Improving TCNs for Image Captioning

General Project Information

For our project, we derive the general order 4 formula for a convolutional neural network (CNN), and implement both forward and backpropagation with only elementary math operations in Python and C++. We experiment using CNNs in a number of different domains, including, but not limited to: real time face region detection with spatial CNNs, and sequential modeling with temporal CNNs.

Introduction and Rationale

One aim of this project is to improve temporal convolutional networks (TCN) in two areas which we have found the original implementation in [1] to be lacking. We propose a new zero-padding and dilation scheme which increases the efficiency and accuracy of the TCN model. The proposed changes are empirically evaluated together with three residual blocks. We then compare our improved TCN to Gated Recurrent Units (GRU) on a complex task which TCNs have never been used for before, namely, automatic image captioning. Showing empirical evidence that TCNs outperform their recurrent counterparts on more complex tasks can lead to the eventuality of replacing every recurrent network with a temporal convolutional model, increasing the accuracy on all sequential problems.

Background

In conventional spatial CNNs, the model learns to transform a signal by applying a number of discrete convolutions on the input data. Instead of convolving along the spatial dimensions, a CNN can be used to convolve along a temporal dimension, and can thus be used as a sequential model, as long as there is no information leakage from the future into the past.

Method

Given a number of data points at timesteps $t \in \{1, 2, 3, ..., i\}$, a sequential model sets out to predict the value at timestep $t = i + 1$, through the use of data from the previous timesteps. For a TCN, the activations at layer l is a tensor of order 3, $X^{(l)} \in \mathbb{R}^{R \times C^{(l)} \times T^{(l)}}$ where $C^{(l)}$ is the number of feature channels and $T^{(l)}$ is the length of the sequence. The kernel weights are also a tensor of order 3, $W^{(l)} \in \mathbb{R}^{C^{(l+1)} \times C^{(l)} \times k}$, where k is the kernel size. In addition, a kernel has a stride s and dilation d (see figure 1).

$$X^{(l+1)}_{r,c',t'} = \sum_{c=1}^{C^{(l)}} \sum_{i=1}^{k} X^{(l)}_{r,c,(st'-s+di)} W^{(l)}_{c',c,i} \tag{1}$$

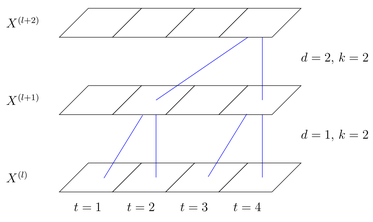

Fig. 1: An illustration of a TCN. Squares represents activations at a specific timestep t. Blue lines represent the convolutional kernel applied at a single position on the tensor of activations.

Improved Zero-padding

The temporal dimension size T and T' of arbitrary layers l and $l + 1$ varies dependent on the zero-padding p, kernel size k, dilation d and stride s:

$$T^{(l+1)} = \frac{T^{(l)} - d(k-1) + p_{left} + p_{right} - 1}{s} + 1 \tag{2}$$

To prevent information leakage from previous timesteps into future timesteps, the activation tensor has to be zero-padded such that $T^{(l+1)} \geq T^{(l)}$. In [1] this problem was solved by padding both sides of the tensor with $p = p_{left} = p_{right} = d(k-1)$ and removing the last $d(k-1)$ neurons which break the rule of information flow. We propose zero-padding only for the beginning of the activation tensor with $p_{left} = d(k-1)$ and $p_{right} = 0$. The result is that the activation at timestep $t = i$ can efficiently use information from all timesteps $t < i$. At $t = 1$, the network effectively computes convolutions with $k = 1$, since all other timesteps are zero-padded neurons (see figure 2). At later timesteps, the network can make use of the whole kernel, and thus activations and information from earlier timesteps.

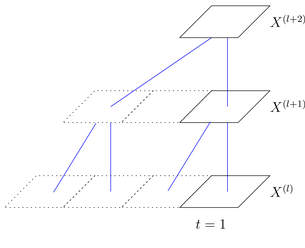

Fig. 2: A temporal convolution at timestep $t = 1$. Bold and dotted squares represent normal and zero-padded activations respectively. Blue lines depict the convolutional kernel. The network effectively computes convolutions with kernel size $k = 1$, since every neuron except at the first timestep is a neuron created by the zero-padding.

Improved Dilations

Let L, d_l and k_l be the total number of layers in the TCN, the dilation at layer l, and the kernel size at layer l respectively. The receptive field r of a single layer and R_{output} of the output neurons are defined as:

$$r(d, k) = d(k-1) + 1 \tag{3}$$

$$R_{output} = 1 + \sum_{l=2}^{L} (r(d_l, k_l) - 1) \tag{4}$$

We suggest replacing the current dilations with a linear schedule, while making the total receptive field of the output neurons to be slightly larger than the maximum sequence length. Exponential dilation increases in [1] causes the last layers with $k \leq 2$ to effectively turn into convolutions with kernel size $k = 1$. If d is large enough, the neurons multiplied by the kernel weights will always contain zero-padded neurons, and thus the whole layer effectively becomes a convolution with kernel size $k = 1$, similar to figure 2. The result of exponential dilations is that only the first layers use information from previous timesteps, while the later layers become equivalent to fully-connected layers. Additionally, our new dilation and zero-padding scheme leads to less memory usage.

Experiments

We evaluated the proposed changes on the original [1], and two additional residual blocks on SequenceMNIST and Permuted SequenceMNIST.

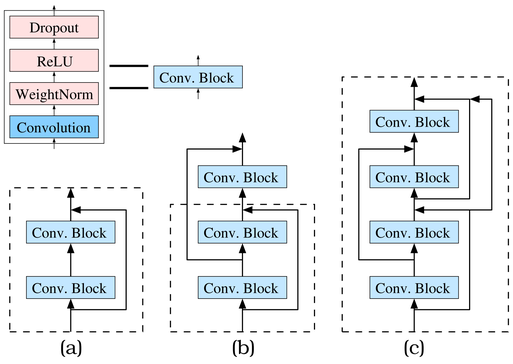

Fig. 3: The original (a) Single and the proposed (b) Double and (c) Radical residual blocks. Lines in bold represent identity connections.

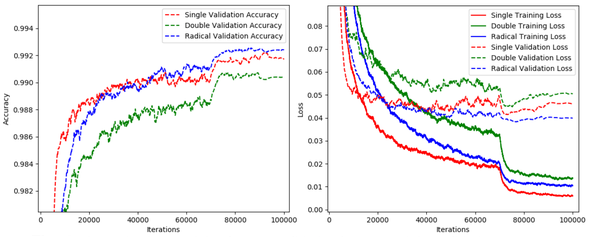

Fig. 4: A graph of the training losses, validation losses and the validation accuracy on SequenceMNIST as a function of the number of training iterations.

Model	Test Acc.	Val. Acc.
TCN (Single)	**99.53%**	99.23%
TCN (Double)	99.34%	99.07%
TCN (Radical)	99.37%	**99.30%**
Dilated GRU [2]	99.0%	N/A
TCN (Original) [1]	99.0%	N/A

Table 1: SequenceMNIST. A comparison with the current state of the art. Data not provided by the authors is marked as *N/A*. Best results are shown in bold.

Automatic Image Captioning

We compared GRU and TCN models on the MSCOCO Captions [3] dataset. The models generate image captions by sequentially suggesting the next word in a sentence. Each word is represented as a one-hot vector encoding, and is given to the model through word embeddings.

A ResNet-18 [4] was used to extract a 512-dimensional feature vector, describing a given image. The GRU initialized its first hidden layer with this vector, and both the TCN and GRU used it as a start of string token. Unknown words were replaced with the *<UKN>* token, and the models stopped after predicting an end of string *<EOS>* token. Let S be the set of all predicted word probabilities $\hat{p}$, and p the associated ground truth to word prediction $\hat{p}$. The total loss $L(\theta)$ which the models used is defined as:

$$L(\theta) = -\frac{1}{|S|} \sum_{p \in S} p \log \hat{p} \tag{5}$$

Fig. 5: Comparative results of the TCN and GRU automatic image annotation models on images from the MSCOCO Captions [3] validation dataset.

Model	CIDEr-D	METEOR	Rouge-L	BLUE-1	BLUE-2	BLUE-3	BLUE-4
TCN (C5)	0.722	0.219	0.482	0.658	0.476	0.338	0.240
GRU (C5)	0.741	0.222	0.490	0.668	0.488	0.347	0.248

Table 2: MSCOCO Results. The TCN and GRU models evaluated on the automatic evaluation metrics of the MSCOCO test set evaluation server [3].

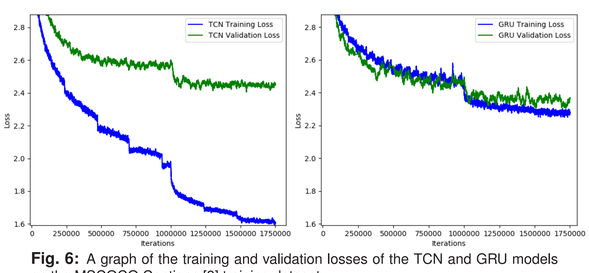

Fig. 6: A graph of the training and validation losses of the TCN and GRU models on the MSCOCO Captions [3] training dataset.

Conclusion

The new suggested zero-padding and dilation scheme heavily outperformed the previous state of the art, halving the previous best test loss on SequenceMNIST. Regarding automatic image captioning, the TCN model differed by a higher quality of generated image captions and faster convergence speeds during training, despite a similarity of TCN and GRU performances on the automated evaluation metrics.