TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
ĐẠI HỌC BÁCH KHOA HÀ NỘI
----------*----------

BÁO CÁO
Nhận dạng chữ viết tay bằng CRNN

GVHD: **Thầy Trần Thế Hùng**
Môn học: **Nhập môn trí tuệ nhân tạo**
Nhóm sinh viên: **Nguyễn Bình An**
Trần Thị Phương
Vũ Đình Dương
Phạm Trung Hiếu
Hồ Văn Đức

Hà Nội 06-2024

Nội dung

1. Giới thiệu

1.1. Phát biểu bài toán

Bài toán nhận diện chữ viết tay là một thách thức hấp dẫn trong học máy, với mục tiêu huấn luyện máy tính phân loại chính xác các ký tự trong hình ảnh chữ viết tay. Thay vì dựa vào quy tắc cứng nhắc như các phương pháp truyền thống, học máy cho phép máy tính tự động học các đặc trưng từ dữ liệu, giúp giải quyết vấn đề biến đổi, độ phức tạp, và biến dạng trong chữ viết tay của con người. Các mô hình học máy, đặc biệt là mạng nơron sâu (DNN), đã đạt được hiệu quả vượt trội trong việc xử lý hình ảnh chữ viết tay, dựa trên việc trích xuất đặc trưng từ các lớp Convolution và Pooling, kết hợp với khả năng xử lý chuỗi thời gian của mạng nơron tuần hoàn (RNN) và bộ nhớ dài hạn của mạng nơron tái diễn (LSTM). Các kỹ thuật học máy cho phép xây dựng các hệ thống tự động hóa nhận diện chữ viết tay, ứng dụng trong nhiều lĩnh vực như quét tài liệu, dịch thuật, phân tích văn bản, tìm kiếm thông tin, và giao tiếp người-máy.

Đặc điểm và thách thức:

Nhận dạng chữ viết tay là một bài toán phức tạp do tính đa dạng và phức tạp của chữ viết tay. Mỗi cá nhân có phong cách viết tay riêng biệt, và ngay cả chữ viết của cùng một người cũng có thể thay đổi theo thời gian hoặc trạng thái tâm lý. Bên cạnh đó, các yếu tố như nhiễu, chất lượng ảnh kém, độ sáng không đồng đều, và các biến dạng khác cũng làm tăng thêm độ khó của bài toán.

Chữ viết tay có tính không đồng đều cao, với sự biến thiên lớn về hình dạng và kích thước của các ký tự. Điều này đòi hỏi các hệ thống nhận dạng phải có khả năng học và phân biệt các đặc trưng tinh vi từ dữ liệu đầu vào. Hơn nữa, sự hiện diện của các yếu tố gây nhiễu như các nét bút không liên quan hoặc dấu mực cũng làm cho việc nhận dạng chính xác trở nên khó khăn hơn.

Phân chia công việc:

Trần Thị Phương: Tiền xử lý dữ liệu

Vũ Đình Dương: nghiên cứu và áp dụng CNN

Phạm Trung Hiếu: nghiên cứu và áp dụng RNN(BiLSTM)

Hồ Văn Đức: nghiên cứu và xử lý CTC

Nguyễn Bình An: Trainning, đánh giá và lựa chọn tham số

1.2. Giải pháp sử dụng: CRNN

CNN (Convolutional Neural Network):

- Vai trò: Trích xuất đặc trưng từ hình ảnh chữ viết tay
- Cách hoạt động:
  - Áp dụng các bộ lọc (kernel) lên hình ảnh để trích xuất các đặc trưng cục bộ như cạnh, đường cong, góc,...
  - Sử dụng các lớp MaxPooling để giảm kích thước bản đồ đặc trưng, loại bỏ thông tin không cần thiết, giảm độ phức tạp tính toán.
- Kết quả: Tạo ra một bản đồ đặc trưng (feature map) chứa các thông tin về các đặc trưng cục bộ đã được trích xuất.

RNN (Recurrent Neural Network) - BiLSTM (Bidirectional Long Short-Term Memory):

- Vai trò: Tạo ra biểu diễn chuỗi (sequence representation) từ đặc trưng hình ảnh, nắm bắt thông tin ngữ cảnh từ các đặc trưng cục bộ.
- Cách hoạt động:
  - BiLSTM sử dụng hai mạng LSTM song song, chạy theo hai hướng (tiến và lùi) để nắm bắt thông tin ngữ cảnh từ cả hai phía của chuỗi.
  - BiLSTM có khả năng ghi nhớ thông tin lâu hơn, giúp xử lý các chuỗi ký tự dài và phức tạp.
- Kết quả: Tạo ra một biểu diễn chuỗi, chứa thông tin về ngữ cảnh của các ký tự trong văn bản.

Kết hợp CNN và BiLSTM trong CRNN:

- Kết quả của CNN (bản đồ đặc trưng) được đưa vào BiLSTM.
- BiLSTM tạo ra biểu diễn chuỗi dựa trên bản đồ đặc trưng.
- Kết quả của BiLSTM được đưa vào lớp Fully Connected để dự đoán nhãn cho từng ký tự trong chuỗi.

2. Quy trình thực hiện

2.1. Khai phá dữ liệu

Thu thập dữ liệu: Dữ liệu được thu thập bao gồm các hình ảnh chứa văn bản viết tay, các nguồn thu thập dữ liệu bao gồm:

- Tập dữ liệu công khai: Bộ orc_dataset từ nguồn https://pbcquoc.github.io/train-crnn/, bộ dữ liệu address từ nguồn https://github.com/TomHuynhSG/Vietnamese-Handwriting-Recognition-OCR

- Tập dữ liệu tự viết tay của các thành viên

Tổng số lượng dữ liệu khoảng 2400 ảnh chữ viết tay.

Tiền xử lý dữ liệu:

- Xử lý nhiễu: Loại bỏ nhiễu trong hình ảnh bằng các kỹ thuật xử lý ảnh như lọc trung bình, lọc Gaussian, lọc median.
- Xử lý các lỗi: Sửa chữa các lỗi trong dữ liệu như lỗi nhãn, lỗi pixel, lỗi định dạng.
- Chuẩn hóa dữ liệu: Hình ảnh sẽ được chuyển đổi về định dạng Grayscale với 1 kênh màu duy nhất. Sau đó, kích thước hình ảnh được điều chỉnh về 32 x 768 pixel.
- Tăng tập dữ liệu huấn luyện: Tiến hành căn chỉnh vị trí câu trong hình ảnh thu được ở bước trên (căn trái, phải, giữa). Kỹ thuật này giúp tăng cường dữ liệu nhằm cải thiện hiệu suất của mô hình khi có ít data.

Chuẩn bị dữ liệu:

Chia tập dữ liệu thành tập huấn luyện (72%), tập kiểm tra(10%) và tập đánh giá (18%).

2.2. Kiến trúc mô hình

2.2.1. Giới thiệu về CNN

Định nghĩa về CNN

CNN là một loại mạng nơron sâu chuyên biệt cho việc xử lý dữ liệu có cấu trúc hình ảnh. Nó được thiết kế để tận dụng cấu trúc không gian của dữ liệu hình ảnh, trích xuất các đặc trưng cục bộ và tổng quát hóa hiệu quả. CNN thường bao gồm các lớp sau:

- Lớp Convolution: Áp dụng các bộ lọc (kernel) lên hình ảnh để trích xuất các đặc trưng cục bộ, ví dụ như cạnh, đường cong, góc,...
- Lớp Pooling: Giảm kích thước của bản đồ đặc trưng, loại bỏ thông tin không cần thiết, giảm độ phức tạp tính toán.
- Lớp Fully Connected: Kết nối tất cả các nút của lớp trước đó với các nút của lớp tiếp theo, thực hiện phân loại dựa trên các đặc trưng đã trích xuất.

Mô hình CNN được Áp dụng

- Kiến trúc: Mô hình CNN được sử dụng trong báo cáo này được xây dựng dựa trên kiến trúc VGG-16, một kiến trúc mạng nơron tích chập phổ biến và hiệu quả. Kiến trúc này bao gồm 13 lớp Convolution và 5 lớp MaxPooling, được thiết kế để trích xuất các đặc trưng đa dạng từ hình ảnh.
- Các lớp:
  - Lớp Convolution: Mỗi lớp Convolution sử dụng các bộ lọc 3x3 để trích xuất các đặc trưng cục bộ, số lượng kênh đầu ra của mỗi lớp Convolution được tăng dần từ 64 lên 512.

- Lớp MaxPooling: Lớp MaxPooling với kích thước cửa sổ 2x2 được sử dụng để giảm kích thước của bản đồ đặc trưng và loại bỏ thông tin không cần thiết, giúp giảm độ phức tạp tính toán.
- Lớp Batch Normalization: Lớp này được thêm vào sau mỗi lớp Convolution để chuẩn hóa giá trị đầu vào của mỗi lớp, giúp mạng học nhanh hơn và ổn định hơn.
- Lớp ReLU: Hàm kích hoạt ReLU được sử dụng sau mỗi lớp Convolution để thêm phi tuyến tính vào mạng, giúp mạng học được các đặc trưng phức tạp hơn.

- Tham số:
  - Kích thước bộ lọc: 3x3.
  - Bước di chuyển (stride): 1.
  - Padding: 1.
  - Số lượng kênh vào: 1 (cho lớp Convolution đầu tiên) hoặc số lượng kênh đầu ra của lớp Convolution trước đó (cho các lớp Convolution tiếp theo).
  - Số lượng kênh ra: Tăng dần từ 64 lên 512 cho các lớp Convolution.
  - Hàm kích hoạt: ReLU.

2.2.2. Giới thiệu về BiLSTM

BiLSTM (Bidirectional Long Short-Term Memory) là một biến thể của LSTM (Long Short-Term Memory), một loại mạng nơron hồi quy (RNN) cải tiến. BiLSTM được thiết kế để xử lý dữ liệu tuần tự, như văn bản hoặc chuỗi thời gian, theo cả hai hướng (tiến và lùi), giúp nắm bắt thông tin ngữ cảnh tốt hơn so với các RNN thông thường.

Vai trò của BiLSTM trong CRNN:

BiLSTM đóng vai trò quan trọng trong CRNN (Convolutional Recurrent Neural Network) bằng cách tạo ra biểu diễn chuỗi từ các đặc trưng hình ảnh đã được trích xuất bởi các lớp CNN (Convolutional Neural Network). Điều này giúp mô hình có khả năng nắm bắt thông tin ngữ cảnh từ cả phía trước và phía sau của chuỗi dữ liệu, cải thiện độ chính xác trong việc nhận dạng chữ viết tay.

Cách hoạt động của BiLSTM:

BiLSTM sử dụng hai mạng LSTM song song chạy theo hai hướng: một mạng xử lý chuỗi từ trái sang phải (tiến) và một mạng xử lý chuỗi từ phải sang trái (lùi). Kết quả từ hai mạng này được kết hợp lại để tạo ra một biểu diễn chuỗi duy nhất, giúp mô hình nắm bắt thông tin từ cả hai phía của chuỗi ký tự.

- BiLSTM tiến: Xử lý chuỗi theo hướng từ trái sang phải, giúp nắm bắt ngữ cảnh từ phía trước.
- BiLSTM lùi: Xử lý chuỗi theo hướng từ phải sang trái, giúp nắm bắt ngữ cảnh từ phía sau.
- Kết hợp: Kết quả của hai mạng BiLSTM được kết hợp để tạo ra một biểu diễn chuỗi chứa thông tin ngữ cảnh đầy đủ từ cả hai phía.

Ưu điểm của BiLSTM:

- Nắm bắt ngữ cảnh toàn diện: BiLSTM có khả năng nắm bắt thông tin ngữ cảnh từ cả phía trước và phía sau của chuỗi, giúp cải thiện độ chính xác trong các tác vụ nhận dạng.
- Khả năng ghi nhớ dài hạn: Nhờ vào cấu trúc LSTM, BiLSTM có thể ghi nhớ thông tin lâu hơn, xử lý tốt các chuỗi ký tự dài và phức tạp.
- Giảm hiện tượng trôi gradient: BiLSTM giúp giảm hiện tượng trôi gradient, một vấn đề thường gặp trong các RNN thông thường, giúp mô hình học hiệu quả hơn.

Nhược điểm của BiLSTM:

- Tính toán phức tạp: BiLSTM yêu cầu nhiều tài nguyên tính toán hơn so với các RNN thông thường do phải xử lý chuỗi theo hai hướng.
- Thời gian huấn luyện dài: Do tính phức tạp của mô hình, BiLSTM có thể mất nhiều thời gian hơn để huấn luyện so với các mô hình khác.
- Khó khăn trong việc triển khai: Việc triển khai và tối ưu hóa BiLSTM có thể phức tạp hơn, đặc biệt khi áp dụng vào các bài toán lớn và đa dạng.

Kết hợp CNN và BiLSTM trong CRNN:

Trong CRNN, các đặc trưng cục bộ được trích xuất bởi các lớp CNN được đưa vào BiLSTM để tạo ra biểu diễn chuỗi. Kết quả của BiLSTM sau đó được đưa vào lớp Fully Connected để dự đoán nhãn cho từng ký tự trong chuỗi. Sự kết hợp này giúp mô hình tận dụng được cả khả năng trích xuất đặc trưng của CNN và khả năng nắm bắt ngữ cảnh của BiLSTM, tạo ra một hệ thống nhận dạng chữ viết tay hiệu quả và chính xác.

2.2.3. Kiến trúc sử dụng

```
class CRNN(nn.Module):
    def __init__(self, nclass, num_hidden, dropout = 0.1):
        print('>>>> use Crnn-------------\n')
        super(CRNN, self).__init__()

        ks = [ 3,  3,   3,   3,   3,   3,  2]
        ss = [ 1,  1,   1,   1,   1,   1,  1]
        ps = [ 1,  1,   1,   1,   1,   1,  0]
        nm = [32, 64, 128, 128, 256, 256, 256]

        cnn = nn.Sequential()
        def convRelu(i):
            nIn = 1 if i == 0 else nm[i - 1]
            nOut = nm[i]
            cnn.add_module('conv{0}'.format(i),
                           nn.Conv2d(nIn, nOut, ks[i], ss[i], ps[i]))
            cnn.add_module('batchnorm{0}'.format(i), nn.BatchNorm2d(nOut))
            cnn.add_module('relu{0}'.format(i), nn.ReLU(True))

        # input : (C, H, W) - (1, 32, 768)
        convRelu(0)
        cnn.add_module('pooling{0}'.format(0), nn.MaxPool2d((2, 2)))  # 32, 16, 384
        convRelu(1)
        cnn.add_module('pooling{0}'.format(1), nn.MaxPool2d((2, 2)))  # 64, 8, 192
        convRelu(2)
        convRelu(3)
        cnn.add_module('pooling{0}'.format(2), nn.MaxPool2d((2, 1)))  # 128, 4, 192
        convRelu(4)
        convRelu(5)
        cnn.add_module('pooling{0}'.format(3), nn.MaxPool2d((2, 1)))  # 256, 2, 192
        convRelu(6) # 256, 1, 191
        self.cnn = cnn
        self.dropout_cnn = nn.Dropout(dropout)
```

```
        # BiLSTM
        self.biLSTM1 = nn.LSTM(nm[-1], num_hidden, bidirectional=True, batch_first = True)
        self.dropout1 = nn.Dropout(dropout)

        self.biLSTM2 = nn.LSTM(num_hidden*2, num_hidden, bidirectional=True, batch_first = True)
        self.dropout2 = nn.Dropout(dropout)

        # Linear
        self.linear = nn.Linear(num_hidden * 2, nclass)
```

- Đầu tiên, ta cần đưa input (kích thước 32 x 768) qua một lớp mạng CNN. Mục đích của phần này là trích chọn các đặc trưng của ảnh và thu được output cuối cùng là chuỗi các feature vector, là một tensor 3 chiều có kích thước (batch_size x 1 x f). Điều quan trọng ở đây là thu được một chiều là 1, vì vậy kiến trúc bên trong lớp CNN này được điều chỉnh để nhằm xuất hiện điều đó.
- Khi đã thu được Output từ CNN , ta tiếp tục cho qua 2 lớp BiLSTM.
- Cuối cùng, đầu ra của từng frame của BiLSTM ta cần một lớp Linear để thu được phân phối theo đúng số lượng nhãn (số lượng chữ cái- nClass) mà ta cần.

2.3. CTC

CTC Loss, hay Connectionist Temporal Classification Loss, là một hàm mất mát (loss function) được sử dụng trong các mô hình học sâu để nhận dạng chuỗi có độ dài thay đổi và không được sắp xếp một cách rõ ràng. Hàm mất mát này được thiết kế đặc biệt cho các bài toán như nhận dạng giọng nói tự động (ASR) và nhận dạng chữ viết tay, nơi mà độ dài của đầu ra (chuỗi ký tự hoặc từ) có thể khác với độ dài của đầu vào (chuỗi tín hiệu âm thanh hoặc hình ảnh).

Đặc điểm chính của CTC Loss:

- Cho phép sự sắp xếp mềm dẻo:
  CTC cho phép mô hình học cách ánh xạ một chuỗi đầu vào dài và không có cấu trúc rõ ràng (ví dụ: chuỗi âm thanh) thành một chuỗi đầu ra ngắn hơn và có cấu trúc (ví dụ: chuỗi từ hoặc ký tự).
- Biểu tượng trống (Blank Symbol):
  CTC sử dụng một biểu tượng đặc biệt gọi là "blank" để xử lý các phần của đầu vào không ánh xạ trực tiếp đến bất kỳ ký tự đầu ra nào. Điều này giúp xử lý khoảng trắng và các phần dư thừa trong chuỗi đầu vào.
- Tính toán xác suất tổng hợp:
  CTC tính toán xác suất tổng hợp của tất cả các ánh xạ hợp lệ từ đầu vào đến đầu ra, thay vì yêu cầu ánh xạ một-một. Điều này được thực hiện thông qua thuật toán forward backward để tổng hợp các xác suất.

Trong bối cảnh bài toán nhận dạng chữ viết tay, hàm CTC Loss tham gia vào 2 quá trình chính:

2.3.1. Tính Loss

Công thức CTC Loss

CTC Loss được tính toán dựa trên xác suất của chuỗi đầu ra đúng y cho trước chuỗi đầu vào x. Cụ thể, nó tính tổng xác suất của tất cả các sắp xếp có thể có của y trong x.

Giả sử chúng ta có chuỗi đầu vào x = (x1, x2, …, xT) và chuỗi đầu ra y= (y1, y2, …, yU), với T là độ dài của chuỗi đầu vào và U là độ dài của chuỗi đầu ra. CTC Loss tìm cách cực đại hóa xác suất P(y|x), được tính như sau:

$$P(y|x) = \sum_{\pi \epsilon B^{-1}(y)} P(\pi|x)$$

Trong đó:

- π là một sắp xếp hợp lệ của y với các biểu tượng trống xen kẽ.
- B là một hàm ánh xạ loại bỏ các biểu tượng trống và các ký tự lặp lại liên tiếp.
- P(π|x) là xác suất của một sắp xếp cụ thể π, được tính dựa trên đầu ra của mô hình (ví dụ: mạng nơ-ron hồi quy RNN hoặc mạng nơ-ron tích chập CNN).

Hàm ánh xạ B là một hàm ánh xạ loại bọ các biểu tượng trống và các ký tự lặp lại liên tiếp, với chuỗi đầu vào được thêm các ký tự trống thỏa mãn điều kiện không có 3 ký tự (khác ký tự trống) đứng liền nhau. Ví dụ, nếu:

- Π = "__C_AA__TT_" thì B(π)="CAT"

Mỗi chuỗi y có độ dài n sẽ có rất nhiều chuỗi $\pi \epsilon B^{-1}(y)$, vì vậy để tìm được xác suất cần tính mà không cần liệt kê toàn bộ chuỗi π, ta sử dụng thuật toán forward-backward với các bước như sau:

1. **Chuẩn bị các biến trạng thái**:
   - Giả sử chúng ta có chuỗi đầu vào x=(x1, x2,…,xT) với độ dài T và chuỗi đầu ra y=(y1,y2,…,yU) với độ dài U.
   - Tạo ra chuỗi mục tiêu mở rộng y′ bằng cách xen kẽ các biểu tượng trống. Ví dụ, nếu y="CAT", thì y′ = (_, C, _, A,_,T,_)
2. **Tính toán forward probabilities (α):**
   - αt(s) là xác suất của việc đạt đến trạng thái s của y′ tại thời điểm t của x.
   - Khởi tạo: α1(1) = p (x1=_), α1(2)=p(x1=y1) (các vị trí còn lại là 0).
   - Cập nhật: Sử dụng công thức để cập nhật các trạng thái tiếp theo, dựa trên các trạng thái trước đó và xác suất của các ký tự tại thời điểm hiện tại.
3. **Tính toán backward probabilities (β):**
   - βt(s) là xác suất của việc đạt đến trạng thái cuối cùng từ trạng thái s tại thời điểm t.
   - Khởi tạo: βT(S)=1 βT(S−1) =1 (các vị trí còn lại là 0).
   - Cập nhật: Sử dụng công thức để cập nhật các trạng thái trước đó, dựa trên các trạng thái sau và xác suất của các ký tự tại thời điểm hiện tại
4. **Kết hợp forward và backward probabilities:**
   - Tính xác suất của toàn bộ chuỗi y với chuỗi x cho trước:

$$P(y|x) = \sum_t \alpha_t(s).\beta_t(s)$$

5. **Tính CTC Loss**
   - CTC LOSS là log-likelihood âm của xác suất này:

$$CTC\ LOSS = -\log\left(P(y|x)\right)$$

2.3.2. Suy diễn (phán đoán)

Trong bối cảnh bải toán nhận dạng chữ viết tay, CTC còn tham gia vào quá trình dự đoán bằng cách sử dụng hàm ánh xạ B.

Với đầu vào là ma trận xác suất của các ký tự có thể xuất hiện tại time step t. Mô hình sẽ tìm được 1 chuỗi đầu ra y* là các ký tự có xác suất lớn nhất tại mỗi time step, thỏa mãn điều kiện không có 3 ký tự (khác ký tự trống) nào đứng liên tiếp. Đưa chuỗi y* vào hàm ánh xạ B, ta sẽ có được đầu ra y cần tìm.

2.4. Chỉ số đánh giá

Để đánh giá hệ thống nhận dạng chữ viết tay qua hình ảnh, các chỉ số được sử dụng để đánh giá bao gồm Tỷ lệ lỗi kí tự (CER – Character Error Rate) và Tỷ lệ lỗi từ (WER – Word Error Rate). Đây là những chỉ số quan trọng giúp đánh giá được độ chính xác của mô hình chuyển đổi ảnh chữ viết tay thành văn bản số.

Tỷ Lệ Lỗi Ký Tự (CER):

CER là một chỉ số đo lường số lượng lỗi ký tự trong chuỗi ký tự được dự đoán so với chuỗi ký tự tham chiếu. Nó được tính toán dựa trên số lượng thao tác chỉnh sửa cần thiết (chèn, xóa, thay thế) để chuyển đổi chuỗi ký tự dự đoán thành chuỗi ký tự tham chiếu.

Công thức tính CER:

$$\mathrm{CER} = \frac{S+D+I}{N}$$

Trong đó:

- S là số ký tự thay thế (substitutions)
- D là số ký tự bị xóa (deletions)
- I là số ký tự được chèn (insertions)
- N là tổng số ký tự trong chuỗi tham chiếu

CER thường được sử dụng để đánh giá mức độ chi tiết, đặc biệt khi cần nhận dạng chính xác từng ký tự, như trong các ứng dụng lưu trữ tài liệu quan trọng hoặc mã hóa. CER càng thấp thì mô hình càng chính xác.

Tỷ Lệ Lỗi Từ (WER):

WER là một chỉ số đo lường số lượng lỗi từ trong chuỗi từ được dự đoán so với chuỗi từ tham chiếu. Tương tự như CER, WER cũng được tính dựa trên số lượng thao tác chỉnh sửa cần thiết để chuyển đổi chuỗi từ dự đoán thành chuỗi từ tham chiếu.

Công thức tính WER:

$$\text{WER} = \frac{S+D+I}{N}$$

Trong đó:

- S là số từ thay thế (substitutions)
- D là số từ bị xóa (deletions)
- I là số từ được chèn (insertions)
- N là tổng số từ trong chuỗi tham chiếu

WER hữu ích trong việc đánh giá độ chính xác tổng thể của văn bản. WER càng thấp thì mô hình càng chính xác.

2.5. Lựa chọn tham số và đánh giá model

Trong quá trình xây dựng mô hình nhận dạng chữ viết tay bằng CRNN, việc lựa chọn tham số là một bước quan trọng để tối ưu hóa hiệu suất của mô hình. Các tham số cần được chọn một cách cẩn thận và có cơ sở khoa học để đảm bảo rằng mô hình có thể học được các đặc trưng quan trọng từ dữ liệu và tổng quát hóa tốt trên các tập dữ liệu mới. Các tham số được lựa chọn:

- **Số Lượng Đơn Vị Ẩn trong BiLSTM (numhidden):** là một biến thể của LSTM có khả năng học các phụ thuộc trong cả hai hướng thời gian. Số lượng đơn vị ẩn trong lớp BiLSTM quyết định khả năng học các đặc trưng từ dữ liệu.

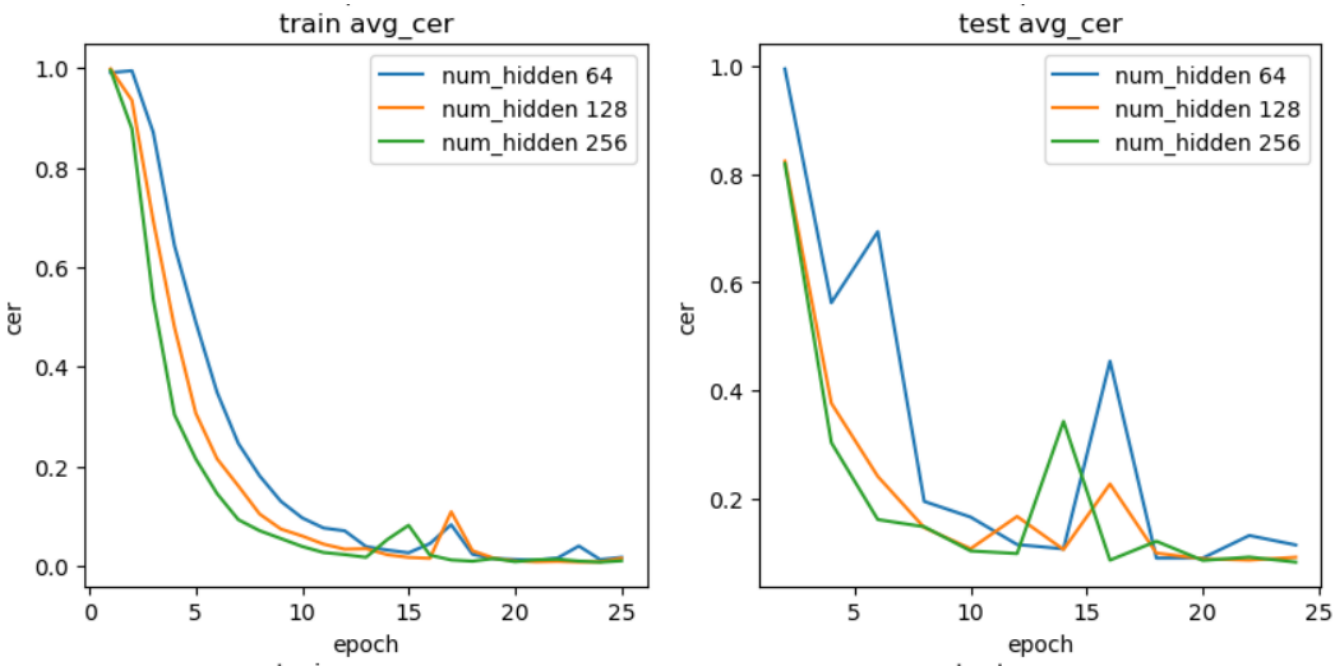

Figure 1: Tỷ lệ lỗi kí tự (CER) của tham số num_hidden

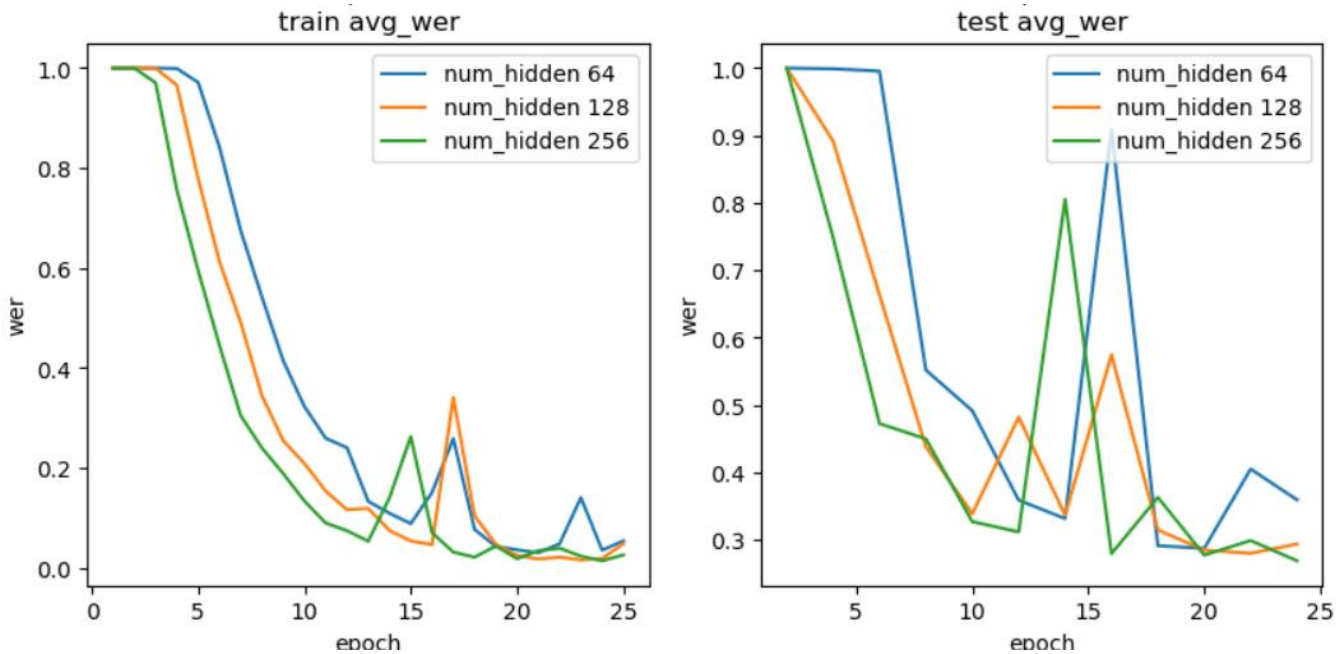

Figure 2: Tỷ lệ lỗi từ (WER) của tham số num_hidden

- **Tỷ lệ Dropout:** Dropout là một kỹ thuật regularization được sử dụng để ngăn chặn mô hình quá khớp (overfitting). Tỷ lệ dropout quyết định tỷ lệ các đơn vị trong mạng sẽ bị bỏ qua ngẫu nhiên trong quá trình huấn luyện.

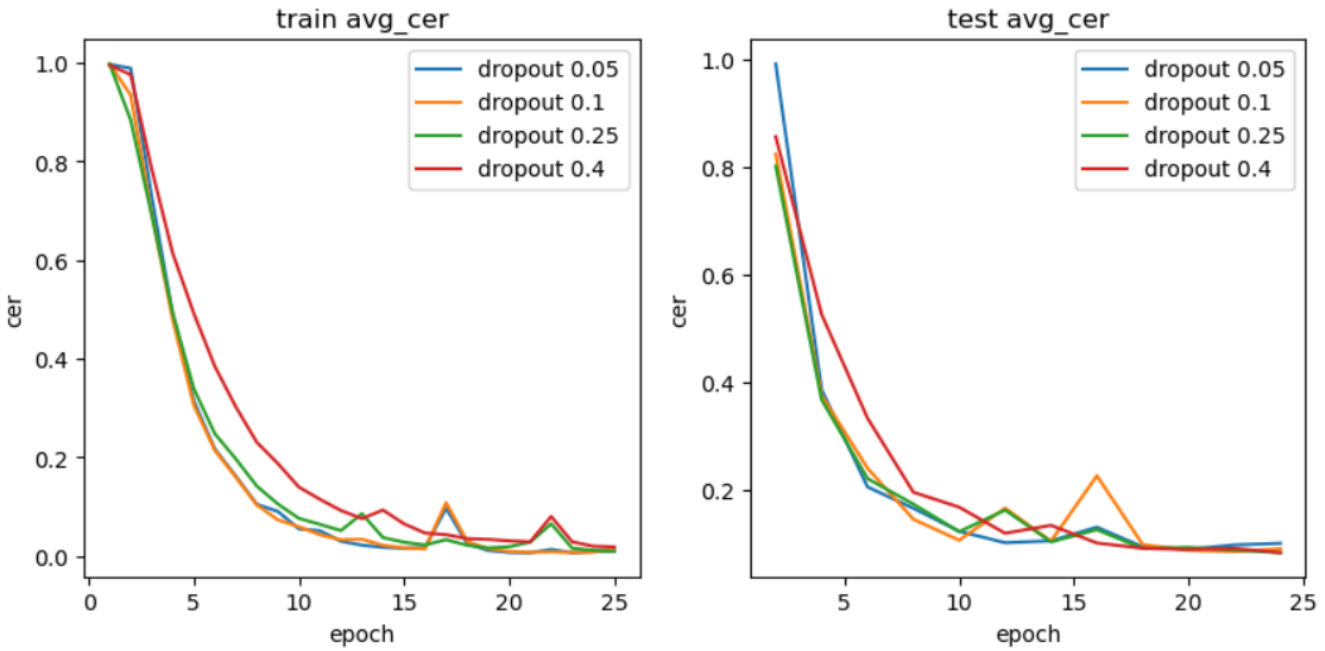

Figure 3: Tỷ lệ lỗi kí tự (CER) của các tham số dropout

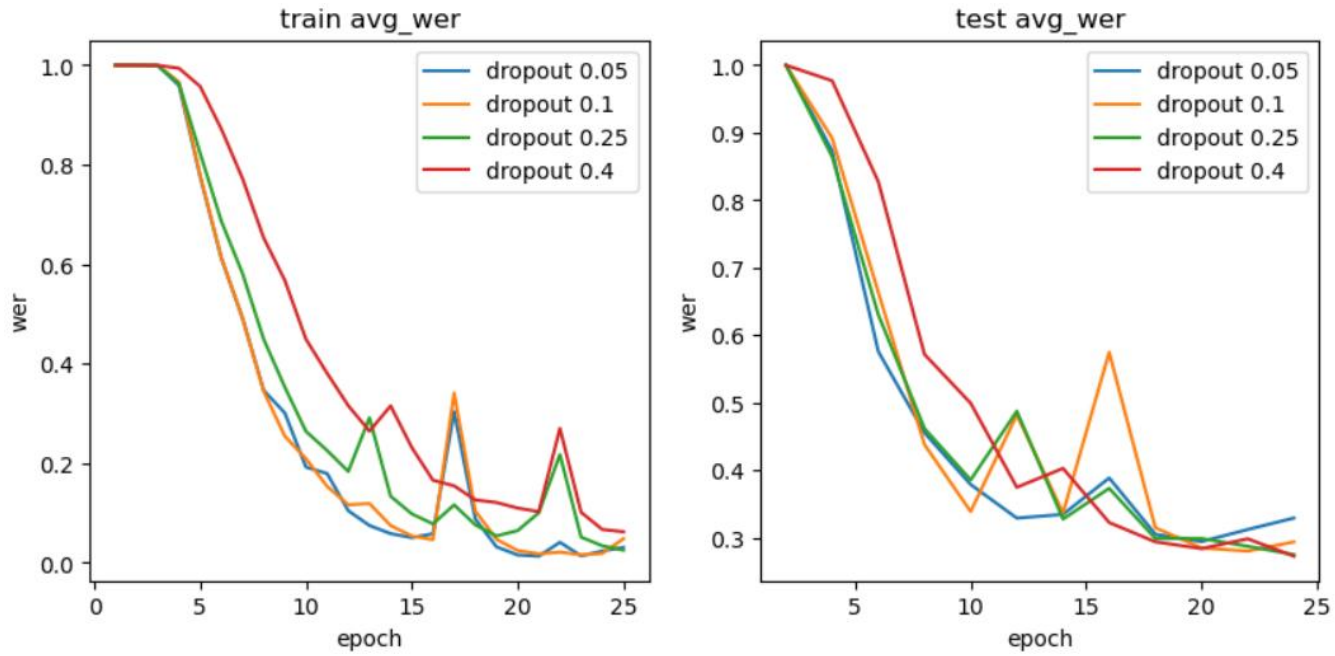

Figure 4: Tỷ lệ lỗi từ (WER) của các tham số dropout

3. Kết quả thu được

Sau các bước thử nghiệm và đánh giá các giá trị khác nhau của tham số. Hai giá trị:

- Dropout = 0.4
- Num_hidden = 256

Là hai giá trị giúp hiệu suất của mô hình nhận diện chữ viết tay đạt kết quả tối ưu nhất. Sau khi sử dụng 2 giá trị này, đã mang lại kết quả đánh giá trên tập dữ liệu kiểm tra thông qua các chỉ số như sau:

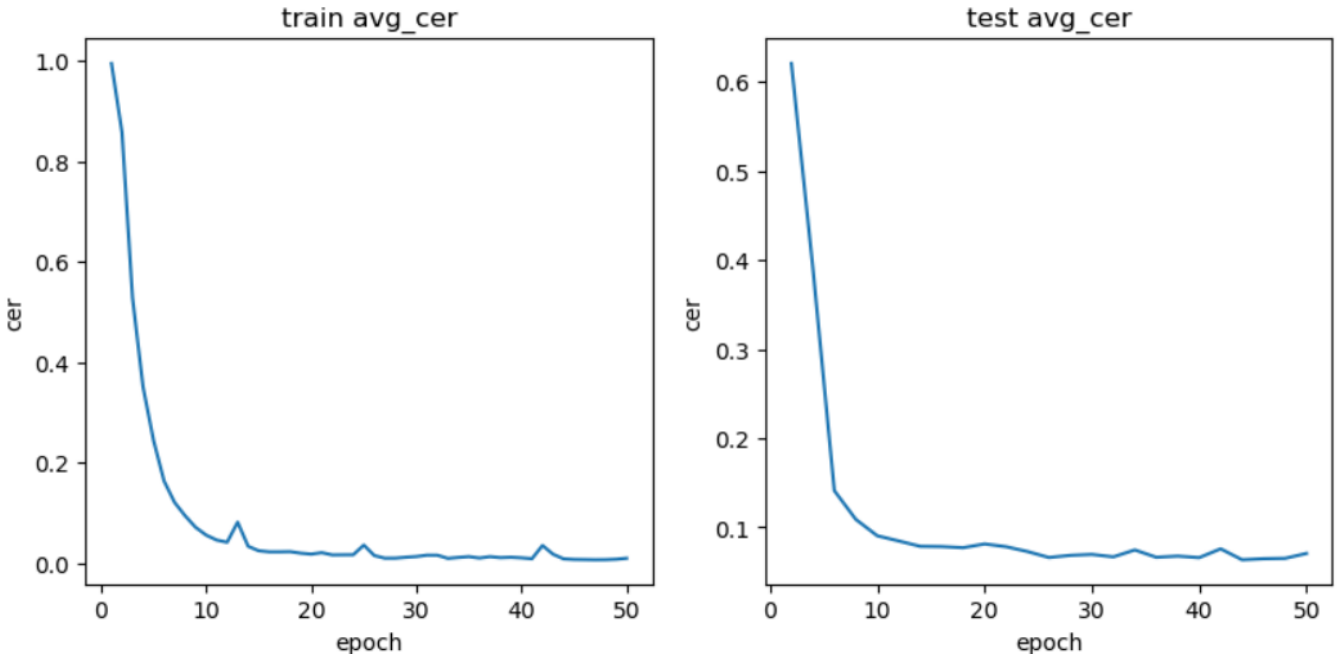

Figure 5: Tỷ lệ lỗi kí tự trên tập huấn luyện và tập thử nghiệm

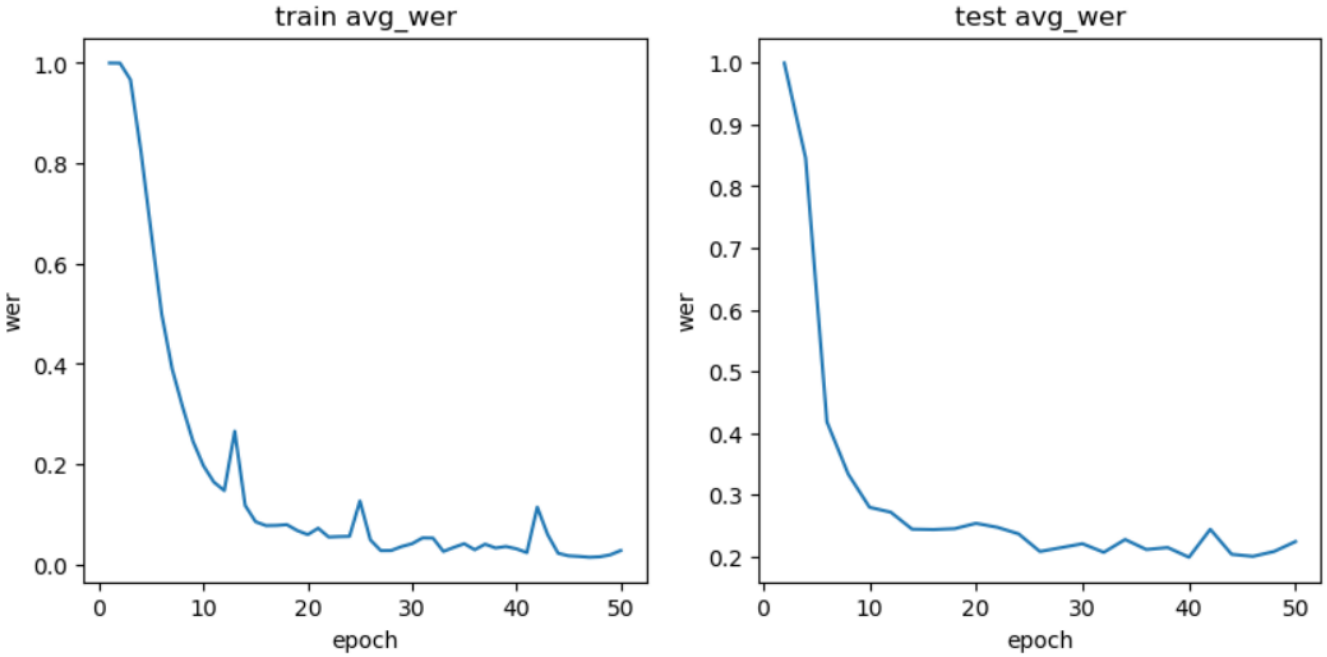

Figure 6: Tỷ lệ lỗi từ trên tập huấn luyện và tập thử nghiệm

Một số kết quả nhận dạng thực tế:

Qua kết quả thu được cho thấy mô hình CRNN với các tham số tối ưu đã đạt được hiệu suất khá cao trong nhận dạng văn bản viết tay trên hình ảnh. Tỷ lệ lỗi ký tự (CER) và tỷ lệ lỗi từ (WER) ở mức có thể chấp nhận được. Cho thấy mô hình có khả năng nhận dạng đa số các ký tự và từ trong hình ảnh. Tuy nhiên, mô hình vẫn chưa thực sự đạt được hiệu suất quá tốt. Nếu số lượng dữ liệu lớn hơn thì mô hình sẽ đạt được hiệu suất cao hơn.