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

J. Zico Kolter (this lecture) and Nihar Shah
Carnegie Mellon University
Spring 2019
Outline

Convolutional neural networks

Applications of convolutional networks

Recurrent networks

Applications of recurrent networks
Convolutional neural networks

Applications of convolutional networks

Recurrent networks

Applications of recurrent networks
The problem with fully-connected networks

A 256x256 (RGB) image $\implies$ ~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 $y = z \ast w$.
Convolutions in image processing

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

Original image $z$

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

Gaussian blur

\[
\left( z * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \right)^2 + \left( z * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \right)^2 \right)^{1/2}
\]

Image gradient
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.
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)
Poll: 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. $\approx 10^3$
2. $\approx 10^4$
3. $\approx 10^5$
4. $\approx 10^6$
Learning with convolutions

How do we apply backpropagation to neural networks with convolutions?

\[ z_{i+1} = f_i(z_i \ast w_i + b_i) \]

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 transpose.
Convolutions as matrix multiplication

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

Then $z_i \ast w_i = W_i z_i$ 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

\[
W_i^T g_{i+1} = \begin{bmatrix}
  w_1 & 0 & 0 & 0 \\
  w_2 & w_1 & 0 & 0 \\
  w_3 & w_2 & w_1 & 0 \\
  0 & w_3 & w_2 & w_1 \\
  0 & 0 & w_3 & w_2 \\
  0 & 0 & 0 & w_3
\end{bmatrix} \quad g_{i+1} = \begin{bmatrix}
  0 \\
  0 \\
  g_{i+1} \\
  0 \\
  0
\end{bmatrix} \ast \overline{w_i}
\]

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
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
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

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%

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

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.

\[
\hat{y}^{(1)} = f_y(W_{zy}x^{(1)} + b_y) \\
\hat{y}^{(2)} = f_y(W_{zy}z^{(1)} + b_y) \\
\hat{y}^{(3)} = f_y(W_{zy}z^{(2)} + b_y) \\
z^{(1)} = f_z(W_{xz}x^{(1)} + W_{zz}z^{(1)} + b_z) \\
z^{(2)} = f_z(W_{xz}x^{(2)} + W_{zz}z^{(2)} + b_z) \\
z^{(3)} = f_z(W_{xz}x^{(3)} + W_{zz}z^{(3)} + b_z)
\]
Most common training approach is to “unroll” the RNN on some dataset, and minimize the loss function

$$\min_{W_x, W_z, W_y} \sum_{i=1}^{m} \ell(\hat{y}(t), y(t))$$

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

\[
\begin{align*}
    i_t &= \tanh(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
    j_t &= \text{sigm}(W_{sj}x_t + W_{hj}h_{t-1} + b_j) \\
    f_t &= \text{sigm}(W_{sf}x_t + W_{hf}h_{t-1} + b_f) \\
    o_t &= \tanh(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
    c_t &= c_{t-1} \odot f_t + i_t \odot j_t \\
    h_t &= \tanh(c_t) \odot o_t
\end{align*}
\]

Figure from (Jozelewicz et al., 2015)
Outline

Convolutional neural networks

Applications of convolutional networks

Recurrent networks

Applications of recurrent networks
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

```c
/*
 * 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 $\Theta_{n+1, \ldots, m}$ where $L_{mn} = 0$, 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

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

and the comparibility in the fibre product covering we have to prove the lemma generated by $\prod Z \times_U V \to V$. Consider the maps $M$ and the set of points $\text{Spec}(\mathcal{O})$ 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 $\text{Spec}(\mathcal{O})$ such that $\text{Spec}(\mathcal{O}) \to S$ 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 finitely presented over $S$. We claim that $\mathcal{O}_{X_{x_i}}$ is a scheme where $x, x', x'' \in S$ such that $\mathcal{O}_{X_{x_i}} \to \mathcal{O}_{X_{x''_i}}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_R(\mathcal{F}')$ and we win.

To prove study we see that $\mathcal{F}_U$ is a covering of $X'$, and $T_i$ is an object of $\mathcal{F}_{X/S}$ for $\mathcal{F}_i \in \mathcal{S}$ exists and let $\mathbb{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $C$ as a $\mathcal{O}_X$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\mathcal{M} = \mathcal{F} \otimes \text{Spec}(\mathcal{O}) \to \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf} \to (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \to (U, \text{Spec}(\mathcal{O}))$$

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 any open covering follows from the less of Example ??, it may replace $S$ by $X_{\text{space}, \text{etale}}$ which gives an open subspace of $X$ and $T$ equal to $S_{\text{zar}}$, 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 $\text{Proj}_X(A) = \text{Spec}(B)$ over $U$ compatible with the complex

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

When in this case of to show that $Q \to C_{2/N}$ 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 connary, 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 $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on $X$. But given a scheme $U$ and a surjective étale morphism $U \to X$. Let $S \cap U \subset S$ be the scheme $X$ over $S$ at the schemes $X_i \to X$ and $U = \lim S_i$.

The following lemma surjective restrincces of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{x_0} = S$.

Lemma 0.2. Let $X$ be a locally Noetherian scheme over $S$, $E = E_{X/S}$. Set $\mathcal{F} = \mathcal{F}_{x_i}$. Since $\mathcal{F} \subset \mathcal{F}$ are nonzero over $i_0 \leq p$ is a subset of $\mathcal{F}_{x_0} = \mathcal{F}_{x_0}$ works.

Lemma 0.3. In Situation ??, hence we may assume $q' = 0$.

Proof. We will use the property we see that $p$ is the next function (??). On the other hand, by Lemma ?? we see that

$$D(O_X) = O_D(D)$$

where $K$ is an $F$-algebra where $\delta_{n+1}$ 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)

Figure 1: Our model reads an input sentence “ABC” and produces “WXYZ” as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

The main result of this work is the following. On the WMT'14 English to French translation task, we obtained a BLEU score of 34.81 by directly extracting translations from an ensemble of 5 deep LSTMs (with 380M parameters each) using a simple left-to-right beam-search decoder. This is by far the best result achieved by direct translation with large neural networks. For comparison, the BLEU score of a SMT baseline on this dataset is 33.30.

The 34.81 BLEU score was achieved by an LSTM with a vocabulary of 80k words, so the score penalizes when the reference translation contained a word not covered by these 80k. This result shows that a relatively unoptimized neural network architecture which has much room for improvement to outperform a mature phrase-based SMT system.

Finally, we used the LSTM to rescore the publicly available 1000-best lists of the SMT baseline on the same task. By doing so, we obtained a BLEU score of 36.5, which improves the baseline by 3.2 BLEU points and is close to the previous state-of-the-art (which is 37.0).

Surprisingly, the LSTM did not suffer on very long sentences, despite earlier experience of other researchers with related architectures. We were able to do so because we reversed the order of words in the source sentence but not the target sentences in the training and test set. By doing so, we introduced many short term dependencies that made the optimization problem much simpler (see sec. 2 and 3.3). As a result, SGD could learn LSTMs that had no trouble with long sentences. The simple trick of reversing the words in these source sentences is one of the key technical contributions of this work.

A useful property of the LSTM is that it learns to map a variable length sentence into a fixed dimension vector representation. Given that translations tend to be paraphrases of the source sentences, the translation objective encourages the LSTM to find sentence representations that capture their meaning, as sentences with similar meanings are close to each other while different
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.