Handwritten OCR - Recognitio

Nguyễn Thế An
Trường Công nghệ Thông tin và Truyền thông
Đại học Bách Khoa Hà Nội

Nguyễn Thế Minh Đức
Trường Công nghệ Thông tin và Truyền thông
Đại học Bách Khoa Hà Nội

Nguyễn Mạnh Dương
Trường Công nghệ Thông tin và Truyền thông
Đại học Bách Khoa Hà Nội

Lê Đức Minh
Trường Công nghệ Thông tin và Truyền thông
Đại học Bách Khoa Hà Nội

Nguyễn Quang Tri
Trường Công nghệ Thông tin và Truyền thông
Đại học Bách Khoa Hà Nội

1 Tổng quan giải pháp

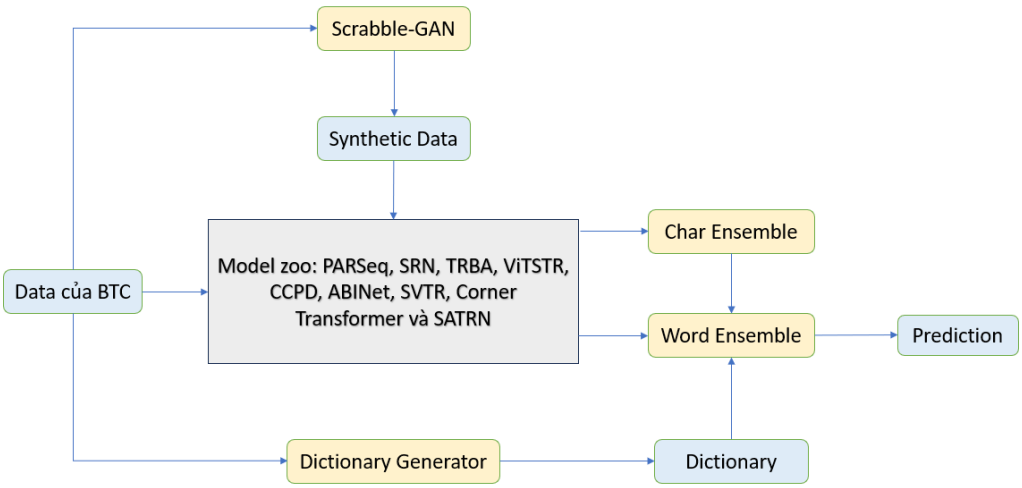


Figure 1: Tổng quan phương pháp

Giải pháp của chúng tôi là sự kết hợp của nhiều model với kiến trúc được tham khảo từ các paper như **PARSeq** [4], **SRN** [17], **TRBA** [3], **ViTSTR** [2], **CCPD** [6], **ABINet** [9], **SVTR** [7], **Corner Transformer** [16] và **SATRN** [12]. Để tăng tốc độ hội tụ cũng như tính tổng quát hóa của một số mô hình, chúng tôi sử dụng **Scrabble GAN** [10] để sinh thêm 100,000 ảnh. Bên cạnh **Cross Entropy loss**, **CTC loss**, chúng tôi sử dụng thêm **Label Smoothing** và **Focal Loss** nhằm cải thiện chất lượng model. Ngoài ra, chúng tôi cũng đề xuất một module mới đếm dấu câu để hỗ trợ mô hình trong nhận dạng dấu câu và một cách đánh nhãn mới cho từ tiếng Việt.

Ở bước ensemble, chúng tôi thực hiện 2 cách ensemble khác nhau. Cách đầu tiên là thực hiện voting cho từng chữ cái một. Cách thứ hai là max voting dựa trên confidence của model, mức độ ngữ nghĩa của từ được dự đoán và output của cách ensemble đầu tiên.

Preprint. Under review.

Giải pháp của chúng tôi cho thấy độ hiệu quả với bài toán OCR khi đạt được CER trên tập valid là 0.0302 và 0.0317 trên tập public test và 0.0915 trên tập private test. Đặc biệt tất cả model của chúng tôi đều được khởi tạo ngẫu nhiên và train từ đầu, trừ một số mô hình sử dụng **ViT** [5] thì weights sẽ được khởi tạo sử dụng pretrain trên tập **ImageNet** [13]. Tổng quan phương pháp của chúng tôi thể hiện ở **Hình 1**.

2 Validation

Dữ liệu huấn luyện BTC cung cấp gồm 103,000 ảnh trong đó gồm 51,000 ảnh form, 48,000 ảnh wild và 4,000 ảnh GAN. Do tập public test chỉ gồm ảnh form và ảnh wild nên tập validation của chúng tôi không có ảnh GAN. Cụ thể tập validation gồm 9,900 ảnh với 5,100 ảnh form và 4,800 ảnh wild được sample ngẫu nhiên như Hình 2. Tập validation sẽ được sử dụng để đánh giá độ hiệu quả của các model và lựa chọn model tham gia quá trình ensemble.

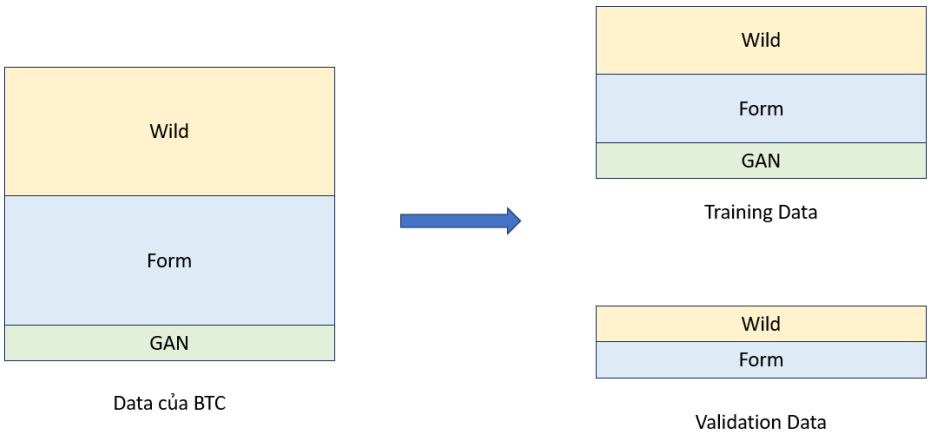


Figure 2: Chia tập train và val

3 Sinh dữ liệu

Sau khi phân tập dữ liệu BTC cung cấp thành tập training và validation, chúng tôi nhận thấy sự thiếu hụt của tập training dẫn đến các model không thể học tốt được phân phối của dữ liệu. Vì thế, chúng tôi đề xuất tham khảo kiến trúc **Scrabble Gan** [10], là một mô hình nổi tiếng với khả năng sinh dữ liệu chữ viết tay.

Chúng tôi huấn luyện model trong 15 epochs với dữ liệu BTC cung cấp. Trong quá trình huấn luyện, random noise được đưa vào model để làm đa dạng ảnh được sinh ra. Ngoài ra, tác giả của cấu trúc có cung cấp pretrained model với bộ **IAM** - chữ viết tay tiếng Anh, tuy nhiên, cân nhắc sự khác nhau giữa tiếng Anh và tiếng Việt cũng như tính đặc thù của dấu câu, chúng tôi quyết định train lại từ đầu để tránh hiện tượng nhũng nhiễu trong phân phối của 2 thứ tiếng.

Cuối cùng chúng tôi sử dụng model sinh ra 221,442 ảnh. Tuy nhiên do giới hạn về bộ nhớ trên Kaggle nên trong quá trình huấn luyện, chúng tôi chỉ sử dụng 100,000 ảnh trong số các ảnh được sinh ra. Một số model sẽ sử dụng thêm data synthetic trong quá trình huấn luyện và ngược lại một số model chỉ sử dụng data do BTC cung cấp.

4 Tiền xử lý dữ liệu và augmentation

Preprocessing: Các ảnh đầu vào sẽ được resize thành các kích thước 32x128, 32x256, 64x256 hoặc 224x224 tùy từng mô hình khác nhau. Khi resize các ảnh có thể vẫn giữ đúng tỉ lệ và được padding

về kích thước định trước hoặc sẽ được resize trực tiếp mà không padding. Với một số mô hình thì ảnh sẽ được grayscale về 1 channel duy nhất. Chi tiết tiền xử lý của từng mô hình xem ở phần 7.

Augmentation: Ảnh sẽ được random rotation từ 0-15 độ, ngoài ra mỗi ảnh sẽ được augment thêm với xác suất 0.2. Các phép augment chúng tôi sử dụng là Equalize, Auto Contrast, Sharpness, Color, Brightness, Jpeg Compression, Gaussian Blur và Motion Blur tham khảo từ [1]. Do chất lượng ảnh sinh ra có thể không tốt nên chúng tôi không augment với data synthetic mà chỉ áp dụng với dữ liệu gốc của BTC.

HOG Preprocess: Trong vòng private test, ảnh BTC cung cấp lúc này ảnh không còn được bo góc, sát với chữ nữa nên nhóm cần thực hiện thêm bước text detection. Do không thể huấn luyện thêm model nên chúng tôi sử dụng phương pháp **Historgram of Oriented Gradient (HOG)** thay thế cho phương pháp text detection bằng Deep Learning thông thường. Đầu vào là ảnh với kích thước $H \times W$ được chuẩn hóa để hạn chế ảnh hưởng của hiệu ứng ánh sáng nói chung tới các đường nét của vật thể. Sau đó, gradient của ảnh được tính toán sơ lượt đầu và tạo ra histogram của gradient với 18 orientation bins trên toàn ảnh, chúng tôi tạm gọi là gradient image. Tiếp theo, gradient iamge được chia thành nhiều cells có kích cỡ 8x8 pixels, sau đó tăng cường histogram của từng pixel dựa trên thông tin của các pixel còn lại trong cell đó. Tại bước này chúng tôi đã thu được biểu diễn sơ bộ của các cạnh tại từng vùng ảnh. Để tăng tính ổn định của phương pháp, các cells được nhóm thành từng block (4 cells / block) và thực hiện chuẩn hóa theo từng block một cách độc lập. Sau khi tổng hợp thông tin tại từng vùng ảnh, nhóm đã thu được kết quả là một segmentation mask cùng kích thước đầu vào $H \times W$ mô tả các cạnh và các "key point" của vật thể trong ảnh, cụ thể là vùng text mà nhóm cần phân tách. Dựa vào mask này, chúng tôi thực hiện cắt ảnh gốc với mức threshold phụ thuộc vào **mean gradient** của từng ảnh (adaptive threshold) để giảm thiểu trường hợp cắt nhầm vào vùng text. Chúng tôi đã thử nhiều phương pháp threshold khác nhau, bao gồm adaptive và constant threshold nhưng mean gradient là phương pháp hiệu quả nhất.

5 Mô hình

Các mô hình của chúng tôi phần lớn gồm 1 Vision Encoder và 1 Language Decoder, với kiến trúc được tham khảo từ các paper như **PARSeq** [4], **SRN** [17], **TRBA** [3], **ViTSTR** [2], **CCPD** [6], **ABINet** [9], **SVTR** [7], **Corner Transformer** [16] và **SATRN** [12] với một số điều chỉnh nhất định về loss cũng như kiến trúc. Với một vài model, ảnh sẽ đi qua **Spatial Transformation Network** (STN) [11] để nắn chỉnh ảnh trước khi đi qua Vision Encoder. Chi tiết từng mô hình xem trong phần 7.

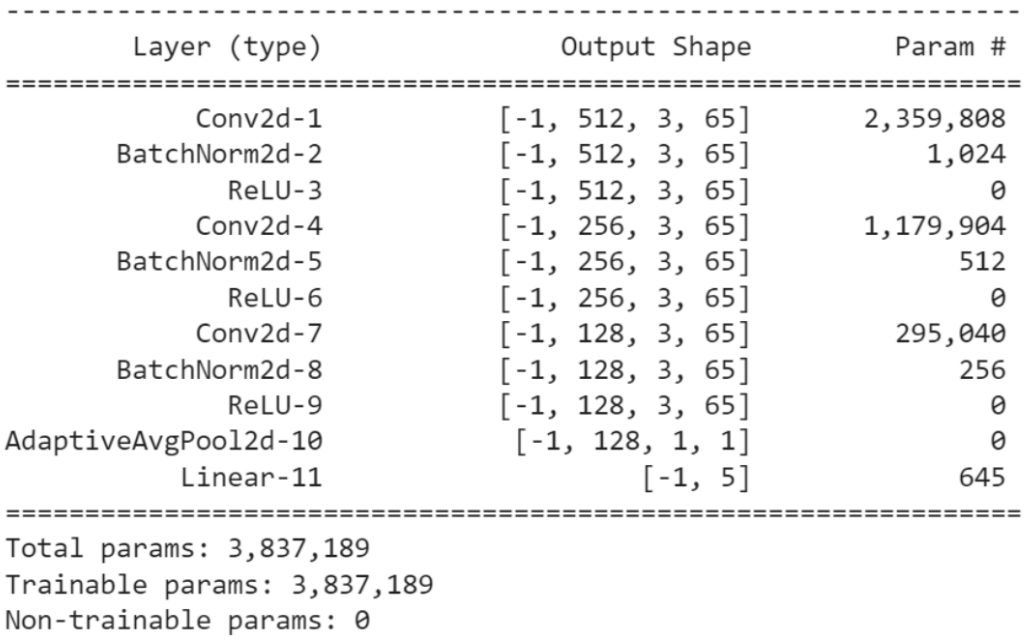
Output của mỗi mô hình sẽ là 1 vector y với kích thước $T \times V$, trong đó T là độ dài tối đa của chuỗi dự đoán (ở đây T là 25), V là kích thước của bảng chữ cái. Chúng tôi sử dụng 2 cách đánh nhãn (labeling) cho từ tiếng Việt. Cách đầu tiên là giữ nguyên các kí tự, bao gồm dấu câu trong nhãn ban đầu của ban tổ chức, tức ta yêu cầu model dự đoán các chữ cái “a”, “ă”, “â”, “B”, “C”, “á”, “Ọ”, “ể”... ; lúc này kích thước của V là 186, tức tổng số ký tự khác nhau xuất hiện trang nhãn do BTC cung cấp. Cách thứ hai là tách ký tự và dấu câu ra, ví dụ chữ “ồ” sẽ được tách thành 3 ký tự `(o, ^, `)`. Kích thước của V sẽ giảm đi đáng kể từ 186 xuống còn 66. Chúng tôi gọi cách đánh nhãn này là **Tone Encoding**.

Bên cạnh một số hàm loss phổ biến như **Cross Entropy** hay **CTC Loss** thì chúng tôi có sử dụng thêm **Focal Loss**[17] và **Label Smoothing** để cải thiện chất lượng mô hình.

Nhận thấy nhiều mô hình gặp khó khăn trong nhận dạng dấu câu, chúng tôi đề xuất sử dụng một module đếm dấu câu là 1 **shallow ConvNet**. Đầu vào của module này là feature sau khi đưa ảnh qua Vision Encoder. Đầu ra của module đếm dấu câu là 5 số thực tương ứng với số dấu sắc, huyền, hỏi, ngã, nặng mà model dự đoán có trong ảnh. Sau đó chúng tôi sử dụng **L2 loss** cho output của module này, gọi là **Count Mark Loss**. Hàm loss cuối cùng là sự kết của Count Mark Loss hợp với hàm loss phân loại ở output Language Decoder (có thể là CTC, Cross Entropy...). Count Mark Loss đóng vai trò hiệu chỉnh giúp model nhận diện dấu câu tốt hơn. Kiến trúc chi tiết module này ở **Hình 3**.

Kiến trúc tổng quan các mô hình chúng tôi sử dụng được thể hiện ở Hình 4

Chúng tôi thực hiện huấn luyện một tập các mô hình với kiến trúc, hàm loss và data huấn luyện khác nhau (sử dụng synthetic hoặc không). Mỗi model sẽ output ra từ được dự đoán cùng với xác suất, độ tự tin của mô hình về từ đó. Tất cả mô hình đều được train sử dụng 1 GPU P100 của kaggle. Do giới hạn về thời gian nên tất cả model đều được train không quá 12 tiếng.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 512, 3, 65]       2,359,808
       BatchNorm2d-2          [-1, 512, 3, 65]           1,024
              ReLU-3          [-1, 512, 3, 65]               0
            Conv2d-4          [-1, 256, 3, 65]       1,179,904
       BatchNorm2d-5          [-1, 256, 3, 65]             512
              ReLU-6          [-1, 256, 3, 65]               0
            Conv2d-7          [-1, 128, 3, 65]         295,040
       BatchNorm2d-8          [-1, 128, 3, 65]             256
              ReLU-9          [-1, 128, 3, 65]               0
AdaptiveAvgPool2d-10           [-1, 128, 1, 1]               0
           Linear-11                  [-1, 5]             645
================================================================
Total params: 3,837,189
Trainable params: 3,837,189
Non-trainable params: 0
```

Figure 3: Kiến trúc module đếm dấu câu

6 Ensemble

6.1 Character-based Ensemble

Sau khi tiến hành huấn luyện tổng cộng 89 model, chúng tôi tiến hành lựa chọn mô hình để tham gia vào 2 quá trình ensemble. Chú ý là danh sách model tham gia quá trình ensemble đầu có thể khác các model tham gia quá trình ensemble thứ 2. Mỗi model sẽ output ra prediction cùng với xác suất của từ đó. Chiến lược ensemble đầu tiên là **Character-based Ensembling**, trong đó chúng tôi thực hiện soft voting, dựa trên từng chữ cái một. Cụ thể thuật toán xem ở Algorithm 1. Để hiểu hơn thuật toán xem ví dụ ở hình 5

Algorithm 1 Character Ensemble Algorithm

Require: Predictions c_i, probability p_i, prediction length l_i $(i = 1, n)$
Ensure: Final prediction p

Initialize $final\_pred = ''$
$l \leftarrow \max l_i$
for $j = 1$ to l **do**
    Initialize $vote\_dict = \{\}$
    Predicted character $char_i \leftarrow c_i[j]$
    for $i = 1$ to n **do**
        if $char_i$ is empty **then**
            continue
        else if $char_i$ not in $vote\_dict$ **then**
            $vote\_dict[char_i] = p_i$
        else
            $vote\_dict[char_i] += p_i$
        end if
    end for
    $final\_pred +=$ argmax $vote\_dict$
end for
return $final\_pred$

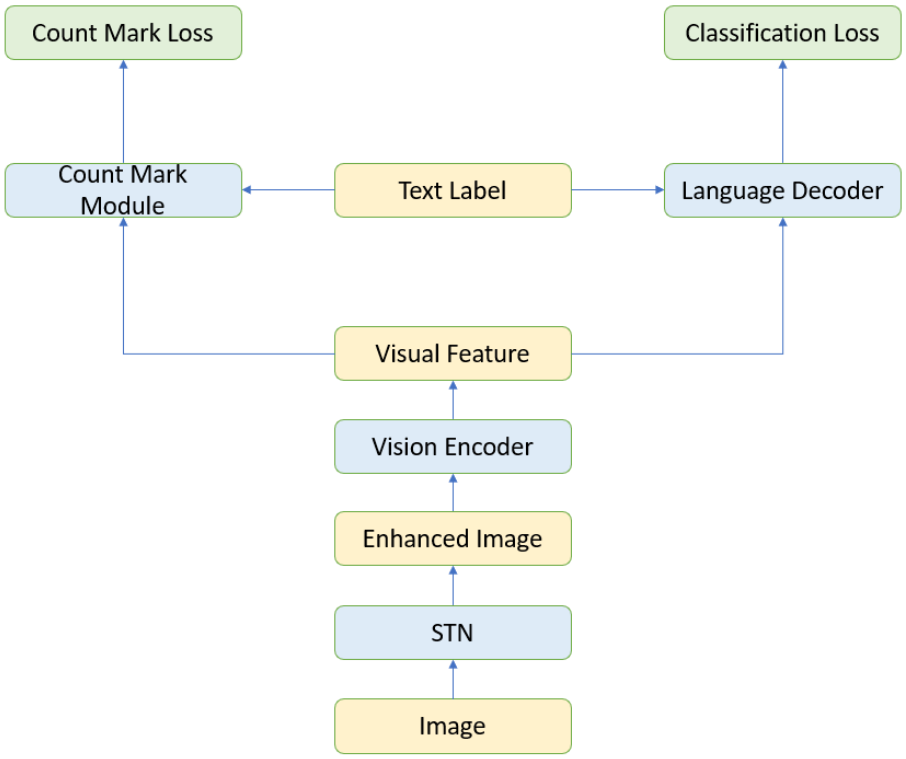


Figure 4: Kiến trúc tổng quan mô hình

Algorithm 2 Word Ensemble Algorithm

procedure COMPUTEPREDICTION($c_i, p_i, char\_pred, dictionary$)
  Initialize word pred probability $pw_i = 0$
  for $i = 1$ **to** n **do**
    $pw_i \mathrel{+}= p_i$
    if c_i **in** $dictionary$ **then**
      $pw_i \mathrel{+}= \alpha$
    end if
    if $c_i == char\_pred$ **then**
      $pw_i \mathrel{+}= \beta$
    end if
  end for
  return c_i **with max** pw_i
end procedure

Việc thực hiện thuật toán trên cho thấy độ hiệu quả khi nó giúp chúng tôi đạt CER trên tập valid là 0.0330 trên tập valid và 0.0333 trên tập public test.

6.2 Dictionary-guided Ensemble

Sau đó chúng tôi nhận thấy nhiều từ prediction trên tập valid khá vô nghĩa và chiếm tỉ trọng lỗi rất cao nên chúng tôi muốn điều chỉnh mô hình sao cho prediction mang tính ngữ nghĩa cao hơn. Do đó chúng tôi đề xuất ra cách ensemble thứ 2 là **Dictionary-Guided Ensembling**. Chúng tôi sẽ thực hiện max voting cho cả cụm từ thay vì từng chữ cái một dựa trên xác suất output của từ đó, từ điển xây dựng từ nhãn của training data và output của character-based ensemble.

Đầu tiên về xây dựng từ điển, nếu sử dụng tất cả nhãn do ban tổ chức cung cấp thì kích thước từ điển sẽ rất lớn và sẽ khó có thể phủ hết được tập các từ có ý nghĩa. Do đó để xây dựng từ điển thì chúng

Output	Probability	Length	1th	2th	3th	4th	5th
Black	0.6	5	B: 0.6	l: 0.6	a: 0.6	c: 0.6	k: 0.6
Pink	0.2	4	P: 0.2	i: 0.2	n: 0.2	k: 0.2	
Black	0.7	5	B: 0.7	l: 0.7	a: 0.7	c: 0.7	k: 0.7
Blacka	0.5	6	B: 0.5	l: 0.5	a: 0.5	c: 0.5	k: 0.5
Blach	0.9	5	B: 0.9	l: 0.9	a: 0.9	c: 0.9	h: 0.9
Final Prediction		5	B: 2.7	l: 2.7	a: 2.7	c: 2.7	k: 1.8

Figure 5: Ví dụ về Character-based Ensembling. Đầu tiên chúng tôi sẽ tìm ra độ dài của từ được dự đoán. Sau đó tại mỗi vị trí, chúng tôi tìm tập ứng cử viên cho chữ cái ở vị trí đó thông qua output của các model. Với mỗi ứng cử viên, chúng tôi tính score cho nó bằng cách cộng tổng xác suất của các model lại. Lặp lại tương tự quá trình trên cho các vị trí khác.

tôi sẽ đưa tất cả các nhãn về lowercase và thực hiện thêm một phép biến đổi nữa. Cụ thể phép biến đổi sẽ xóa đi dấu nặng của tất cả các từ và thực hiện xóa dấu của các chữ có "â" hoặc "ê", tức chữ "ế" -> "ê", "ầ" -> "â". Các phép biến đổi trên được tìm ra sau khi chúng tôi thực hiện error analysis trên tập valid. Tổng quan quá trình xây dựng từ điển xem ở Hình 6

Figure 6: Quá trình xây dựng từ điển

Sử dụng từ điển, thay vì chỉ đơn thuần thực hiện max voting dựa trên xác suất của từ được dự đoán, chúng tôi cộng thêm vào xác suất của từ dự đoán một hằng số α cho những prediction nằm trong từ điển (lưu ý những prediction sẽ được đưa về lowercase và transform như quá trình xây dựng từ điển).Gọi V là tập vocab, từ điển xây dựng từ trên dữ liệu BTC, N là số điểm dữ liệu trong tập valid, K là số model của chúng ta, P là ma trận $N \times K$ với $P_{i,j}$ là dự đoán của model j trên sample thứ i của tập valid. Ta xây dựng mask M là ma trận $N \times K$ trong đó $M_{i,j} = 1$ nếu $P_{i,j}$ thuộc V. Gọi S là ma trận $N \times K$ cho xác suất của mỗi từ dự đoán.

Sau đó chúng tôi thay $S = S + \alpha \dot{M}, \alpha \geq 0$. Tuy nhiên chúng tôi nhận thấy là nếu chỉ sử dụng từ điển được xây dựng từ nhãn của tập huấn luyện do ban tổ chức cung cấp thì khả năng phủ của từ điển vẫn sẽ bị hạn chế mặc dù đã thực hiện các phép đổi để cải thiện. Do đó chúng tôi sử dụng thêm output của character-based ensemble để giải quyết vấn đề trên. Cụ thể gọi $\hat{y}$ là output của character-based ensemble là một vector có kích thước N, nếu $P_{i,j}$ giống $\hat{y}_i$ thì chúng tôi sẽ tiếp tục cộng thêm vào $S_{i,j}$ một hằng số β. Cuối cùng voting cũng được thực hiện trên ma trận S mới này. Prediction $Y_i = P_{i,k}$, với $k = argmax(S_{i,j})$. Ở đây chúng tôi để $\alpha = 1.25$ và $\beta = 0.25$. Có thể xem ví dụ của thuật toán ở hình 7, chi tiết thuật toán xem Algorithm 13

7 Quá trình huấn luyện

Với một số mô hình sử dụng **ViT**, chúng tôi sử pretrained weights trên tập **Imagenet**. Còn không thì tất cả các weights sẽ được khởi tạo ngẫu nhiên từ đầu.

AdamW được sử dụng làm optimizer cho tất cả mô hình. Learning rate scheduler với một số mô hình là **1cycle** [15] hoặc **step decay**.

Với một số mô hình, chúng tôi sử dụng thêm **label smoothing** với tham số $\epsilon = 0.1$ hoặc **focal loss** với tham số $\alpha = 1$ và $\gamma = 0.5$ hoặc $\gamma = 2$.

Model	Output	Probability	Dictionary Score	Character Output	Character Score	Final Score
Model1	An	0.8	1.25	Ăn	0.0	2.05
Model2	Ăn	0.7	1.25	Ăn	0.25	2.20
Model3	Am	0.4	0.0	Ăn	0.0	0.4

Figure 7: Ví dụ về Dictionary-Guided Ensembling. Đầu tiên các từ sẽ được kiểm tra xem là có xuất hiện trong từ điển không và sau đó sẽ check xem có trùng với output của character-based không. Cuối cùng output với score cao nhất sẽ được chọn làm final prediction.

Qua thí nghiệm chúng tôi nhận thấy hệ số nhân vào count mark loss không ảnh hưởng nhiều đến kết quả cuối cùng nên chúng tôi đặt hệ số bằng **1** cho loss này. Loss cuối cùng để huấn luyện mô hình là:

$$L = L_{classify} + L_{countmark} \tag{1}$$

Tất cả thông số huấn luyện của mô hình được tuning sử dụng tập huấn luyện 10,000 ảnh sample ngẫu nhiên, train trong 3 epochs. Do hạn chế về tài nguyên tính toán, chúng tôi sử dụng GPU P100 của Kaggle, số epochs được hiệu chỉnh để có thể huấn luyện trong 12 tiếng, thời gian chạy tối đa cho 1 notebook trên kaggle.

Sau đây là chi tiết tham số huấn luyện của 52 model mà chúng tôi sử dụng để produce kết quả cuối cùng:

model_name	model6_new_full	model17_new_full	model5_synth_tone_full	model1_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Count Mark Loss	No	Yes	Yes	No
Label smoothing loss	No	No	No	No
Use synthetic data	No	No	Yes	No
Epoch	35	30	25	30
Learning rate	1e-4	3e-4	3e-4	5e-4
Weight decay	1e-4	1e-5	1e-5	1e-3
Batch size	128	64	64	64
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	SRN	SRN	TRBA	CTC
Tone Labelling	No	No	Yes	No

model_name	model35_tone_full	model3_new_full	model10_synth_new_full	model1_synth_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	No	No	No	No
Use synthetic data	No	No	Yes	Yes
Epoch	15	29	17	26
Learning rate	2e-4	3e-4	3e-4	1e-3
Weight decay	1e-5	1e-5	1e-5	1e-4
Batch size	32	64	64	64
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	Corner Transformer	TRBA	TRBA	CTC
Tone Labelling	Yes	No	No	No

References

[1] R. Atienza. Data augmentation for scene text recognition, 2021.

[2] R. Atienza. Vision transformer for fast and efficient scene text recognition, 2021.

[3] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis, 2019.

[4] D. Bautista and R. Atienza. Scene text recognition with permuted autoregressive sequence models, 2022.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

model_name	model5_synth_full	model4_synth_new_full	model5_synth_new_full	model15_new_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	Yes	Yes	Yes	No
Use synthetic data	Yes	Yes	Yes	No
Epoch	25	16	16	40
Learning rate	3e-4	3e-4	3e-4	3e-4
Weight decay	1e-5	1e-5	1e-5	1e-5
Batch size	64	64	64	64
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	TRBA	TRBA	TRBA	ParSeq
Tone Labelling	No	No	No	No

model_name	model5_new_full	model10_full	model35_full	model19_synth_new_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	Yes	No	No	No
Use synthetic data	No	No	No	Yes
Epoch	30	31	15	9
Learning rate	3e-4	1e-4	2e-4	5e-4
Weight decay	1e-5	0	1e-5	1e-5
Batch size	64	64	32	64
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	TRBA	TRBA	Corner Transformer	ParSeq
Tone Labelling	No	No	No	No

model_name	model35_synth_full	model2_synth_full	model27_full	model33_tone_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	No	No	No	No
Use synthetic data	Yes	Yes	No	No
Epoch	7	16	65	29
Learning rate	2e-4	1e-4	5e-4	3e-4
Weight decay	1e-5	1e-4	1e-3	1e-5
Batch size	32	64	128	64
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	Corner Transformer	0	abinet	TRBA
Tone Labelling	No	No	No	Yes

model_name	model17_full	model30_synth_full	model18_new_full	model10_synth_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	No	No	No	No
Use synthetic data	Yes	Yes	No	Yes
Epoch	24	50	30	25
Learning rate	3e-4	5e-4	3e-4	3e-4
Weight decay	1e-5	1e-3	1e-4	1e-5
Batch size	64	128	128	64
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	SRN	abinet	0	TRBA
Tone Labelling	No	No	No	No

model_name	model19_new_full	model30_tone_full	model15_full	model13_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	Yes	No	No	No
Use synthetic data	No	No	Yes	Yes
Epoch	17	80	40	40
Learning rate	5e-4	5e-4	7e-4	7e-4
Weight decay	1e-5	1e-3	1e-5	1e-5
Batch size	64	128	128	128
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	ParSeq	abinet	ParSeq	ParSeq
Tone Labelling	No	Yes	No	No

model_name	model2_new_full	model34_tone_full	model15_synth_new_full	model34_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	No	No	No	No
Use synthetic data	No	No	Yes	No
Epoch	20	20	25	21
Learning rate	1e-4	2e-4	3e-4	2e-4
Weight decay	1e-4	1e-5	1e-5	1e-5
Batch size	64	32	64	32
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	0	satrn	ParSeq	satrn
Tone Labelling	No	Yes	No	No

model_name	model31_full	model7_full	model32_full	model19_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	Yes	Yes	Yes	No
Use synthetic data	No	No	No	Yes
Epoch	65	30	29	13
Learning rate	3e-4	1e-4	2e-4	5e-4
Weight decay	5e-4	0	1e-5	1e-5
Batch size	128	64	64	64
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	cppd	TRBA	cppd	ParSeq
Tone Labelling	No	No	No	No

model_name	model14_full	model3_synth_full	model26_synth_full	model5_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	No	No	No	Yes
Use synthetic data	Yes	Yes	Yes	No
Epoch	55	26	42	30
Learning rate	5e-4	3e-4	3e-4	1e-4
Weight decay	1e-3	0	5e-4	0
Batch size	128	64	128	64
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	ParSeq	TRBA	CTC	TRBA
Tone Labelling	No	No	No	No

model_name	model5_tone_full	model3_tone_full	model8_synth_full	model19_synth_new_tone_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	Yes	No	No	No
Use synthetic data	No	No	Yes	Yes
Epoch	33	29	25	9
Learning rate	3e-4	3e-4	3e-4	5e-4
Weight decay	1e-5	1e-5	1e-5	1e-5
Batch size	64	64	64	64
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	TRBA	TRBA	TRBA	ParSeq
Tone Labelling	Yes	Yes	No	Yes

model_name	model9_full	model30_full	model2_full	model29_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	Yes	No	No	No
Use synthetic data	No	No	No	No
Epoch	35	70	15	60
Learning rate	1e-4	5e-4	1e-4	5e-4
Weight decay	1e-4	1e-3	1e-3	1e-3
Batch size	128	128	64	128
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	SRN	abinet	0	abinet
Tone Labelling	No	No	No	No

model_name	model20_full	model5_synth_tone_full	model20_tone_full	model4_synth_full
Tiền xử lí	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,	Resize to 64x256,
Focal Loss	No	No	No	No
Label smoothing loss	No	No	No	No
Count Mark Loss	Yes	No	Yes	Yes
Use synthetic data	Yes	Yes	No	Yes
Epoch	45	38	60	25
Learning rate	5e-4	3e-4	5e-4	3e-4
Weight decay	1e-4	1e-5	1e-4	1e-5
Batch size	128	64	128	64
Scheduler	Step decay	Step decay	Step decay	Step decay
Kiến trúc gốc	TRBA	ParSeq	TRBA	TRBA
Tone Labelling	No	Yes	Yes	No

[6] Y. Du, Z. Chen, C. Jia, X. Yin, C. Li, Y. Du, and Y.-G. Jiang. Context perception parallel decoder for scene text recognition, 2023.

[7] Y. Du, Z. Chen, C. Jia, X. Yin, T. Zheng, C. Li, Y. Du, and Y.-G. Jiang. Svtr: Scene text recognition with a single visual model, 2022.

[8] H. N. Duc. Vietnamese word list: Ho ngoc duc's word list. `http://www.informatik.unileipzig.de/~duc/software/misc/wordlist.html`, 2004.

[9] S. Fang, H. Xie, Y. Wang, Z. Mao, and Y. Zhang. Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition, 2021.

[10] S. Fogel, H. Averbuch-Elor, S. Cohen, S. Mazor, and R. Litman. Scrabblegan: Semi-supervised varying length handwritten text generation, 2020.

[11] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks, 2016.

[12] J. Lee, S. Park, J. Baek, S. J. Oh, S. Kim, and H. Lee. On recognizing texts of arbitrary shapes with 2d self-attention, 2019.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[14] D. Sang and N. Thuan. An efficientnet-like feature extractor and focal ctc loss for image-base sequence recognition. pages 326–331, 11 2020.

[15] L. N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, 2018.

[16] X. Xie, L. Fu, Z. Zhang, Z. Wang, and X. Bai. Toward understanding wordart: Corner-guided transformer for scene text recognition, 2022.

[17] D. Yu, X. Li, C. Zhang, J. Han, J. Liu, and E. Ding. Towards accurate scene text recognition with semantic reasoning networks, 2020.