Improving TCNs for Image Captioning

Introduction and Rationale

The aim of this project is to improve temporal convolutional networks (TCN) in two areas which we have found the original implementation in [**?**] to be lacking. We propose a different zero-padding and dilation scheme which still keeps information from not leaking from the future into the past, drawing inspiration from spatial CNNs. These changes will increase both the efficiency and accuracy of the models, and we empirically evaluate them together with three different residual blocks. We then compare our improved TCN with a Gated Recurrent Unit (GRU) on a complex task which TCNs have never been used before, namely, automatic image captioning.

Showing that TCNs outperform their recurrent counterparts on more complex tasks can lead to the eventuality of replacing every recurrent network with a temporal convolutional counterpart, increasing the accuracy of any sequential problem.

Background

In conventional 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. The activations at layer l is a tensor of order 3 $X^{(l)} \in \mathbb{R}^{R \times C \times T}$ where C is the number of feature channels and T is the maximum length of the sequence. The kernel weights are also a tensor of order 3 $W^{(l)} \in \mathbb{R}^{C' \times C \times k}$, where C' is the number of feature channels of the proceeding layer, and k is the kernel size. In addition, a kernel has a stride s and dilation d (see figure 1).

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

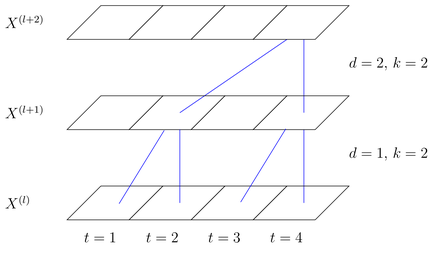

Fig. 1: An illustration of a TCN. Squares represents activations at a specific timestep t. The levels of height show the layers of depth of a TCN. Blue lines represent the kernel applied at a single position on the activation tensor.

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:

$$T' = \frac{T - d(k-1) + p_{left} + p_{right} - 1}{s} + 1 \tag{2}$$

To not cause any information leakage from previous timesteps into future timesteps, the activation tensor has to be zero-padded such that $T' \geq T$. In [**?**] 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 information flow rule. 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 effectively 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.

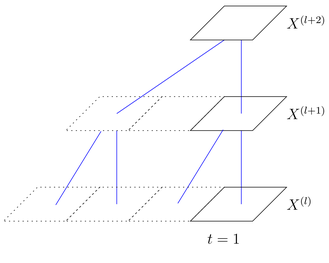

Fig. 2: A temporal convolution at timestep $t = 1$. Dotted squares represent zero-padded activations, while bold squares represent normal activations. Blue lines represent the convolutional kernel. The network effectively computes convolutions with kernel size $k = 1$ since every neuron except at the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

Improved Dilations

Let L, d_l and k_l are the total number of layers in the TCN, 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 by:

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

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

We suggest replacing the current dilations with a linear schedule. Exponential dilation increases in [**?**] 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 mathematically equivalent to fully-connected layers. Additionally, our suggestion leads to less memory usage due to decreased dilations.

Experiments

We evaluate the proposed changes on the original [**?**] and two additional residual blocks on SequenceMNIST and Permuted SequenceMNIST.

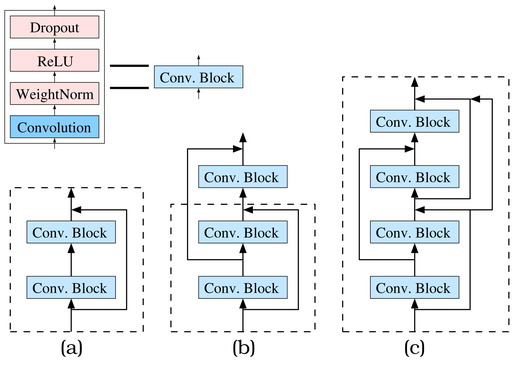

Fig. 3: The original (a) Single and the proposed (b) Double and (c) Radical residual blocks. Lines in bold represent identity connections.

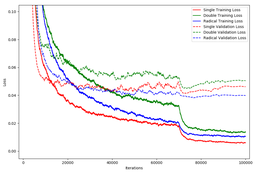

Fig. 4: SequenceMNIST training and validation losses

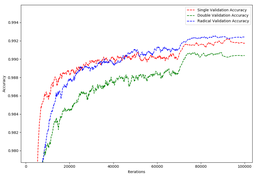

Fig. 5: SequenceMNIST validation accuracy.

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 [**?**]	99.0%	N/A
TCN (Original) [**?**]	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*.

Automatic Image Captioning

We compare a GRU and TCN on the MSCOCO Captions [**?**] 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 is used to extract a 512-dimensional feature vector, describing a given image. The GRU initializes its first hidden layer with this vector, and both the TCN and GRU uses it as a start of string token. Unknown words are replaced with the *<UKN>* token, and the models stops after predicting an end of string *<EOS>* token. Let S be the set of all predicted word probabilities p, and $\hat{p}$ the associated ground truth to word prediction p. The total loss $L(\theta)$ is given by:

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

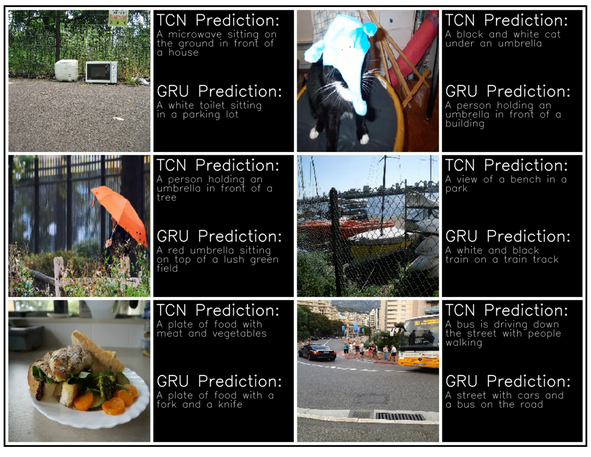

Fig. 6: Comparative results of the TCN and GRU model on images from the MSCOCO Captions validation set [**?**].

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 model evaluated on the automatic evaluation metrics of the MSCOCO test server [**?**].

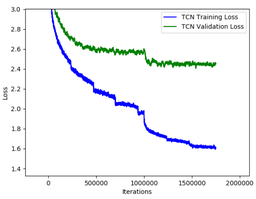

Fig. 7: The training and validation loss of the TCN.

Fig. 8: The training and validation loss of the GRU.

Conclusion

The new suggested zero-padding and dilation scheme heavily outperformed the previous state of the art, halving the 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, in spite of a similarity of TCN and GRU performances on the automated evaluation metrics.