Nhận diện biểu thức toán học viết tay bằng phương pháp Watch - Attention - Parse

Phan Tấn Phúc - Bùi Khánh Ngọc

Ho Chi Minh City University of Technology

{phantanphuc2512, buikhanhngoc142}@gmail.com

Ngày 21 tháng 8 năm 2017

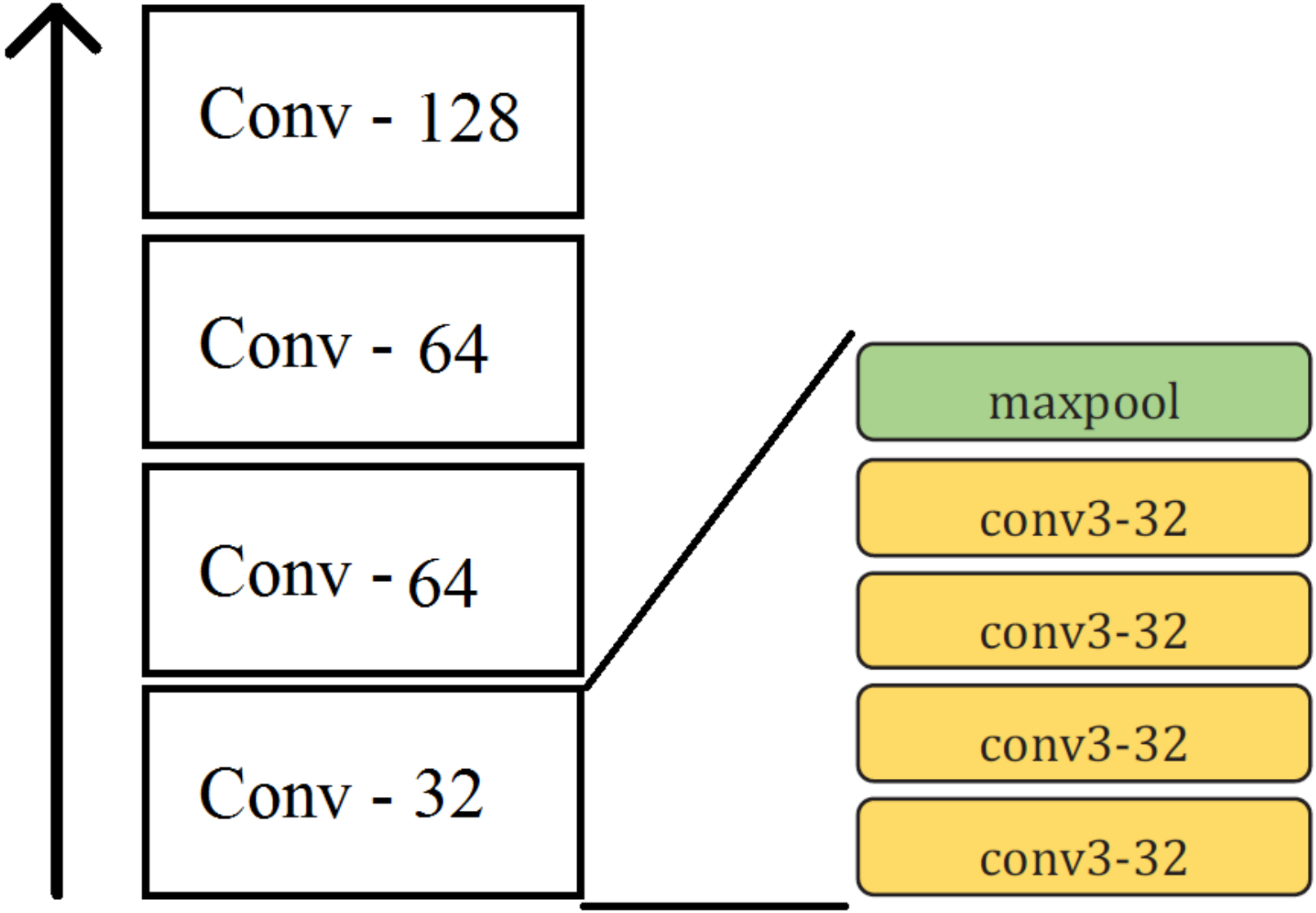
Overview

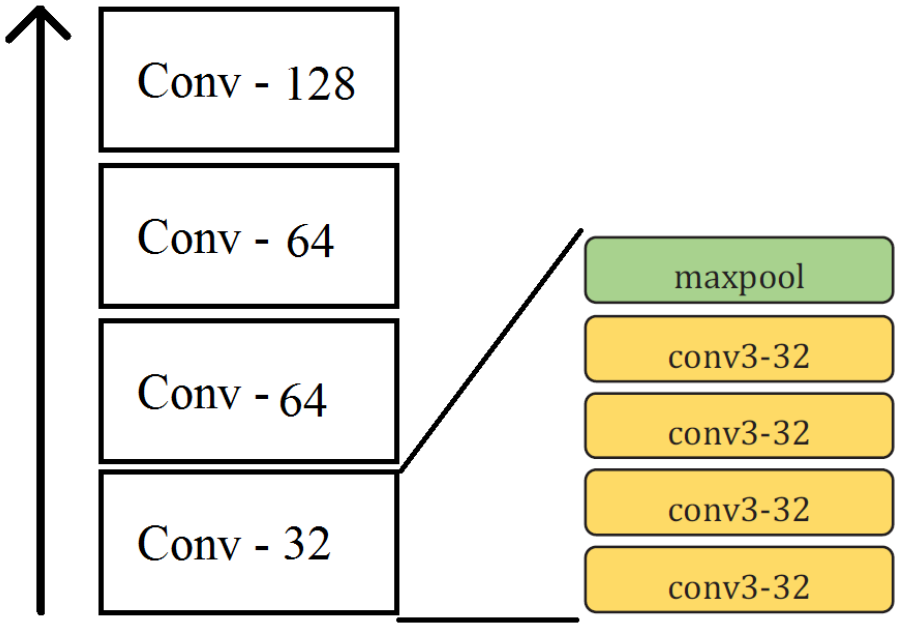
Giới thiệu Mạng nơron hồi quy (RNN)

[1] Thuật ngữ tiếng anh: Recurrent Neural Network

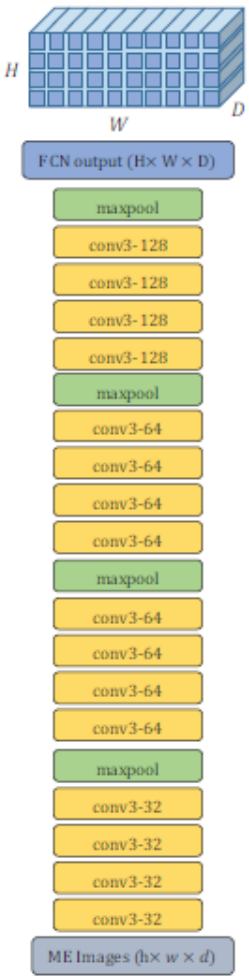
Watcher

Watcher bản chất là một hệ thống các lớp convolution tạo thành một mạng Fully Convolutional Network (FCN). Cấu trúc của khối watcher được thể hiện qua sơ đồ sau:





Trong đó, mỗi khối conv được cấu thành bởi 4 lớp convolution liên tiếp và một lớp max-pooling. Các lớp convolution này sử dụng kernel có kích thước 3×3 và sử dụng stride bắng 1 và padding bằng 1. Sau mỗi lớp convolution thì tensor sẽ được đi qua hàm relu. Sau đó, ở cuối mỗi khối Conv là một lớp max-pooling với stride bằng 2 và sử dụng kernel 2×2. Có 4 khối Conv chồng lên nhau có bản chất khá giống nhau (đều là những lớp convolution với kenel 3×3) nhưng với mỗi khối thì có số kernel khác nhau (từ đó thì đầu ra của mỗi layer trong các khối sẽ có số kênh khác nhau - 32, 64 hoặc 128)



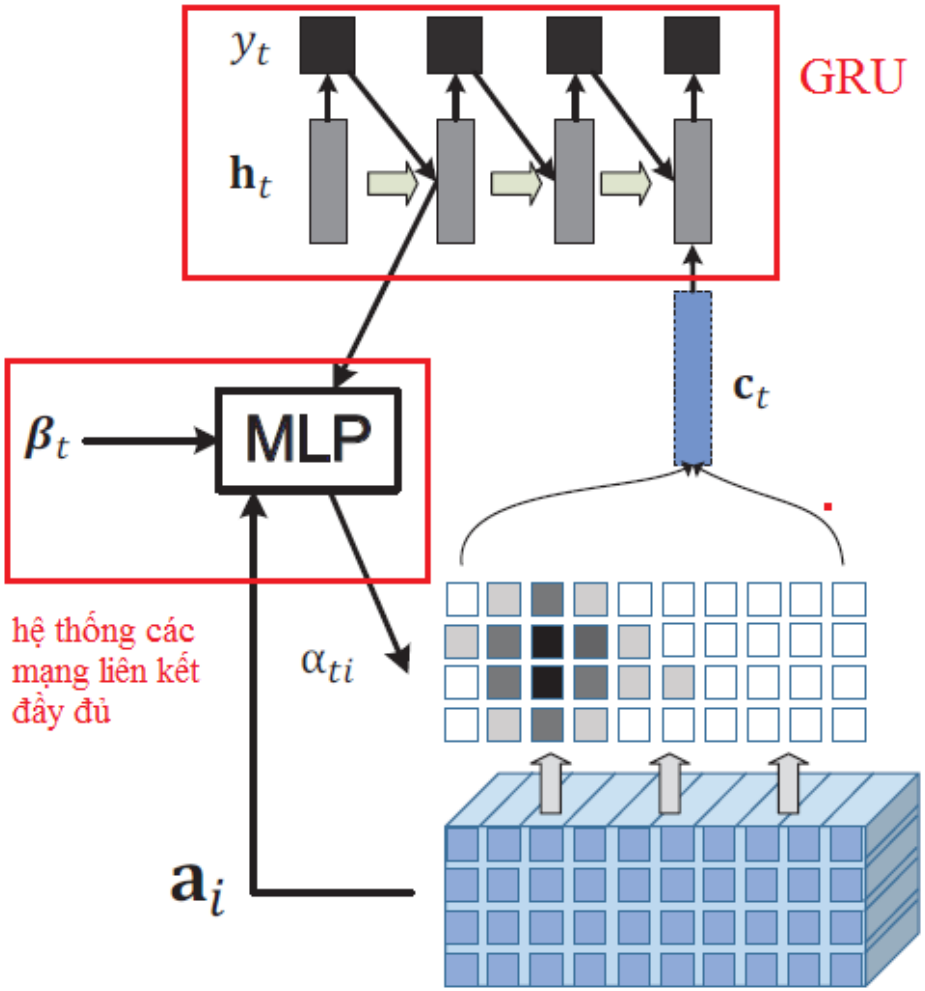
Hình: Caption1

Hình ảnh bên là mô
hình đầy đủ của khối Watcher với một số đặc điểm:

- Mạng nhận dữ liệu đầu vào là khối tensor $9 \times w \times h$. Với w, h là kích thước ảnh. 9 kênh của dữ liệu đầu vào bao gồm ảnh 8 hướng của ảnh biểu thức (chiếm 8 kênh) và chính bản thân ảnh đầu vào. Mức xám của ảnh đầu vào là 0 hoặc 1.
- Mạng cho dữ liệu đầu ra
  là một khối tensor a có kích thước $H \times W \times D$.

Attentions

Trung tâm của khối Attention là một mạng GRU biến thể kết hợp với một số lớp liên kết đầy đủ[2] được cấu trúc đặc biệt theo sơ đồ sau:



[2]Thuật ngữ tiếng anh: Fully Connected Layer

Vector C_t và ma trận α

Cơ chế Attention là từ trạng thái hiện tại, mạng sẽ sinh ra một "ảnh" α, đây là một phân phối xác suất của vị trí của ký tự tiếp theo cần được nhận diện. Ma trận α có kích thước chiều dài và rộng bằng với khối tensor a nhưng chỉ có một kênh, kênh này có các phần tử có giá trị từ $0 - 1.0$ biểu thị xác suất xuất hiện của ký tự tiếp theo cần nhận diện. Do α là một phân phối xác suất, tổng giá trị các phần tử trong α bằng 1.0.
C_t là dữ liệu đầu vào của mạng GRU, C_t được sinh ra theo các bước:

- Nhân từng vector của a (lấy theo chiều sâu) nhân với từng giá trị tương ứng trong α, giả sử ta gọi kết quả của phép nhân này là a'. Vậy ta sẽ có công thức: $a'_i = a_i \times \alpha_i$
- Từ a', ta cộng dồn các vector a'_i để được vector C_t. Ta có công thức: $C_t = \sum_i a'_i = \sum_i a_i \alpha_i$

Khối GRU

Như đã được trình bày, GRU là một biến thể của mạng RNN, vì vậy, nó có khả năng sinh ra một chuỗi từ một giá trị đầu vào (one-to-many), điều này phù hợp với bài toán nhận diện biểu thức toán học do ta phải sinh ra một biểu thức toán học gồm nhiều ký tự từ một hình ảnh.
Các cổng của GRU được hoạt động theo công thức:

$$\mathbf{z}_t = \sigma(\mathbf{W}_{yz}\mathbf{E}\mathbf{y}_{t-1} + \mathbf{U}_{hz}\mathbf{h}_{t-1} + \mathbf{C}_{cz}\mathbf{c}_t)$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_{yr}\mathbf{E}\mathbf{y}_{t-1} + \mathbf{U}_{hr}\mathbf{h}_{t-1} + \mathbf{C}_{cr}\mathbf{c}_t)$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_{yh}\mathbf{E}\mathbf{y}_{t-1} + \mathbf{U}_{rh}(\mathbf{r}_t \otimes \mathbf{h}_{t-1}) + \mathbf{C}_{cz}\mathbf{c}_t)$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \otimes \mathbf{h}_{t-1} + \mathbf{z}_t \otimes \tilde{\mathbf{h}}_t$$

Trong đó, h là hidden state của GRU, y là giá ký hiệu đã dự đoán và chỉ số dưới t chỉ vòng lặp GRU thứ t.

Dự đoán ký tự

Từ giá trị h_t ở trên, để dự đoán ký tự, ta theo công thức:
$p(y_t|x, y_{t-1}) = \sigma(W_o(Ey_{t-1} + W_h h_t + W_c c_t))$
Trong đó:
σ là hàm sigmoid.

Hệ thống các mạng liên kết đầy đủ

Hệ thống này góp phần dự đoán vị trí của ký tự cần dự đoán tiếp theo (cụ thể là tạo ra ma trận α), từ đó góp phần sinh vector c_t. Việc tạo ma trận α được sinh theo công thức:

- $e_{ti} = \nu_a^T tanh(W_a h_{t-1} + U_a a_i)$: Ở đây, ta lấy giá trị h_t đã được tính ở khối GRU cộng với vector a_i, kết quả sẽ được đi qua hàm *tanh* và tiếp tục đi qua một lớp liên kết đẩy đủ nữa để thu được giá trị e_{ti} tương ứng. Ma trận e_t sẽ có kích thước chiều dài và rộng bằng kích thước của a và chiều sâu bằng 1.
- $\alpha_{ti} = \frac{e^e_{ti}}{\sum_k e^e_{tk}}$: Sau khi tính được ma trận e_t, ta cho e_t đi qua một hàm softmax để cuối cùng thu được một phân phối xác suất, kết quả này chính là ma trận α

Coverage

Tuy nhiên, nếu chỉ sử dụng mạng với cấu trúc nêu trên, thì việc huấn luyện mạng để sinh ra alpha sẽ gặp khá nhiều khó khăn và từ đó dẫn đến một ký hiệu có thể bị tập trung nhiều lần. Từ đó, tác giả đã đưa vào một nhân tố gọi là Coverage, nhân tố này giúp mạng nắm được các trạng thái tập trung trước đó và điều chỉnh vị trí tập trung tốt hơn.
Nguyên lý hoạt động của nhân tố này là ta sẽ phải cộng dồn các ma trận α lại được ma trận β, ma trận β này sẽ góp phần sinh ra α kế tiếp thông qua việc can thiệp vào công thức: $e_{ti} = \nu_a^T tanh(W_a h_{t-1} + U_a a_i)$. Cụ thể:

- $\beta_t = \sum_l^{t-1} \alpha_l$: Cộng dồn các ma trận α.
- $F = Q * \beta_t$: trong đó, * là phép convolution, ở đây, ta sẽ cho β_t đi qua một lớp convolution, ta sẽ thu được một tensor (thường có cùng kích thước với a).
- $e_{ti} = \nu_a^T tanh(W_a h_{t-1} + U_a a_i + U_f f_i)$: Ta tính tổng các thành đã có lại như đã giải thích ở trên và thêm $U_f f_i$, kết quả sẽ đi qua hàm tanh và tiếp tục đi qua một lớp liên kết đầy đủ ν_a^T

References

Một số link tham khảo: Karpathy github:
https://gist.github.com/karpathy/d4dee566867f8291f086
LSTM - GRU:
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The End