Long-term Recurrent Convolutional Networks for Visual Recognition and Description

1 Citation

Donahue, Jeffrey, et al. "Long-term recurrent convolutional networks for visual recognition and description." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

`https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Donahue_Long-Term_Recurrent_Convolutional_2015_CVPR_paper.pdf`

2 Abstract

We combine CNNs with LSTMs to create models that can solve activity recognition (sequential input, fixed-length output), image captioning (fixed-length input, sequential output), and video narration (sequential input, sequential output).

3 Introduction

Existing CNNs will use optical flow features and/or do 3D convolution to incorporate the time dimension in videos. People have avoided using Recurrent Neural Networks (RNNs) to solve this problem because of the vanishing gradient problem, but Long Short-Term Memory networks (LSTM) mitigate this problem somewhat by allowing the network to forget.

We try various combinations of convolutional neural networks (CNNs) and LSTMs to solve sequence learning problems.

4 Background: Recurrent Neural Networks (RNNs)

A typical RNN is governed by these equations:

$$h_t = g(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$
$$z_t = g(W_{hz}h_t + b_z)$$

for nonlinearity g. If you use this, you run into the vanishing gradient problem. Thus, we use LSTMs instead.

Letting $\phi(x) = \tanh(x)$ and $\sigma(x)$ be the sigmoid, we have:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$g_t = \phi(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$
$$h_t = o_t \cdot \phi(c_t)$$

where h_t is the hidden unit, c_t is the memory cell, i_t is the input gate, f_t is the forget gate, o_t is the output gate, and g_t is the input modulation gate.

You can stack LSTMs by making h_t be the input (x_t) to another LSTM.

5 Long-Term Recurrent Convolutional Network (LRCN) Model

Apply a CNN (parametrized by V) to compute image features: $\phi_V(x)$. We can apply these in parallel to the sequence $[x_1, x_2, ..., x_T]$ and get $[\phi_V(x_1), \phi_V(x_2), ..., \phi_V(x_T)]$.

After the CNN, we feed the image features (and maybe some other input, like the ground truth output at the given timestep) to a stack of LSTMs. At the end of the LSTM stack, we apply a fully-connected layer and softmax to do classification. Thus, our model takes in an sequence $[x_1, x_2, ..., x_T]$ and outputs a sequence of equal length $[y_1, y_2, ..., y_T]$.

We consider three problems: sequential input/fixed output ($[x_1, x_2, ..., x_T] \rightarrow y$), fixed input/sequential output ($x \rightarrow [y_1, y_2, ..., y_T]$), and sequential input/sequential output ($[x_1, x_2, ..., x_T] \rightarrow [y_1, y_2, ..., y_{T'}]$) (notice $T \neq T'$ is possible).

For sequential input/fixed output, we take a late fusion approach to combine $[y_1, y_2, ..., y_T]$ to produce y. For fixed input/sequential output, we simply set $x_t = x$ for all t. For sequential input and output, we have an encoder network that processes $[x_1, x_2, ..., x_T]$ and produces a fixed size state. Then, a decoder network processes that state and produces $[y_1, y_2, ..., y_{T'}]$.

Assuming our CNN is parametrized by V and our LSTMs are parametrized by W, our loss is:

$$L(V, W) = -\sum_{t=1}^{T} \log P_{V,W}(y_t | x_{1:t}, y_{1:t-1})$$

6 Activity Recognition

Our CNN is tweaked AlexNet (pretrained on ImageNet) and we use one 256-unit LSTM. We experimented with the last, and second to last, fully connected layer of the CNN as the image descriptor. We tried representing images as RGB and as flow-x, flow-y, flow-magnitude. For each video, we extract 16-frame clips with a stride of 8 between frames and average the final class scores across clips.

Fine-tuning and evaluating on the UCF-101 dataset, we find that the flow inputs give a 5% improvement while RGB gives 0.5% boost. The second to last fully connected layer is the better feature descriptor.

7 Image Description

At each timestep, the input is the image and previous word (this is the ground truth word at train time and the predicted word a test time). The first LSTM receives the embedded word. The second LSTM receives the output of the first LSTM and the image descriptor. The third LSTM just processes the output of the second LSTM. The fourth LSTM feeds into a fully-connected layer and softmax over the words. Note that the first LSTM only processes language - this factored approach gives good performance.

We evaluate on Flickr30k and MS COCO, beating baselines (our metric is BLEU). We also crowdsource human evaluation with Amazon Mechanical Turk - we do pretty well.

8 Video Description

There aren't many video description datasets, so we won't use our sequence-to- sequence model. Instead, we will assume that we have a conditional random field input (CRF) that indicates the subjects, objects, and verbs in the video with their probabilities. Usually, this goes into a statistical machine translation (SMT) model, but we replace that with our LSTMs. We consider three architectures. First, we take the max-probability word at each time step, create a sequence of words, and one-hot encode the words. These one-hot vectors are the input to our LSTM encoder network. The encoder produces a state that the LSTM decoder network unravels into the description. Second, we take the max-probability words, one-hot encode them, and provide that as input for every timestep and produce an output word for each timestep. Third, we use the same approach as in the previous sentence, but we use the probability distribution over the words instead of one-hot encoding them.

We evaluate on the TACoS dataset (45K video/description pairs). The second and third approaches beat the first (the encoder-decoder). We beat the baseline SMT (we get 28.8% BLEU-4 while the baseline got 26.9%).

9 Conclusions

LRCNs combine CNNs and LSTMS and can solve many sequence learning problems. You can also just use the LSTMs with traditional computer vision techniques (e.g. conditional random fields).