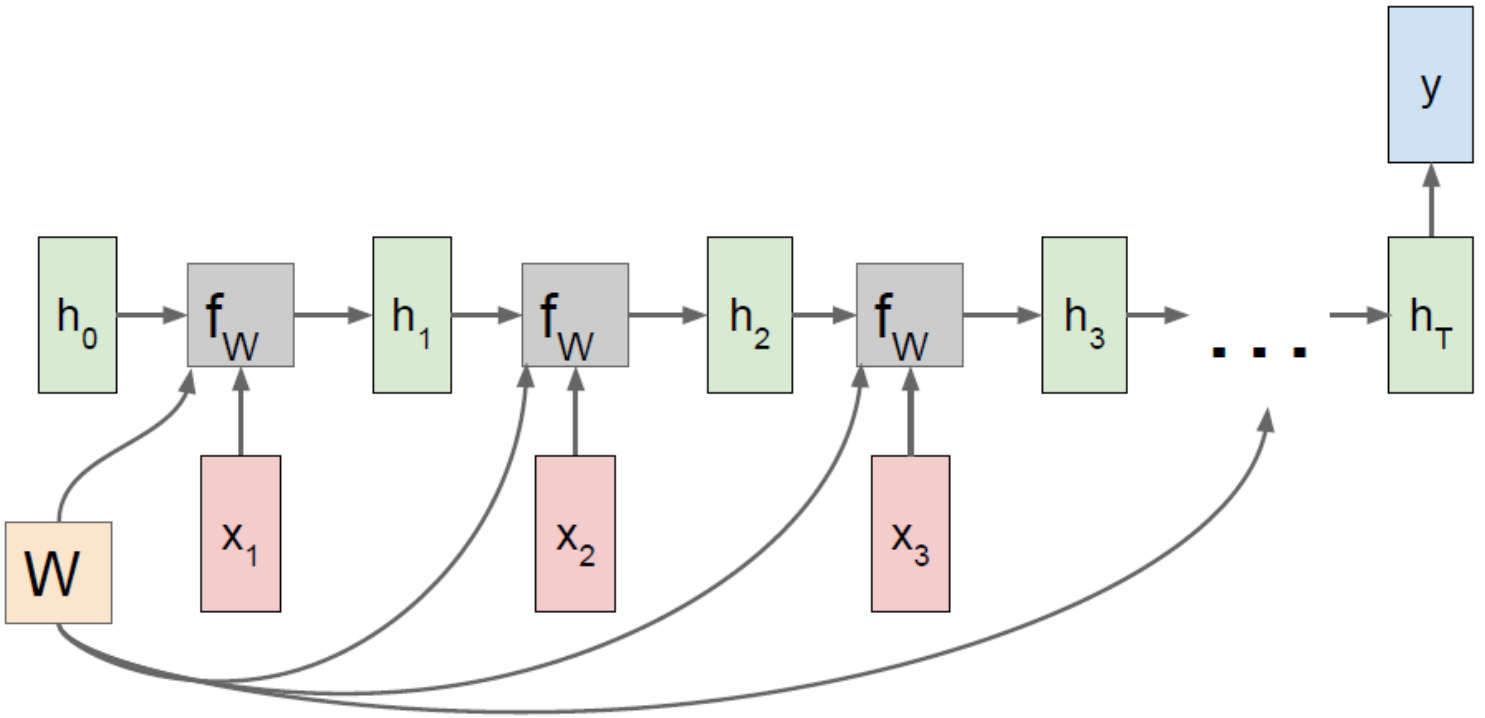
- The loss will be the sum of all losses
- We need to compute the gradient of the Loss with respect to W (to backpropagate)
- Each time-step will calculate a local gradient on the weight and this will be summed up

RNN – many to one



Sentiment analysis from text

RNN: Computational Graph: Many to One



- In a case such as sentiment analysis we make the decision depending on the final hidden state
- Final hidden state summarizes the entire sequence

RNN – One to many

Image Captioning: Microsoft COCO Dataset

The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.

- The captions in the training data is written by humans.
- The training phase involves finding the parameters in the model that maximizes the probability of captions given the image in the training set.

RNN – One to many

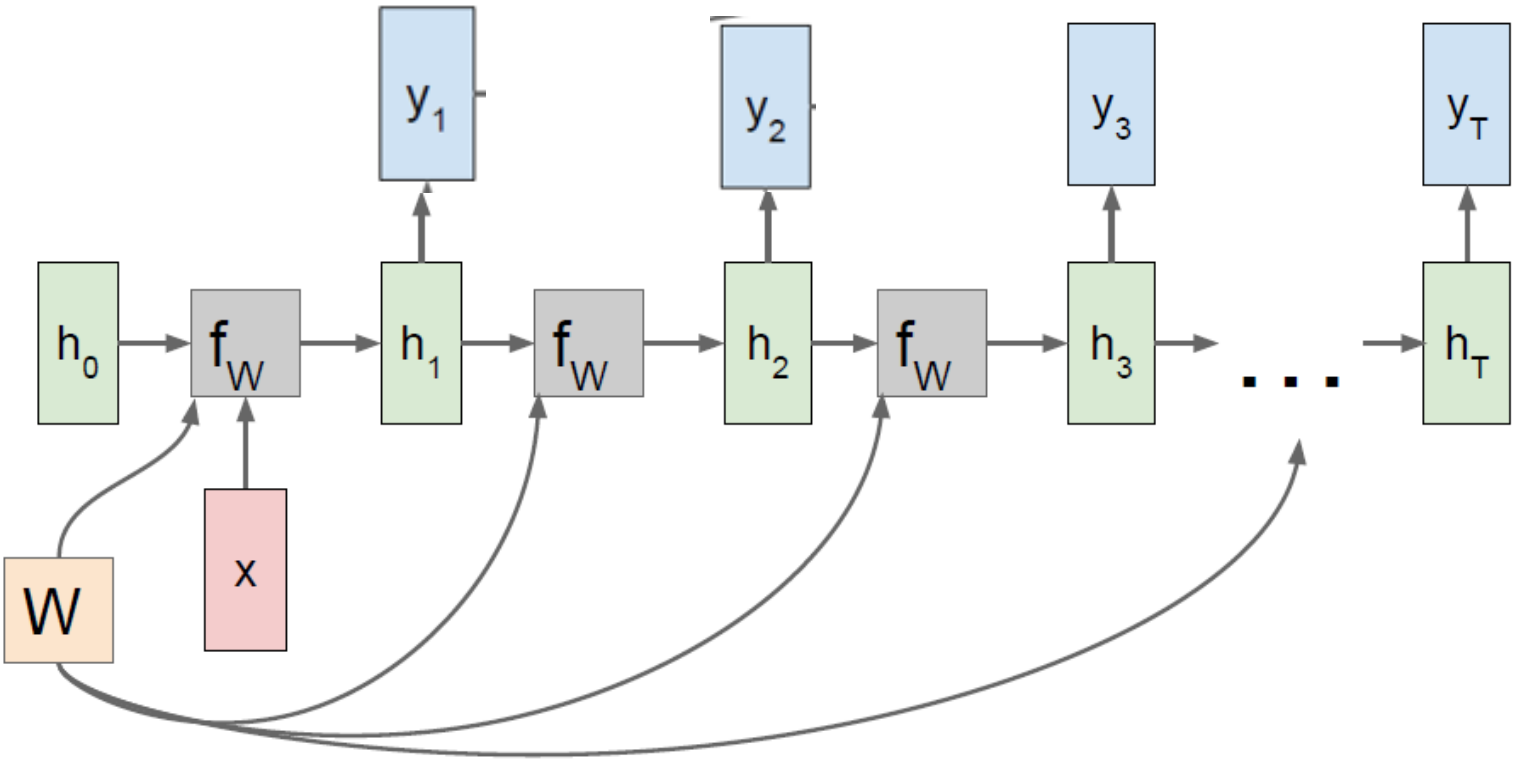
Image Captioning: Microsoft COCO Dataset

The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.

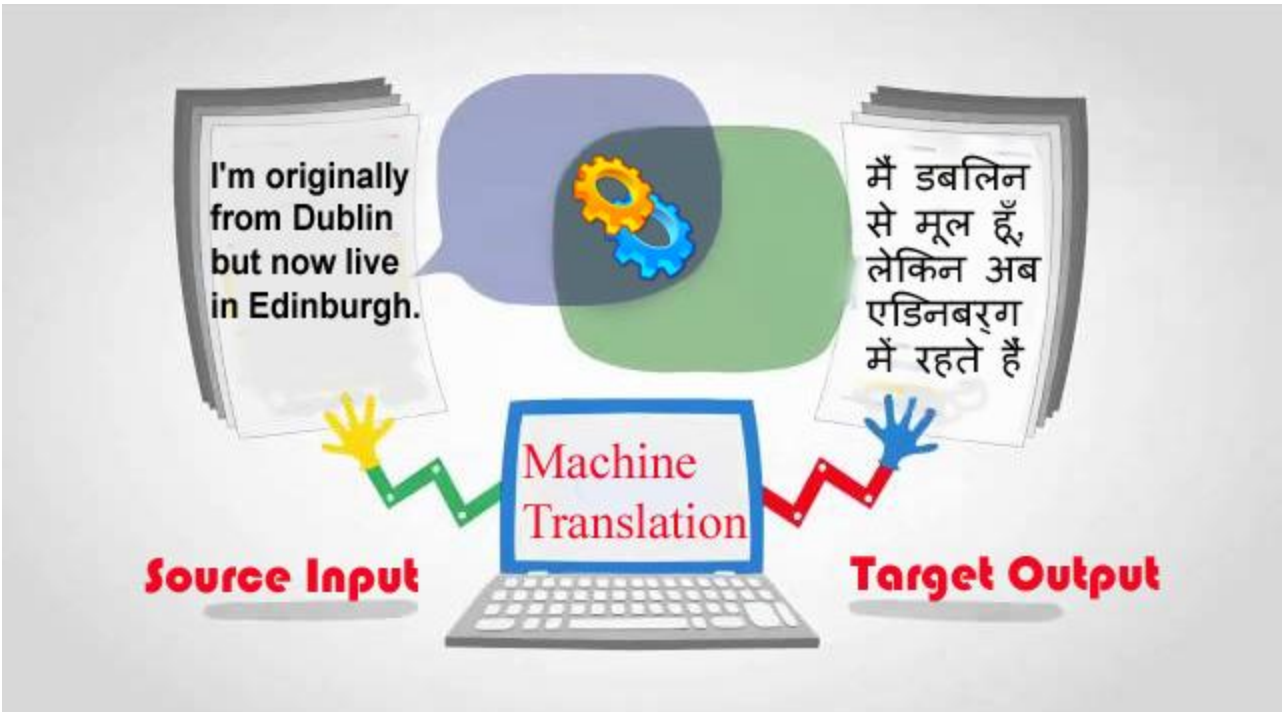
- The probability of a word depends on the previously generated words and the image
- Hence we can condition on these variables in the equation.

RNN: Computational Graph: One to Many



- Fixed sized input -> variable size output
- One thing to note in Image captioning RNN is that the image features from a trained convolution network is transferred to the first hidden **state** as an input.
- We unroll the graph for each element of the output

RNN – Many to many



Machine Translation

Architecture: simple RNN

Machine Translation

- Two stages: encoder (many-to-one type) and decoder (one-to-many type).
- The encoder receives a variable size input (sentence in one language).
- Then summarizes that entire sentence using the final hidden state of the encoder network.

Architecture: simple RNN

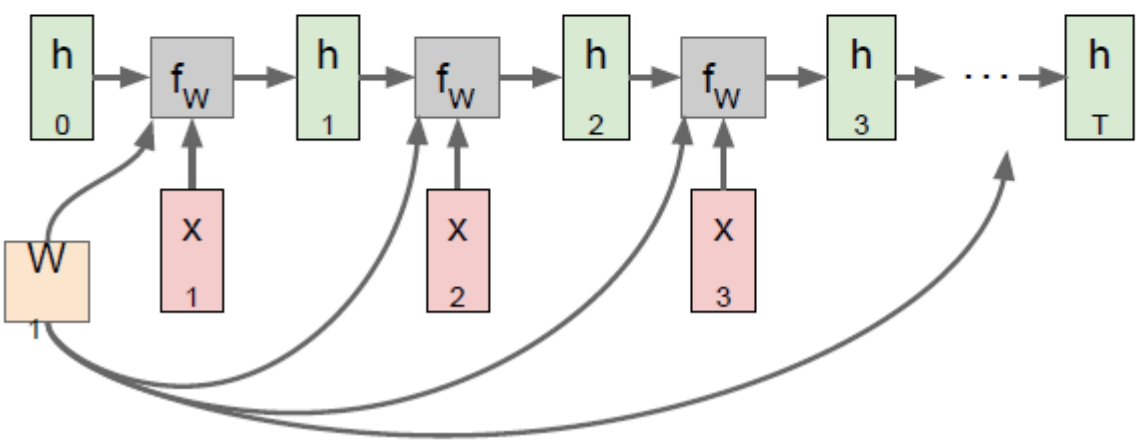
Machine Translation

- Decoder network gets a single vector summarizing the input sentence and produces variable size output (sentence in another language).
- In this variable size output we might make some predictions at every time step about what word to use.
- Train by unrolling this computational graph summing the losses at the output sequence and performing propagation as usual.

Architecture: simple RNN

Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

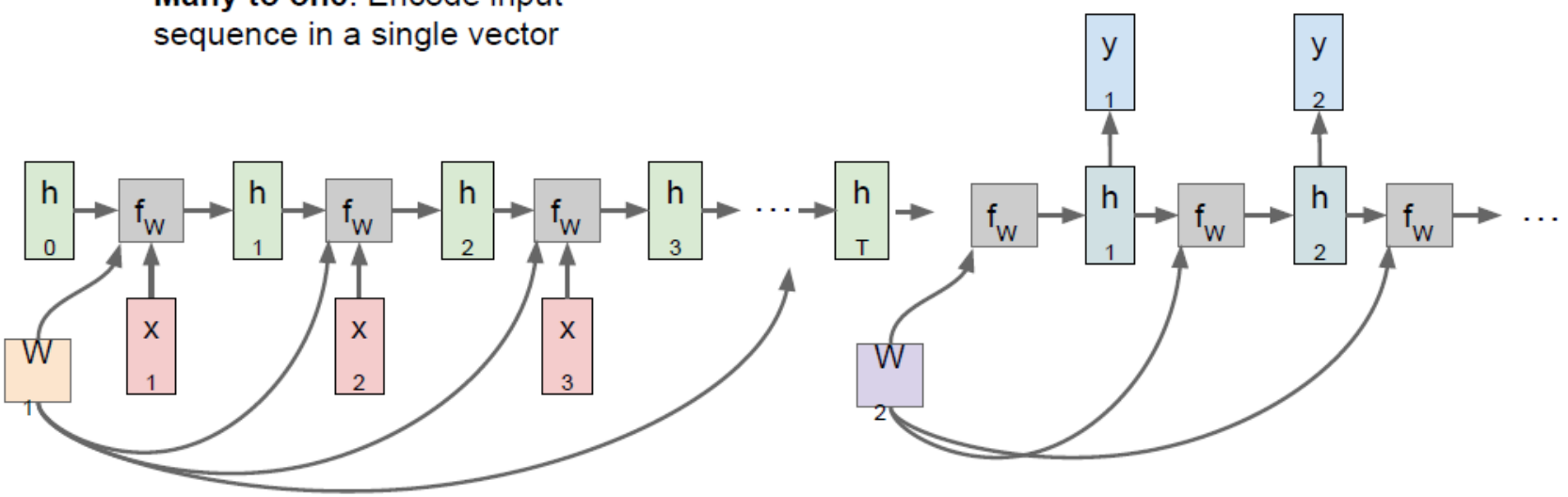


Architecture: simple RNN

Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

One to many: Produce output sequence from single input vector



Architecture: simple RNN

Character Level Language Model

- The network will get some sequence of characters and then will predict what will the next character be in this stream of text

- In this example we have this very small vocabulary of four letters H,E,L and O and we have training sequence of the word “h-e-l-l-o”

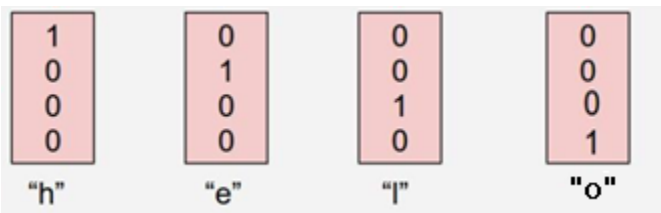
- During training we'll feed the characters of our training sequence these will be the X_t that we feed in as the inputs to our current neural network

Architecture: simple RNN

Character Level Language Model

- Each of these inputs is a letter - we need to find a way to represent letters in our network

- Our vocabulary has four elements and each letter will be represented by a vector that has zeros in every slot but one (one-hot encoding)

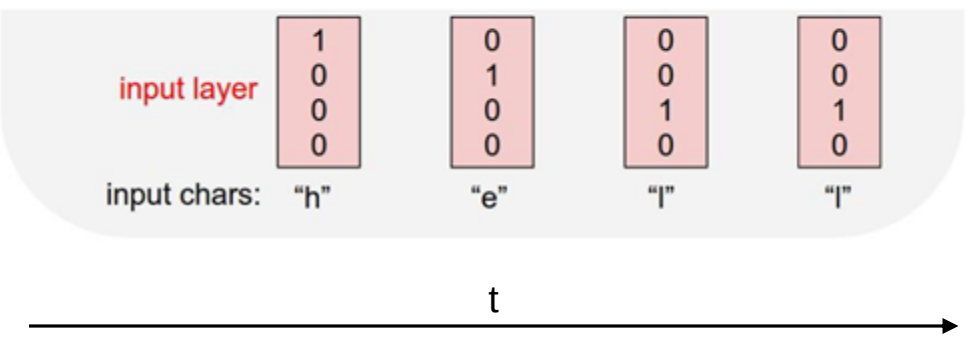


Character-Level Language Model

- We will feed a character at a time to the network and ask it to predict the next character
- So will predict an entire distribution for what it thinks should come next in the sequence that it has seen so far

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

Character-Level Language Model Training

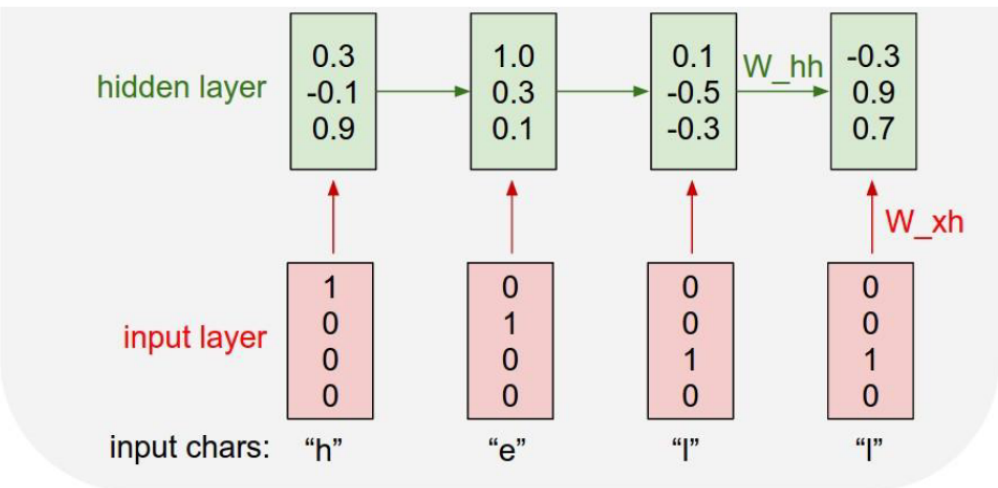
- hidden state vector at every single time step is calculated using this fixed recurrence formula

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

Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



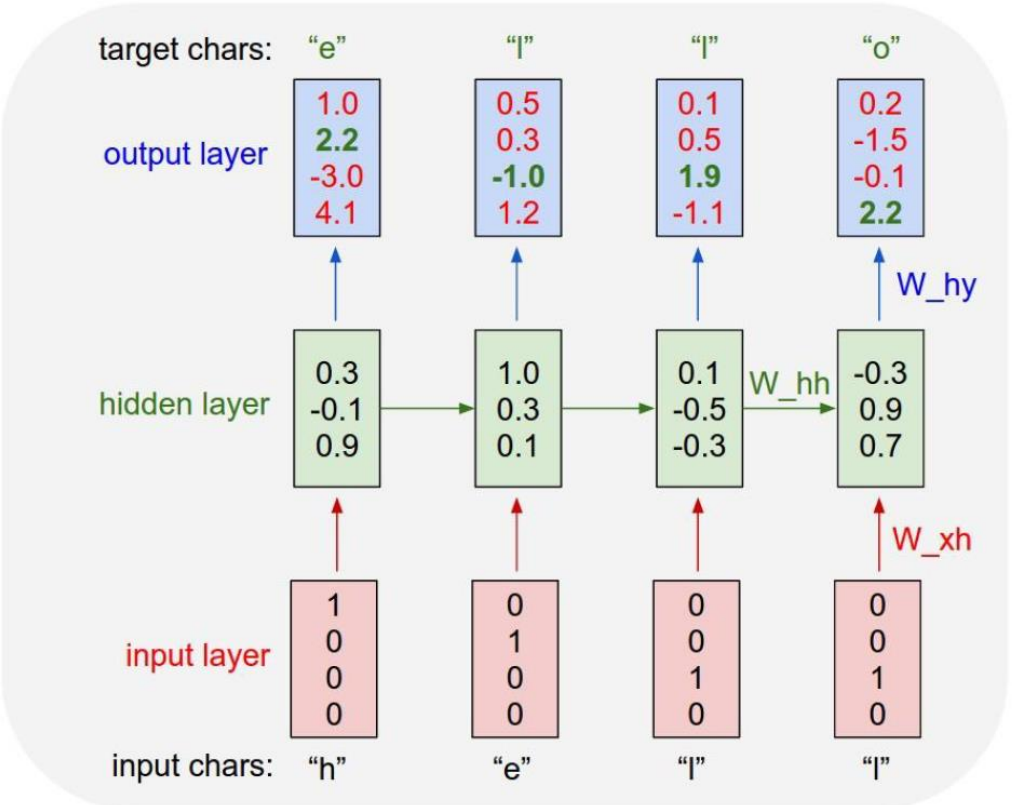
Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

Character-Level Language Model Training

Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



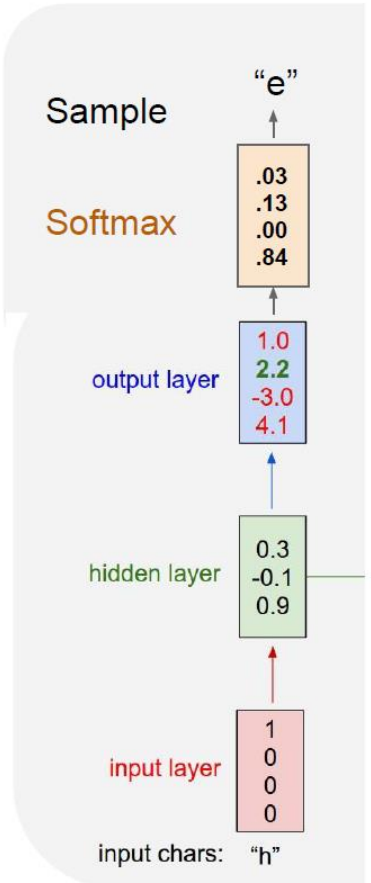
Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

Character-Level Language Model Testing

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



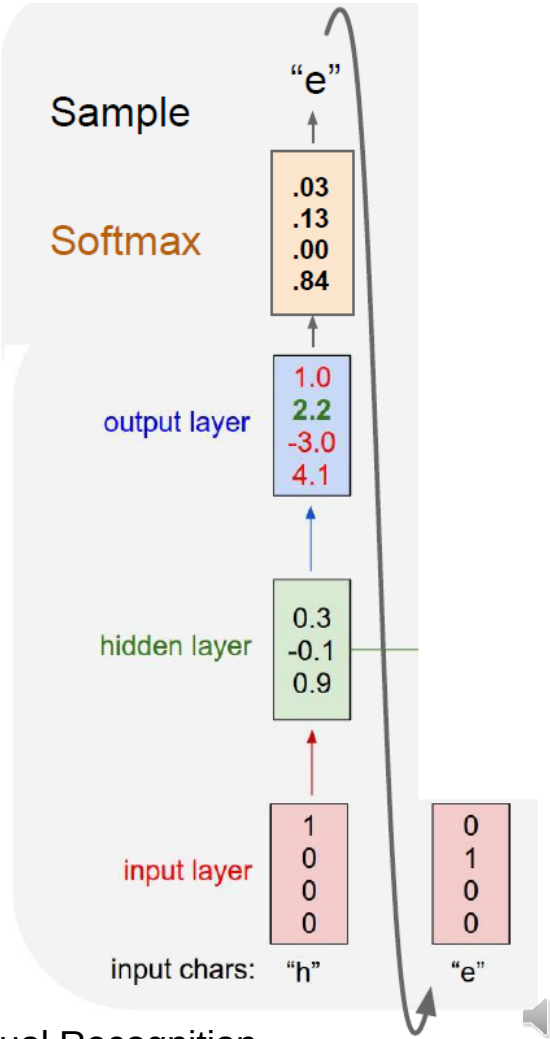
Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

Character-Level Language Model
Testing

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model

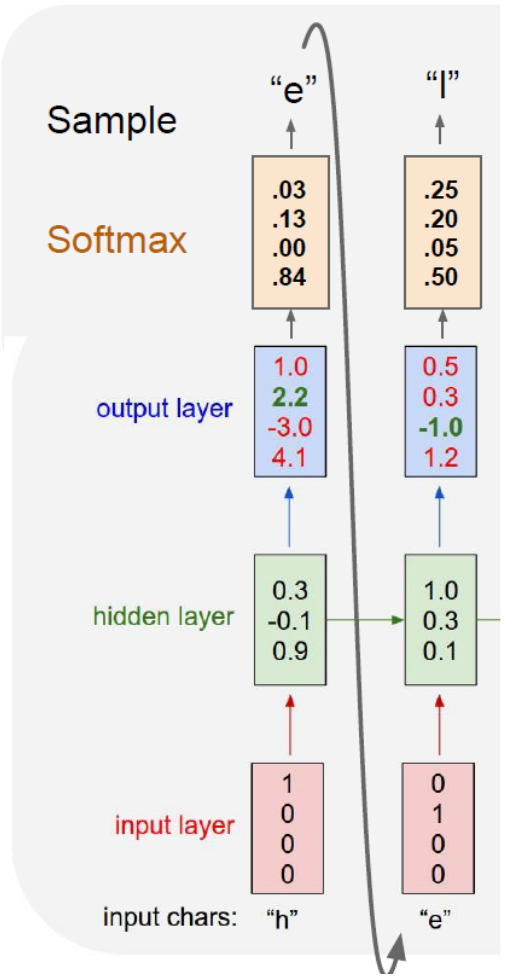


Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model

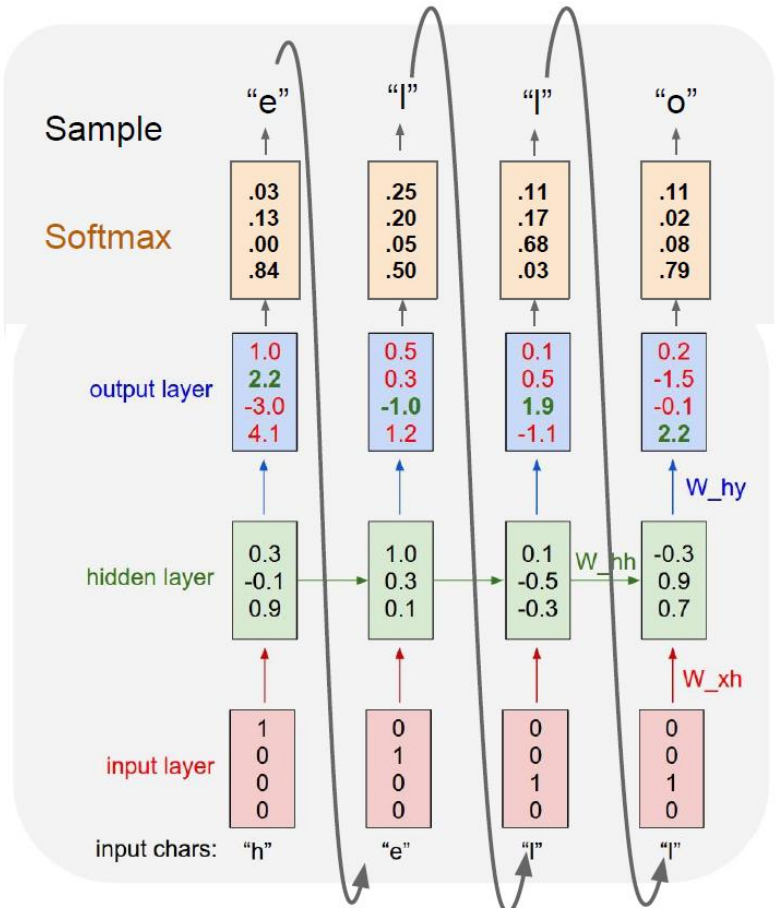


Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

Character-Level Language Model – Some Notes

- In this particular example, we do not select the letter with the highest probability. Rather, we sample the letters from the probability distribution.
- The example shows a "lucky" case where the system outputs "hello", though it might have been a different word as it involves sampling.
- If you do sampling you will have a variety of outputs every time you run the system with the same seed letter.
- You could also select the letter with the highest probability (argmax). This would also work, but in that case the outcome would be the same every time you run the system with the same seed letter.

RNNs

- Q: Does order of the characters fed in matter?
- Yes, because it is a function on everything comes before that.

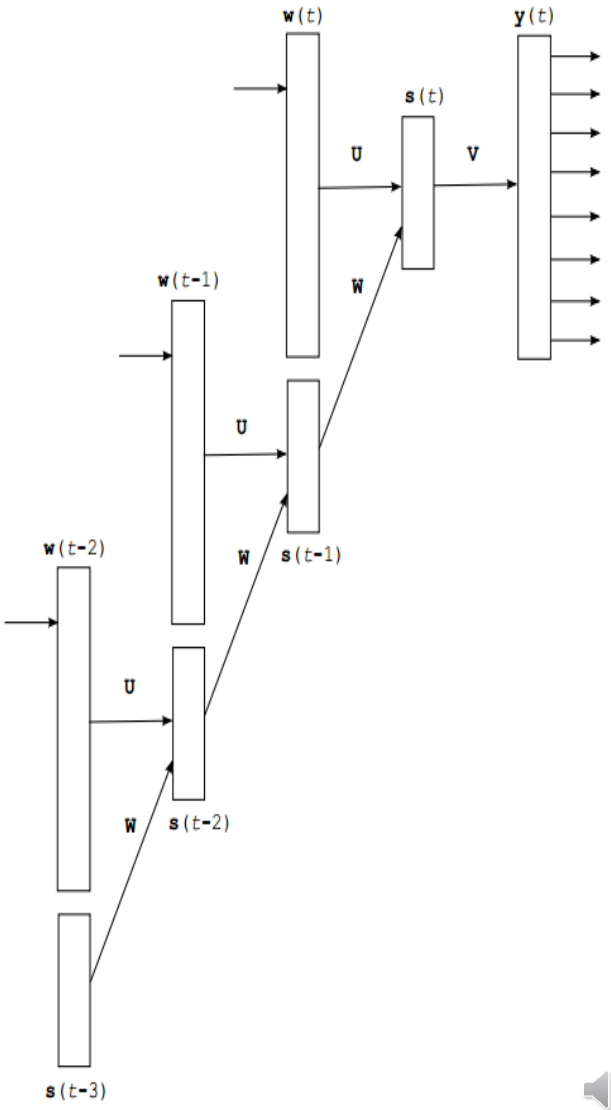
- Q: How to initialize the first h_0?
- Setting it to zero is quite common,

Training: Backpropagation through time (BPTT)

- How to train the recurrent nets?
- The output value depends on the state of the hidden layer, which depends on all previous states of the hidden layer (and thus, all previous inputs)
- Recurrent net can be seen as a (very deep) feedforward net with shared weights

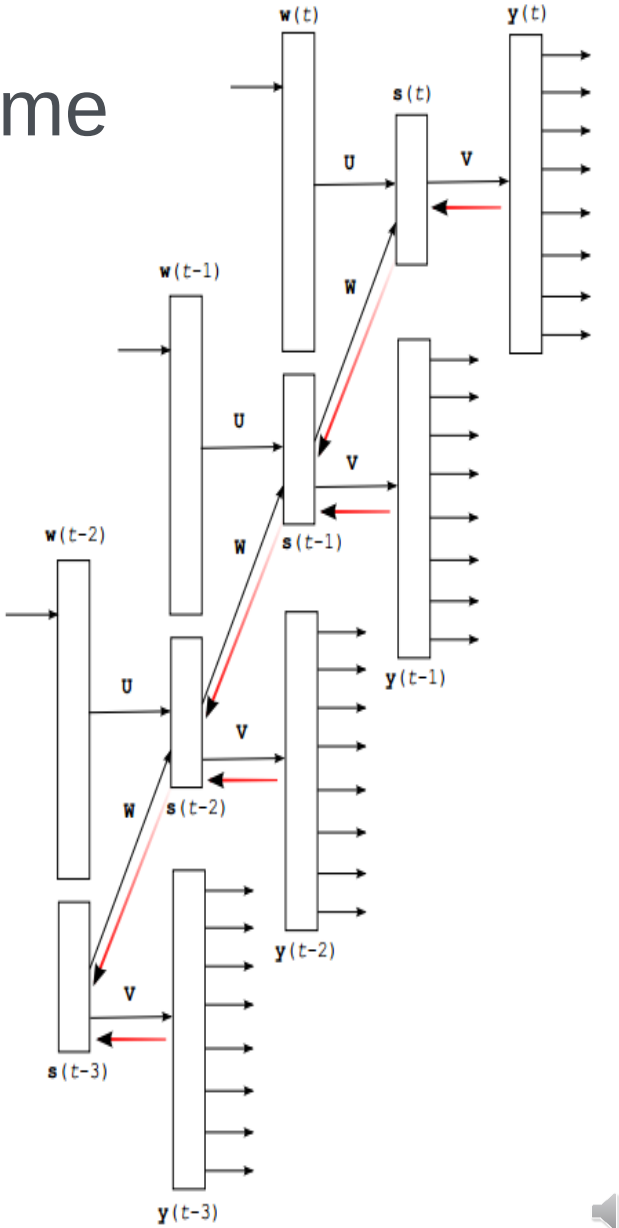
Backpropagation Through Time (BPTT)

- The intuition is that we unfold the RNN in time
- We obtain deep neural network with shared weights **U** and **W**
- But this becomes problematic if we train with long sequences
- Unfolding for several steps is often enough (*Truncated BPTT*)



Backpropagation through time

- We train the unfolded RNN using normal backpropagation + SGD
- In practice, we limit the number of unfolding steps to 5 – 10
- It is computationally more efficient to propagate gradients after few training examples (batch training)



Character-level language model in Python/numpy

Minimal character-level language model with a Vanilla Recurrent Neural Network, in Python/numpy:

https://gist.github.com/karpathy/d4dee566867f8291f086

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

Deep RNNs

– By carefully analyzing and understanding the architecture of an RNN, however, we find three points of an RNN which may be made deeper;

(1) input-to-hidden function

(2) hidden-to-hidden transition

(3) hidden-to-output function.

– Based on this observation, we propose two novel architectures of a deep RNN which are orthogonal to an earlier attempt of stacking multiple recurrent layers to build a deep RNN (Schmidhuber, 1992; El Hihi and Bengio, 1996).

Pascanu, R., Gulcehre, C., Cho, K. and Bengio, Y., 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Deep RNNs

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

$h \in \mathbb{R}^n$ W^l $[n \times 2n]$

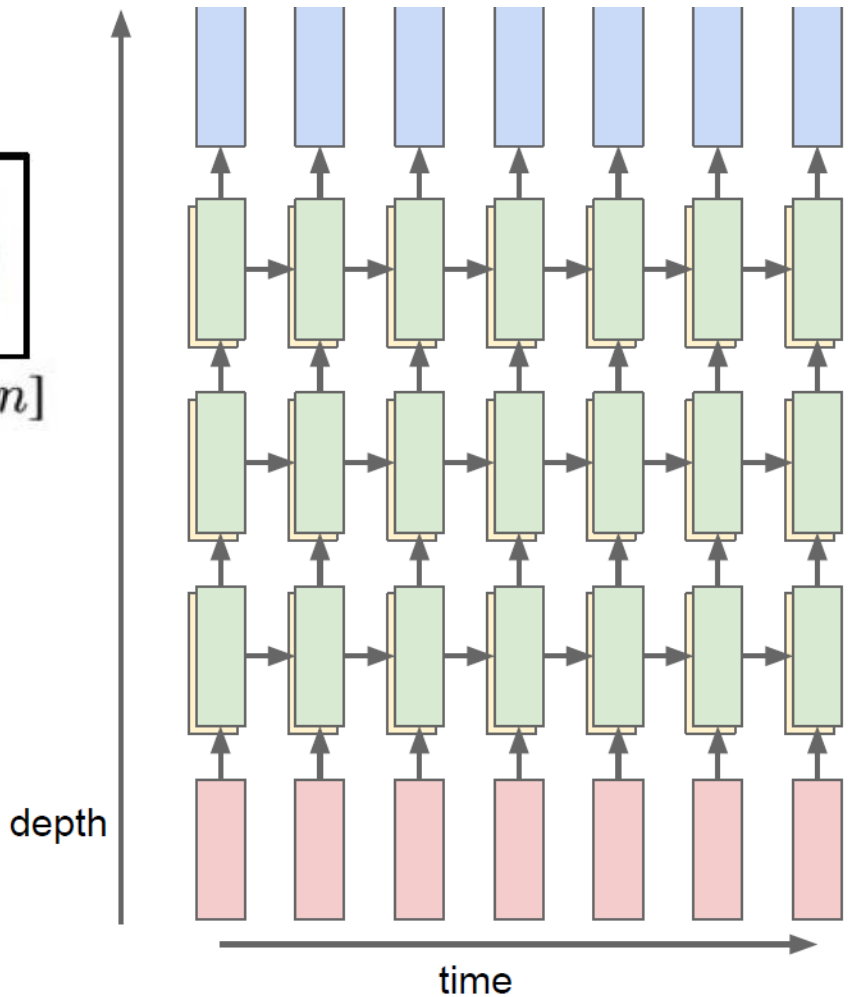
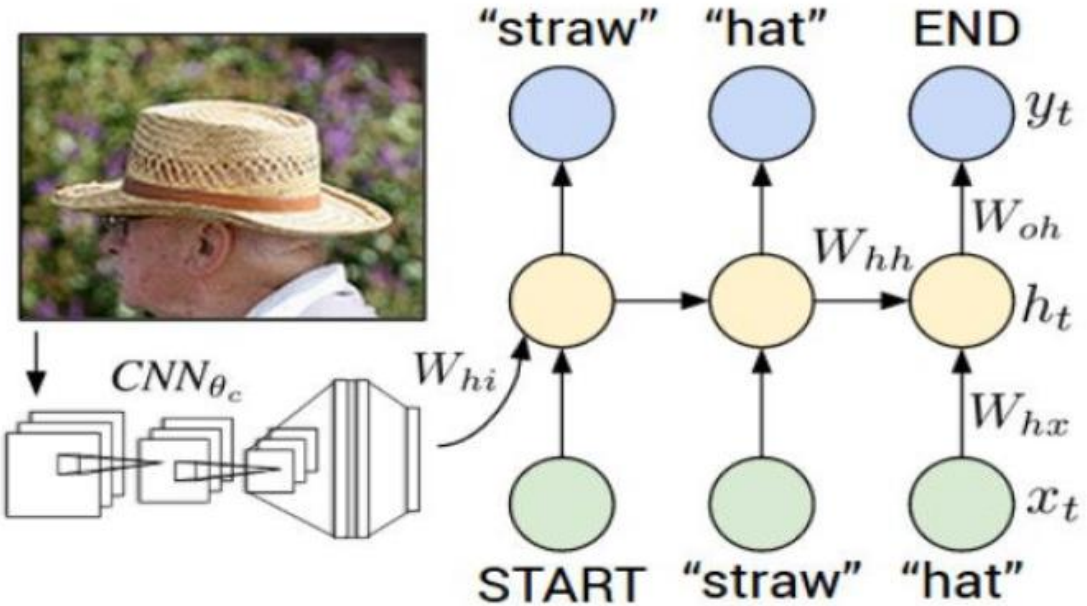


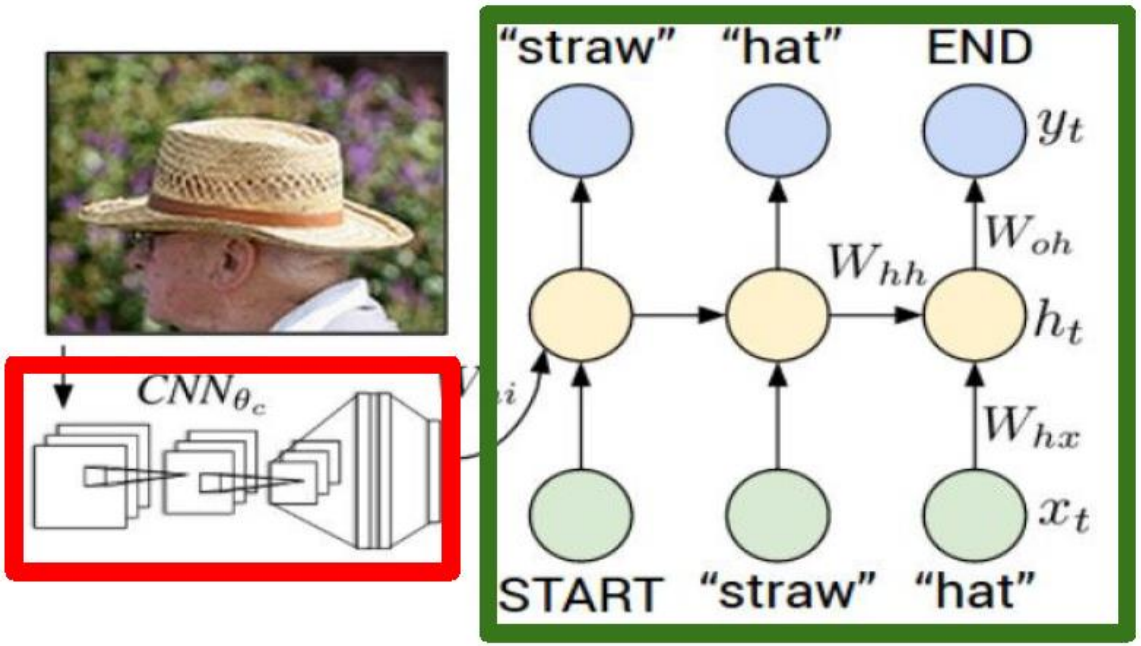
Image Captioning



- Aim: Take a single image and describe with a sequence of words
- 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
- W_{hi}, W_{hx}, W_{hh}, W_{oh} are learnable parameters

Karpathy et al., "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015.

Recurrent Neural Network

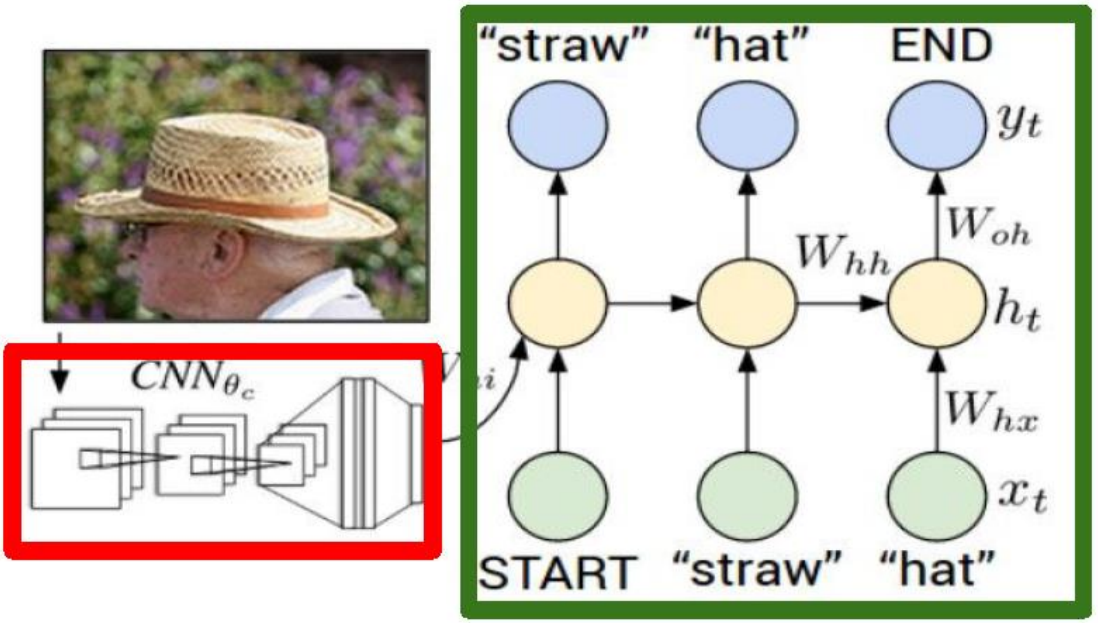


Convolutional Neural Network

- CNN is processing the image -> In this example VGGNet
- We're only plugging in the image at the very first time step
- At the output we have a softmax giving a probability distribution over 1000 categories of Imagenet

Karpathy et al., "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015.

Recurrent Neural Network



Convolutional Neural Network

- RNN will model the sequences
- We backpropagate through the the whole framework as a single model and train all jointly.
- Use a CNN pre-trained with Imagenet and start with these weights
- Then train everything jointly to get a caption (so we can find out features that are better at describing the images at the end)
- In practice we train on image sentence datasets (Microsoft COCO)

Karpathy et al., "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015.

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

FC-1000

softmax

X

test image

image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

FC-4096
FC-4096

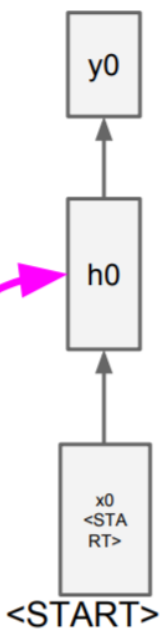
4096 dimensional vector describing the image

test image

x0 <START>

<START>

test image



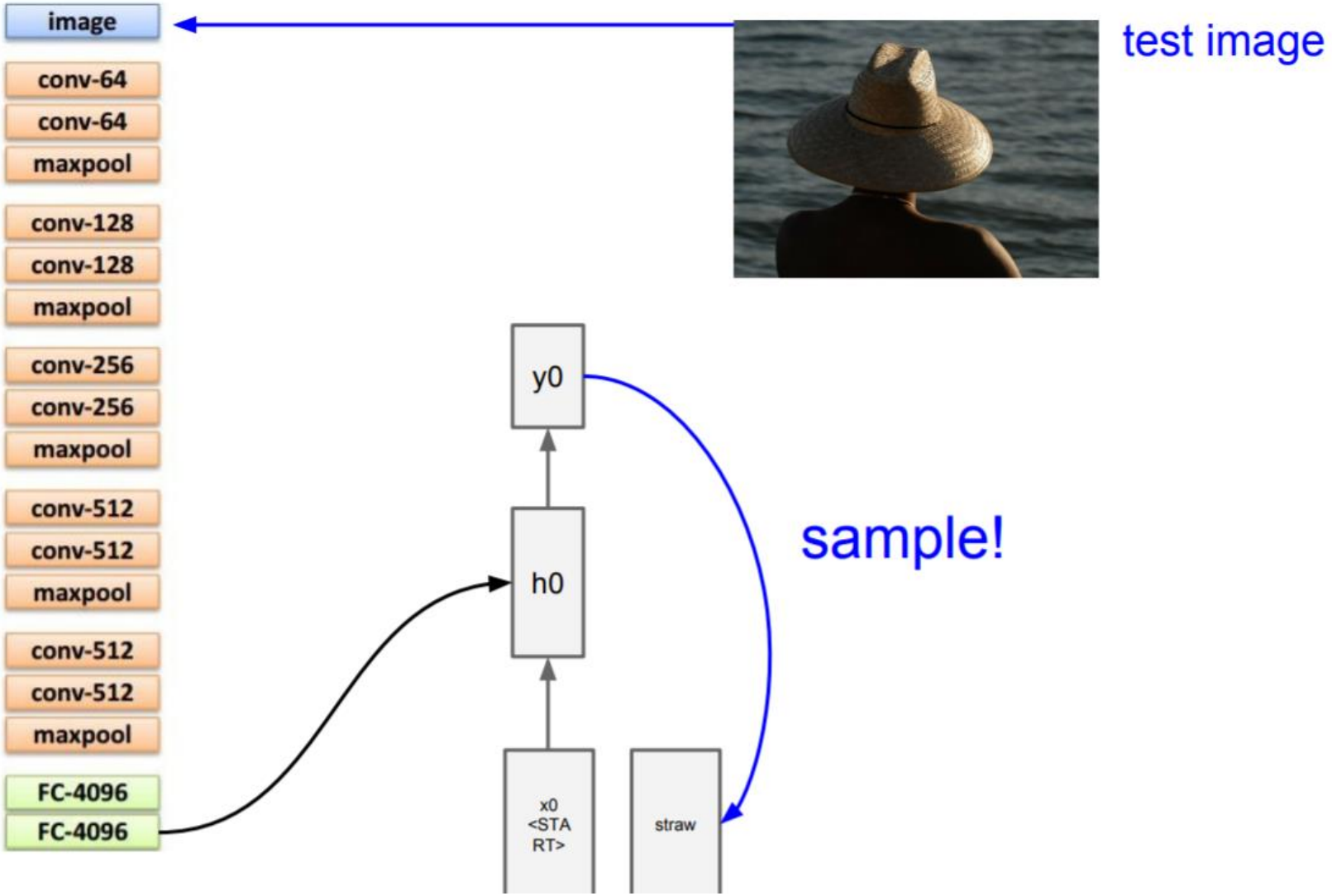
Wih

before:

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

now:

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



Sample over all words in the vocabulary (typically quite large)

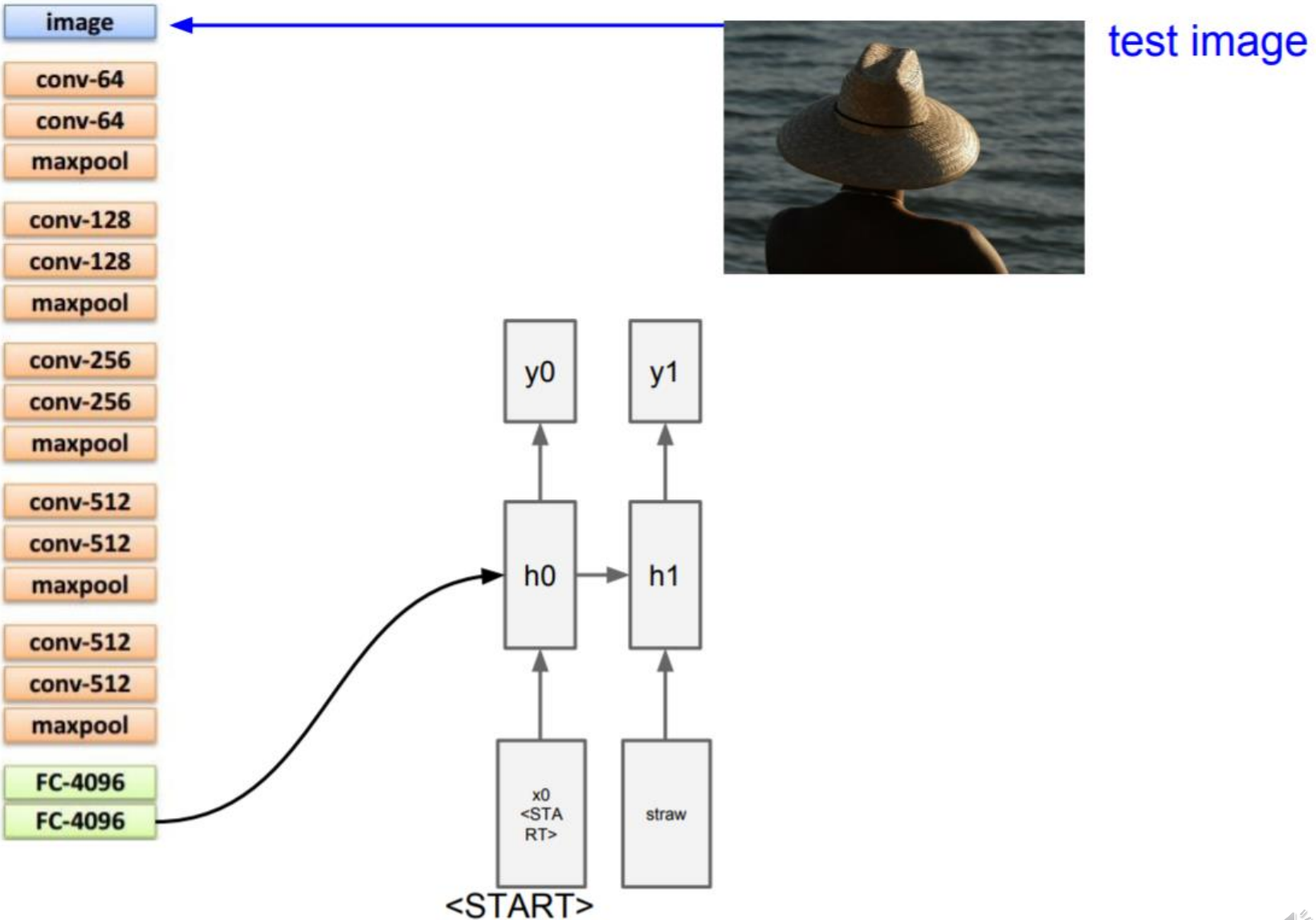
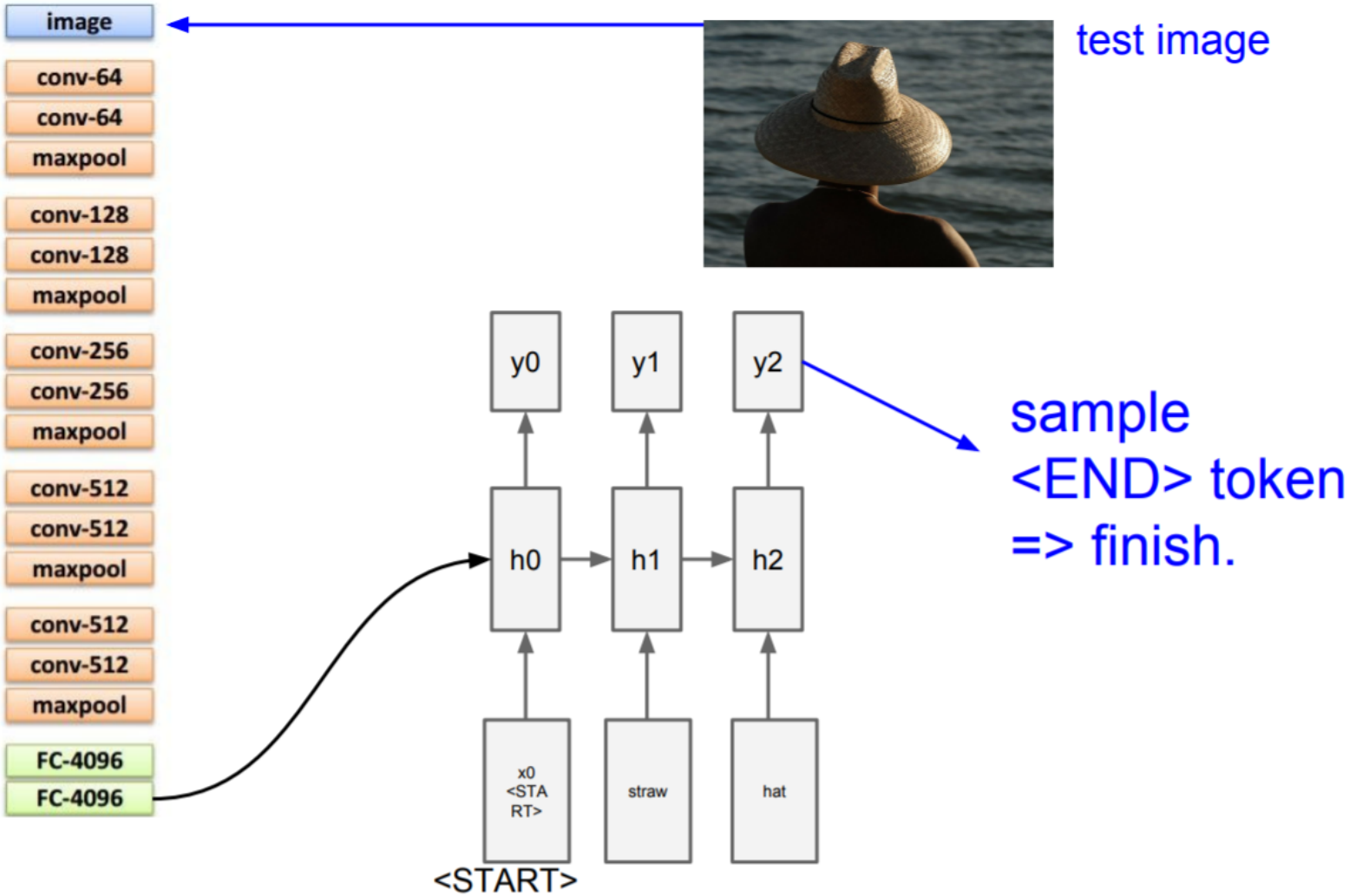


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
x0
<STA
RT>
straw
<START>



- At the training time we also feed captions with END token, so the network learns to put END token when finished and stop generating
- During training the whole network is trained together, i.e. backpropagates through RNN and then CNN

Example Results

A cat sitting on a suitcase on the floor

A woman is holding a cat in her hand

Two people walking on the beach with surfboards

A person holding a computer mouse on a desk

Image Captioning with Attention

Different colors show a correspondence between attended regions and underlined words, i.e. where the network focuses its attention in generating the prediction.

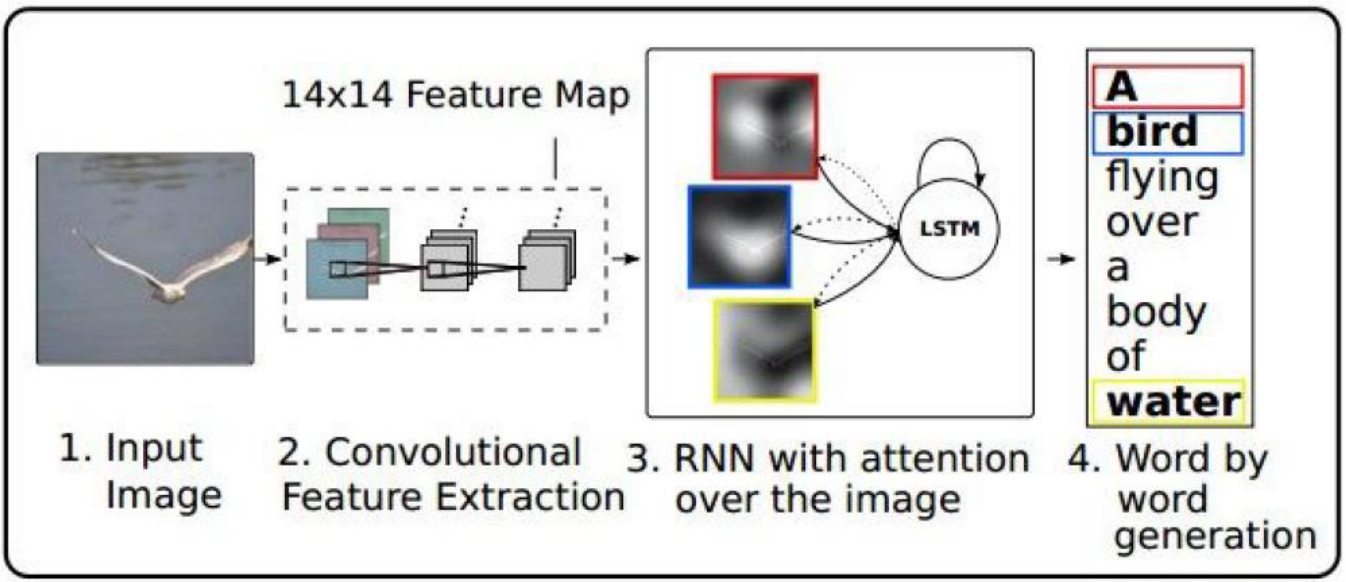
- Attention: The ability **to weight regions of the image differently**.
- Rather than summing up the image as a whole, with attention the network can add more weight to the ‘salient’ parts of the image.
- Additionally, for each output word the network can recompute its attention to focus on a different part of the image.

Image Captioning with Attention

- In the previous model we only feed in the image a single time at the beginning
- Idea: Make RNN to look back to the image and reference parts of the image while it's describing the words
- While generating every single word, RNN looks back to the image and looks for different features to find out what to describe next
- You can actually do this in a fully trainable way so the RNN not only creates these words **but also decides where to look next in the image**

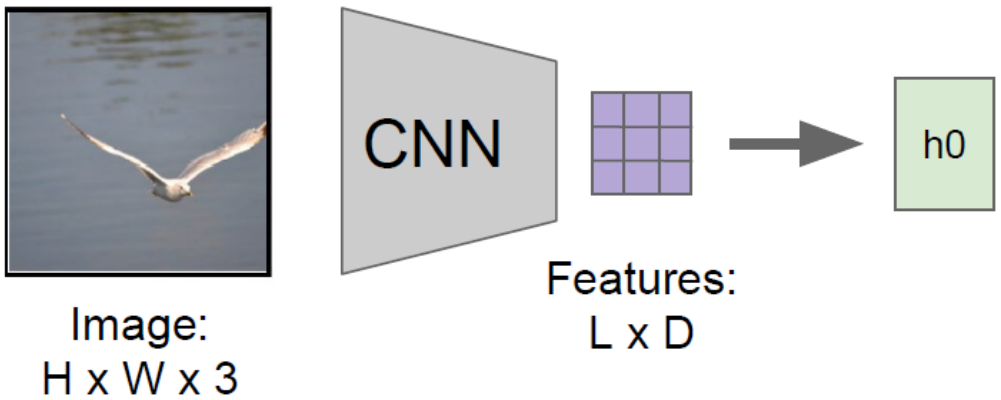
Image Captioning with Attention

RNN focuses its attention at a different spatial location when generating each word



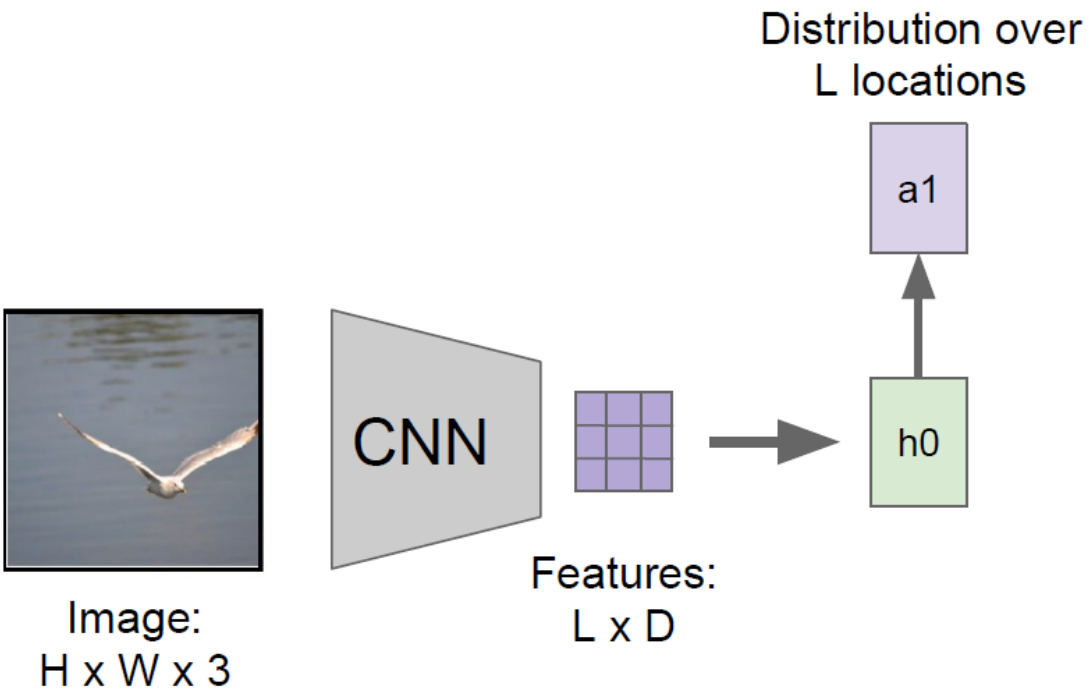
Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



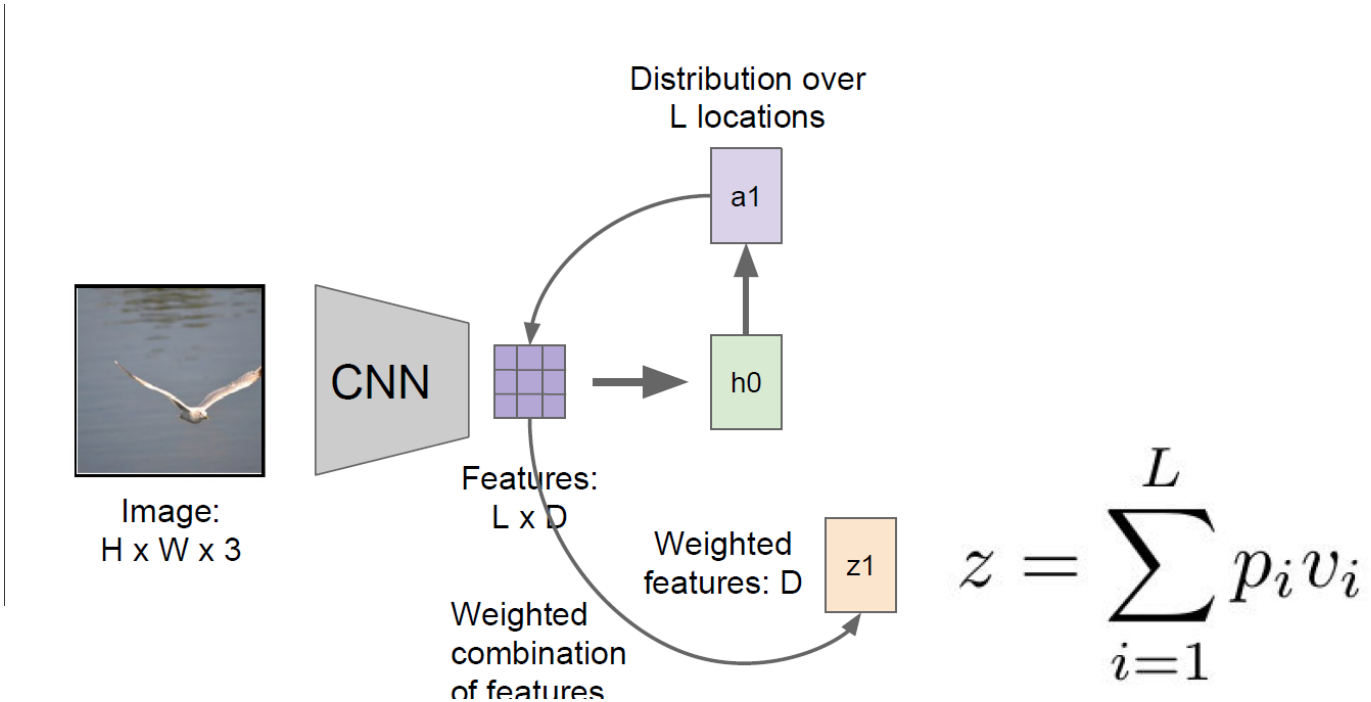
- CNN now produces a separate vector for each spatial location in the image rather than a single vector describing the whole image
- The first hidden state computes this distribution over image locations to give a single summary vector that focuses the attention on one part of that image
- This summary vector gets fed as an as an additional input at the next time step of the neural network

Image Captioning with Attention



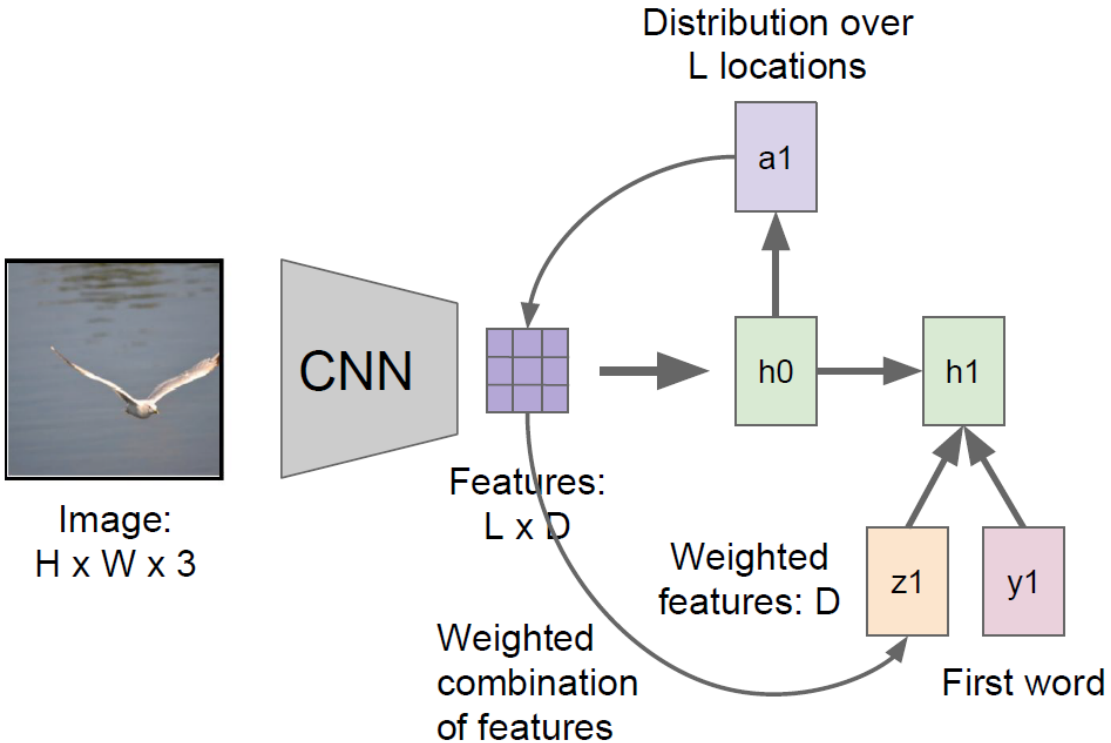
- When this model runs forward, in addition to sampling the vocabulary at every time step it also produces a distribution over the locations in the image where it wants to look
- This distribution over image locations can be seen as a kind of attention of where the model should look during training

Image Captioning with Attention



- This summary vector gets fed as an as an additional input at the next time step of the neural network

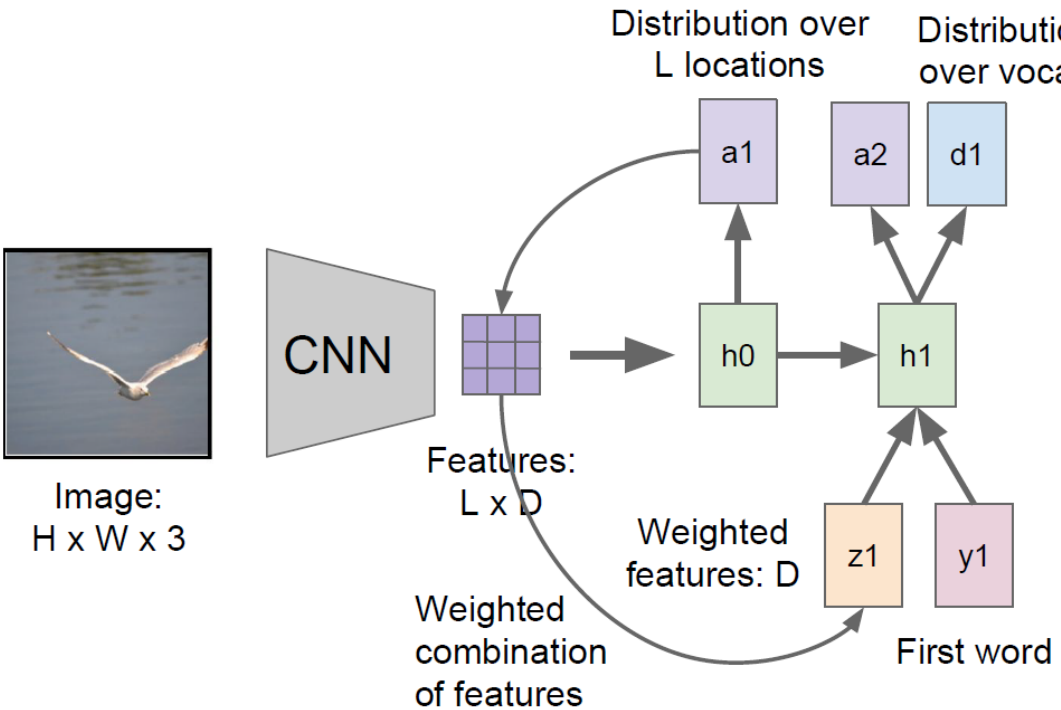
Image Captioning with Attention



Again it will produce two outputs:

- Distribution over image locations
- Distribution over vocabulary words

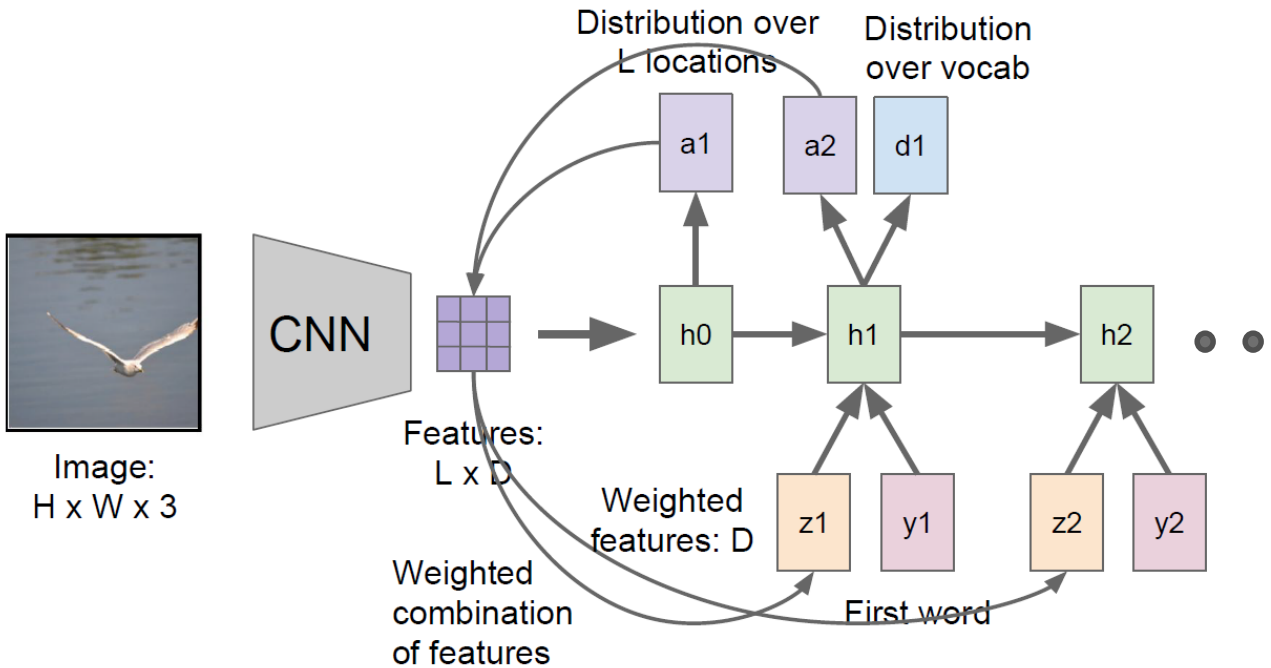
Image Captioning with Attention



Again it will produce two outputs:

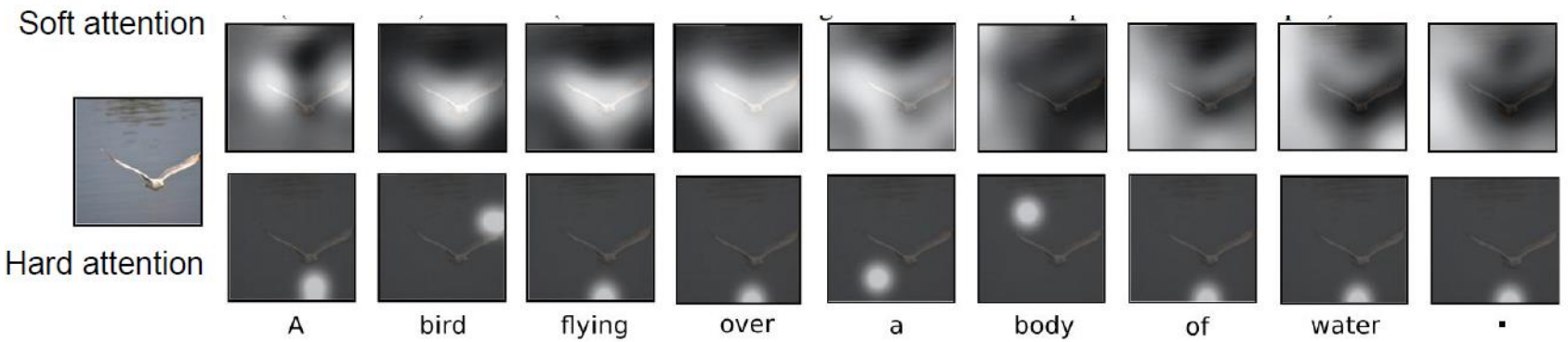
- Distribution over image locations
- Distribution over vocabulary words

Image Captioning with Attention



- This whole process will continue to do these two different things at every time step
- After we train the model, it will shift its attention around the image for every word that it generates in the caption

Image Captioning with Attention



Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

- Soft attention (Top row): taking a weighted combination of features from all image locations
- Hard attention (Bottom row): the model is forced to select one location to look at in the image at each time step

Natural Language Processing (NLP)

- Natural Language Processing (NLP) key terms: https://www.kdnuggets.com/2017/02/natural-language-processing-key-terms-explained.html
- Statistical language modeling is one of the oldest and most important NLP tasks
- The language models are core of machine translation, speech recognition and many other applications
- Historically, it was difficult to beat N-grams

Natural Language Processing (NLP)

How do we prepare text data in NLP before feeding into training?

- We first tokenize the data: Splitting strings of text into smaller pieces (tokens).
  - For example text is tokenized into sentences and sentences are tokenized into words.

- Then we normalize, i.e. make all the text represented similarly such as by converting all letters to lowercase, removing punctuation, converting numbers into text. This step might also include stemming and lemmatization.

- Finally, if the data is noisy, we do noise removal.

Natural Language Processing (NLP)

- A corpus (plural *corpora*) is a large and structured set of texts

- WordNet (Fellbaum, 1998) is a lexical database that stores words and relationships among them such as synonymy (when two words can mean the same thing) and hyponymy (when one word's meaning is a more specific case of another's).

- WordNet also explicitly captures the different senses of words that take multiple meanings, such as fan (a machine for blowing air, or someone who is supportive of a sports team or celebrity).

Language modeling: example

A B C A B C A B _

– What symbol comes next?
– What is its probability?

Yesterday it was Sunday, so today it must be _

– How to predict the next word?

N-grams

– An n-gram is a contiguous sequence of *n* items (such as consecutive words) from a given linguistic sequence;

– Particular versions: unigram, bigram, trigram, four-gram or five-gram.

– A skip-gram simply drops items from the n-gram.

N-grams

– Task: compute probability of a sentence W

$$P(W) = \prod_i P(w_i | w_1 \dots w_{i-1})$$

– Often simplified to trigrams:

$$P(W) = \prod_i P(w_i | w_{i-2,} w_{i-1})$$

– The probabilities are computed using counting on training data

Language modeling with neural networks

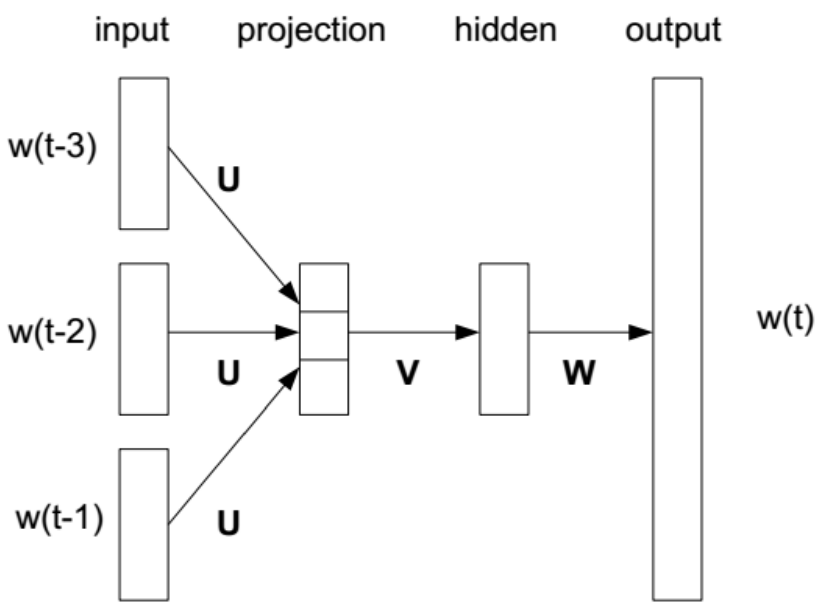
- Main deficiency of N-grams is the exponential growth of number of parameters with the length of the context
- Neural networks address this problem by performing dimensionality reduction and parameter sharing

Language modeling with neural networks

- Neural network language models are today state of the art, often applied to systems participating in competitions (ASR, MT)
- Two main types of neural network architectures for language modeling: feedforward and recurrent

Feedforward neural network language model

- Proposed by Bengio et al, 2003
- The projection layer is linear
- The hidden layer is non-linear
- Softmax at the output computes probability distribution over the whole vocabulary
- The basic architecture is computationally very expensive



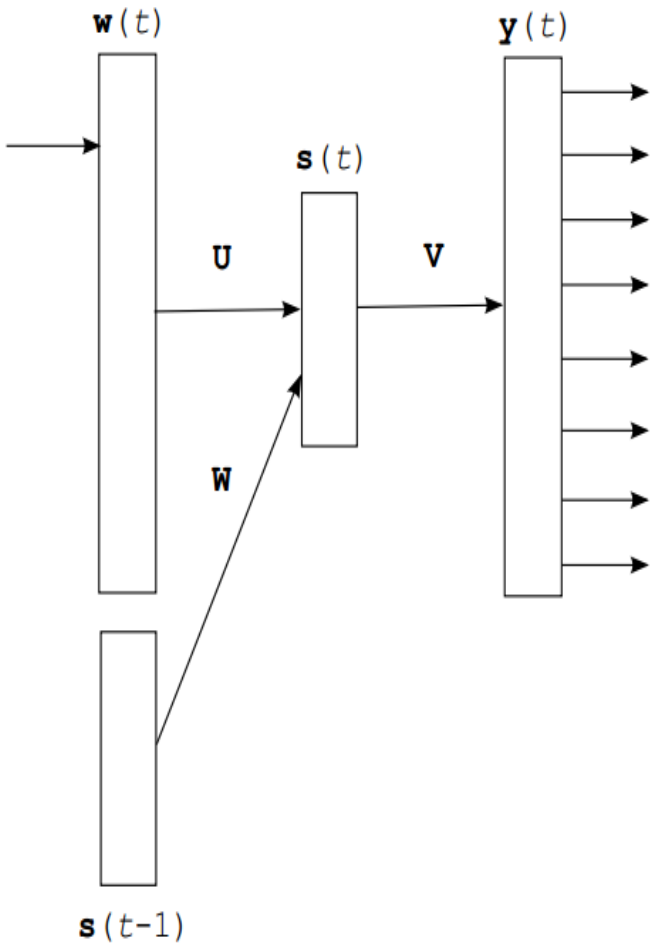
RNN language model

$$\mathbf{s}(t) = f\left(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t\text{-}1)\right)$$

$$\mathbf{y}(t) = g(\mathbf{V}\mathbf{s}(t))$$

$$f(z) = \frac{1}{1 + e^{-z}}$$

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$



Comparison of performance on small data: Penn Treebank

- Small, standard dataset, ~1M words
- In information theory, “perplexity” is a measurement of how well a probability distribution or probability model predicts a sample.
- It may be used to compare probability models.
- A low perplexity indicates the probability distribution is good at predicting the sample.

Model	Perplexity
Kneser-Ney 5-gram	141
Maxent 5-gram	142
Random forest	132
Feedforward NNLM	140
Recurrent NNLM	**125**

Word2Vec

- Word2vec is a two-layer neural net (not a deep network) that processes text.
- Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus.
- These vectors are also known as *word embeddings*
- It turns text into a numerical form that deep nets can understand.

Source: https://skymind.ai/wiki/word2vec

Word2Vec

- Word2vec's applications extend beyond parsing sentences.
- It can be applied just as well to genes, code, likes, playlists, social media graphs and other verbal or symbolic series in which patterns may be discerned.
- Because words are simply discrete states like the other data mentioned above, and we are simply looking for the transitional probabilities between those states: the likelihood that they will co-occur.

Source: https://skymind.ai/wiki/word2vec

Word2Vec

- The purpose and usefulness of Word2vec is to group the vectors of similar words together in vector space.

- Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words. It does so without human intervention.

- Given enough data, usage and contexts, Word2vec can make highly accurate guesses about a word's meaning based on past appearances.

Source: https://skymind.ai/wiki/word2vec

Word2Vec

- Those guesses can be used to establish a word's association with other words (e.g. "man" is to "boy" what "woman" is to "girl"), or cluster documents and classify them by topic.

- These clusters can form the basis of search, sentiment analysis and recommendations in such diverse fields as scientific research, legal discovery, e-commerce and customer relationship management.

- The output of the Word2vec neural net is a vocabulary in which each item has a vector attached to it, which can be fed into a deep-learning net or simply queried to detect relationships between words.

Source: https://skymind.ai/wiki/word2vec

Word2Vec

- Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.
- The cosine of 0° is 1, and it is less than 1 for any angle in the interval (0,π] radians.
- It is thus a judgment of orientation (not magnitude): two vectors with the same orientation have a cosine similarity of 1, two vectors oriented at 90° relative to each other have a similarity of 0.

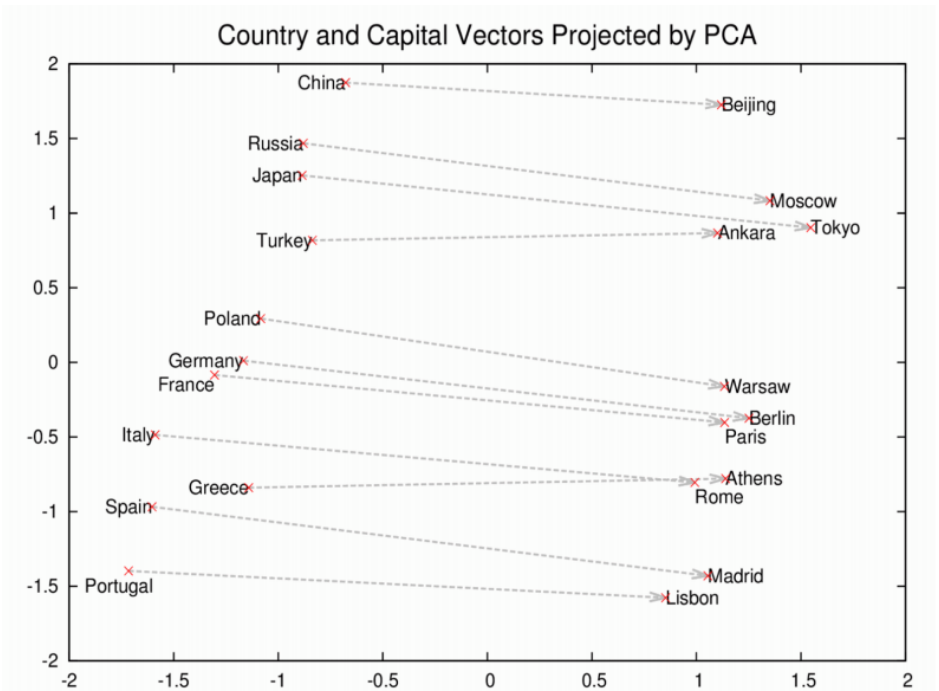
Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

Proximity of words associated with “Sweden” using Word2vec

Source: https://skymind.ai/wiki/word2vec

Word2Vec

- Rome, Paris, Berlin and Beijing cluster near each other
- Also they will each have similar distances in vector space to the countries whose capitals they are
- If you only knew that Rome was the capital of Italy, and were wondering about the capital of China, then you can use the following equation to find out it is Beijing: Rome -Italy + China.



Source: https://skymind.ai/wiki/word2vec

Example

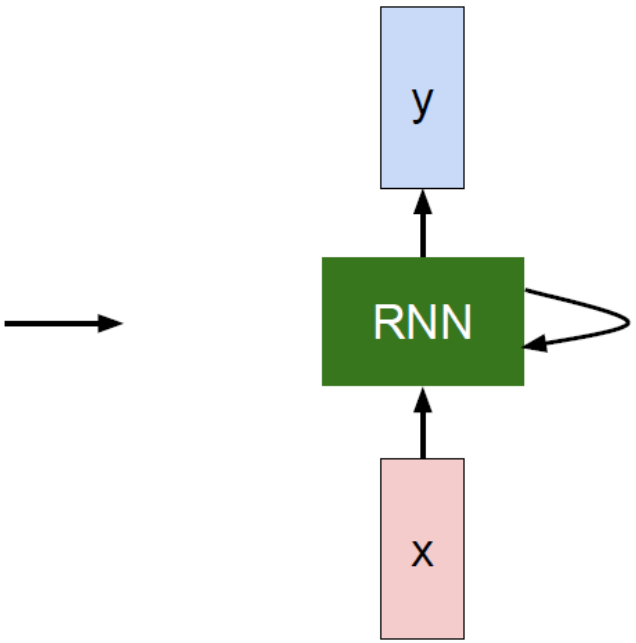
- Concatenate all of William Shakespeare's works to obtain a giant sequence of characters
- Then put it into the RNN and you try to predict the next character in a sequence

THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
This were to be new made when thou art old,
And see thy blood warm when thou feel'st it cold.



Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

Example

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

Example

```
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.
```

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

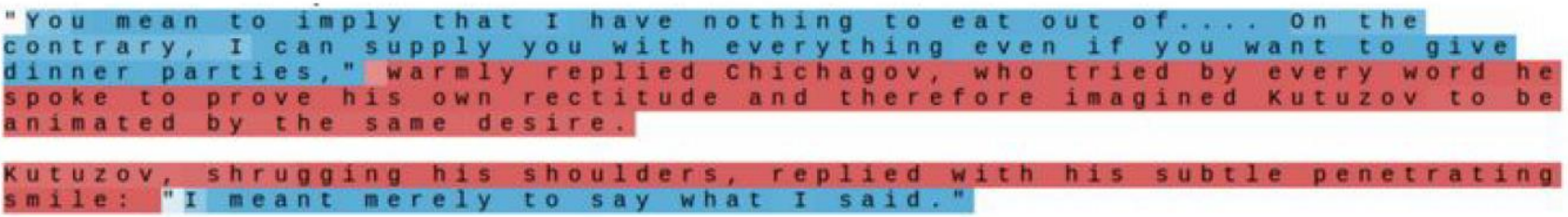
Example – Generated C Code

– Take all Linux source (C code) and concatenate: 700 megabytes of C code and header files

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Char-rnn: Multi-layer Recurrent Neural Networks (LSTM, GRU, RNN) for character-level language models in Torch
https://github.com/karpathy/char-rnn

Searching for interpretable cells



quote detection cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

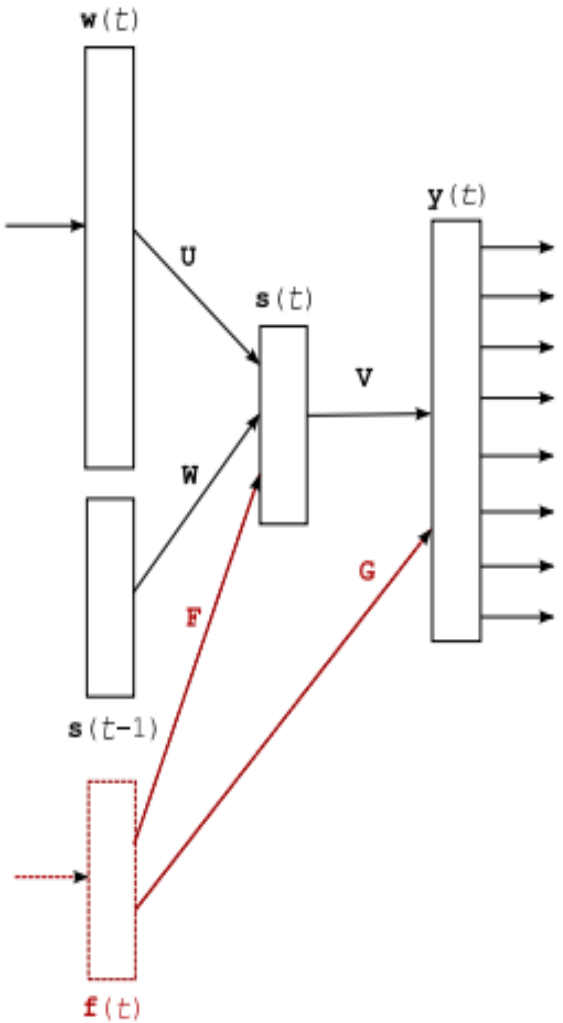
Source: Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

Language modeling summary

- RNN outperforms FNN on language modeling tasks, both are better than n-grams
- The question “are neural nets better than n-grams” is incomplete: the best solution is to use both
- Joint training of RNN and maxent with n-gram features works great on large datasets

Recurrent neural network with additional features

We can use extra features (Part-Of-Speech-POS, external information, long context information, …) represented as additional inputs



Context dependent recurrent neural network language model (Mikolov & Zweig, 2012)

Recurrent neural network with slow features

– We can define the extra features f(t) as exponentially decaying sum of word vectors w(t):

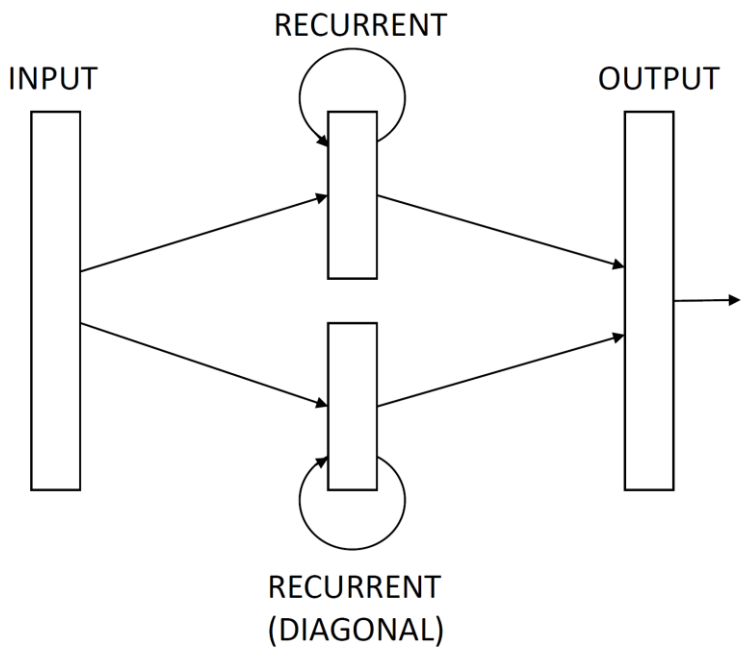
$$f(t) = f(t-1)\gamma + w(t)(1-\gamma)$$

– This will give the model longer term memory

– Performance in language modeling similar to Long Short-Term Memory (LSTM) RNN – about 10% reduction of perplexity over simple RNN

Recurrent neural network with slow features

- 2 hidden layers
- One usual, fully connected
- The other constrained (diagonal)
- The values in diagonal can be fixed (close to 1)
- Learns longer term patterns



RNNLM toolkit

- Available at rnnlm.org
- Based on the word2vec toolkit, it is a small extension of word2vec that makes possible to apply it for language modeling.
- Allows training of RNN and RNNME language models
- Extensions are actively developed, for example multi-threaded version with hierarchical softmax: http://svn.code.sf.net/p/kaldi/code/trunk/tools/rnnlm-hs-0.1b/

CSLM: Feedforward NNLM code

- Continuous Space Language Model toolkit: http://www-lium.univ-lemans.fr/cslm/
- Implementation of feedforward neural network language model by Holger Schwenk

Other neural net SW

– List available at http://deeplearning.net/software_links/

– Mostly focuses on general machine learning tools

Large text corpora

- Short list available at the word2vec project: https://code.google.com/p/word2vec/#Where_to_obtain_the_training_data
- Sources: Wikipedia dump, statmt.org, UMBC webbase corpus
- Altogether around 8 billion words can be downloaded for free

Benchmark datasets for language modeling

- The Penn Treebank setup including the usual text normalization is part of the example archive at rnnlm.org

- WSJ setup (simple ASR experiments, includes N-best list rescoring): http://www.fit.vutbr.cz/~imikolov/rnnlm/kaldi-wsj.tgz

Further Reading

- Noah A. Smith, “Contextual Word Representations: A Contextual Introduction”, Feb. 2019.

- Mikolov Tomá, Deoras Anoop, Povey Daniel, Burget Luká, Èernocký Jan: Strategies for Training Large Scale Neural Network Language Models, In: Proceedings of ASRU 2011