To: David L. Banks

From: Qi QIAN

Date: Aug 17th, 2019

Subject: A comparison of different RNN Structures on Cifar-10 Dataset

Introduction:

"CIFAR-10 is an established computer-vision dataset, which is regarded as a simplified version of ImageNet, used for object recognition. It consists of 60,000 32x32 color images containing one of 10 object classes, with 6000 images per class. For previous work, we used NN and CNN structure, in this report, I implemented several different LSTM structures and compared the performances of different models.

Hyperparameters:

- Learning Rate: 0.001
- Batch Size: 1024
- Epochs: 10
- Number of Hidden Units: 128

Part I: Pixel by Pixel (Flatten) Method

In this method, we consider each pixel in three dimensions as a cell of RNN structure. We will then handle 1024 sequences of 3 timesteps for every sample. Especially, we need to flatten each pixel in 3 channels (RGB). To implement this, we need firstly reshape the dataset dimensions from train: (50000, 32, 32, 3); test: (10000, 32, 32, 3) to train: (50000, 3, 1024); test: (10000, 3, 1024). Then, we set the parameters as:

Numbers of input: n_step = 3
The dimension of n_input = 1024

Model Summary:

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 128)               590336
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_4 (Activation)    (None, 10)                0
=================================================================
Total params: 591,626
Trainable params: 591,626
Non-trainable params: 0
```

Part II: Pixel by Pixel (Non-Flatten) Method

In this method, we also consider each value in each channel of one pixel in RGB as a cell of RNN structure. We will then handle 3072 sequences of 1 timesteps for every sample. To implement this, we need firstly reshape the dataset dimensions from train:

(50000, 32, 32, 3); test: (10000, 32, 32, 3) to train: (50000, 1, 3072); test: (10000, 1, 3072). Then, we set the parameters as:

Numbers of input: n_step = 1
The dimension of n_input = 3072

Model Summary:

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_11 (LSTM)               (None, 128)               1638912
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_8 (Activation)    (None, 10)                0
=================================================================
Total params: 1,640,202
Trainable params: 1,640,202
Non-trainable params: 0
```

Part III: Line by Line (Flatten) Method

In this method, we consider each line in three dimensions as a cell of RNN structure. Especially, we need to flatten each pixel in 3 channels (RGB). We will then handle 32 sequences of 96 timesteps for every sample. To implement this, we need firstly reshape the dataset dimensions from train: (50000, 32, 32, 3); test: (10000, 32, 32, 3) to train: (50000, 96, 32); test: (10000, 96, 32). Then, we set the parameters as:

Numbers of input: n_step = 96
The dimension of n_input = 32

Model Summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128)               82432
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 83,722
Trainable params: 83,722
Non-trainable params: 0
```

Part IV: Line by Line (Non-Flatten) Method

In this method, we consider each line in each channel of RGB as a cell of RNN structure. We will then handle 96 sequences of 32 timesteps for every sample. To implement this, we need firstly reshape the dataset dimensions from train: (50000, 32, 32, 3); test: (10000, 32, 32, 3) to train: (50000, 32, 96); test: (10000, 32, 96). Then, we set the parameters as:

Numbers of input: n_step = 32
The dimension of n_input = 96

Model Summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 128)               115200
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0
=================================================================
Total params: 116,490
Trainable params: 116,490
Non-trainable params: 0
```

Results:

	Train Accuracy	Test Accuracy
Model I	0.4787	0.4731
Model II	0.4577	0.4455
Model III	0.3969	0.4070
Model IV	**0.5135**	**0.5016**

From the table above, we can see after 10 epochs' training, Model IV performs the best with over 50% accuracy. Overall speaking, RNN structure is not a good fit for cifar-10 or image classification problem with generally low accuracy. A further combination of CNN and RNN may achieve far higher accuracy and this method should apply to more CV datasets for generalization validation.