15-780 – Graduate Artificial Intelligence: Convolutional and recurrent networks

J. Zico Kolter (this lecture) and Ariel Procaccia
Carnegie Mellon University
Spring 2017

Outline

Convolutional neural networks

Applications of convolutional networks

Recurrent networks

Applications of recurrent networks

Outline

Convolutional neural networks

Applications of convolutional networks

Recurrent networks

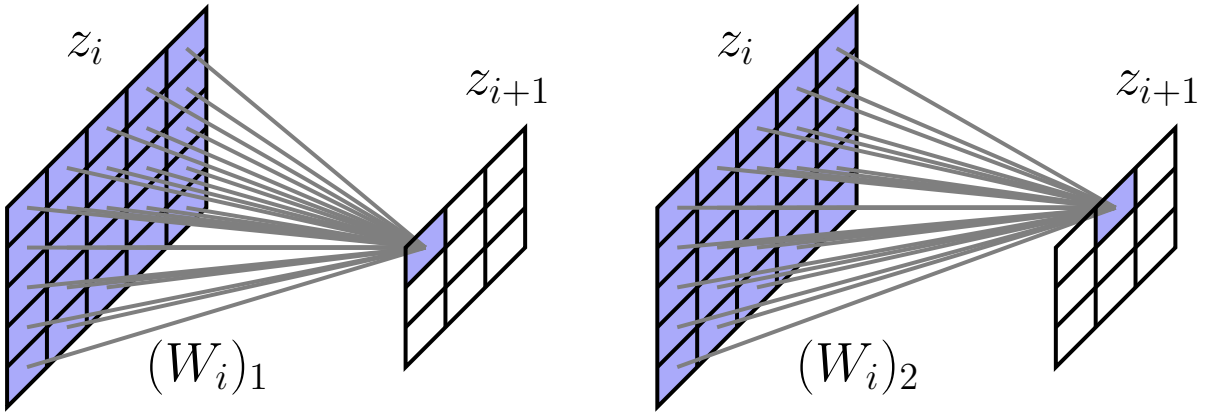
Applications of recurrent networks

The problem with fully-connected networks

A 256x256 (RGB) image $\Longrightarrow$ ~200K dimensional input x

A fully connected network would need a very large number of parameters, very likely to overfit the data

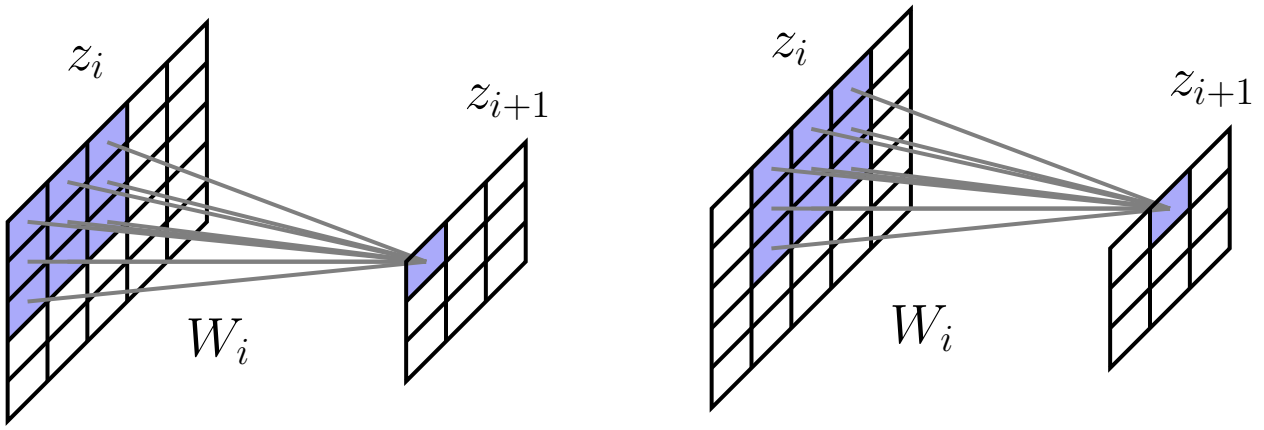
Generic deep network also does not capture the “natural” invariances we expect in images (translation, scale)



Convolutional neural networks

To create architectures that can handle large images, restrict the weights in two ways

1. Require that activations between layers only occur in “local” manner
2. Require that all activations share the same weights



These lead to an architecture known as a convolutional neural network

Convolutions

Convolutions are a basic primitive in many computer vision and image processing algorithms

Idea is to "slide" the weights w (called a filter) over the image to produce a new image, written $y = z * w$

z_{11}	z_{12}	z_{13}	z_{14}	z_{15}
z_{21}	z_{22}	z_{23}	z_{24}	z_{25}
z_{31}	z_{32}	z_{33}	z_{34}	z_{35}
z_{41}	z_{42}	z_{43}	z_{44}	z_{45}
z_{51}	z_{52}	z_{53}	z_{54}	z_{55}

$*$

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

$=$

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	33

$$y_{23} = z_{23}w_{11} + z_{24}w_{12} + z_{25}w_{13} + z_{33}w_{21} + \ldots$$

Convolutions in image processing

Convolutions (typically with *prespecified* filters) are a common operation in many computer vision applications

Original image z

Gaussian blur

$$z * \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 4 & 4 & 1 \end{bmatrix} /273$$

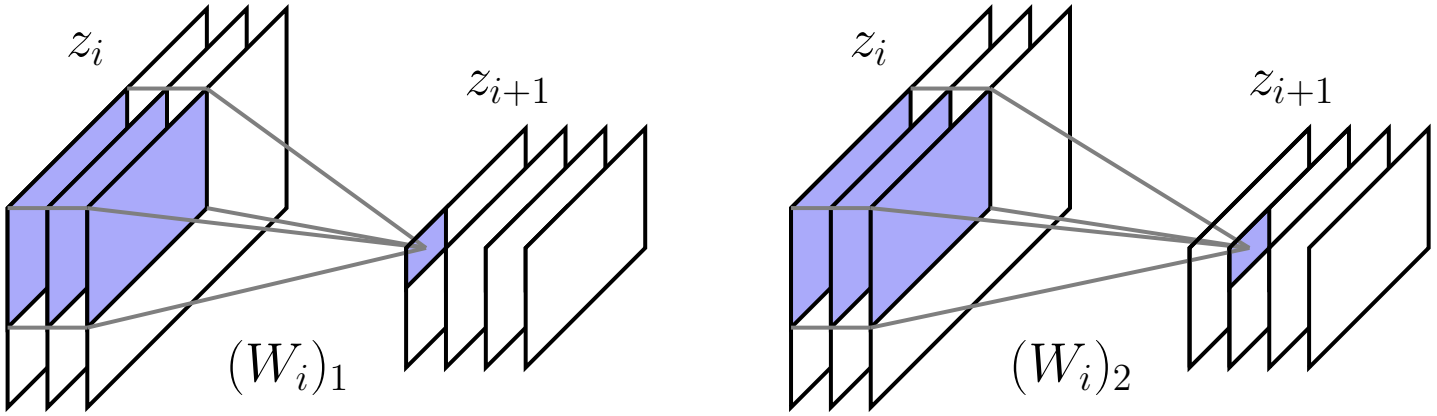
Image gradient

$$\left(\left(z * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \right)^2 + \left(z * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \right)^2 \right)^{\frac{1}{2}}$$

Convolutional neural networks

Idea of a convolutional neural network, in some sense, is to let the network “learn” the right filters for the specified task

In practice, we actually use “3D” convolutions, which apply a separate convolution to multiple layers of the image, then add the results together

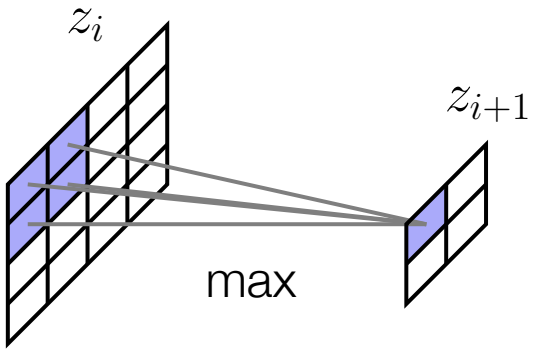


Additional on convolutions

For anyone with a signal processing background: this is actually *not* what you call a convolution, this is a correlation (convolution with the filter flipped upside-down and left-right)

It's common to "zero pad" the input image so that the resulting image is the same size

Also common to use a max-pooling operation that shrinks images by taking max over a region (also common: strided convolutions)



Number of parameters

Consider a convolutional network that takes as input color (RGB) 32x32 images, and uses the layers (all convolutional layers use zero-padding)

1. 5x5x64 convolution
2. 2x2 Maxpooling
3. 3x3x128 convolution
4. 2x2 Maxpooling
5. Fully-connected to 10-dimensional output

How many parameters does this network have?

1. O(10^3)
2. O(10^4)
3. O(10^5)
4. O(10^6)

Learning with convolutions

How do we apply backpropagation to neural networks with convolutions?

$$z_{i+1} = f_i(z_i * w_i + b_i)$$

Remember that for a dense layer $z_{i+1} = f_i(W_i z_i + b_i)$, forward pass required multiplication by W_i and backward pass required multiplication by W_i^T

We're going to show that convolution *is* a type of (highly structured) matrix multiplication, and show how to compute the multiplication by its tranpose

Convolutions as matrix multiplication

Consider initially a 1D convolution $z_i * w_i$ for $w_i \in \mathbb{R}^3$, $z_i \in \mathbb{R}^6$

Then $z_i * w_i = W_i z_i$ for

$$W_i = \begin{bmatrix} w_1 & w_2 & w_3 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & w_3 \end{bmatrix}$$

So how do we multiply by W_i^T?

Convolutions as matrix multiplication, cont

Multiplication by transpose is just

$$W_i^T g_{i+1} = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 \\ w_3 & w_2 & w_1 & 0 \\ 0 & w_3 & w_2 & w_1 \\ 0 & 0 & w_3 & w_2 \\ 0 & 0 & 0 & w_3 \end{bmatrix} g_{i+1} = \begin{bmatrix} 0 \\ 0 \\ g_{i+1} \\ 0 \\ 0 \end{bmatrix} * \overline{w}_i$$

where $\overline{w}_{i+1}$ is just the flipped version of w_i

In other words, transpose of convolution is just (zero-padded) convolution by flipped filter (*correlations* for signal processing people)

Property holds for 2D convolutions, backprop just flips convolutions

Outline

Convolutional neural networks

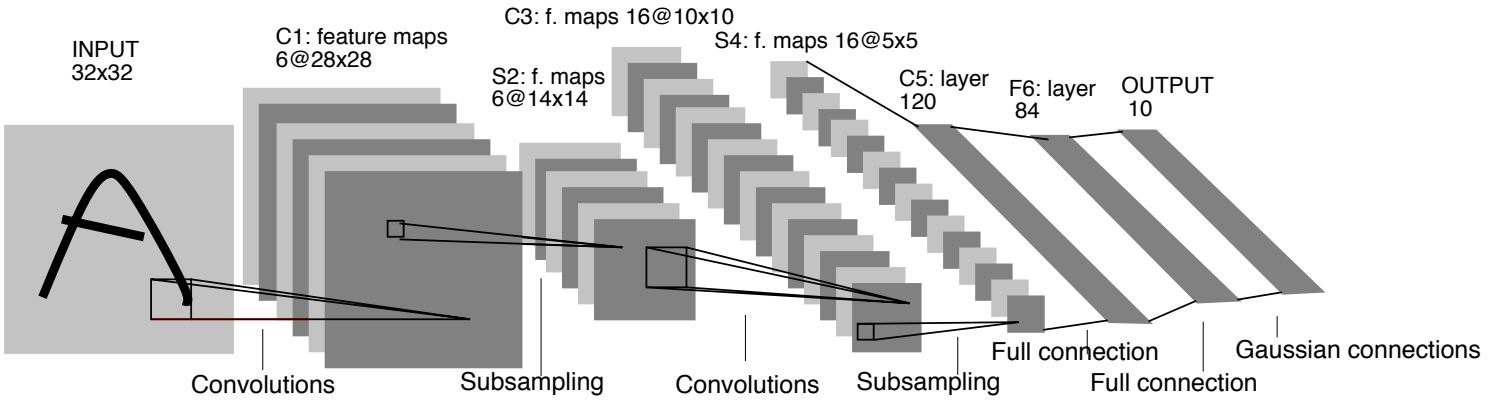
Applications of convolutional networks

Recurrent networks

Applications of recurrent networks

LeNet network, digit classification

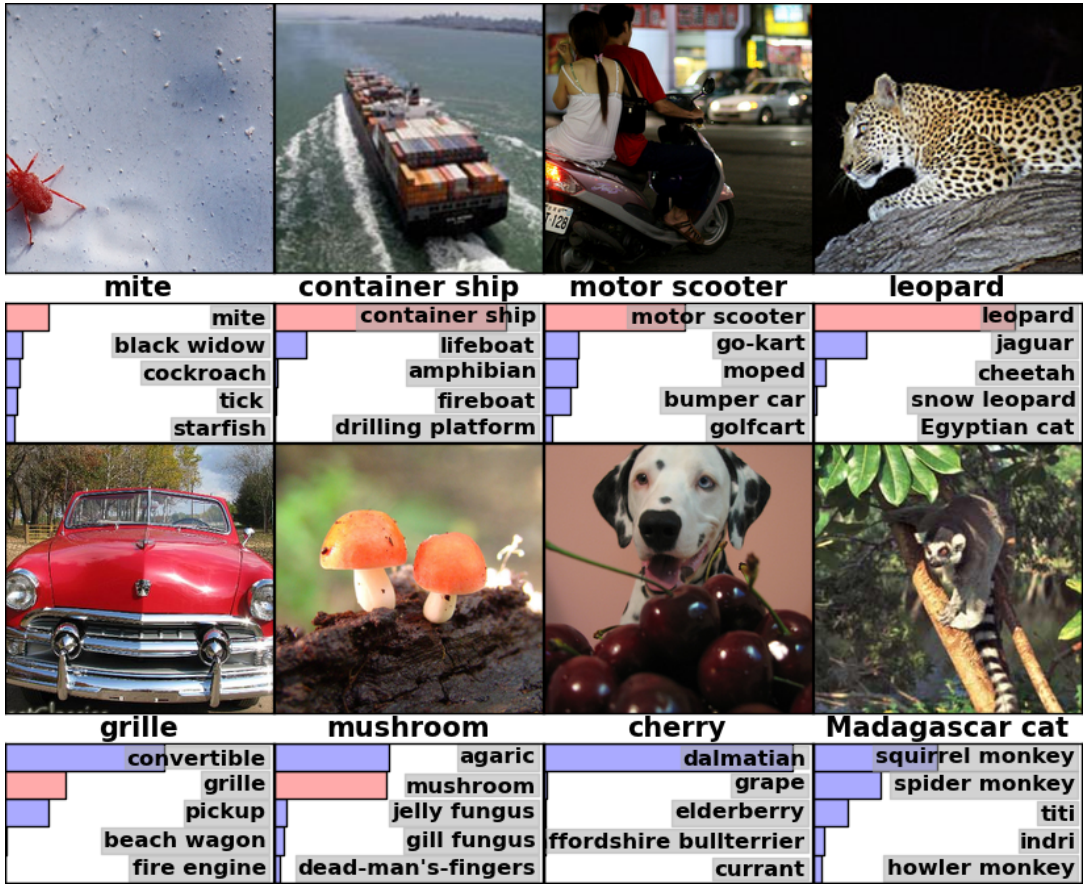
The network that started it all (and then stopped for ~14 years)



LeNet-5 (LeCun et al., 1998) architecture, achieves 1% error in MNIST digit classification

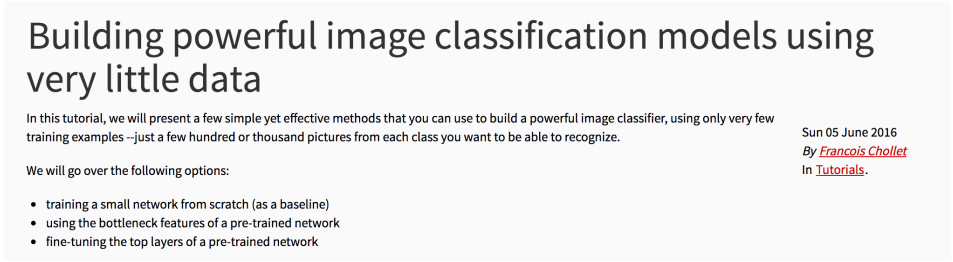
Image classification

Recent ImageNet classification challenges



Using intermediate layers as features

Increasingly common to use later-stage layers of *pre-trained* image classification networks as features for image classification tasks



https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Classify dogs/cats based upon 2000 images (1000 of each class):

Approach 1: Convolution network from scratch: 80%

Approach 2: Final-layer from VGG network -> dense net: 90%

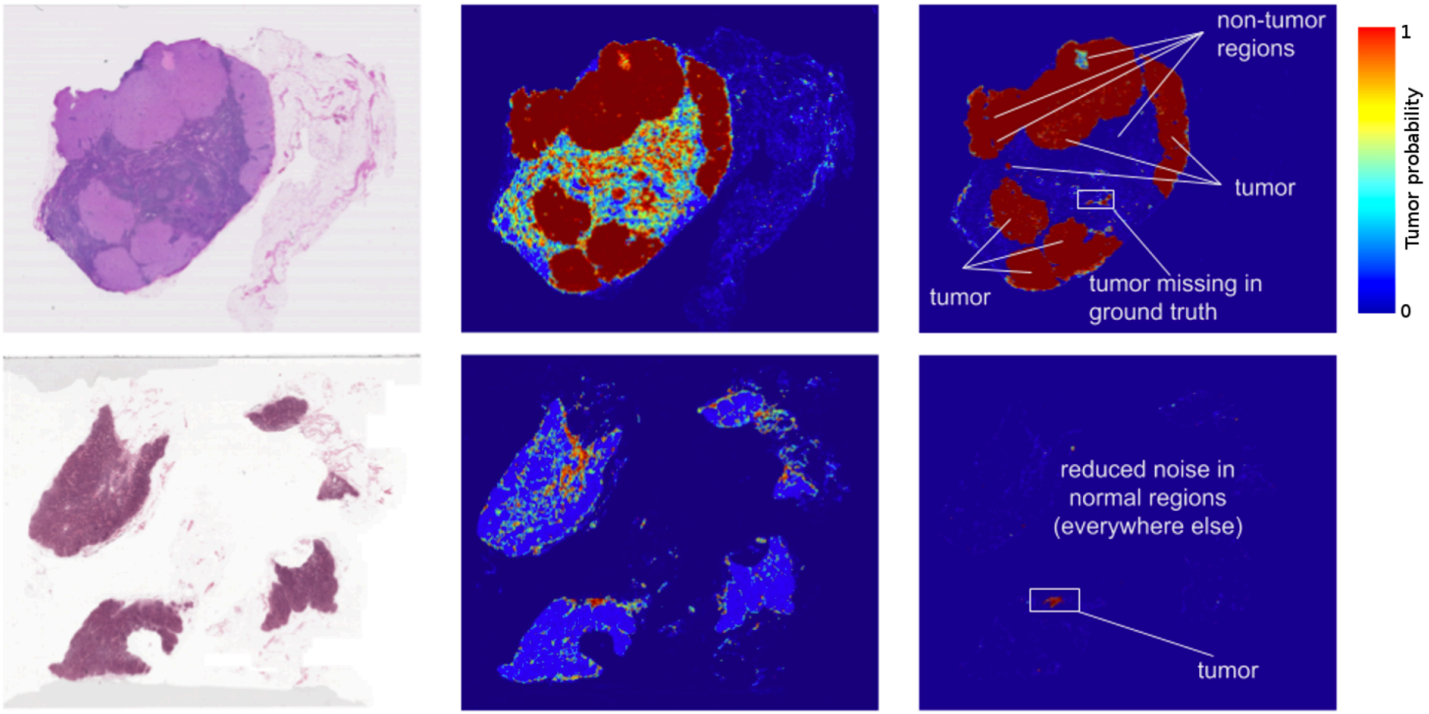
Approach 3: Also fine-tune last convolution features: 94%

Playing Atari games

Neural style

Adjust input image to make feature activations (really, inner products of feature activations), match target (art) images (Gatys et al., 2016)

Detecting cancerous cells in images



Left: Images from two lymph node biopsies. **Middle:** earlier results of our deep learning tumor detection. **Right:** our current results. Notice the visibly reduced noise (potential false positives) between the two versions.

https://research.googleblog.com/2017/03/assisting-pathologists-in-detecting.html

Outline

Convolutional neural networks

Applications of convolutional networks

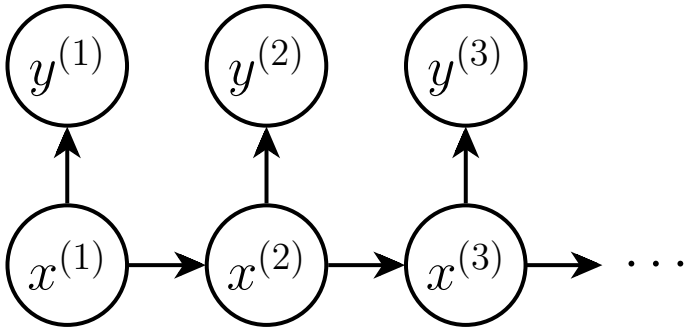
Recurrent networks

Applications of recurrent networks

Predicting temporal data

So far, the models we have discussed are application to independent inputs $x^{(1)}, \ldots, x^{(m)}$

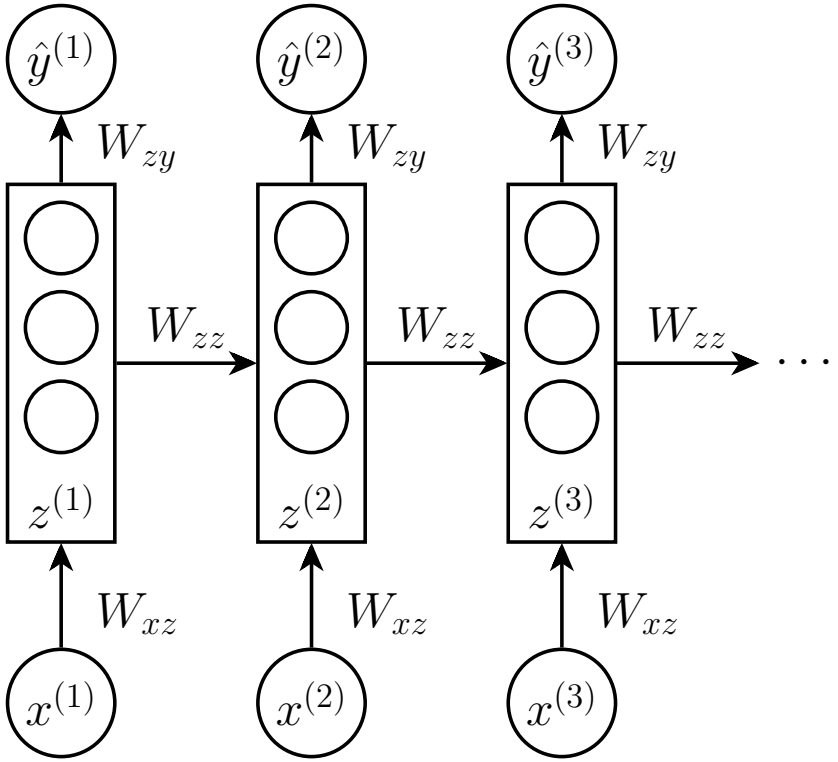
In practice, we often want to predict a *sequence* of outputs, given a sequence of inputs (predicting independently would miss correlations)



Examples: time series forecasting, sentence labeling, speech to text, etc

Recurrent neural networks

Maintain hidden state over time, hidden state is a function of current input and *previous* hidden state



$$z^{(t)} = f_z(W_{xz}x^{(t)} + W_{zz}z^{(t-1)} + b_z)$$
$$\hat{y}^{(t)} = f_y(W_{zy}z^{(t)} + b_y)$$

Training recurrent networks

Most common training approach is to "unroll" the RNN on some dataset, and minimize the loss function

$$\underset{W_{xz}, W_{zz}, W_{zy}}{\text{minimize}} \quad \sum_{i=1}^{m} \ell(\hat{y}^{(t)}, y^{(t)})$$

Note that the network will have the "same" parameters in a lot of places in the network (e.g., the *same* W_{zz} matrix occurs in each step); advance of computation graph approach is that it's easy to compute these complex gradients

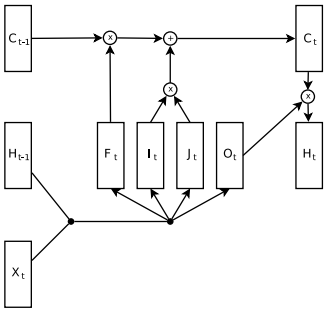
Some issues: initializing first hidden layer (just set it to all zeros), how long a sequence (pick something big, like >100)

LSTM networks

Trouble with plain RNNs is that it is difficult to capture long-term dependencies (e.g. if we see a "(" character, we expected a ")" to follow at some point)

Problem has to do with *vanishing gradient*, for many activations like sigmoid, tanh, gradients get smaller and smaller over subsequent layers (and ReLU's have their own problems)

One solution, long short term memory (Hochreiter and Schmidhuber, 1997), has more complex structure that specifically encodes memory and pass-through features, able to model long-term dependencies



$$
\begin{aligned}
i_t &= \tanh(W_{\mathrm{xi}} x_t + W_{\mathrm{hi}} h_{t-1} + b_{\mathrm{i}}) \\
j_t &= \mathrm{sigm}(W_{\mathrm{xj}} x_t + W_{\mathrm{hj}} h_{t-1} + b_{\mathrm{j}}) \\
f_t &= \mathrm{sigm}(W_{\mathrm{xf}} x_t + W_{\mathrm{hf}} h_{t-1} + b_{\mathrm{f}}) \\
o_t &= \tanh(W_{\mathrm{xo}} x_t + W_{\mathrm{ho}} h_{t-1} + b_{\mathrm{o}}) \\
c_t &= c_{t-1} \odot f_t + i_t \odot j_t \\
h_t &= \tanh(c_t) \odot o_t
\end{aligned}
$$

Figure from
(Jozefowicz et al., 2015)

Outline

Convolutional neural networks

Applications of convolutional networks

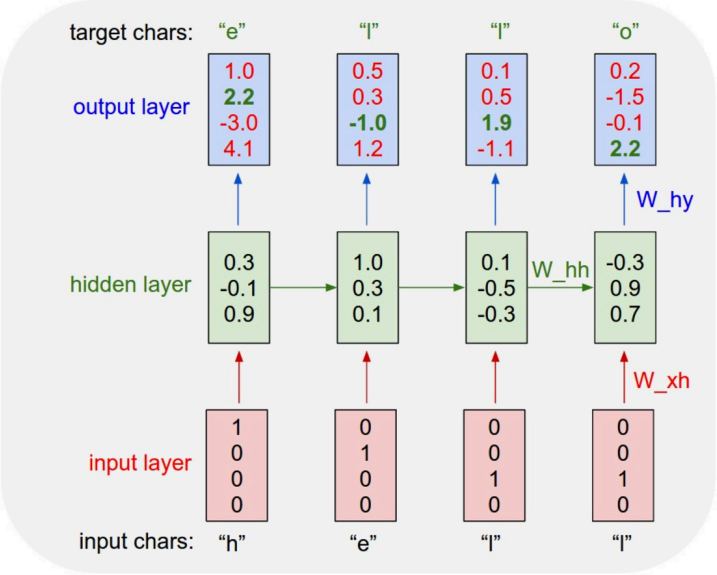
Recurrent networks

Applications of recurrent networks

Char-RNN

Excellent tutorial available at: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Basic idea is to build an RNN (using stacked LSTMs) that predicts the next character from some text given previous characters



Sample code from Char-RNN

Char-RNN trained on code of Linux kernel

```
/*
 * 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;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
…
```

Sample Latex from Char-RNN

Char-RNN trained on Latex source of textbook on algebraic geometry

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariconly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, **??** and the fact that any U affine, see Morphisms, Lemma **??**. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example **??**. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma **??**. Namely, by Lemma **??** we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition* **??** *(without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation* **??**. *Hence we may assume* $\mathfrak{q}' = 0$.

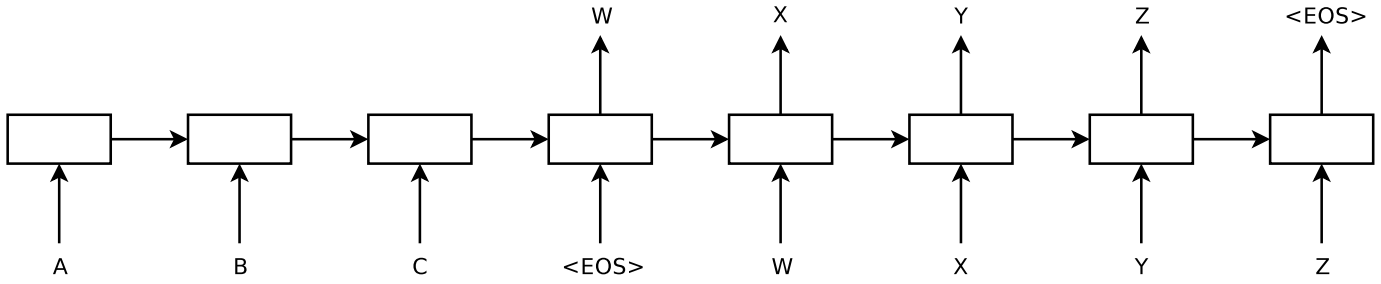
Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (**??**). On the other hand, by Lemma **??** we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

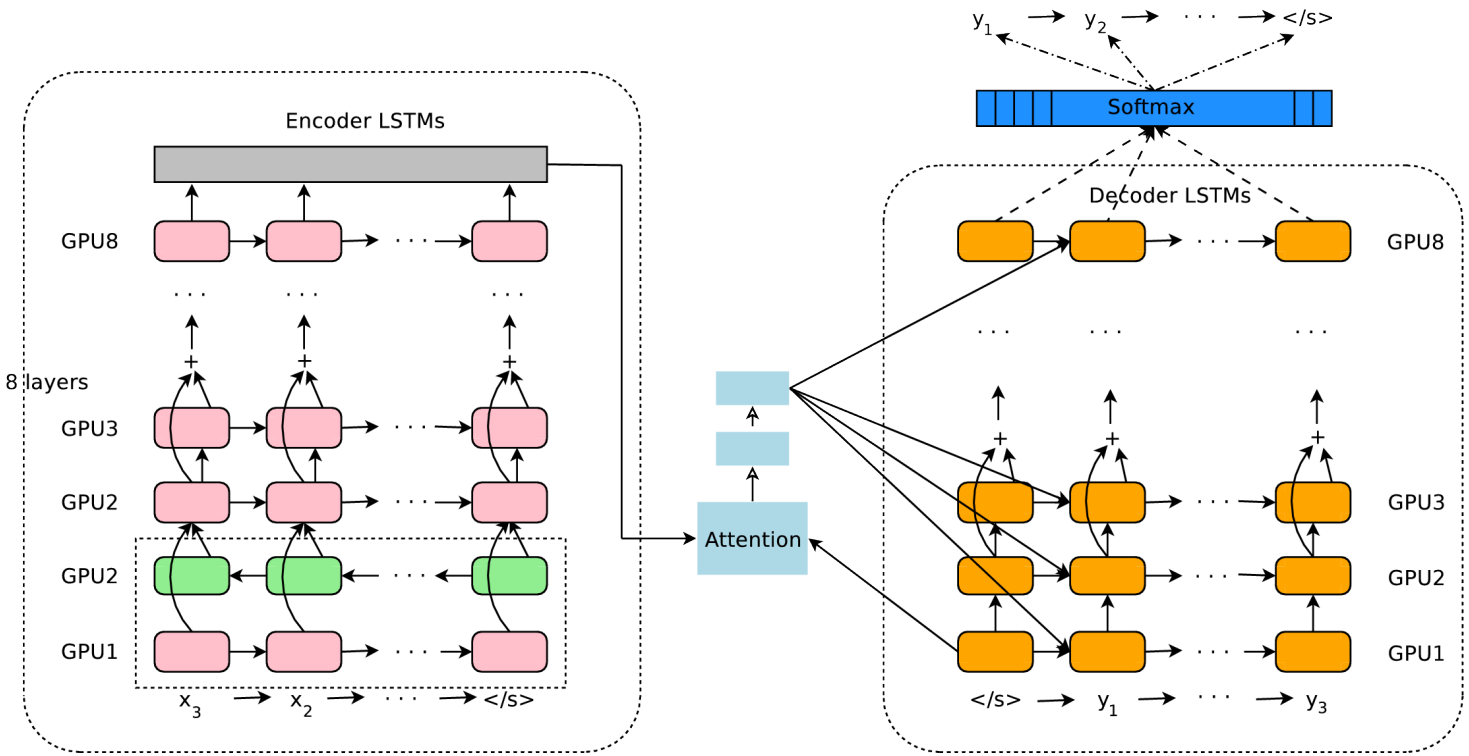
Sequence to sequence models

Idea: use LSTM without outputs on “input” sequence, then auto-regressive LSTM on output sequence (Sutskever et al., 2014)



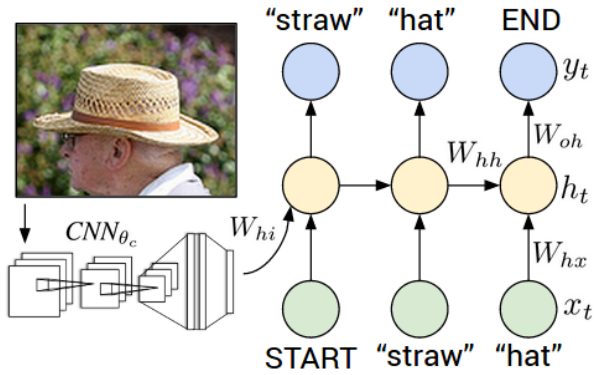
Machine translation

A scale-up of sequence to sequence learning, now underlying much of Google's machine translation methods (Wu et al., 2016)



Combining RNNs and CNNs

Take convolutional network and feed it into the first hidden layer of a recurrent neural network



"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."