NÂNG CAO ĐỘ CHÍNH XÁC TRONG NHẬN DẠNG CHỮ VIỆT ĐỨT, DÍNH

Nguyễn Thị Thanh Tân*
Trường Đại học Điện Lực

TÓM TẮT

Bài báo này đề xuất một giải pháp hiệu quả nhằm nâng cao độ chính xác nhận dạng các văn bản tiếng Việt chứa nhiều ký tự bị đứt, dính. Ý tưởng cơ bản của phương pháp đề xuất dựa trên việc tối ưu quá trình nhận dạng trên từng dòng văn bản, trong đó tập trung vào 3 công đoạn chính: (i) Tăng cường độ chính xác nhận dạng ký tự; (ii) Xây dựng tập lát cắt ứng cử viên rút gọn; (iii) Tối ưu hóa quá trình tìm kiếm lời giải tốt từ tập ứng cử viên. Phương pháp này đã được thử nghiệm trên ba tập dữ liệu tiếng Việt được thu thập từ thực tế với tổng số 15270 dòng văn bản, đa dạng về số lượng, chất lượng và kiểu font chữ. Kết quả thực nghiệm cho thấy phương pháp này có độ chính xác cao và ổn định trên các tập dữ liệu thử nghiệm và hoàn toàn có khả năng ứng dụng để nhận dạng những văn bản đầu vào có chất lượng xấu.

Từ khóa: *Thành phần liên thông; đoạn ảnh chữ; ký tự bị đứt, dính; lát cắt ứng cử viên rút gọn; cắt nhỏ; lát cắt sai; lát cắt nghi ngờ; nhận dạng; phân lớp;độ tin cậy; qui hoạch động; mạng nơron; học sâu; Convolutional Neural Networks; Convolutions; pooling; subsampling*

MỞ ĐẦU

Nhận dạng chữ là quá trình chuyển đổi từ dạng hình ảnh của một hay nhiều trang ảnh chứa các thông tin văn bản thành tệp văn bản thực sự có thể soạn thảo được trên máy tính. Độ chính xác của hệ thống nhận dạng phụ thuộc rất nhiều vào chất lượng của ảnh cần đầu vào [0], [0], [0]. Một ảnh văn bản được coi là có chất lượng tốt nếu hầu hết các ký tự trên đó là rõ ràng, không bị dính, biến dạng, không đứt hay mất nét. Ngược lại, một ảnh văn bản có chất lượng thấp nếu trên đó xuất hiện nhiều ký tự bị dính, biến dạng, bị đứt hay mất nét mà rất khó có thể phân tách chúng một cách chính xác bằng những phương pháp thông thường (chẳng hạn chỉ thuần túy dựa trên histogram hoặc dựa trên phân tích các thành phần liên thông).

Bài báo này đề xuất một giải pháp hiệu quả nhằm nâng cao độ chính xác nhận dạng dựa trên việc tối ưu quá trình nhận dạng trên từng dòng văn bản, trong đó tập trung vào 3 công đoạn chính: i) Tăng cường độ chính xác nhận dạng ký tự; ii) Xây dựng tập lát cắt ứng cử viên rút gọn; iii) Tối ưu hóa quá trình tìm kiếm lời giải tốt từ tập ứng cử viên.

PHƯƠNG PHÁP NGHIÊN CỨU

Ý tưởng cơ bản của phương pháp như sau: Từ mỗi dòng văn bản cần nhận dạng, bước xử lý đầu tiên sẽ tiến hành xác định tập lắt cắt ứng cử viên rút gọn, bước xử lý tiếp theo sẽ tiến hành xây dựng đồ thị phân đoạn định hướng DSG (Directed Segmentation Graph) từ tập lát cắt ứng cử viên rút gọn. Trong đó, mỗi đỉnh của đồ thị biểu diễn một vùng ảnh tương ứng với một lát cắt ứng cử viên, mỗi cạnh nối giữa hai đỉnh của đồ thị được gán nhãn bằng một giá trị xác suất tính được từ một mô hình ngôn ngữ tri-gram mức ký tự dựa trên kết quả nhận dạng (phân lớp) ký tự tương ứng tại các đỉnh. Cuối cùng, việc nhận dạng dòng văn bản được đưa về bài toán tìm kiếm đường đi tốt nhất trên đồ thị DSG.

Thuật toán xây dựng tập lát cắt ứng cử viên rút gọn

Tiêu chí chính của việc xây dựng tập lát cắt ứng cử viên ở đây là bao phủ được tất cả các lát cắt có thể có trên các phần chữ bị dính và hạn chế các lát cắt dư thừa trên các ký tự tốt. Thuật toán xây dựng tập lát cắt ứng cử viên rút gọn được mô tả cụ thể:

Input: *LineImage* (dòng ảnh văn bản đầu vào).
Output: + *CG_ List*: Danh sách các đoạn ảnh chữ.
+ *Cut_List*: Tập lát cắt ứng cử viên
1. *CC_List* ← **CC_Analysis** (LineImage);
2. *CC_ List* ← **Sort_By_LeftTop** (CC_List);
3. *CC* ← **GetNext** (CC_ List);
4. *New_CG* ← CC;
6. while *CC_ List* is not empty do
6.1 CC ← **GetNext** (CC_ List);

*Tel: 0987 279077, Email: thanhtan.nt@gmail.com

6.2 if **Intersect_X**(New_*CG*, *CC*) = True OR
Intersect(New_*CG*, *CC*) = True OR
Inter_Sub (New *CG*,*CC*) = True then
Merge (New_CG, CC);
6.3 else
6.3.1 n_count ← n_count + 1;
6.3.2 **AddTo**(CG_List, New_CG) ;
6.3.3 New_CG ← CC;
7. Cut_List ← **FindCandCut** (*CC_ List*)
8. for each *Cand_Cut* in *Cut_List* do
8.1 *Image* ← **GetImage** (Cand_Cut)
8.2 if **IsTouching**(Image) then
8.2.1 *A_List* ← **OversegCut**(Image);
8.2.2 *Cut_List*←*Cut_List* ∪ *A_List*
9. Return *CG_ List*, *Cut_List*;

Thuật toán bắt đầu bằng bước phân tích và ghép nhóm các thành phần liên thông thành các đoạn ảnh chữ. Bản chất của việc phân tích, ghép nhóm các thành phần liên thông là quá trình "hợp" các thành phần liên thông thành các đoạn ảnh chữ chữ trước khi "chia nhỏ" chúng.

Hàm **CC_Analysis(***LineImage***)** xác định và phân tích các thành phần liên thông trên dòng ảnh đầu vào. Hàm **Sort_By_LeftTop(**CC_List**)** tiến hành sắp xếp các thành phần liên thông theo tọa độ góc trên bên trái các hình bao của chúng. Hàm **Intersect**(CG, CC) nhận giá trị đúng (=1) khi và chỉ khi $Z_1 \cap Z_2 \neq \phi$, trong đó Z_1 và Z_2 lần lượt là hình bao của đoạn ảnh chữ CG và thành phần liên thông CC đang xét, $\cap$ là phép giao của hai hình bao, giả sử $Z = Z_1 \cap Z_2$ thì $(Z \subseteq Z_1) \wedge (Z \subseteq Z2)$. Hàm **Intersect_X**(CG, CC) nhận giá trị đúng (=1) khi và chỉ khi tọa độ các hình bao tương ứng của chúng thỏa mãn: $[(t_1 \geq b_2) \vee (t_2 \geq b_1)] \wedge (l_1 \leq l_2 \leq r_1)$ trong đó (l_1, t_1, r_1, b_1) và (l_2, t_2, r_2, b_2) lần lượt là tọa độ góc trái trên, phải dưới của hình bao đoạn ảnh chữ CG và thành phần liên thông CC.

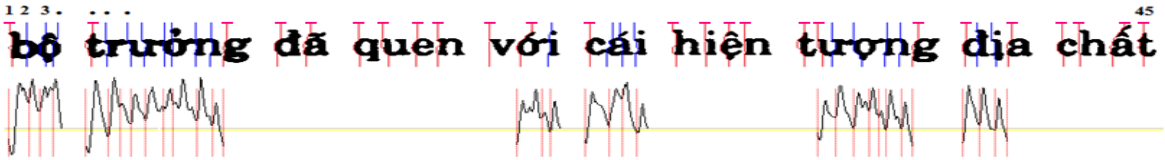
Hàm **Inter_Sub**(CG, CC) nhận giá đúng (=1) khi và chỉ khi $(b_2\text{-}t_2) \leq (1/k)\times(b_1\text{-}t_1)$, trong đó k là một giá trị ngưỡng cho trước. Hàm **FindCandCut** (*CC_ List*) dùng để xác định tập lát cắt ứng cử viên ban đầu dựa trên biên của các đoạn ảnh chữ trong tập *CC_ List*. Hàm **GetImage** (Cand_Cut) trả về vùng ảnh tương ứng với lát cắt Cand_Cut đang xét. Hàm **IsTouching**(Image) nhận giá trị đúng (=1) nếu ảnh đầu vào Image được xác định là vùng ảnh chứa các ký tự bị dính (có kích thước lớn hơn ngưỡng tiêu chuẩn và độ tin cậy của kết quả nhận dạng thấp). Hàm **OversegCut**(Image) có nhiệm vụ bổ sung thêm các lát cắt mới cho đoạn ảnh chữ Image.

Quá trình bổ sung thêm lát cắt cho đoạn ảnh chữ *Image*, kích thước *w*×*h* bất kỳ được bắt đầu bằng việc xây dựng các đường đi tối ưu qua mỗi vị trí x_i ($0 \leq x_i < w$). Đường đi qua một vị trí x_i, kí hiệu $SP(x_i)$, được định nghĩa như sau:

$$SP(x_i) = \left\{ \left(x_i^{'}, i\right) \mid i = \overline{0, h-1} \wedge x_0^{'} = x_i \wedge x_i^{'} = x_i \pm \Delta x \right\}$$

Mỗi đường đi $SP(x_i)$ được xây dựng theo phương pháp qui hoạch động. Mục tiêu của thuật toán là cực tiểu hóa hàm chi phí $F(SP(x_i))$, được xây dựng dựa trên ý tưởng kế thừa các ưu điểm các điểm cực tiểu trên biểu đồ tần suất dọc [0] và các lát cắt cong gần với cạnh biên trái của các thành phần liên thông [0].

$$F\left(SP(x_i)\right) = Cost\left(x_i^{'}, i\right)$$



Hình 1. *Xây dựng tập lát cắt ứng cử viên rút gọn*

$$Cost\left(x_i^{'}, i\right) = Cost\left(x_{i-1}^{'}, i-1\right) + W\left(x_{i-1}^{'}, i-1; I\right) + C(x_{i-1}^{'}, i-1, x_i^{'}, i; I)$$

Trong đó $\left(x_{i-1}^{'}, i-1\right)$ là điểm liền trước của điểm $\left(x_i^{'}, i\right)$ trên đường đi đang xét, $W\left(x_{i-1}^{'}, i-1; I\right)$ là trọng số của điểm đó và $C(x_{i-1}^{'}, i-1, x_i^{'}, i; I)$ là chi phí từ điểm $\left(x_{i-1}^{'}, i-1\right)$ đến điểm $\left(x_i^{'}, i\right)$.

Tập các lát cắt ứng cử viên trên đoạn ảnh chữ đầu vào lúc này được định nghĩa như sau:

$$Cut(i) = SP\left(\arg\min_{x\in[x_i-\Delta x, x_i+\Delta x]}(F(SP(x)))\right)$$

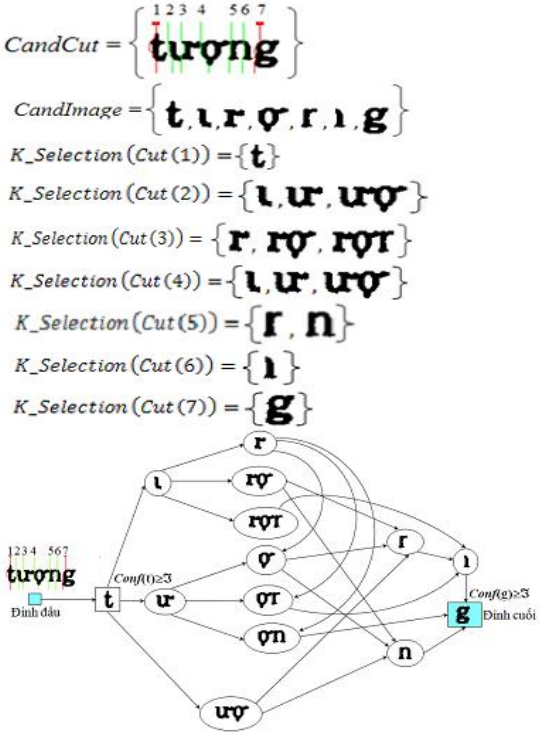
$$CandCuts = \{Cut(i) \mid i = \overline{1,n}\}$$

Trong đó giá trị Δx xác định phạm vi xác định cực tiểu, được lựa chọn từ thực nghiệm ($\Delta x = 3$).

Hình 1 chỉ ra một tập gồm 45 lát cắt ứng cử viên $CandCuts = \{Cut(i) \mid i = \overline{1,45}\}$, xác định được từ một dòng ảnh đầu vào thực tế. Các lát cắt với đầu mút hình chữ nhật phía trên được xác định dựa vào các biên trái của các đoạn ảnh chữ, các lát cắt còn lại là các lát cắt được bổ sung cho từng đoạn ảnh chữ bị dính.

Xây dựng đồ thị phân đoạn DSG

Định nghĩa Đồ thị DSG: Gọi $CandCuts = \{Cut(i) \mid i = \overline{1,n}\}$ là tập gồm n lát cắt ứng cử viên cho trước và $\text{CandImage} = \{SI(i) \mid i = \overline{1,n}\}$ là tập các đoạn ảnh chữ tương ứng. Đồ thị DSG là một đồ thị có hướng, được thể hiện bằng một tập đỉnh V và một tập cạnh E, trong đó:



Hình 2. *Xây dựng đồ thị DSG*

v_0: Là đỉnh đầu, các đỉnh còn lại biểu diễn các đoạn ảnh chữ trong tập tập các giải pháp có thể lựa chọn từ lát cắt i, được ký hiệu là $K\_Selection\ (Cut(i) \mid i = \overline{1,n})$. Giả sử $SI' = \bigcup SI(x+i)$ là khối ảnh thứ j trong tập $K\_Selection\ (Cut(i))$. Khi đó một đỉnh v' thể hiện khối ảnh SI' sẽ có đường đi tới một đỉnh v'' thể hiện một khối ảnh SI'' bất kỳ nếu và chỉ nếu $S'' \in K\_Selection\ (Cut\ (j+1))$.

Một đỉnh không có cạnh đi ra sẽ được gọi là một *đỉnh cuối* của đồ thị DSG.

Hình 2 chỉ ra một đồ thị DSG được xây dựng cho tập đầu vào gồm 7 lát cắt ứng cử viên. Trong đó, các đỉnh hình chữ nhật thể hiện các ảnh ký tự đầy đủ, các đỉnh hình tròn thể hiện các phân đoạn ảnh bị đứt hoặc được cắt ra từ các đoạn ảnh chữ bị dính. Đồ thị có duy nhất một đỉnh cuối, thể hiện phân đoạn ảnh SI(7).

Nhận dạng dòng văn bản

Ở bước này, bài toán nhận dạng dòng văn bản đã được đưa về bài toán tìm đường đi "tốt nhất" trên đồ thị DSG. Quá trình tìm kiếm được định hướng bởi hàm mục tiêu **f(.)**, được xây dựng dựa trên kết quả nhận dạng các phân đoạn ảnh và các thông tin ngữ cảnh liên quan. thông tin ngữ cảnh liên quan đến mỗi trạng thái tìm kiếm u_k được thể hiện bằng xác suất xuất hiện của ký tự nhận dạng được tại trạng thái đó trong xâu ký tự đã được sinh ra trên đường đi từ trạng thái ban đầu u_0 đến trạng thái liền trước của u_k. Các độ đo xác suất ở đây được ước lượng bằng các mô hình tri-gram mức ký tự [0].

Xây dựng hàm đánh giá: Gọi SI^{u_i} là tổ hợp phân đoạn ảnh được thể hiện bởi trạng thái u_i, $ch_i = Classify(SI^{u_i})$ là kết quả phân lớp của SI^{u_i}, $S(u_i)$ là chuỗi ký tự nhận dạng được trên đường đi từ trạng thái khởi tạo u_0 đến trạng thái u_i:

Tại trạng thái khởi tạo $u_0 = v_0$:

$$S(u_0) = ch_0 = EMPTY;$$

$$f(u_0) = g(u_0) = \log[P(ch_0)] = 0;$$

Xét trạng thái thái tìm kiếm u_1 tiếp theo:

$$ch_1 = Classify(SI^{u_1}); \quad S(u_1) = ch_0 ch_1;$$

$$f(u_1) = g(u_1) = \log[P(ch_0 ch_1)]$$

$$= \log[P(ch_0) \times P(ch_1 \mid ch_0)] =$$

$$= \log[P(ch_0)] + \log[P(ch_1 \mid ch_0)]$$

Xét trạng thái tìm kiếm u_2 kế tiếp:

$ch_2 = Classify(SI^{u_2})$; $S(u_2) = ch_0 ch_1 ch_2$;

$f(u_2) = g(u_2) = \log[P(ch_0 ch_1 ch_2)]$

$= f(u_0) + \log[P(ch_1 \mid ch_0)]$

$= \log[P(ch_0) \times P(ch_1 \mid ch_0) \times P(ch_2 \mid ch_1 ch_0)] =$

$= \log[P(ch_0) \times P(ch_1 \mid ch_0)] + \log[P(ch_2 \mid ch_1 ch_0)]$

$= f(u_1) + \log[P(ch_2 \mid ch_1 ch_0)]$

Như vậy, trong trường hợp tổng quát, đối với một trạng thái tìm kiếm u_k bất kỳ ta sẽ có:

$ch_k = Classify(SI^{u_k})$; $S(u_k) = S(u_{k-1}) \oplus ch_k$;

$f(u_k) = f(u_{k-1}) + \log[P(ch_k \mid ch_{k-1} ch_{k-2})]$

Phép toán $\oplus$ thể hiện phép thêm một ký tự vào xâu. Thuật toán tìm kiếm được mô tả cụ thể:

Input:
- (V,E): Đồ thị DSG, u_0: Trạng thái khởi tạo, goal_set: Tập đỉnh cuối của đồ thị.
- NGram_Character_Models: Mô hình ngôn ngữ mức chữ cái.
- $\Im$: Ngưỡng độ tin cậy phân lớp ký tự tối thiểu cho phép.

Output: Đường đi tốt nhất trên đồ thị (Best_Path).

1. Khởi tạo:
   - *Best_Cost* ← -INFINITY; New_Cost←0;
   - $f(u_0) \leftarrow 0$; $S(u_0) \leftarrow \varnothing$; $Parent(u_0) \leftarrow \varnothing$; {f(u) ← -INFINITY |∀u∈V};
2. OPEN← Push(u_0, $f(u_0)$, $S(u_0)$);
3. While OPEN is not empty do
   - 3.1 (u_m, $f(u_m)$, $S(u_m)$) ← Pop(OPEN);
   - 3.2 if $u_m \in$ *goal_set* then thực hiện bước 4;
   - 3.3 for each u_i in Slibling(u_m)
     - 3.3.1 SI ← **Get_Combination_IMG**(u);
     - 3.3.2 (ch, conf) ← **Recognize**(SI);
     - 3.3.3 if conf ≥ $\Im$ then
       - if Parent(u_m) = ∅ then $ch_{m-1} \leftarrow \varnothing$;
       - else $ch_{m-1} \leftarrow$ **Get_Recog_Result**(Parent(u_m))
       - $ch_m \leftarrow$ **Get_Recog_Result**(u_m);
       - New_Cost ← $f(u_m) + \left|\log[P(ch \mid ch_{m-1} ch_m)]\right|$;
       - if $u_i \in$ *OPEN* AND $f(u_i) >$ *New_Cost* then
       - Begin
         - $f(u_i) \leftarrow$ *New_Cost*;
         - $Parent(u_i) \leftarrow u_m$;
         - $S(u_i) \leftarrow$ **Add_ To_String**($S(u_m)$, *ch*);
       - End
       - If $u_i \notin$ *OPEN* then
       - Begin
         - $f(u_i) \leftarrow$ *New_Cost*;
         - $Parent(u_i) \leftarrow u_m$;
         - $S(u_i) \leftarrow$ **Add_ To_String**($S(u_m)$, *ch*);
         - OPEN ← **Push**(u_i, $f(u_i)$, $s(u_i)$);
       - End
4. Best_Path← (u_m, Parent(u_m),...,u_0)
5. Return Best_Path;

Hàm **Add_ To_String**($S(u)$, *ch*) trả về một chuỗi ký tự mới bằng cách thêm ký tự *ch* vào chuỗi S(u). Hàm **Get_Combination_IMG**(u) trả về đoạn ảnh chữ được biểu diễn bởi trạng thái u. Hàm **Recognize**(SI) nhận dạng đoạn ảnh chữ *SI*. Kết quả trả về của hàm là cặp giá trị (ch, conf), trong đó *ch* là ký tự nhận dạng được và *conf* là độ tin cậy của kết quả nhận dạng. Hàm **Get_Reco_Result**(u) trả về ký tự nhận dạng được tại trạng thái đang xét u.

P(t) =0.433; P(l|t) = 0.0002;

P(u|t) = 0.253; P(r|lt) = 0.000015;

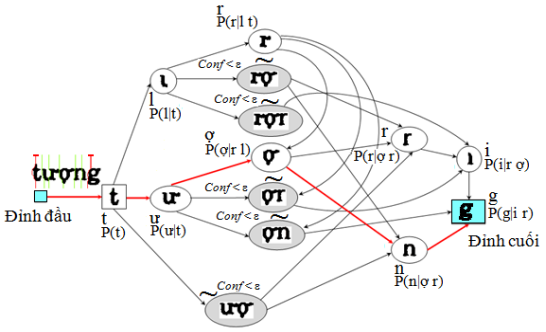
P(ợ|r l) = 0. 00013; P(r|ợ r) = 0.00017;

P(i|r ợ) = 0.00003; P(g|i r) = 0.034;

P(n|ợ r) = 0.254; P(g|n ợ) = 0.223;

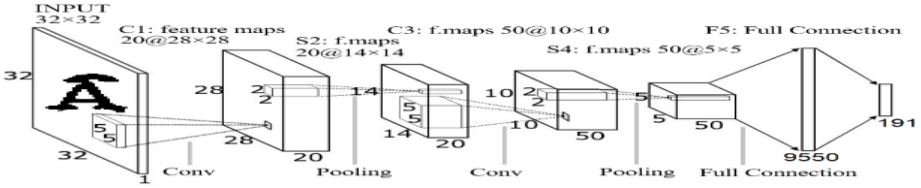
P(ợ|ư t) = 0.218; P(r|ợ ư) = 0.00023;

P(i|r ợ) = 0.000053; P(n |ợ ư) = 0.348



Hình 3. *Tìm kiếm đường đi tốt nhất trên DSG*

Hình 3 thể hiện kết quả của quá trình tìm kiếm đường đi tốt nhất trên đồ thị DSG ở trên. Xác suất của các n-gram tương ứng được cho ở bên trái của hình vẽ. Đường đi tốt nhất tìm được là đường chứa với các mũi tên màu đỏ, đậm. Các đỉnh màu xám là các đỉnh bị loại trong quá trình tìm kiếm.



Hình 4. *Mô hình mạng CNN nhận dạng ký tự*

Bộ nhận dạng ký tự

Để nhận dạng ký tự, chúng tôi sử dụng mô hình mạng tích chập (CNN Convolutional Neural Networks) [0], [0], [**Error! Reference source not found.**] được mô tả cụ thể trên Hình 4. Từ ảnh đầu vào cần nhận dạng, trước tiên sẽ được chuẩn hóa về kích thước 32×32. Mạng được thiết kế bao gồm 5 lớp (không kể lớp đầu vào, đầu ra), bao gồm hai lớp tích chập (Convolutions) C1 và C3, hai lớp lọc pooling (subsampling) S2 và S4, lớp kết nối đầy đủ F5. Lớp Các lớp tích chập kết hợp với các lớp phi tuyến sử dụng các hàm phi tuyến như ReLU để tạo ra thông tin trừu tượng hơn cho các lớp tiếp theo. Các lớp lọc pooling/subsampling dùng để lọc lại các thông tin hữu ích hơn bằng cách loại bỏ các thông tin nhiễu. Trong suốt quá trình huấn luyện, CNNs sẽ tự động học các tham số cho các lớp. Lớp cuối cùng được gọi là lớp kết nối đầy đủ (Fully connect layer) dùng để phân lớp.

Thông tin cụ thể của các lớp mạng như sau: Các lớp tích chập C1 sử dụng 20 ma trận chập, kích thước 5×5, cửa sổ trượt (stride) kích thước 2×2. Lớp S2 sử dụng bộ lọc kích thước 2×2, với hàm AveragePooling (lấy giá trị trung bình). Lớp C3 sử dụng 50 ma trận chập, kích thước 5×5, cửa sổ trượt (stride) kích thước 2×2. Lớp S4 sử dụng bộ lọc kích thước 2×2. Lớp cuối cùng F5 được kết nối đầy đủ với 191 neural đầu ra, tương ứng để nhận dạng 191 lớp ký tự, bao gồm các chữ số (0 đến 9), các chữ cái hoa/thường có dấu và không dấu, các kí hiệu thường xuyên xuất hiện trên văn bản như {. , - : ; “ ” ‘ ’ ( ) # $ @ ! % ? /}. Trong đó, không có sự phân biệt hoa/thường đối với một số cặp ký tự, chẳng hạn như: “c/C, o/O, p/P, s/S, v/V, w/W, x/X, z/Z, ơ/Ơ”.

Để thử nghiệm mô hình, chúng tôi đã tiến hành huấn luyện mạng với tổng số 1.530.000 mẫu trên 191 lớp ký tự. Các mẫu huấn luyện được trích chọn từ các nguồn văn bản, sách, báo, đa dạng về chất lượng và kiểu font chữ (.VnTime, Times New Roman, Arial, Tahoma, Courier New, Verdana – với các thuộc tính normal, normal bold, italic, italic bold).

KẾT QUẢ THỰC NGHIỆM

Dữ liệu thực nghiệm

Hiệu quả của thuật toán được kiểm nghiệm, đánh giá trên ba tập dữ liệu thử nghiệm, bao gồm:

+ Dataset 1: Chứa tổng số 6750 dòng ảnh văn bản tiếng Việt chữ thường có chất lượng thấp, đa dạng về kiểu font chữ, kích cỡ và chất lượng.

+ Dataset 2: Bao gồm tổng số 4920 dòng văn bản tiếng Việt chữ hoa có chất lượng thấp, được in từ nhiều kiểu font chữ với kích cỡ và mức độ đậm, nhạt khác nhau.

+ Dataset 3: Bao gồm tổng số 3600 dòng văn bản chữ nghiêng, được in từ nhiều kiểu font chữ với kích cỡ và mức độ đậm, nhạt khác nhau.

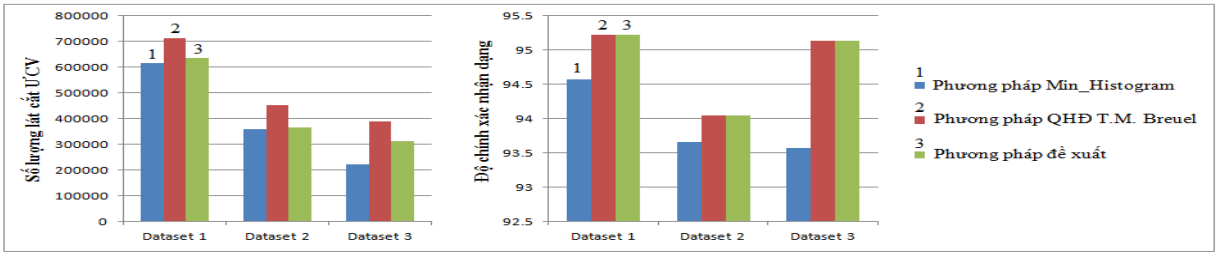
Kết quả thực nghiệm

Mục tiêu của quá trình thực nghiệm ở đây nhằm đánh giá hai độ đo hiệu quả cơ bản: Số lượng các lát cắt ứng cử viên đã sinh ra và độ chính xác nhận dạng. Các kết quả thực nghiệm của phương pháp đề được đối sánh với các kết quả thực nghiệm của phương pháp xác định tập lát cắt ứng cử viên dựa vào các điểm cực tiểu trên biểu đồ tần suất theo chiều dọc của khối chữ (*Min_Histogram*) và phương pháp qui hoạch động được đề xuất bởi T.M. Breuel [0] (*QHĐ T.M. Breuel*). Kết quả thực nghiệm của các phương pháp được thể hiện trên Hình 5. Từ các kết quả thực nghiệm cho thấy đối với phương pháp Min_ Histogram: Mặc dù số lượng các lát cắt ứng cử viên tìm được theo phương pháp này là ít hơn số lát cắt tìm được của *phương pháp đề xuất* trên cả ba tập dữ liệu thử nghiệm. Tuy nhiên, phương pháp này lại bị trả giá bởi độ chính xác nhận dạng do một số lát cắt nghi ngờ đã bị bỏ qua. Ngoài ra, đối với tập chữ in nghiêng, các điểm cắt ứng cử viên tìm được thường lệch đi một số vị trí so với các điểm cắt nghi ngờ dẫn tới các phân đoạn ảnh thu được từ các lát cắt đó thường bị thiếu hoặc thừa ra những phần ảnh nào đó (do dính vào ký tự bên cạnh hoặc ký tự bên cạnh dính vào) làm giảm độ chính xác nhận dạng.

KẾT LUẬN

Bài báo này đã đề xuất một giải pháp tổng thể để nâng cao chất lượng nhận dạng văn bản tiếng Việt chứa nhiều ký tự bị đứt, dính. Điểm mạnh của phương pháp đề xuất được thể hiện ở chỗ đã kết hợp thực hiện luân phiên và đan xen giữa các thao tác cắt và nhận dạng ký tự, các lát cắt được lựa chọn dựa trên kết quả

nhận dạng và các thông tin ngữ cảnh liên quan, đảm bảo luôn chọn được kết quả nhận dạng tốt nhất một cách toàn cục. Hiệu quả của mô hình đã được đánh giá trên trên ba tập dữ liệu tiếng Việt được thu thập từ thực tế, đa dạng về số lượng, chất lượng và kiểu font chữ. Kết quả thực nghiệm cho thấy phương pháp này có độ chính xác cao và ổn định trên các tập dữ liệu thử nghiệm. Hiện tại mô hình này đã được ứng dụng trong sản phẩm nhập liệu tự động thông tin trên chứng minh nhân dân đang được triển khai tại tổng công ty Mobile Phone.



Hình 5. *Kết quả thực nghiệm*

TÀI LIỆU THAM KHẢO

1. A. Poznanski and L. Wolf. Cnn-n-gram for handwriting word recognition. In CVPR, 2016.

2. Cheriet.M, Kharma.N.N, Liu.C.L, Suen.C, Character Recognition Systems: A Guide for Students and Practitioners, Wiley, October 2007.

3. Partha Pratim Roy, Umapada Pal, Josep Lladós, "Multi-Oriented and Multi-Sized Touching Character Segmentation Using Dynamic Programming", ICDAR, 2009, pp. 11-15.

4. M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. IJCV, 116(1):1–20, 2016.

5. Tanzila Saba et. al., "Non-Linear Segmentation of Touched Roman Characters Based on GA", Int. Journal on Computer Science and Engineering, vol. 02, No. 06, 2010, pp 2167-2172.

6. T. M. Breuel, "Segmentation of handprinted letter strings using a dynamic programming algorithm", ICDAR, 2001, pp. 821-826.

7. Ventzislav, "Using Critical Points in Contours for Segmentation of Touching Characters", in Proc. of the 5th Int. Conf. on Computer Systems and Technologies, 2004.

8. Wang, Tao, Wu, David J, Coates, Adam, and Ng, Andrew Y. Endto-end text recognition with convolutional neural networks. In ICPR, pp. 3304–3308. IEEE, 2012.

SUMMARY

IMPROVING THE ACCURACY OF VIETNAMESE OPTICAL TOUCHING AND BREAKING CHARACTER RECOGNITION

Nguyen Thi Thanh Tan*
Electric Power University

This paper propose an efficient method for improving the accuracy of Vietnamese optical touching and breaking character recognition. Basically, the propose method focus on three main step: i) improving the accuracy of character classification algorithm; ii) Determining the optimal set of the cut candidate; iii) Optimizing the searching the best result from cut candidate. The performance of this method has been verified on three Vietnamese data sets, collected from reality with a total of 15270 lines of text, diverse in number, quality and font type. Experimental results show that this method has high accuracy and stability on experiment data sets and is fully capable of recognize poor quality input texts.

Keywords: *Connected component; segment; breaking optical character; touching optical character; candidate cut; neural network; deep learning; Convolutional Neural Networks; Convolutions; pooling; subsampling*

Ngày nhận bài: 05/02/2018; Ngày phản biện: 24/02/2018; Ngày duyệt đăng: 05/3/2018

* *Tel: 0987 279077, Email: thanhtan.nt@gmail.com*