Convolutional Recurrent Neural Network

The network architecture. The architecture consists of  
three parts: 1) convolutional layers, which extract a feature se-  
quence from the input image; 2) recurrent layers, which predict  
a label distribution for each frame; 3) transcription layer, which  
translates the per-frame predictions into the ﬁnal label sequence.

Feature Sequence Extraction

1. In CRNN model, the component of convolutional layers  
   is constructed by taking the convolutional and max-pooling  
   layers from a standard CNN model (fully-connected layers  
   are removed)

Such component is used to extract a sequential

feature representation from an input image.

Before being fed into the network, all the images need to be scaled  
to the same height.

Then a sequence of feature vectors is  
extracted from the feature maps produced by the compo-  
nent of convolutional layers, which is the input for the re-  
current layers. Speciﬁcally, each feature vector of a feature  
sequence is generated from left to right on the feature maps  
by column. This means the i-th feature vector is the con-  
catenation of the i-th columns of all the maps.

(Объяснение:

As the layers of convolution, max-pooling, and element-  
wise activation function operate on local regions, they are  
translation invariant. Therefore, each column of the feature  
maps corresponds to a rectangle region of the original image

and such rectangle regions are in the same order to their

corresponding columns on the feature maps from left to right.)

1. Sequence Labeling

A deep bidirectional Recurrent Neural Network is built  
on the top of the convolutional layers, as the recurrent layers.

The recurrent layers predict a label distribution yt for each frame xt in the feature sequence x = x1, . . . , xT.

Firstly, RNN has a strong capability of capturing contextual information

within a sequence. Using contextual cues for image-based sequence recognition

is more stable and helpful than treating each symbol independently.

Secondly, RNN can back-propagates error differentials to

its input, i.e. the convolutional layer, allowing us to jointly train the recurrent  
layers and the convolutional layers in a uniﬁed network.

Thirdly, RNN is able to operate on sequences of arbitrary  
lengths, traversing from starts to ends.

A traditional RNN unit has a self-connected hidden layer between its input and output layers. Each time it receives a frame xt in the sequence, it updates its internal state ht with a non-linear function that takes both current input xt and past state ht−1 as its inputs: ht = g(xt, ht−1). Then the prediction yt is made based on ht.