***Shape, rectangle

Description automatically generated***

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**



**BÁO CÁO ĐỒ ÁN CUỐI KỲ**  
**Môn: Nhận dạng - CS338.N21**

**Chủ đề: Thực nghiệm một số Kiến trúc chứa CNN**   
**cho bài toán Nhận diện Captcha văn bản**

**Trần Thành**  **20521924**

**Hồ Thanh Tịnh** **20520813**

**Trần Đức Huy** **20521414**

**Nguyễn Đặng Quang Tuấn 20522116**

**Nội dung**

[I. Phân công công việc 4](#_Toc140246208)

[II. Giới thiệu bài toán 5](#_Toc140246209)

[1. Tổng quan 5](#_Toc140246210)

[2. Tập dữ liệu 6](#_Toc140246211)

[III. Các nghiên cứu liên quan 7](#_Toc140246212)

[1. Bài toán Captcha regconition với Keras 7](#_Toc140246213)

[IV. Các thuật toán 9](#_Toc140246214)

[1. Trích xuất đặc trưng 9](#_Toc140246215)

[1.1. Mạng nơ-ron tích chập (CNN) 9](#_Toc140246216)

[1.2. Kiến trúc 2 lớp CNN của mô hình xây dựng bởi Keras 11](#_Toc140246217)

[1.3. Kiến trúc xây dựng dựa trên AlexNet 12](#_Toc140246218)

[1.4. Kiến trúc xây dựng dựa trên VGG16 15](#_Toc140246219)

[1.5. Kiến trúc xây dựng dựa trên ResNet50V2 17](#_Toc140246220)

[2. Kiến trúc BiLSTM 21](#_Toc140246221)

[3. Kiến trúc CTC 24](#_Toc140246222)

[4. Đánh giá 25](#_Toc140246223)

[4.1. Phương pháp đánh giá 25](#_Toc140246224)

[4.2. Hàm mất mát (Loss function) 26](#_Toc140246225)

[4.3. Độ đo 26](#_Toc140246226)

[V. Kết quả thực nghiệm và phân tích 27](#_Toc140246227)

[1. Kết quả 27](#_Toc140246228)

[2. Phân tích 30](#_Toc140246229)

[VI. Kết luận và hướng phát triển 32](#_Toc140246230)

[1. Kết luận 32](#_Toc140246231)

[2. Hướng phát triển 33](#_Toc140246232)

[VII. Ứng dụng 34](#_Toc140246233)

[1. Thực tế hóa các phương pháp rút trích đặc trưng khác nhau 34](#_Toc140246234)

[2. Phục vụ cho lĩnh vực an toàn thông tin 34](#_Toc140246235)

[VIII. Demo 35](#_Toc140246236)

[IX. Tài liệu tham khảo 35](#_Toc140246237)

# I**. Phân công công việc**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thành**  **viên**  **Công**  **Việc** | **Trần Thành** | **Nguyễn Đặng Quang Tuấn** | **Hồ Thanh Tịnh** | **Trần Đức Huy** |
| **20521924** | **20522116** | **20520813** | **20521414** |
| Phân công, quản lý công việc chung | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill |
| Viết file báo cáo .docx | Checkmark with solid fill |  |  |  |
| ­­­Thực nghiệm dựa trên AlexNet | Checkmark with solid fill |  |  |  |
| Thực nghiệm 2 lớp CNN dựa trên Keras |  | Checkmark with solid fill |  |  |
| Thực nghiệm dựa trên VGG |  | Checkmark with solid fill |  |  |
| Thực nghiệm dựa trên ResNet |  |  | Checkmark with solid fill | Checkmark with solid fill |
| Làm slide | Checkmark with solid fill |  |  | Checkmark with solid fill |
| Giao diện demo |  |  |  | Checkmark with solid fill |
| Mức độ hoàn thiện (%) | 100% | 100% | 100% | 100% |

# II. Giới thiệu bài toán

## 1. Tổng quan

Nhu cầu bảo mật thông tin và an toàn cho người dùng khỏi những cuộc tấn công của robot, thư rác (spam) và phần mềm độc hại luôn được lên ưu tiên hàng đầu, từ đó mà CAPTCHA **(Completely Automated Public Turing test to tell Computers and Humans Apart) được sinh ra.**

Đây là một loại bảo mật bằng cách đưa ra những câu hỏi hoặc yêu cầu đơn giản với người sử dụng, mà các chương trình máy tính hoặc robot khó trả lời. Mục đích nhằm xác thực đối tượng đang truy cập là con người thực sự.

Captcha cũng bao gồm nhiều kiểu như kiểm tra về văn bản (text), ảnh (image), âm thanh (sound) và thường hay xuất hiện gần đây là reCAPTCHA.

Trong nội dung đồ án môn học này, nhóm em mong muốn được thử sức với những captcha chứa văn bản. Mục đích là xây dựng được một mô hình có thể dự đoán (predict) được nội dung văn bản của hình ảnh captcha.

Cụ thể, ảnh đầu vào là một captcha chứa văn bản. Kết quả đầu ra là một đoạn văn bản chứa nội dung bên trong captcha.



*Hình. Một ảnh captcha chứa văn bản (text)*

Từ ví dụ trên, bài toán này được mô tả đơn giản như sau:

+ **Input:** Một ảnh RGB chứa captcha văn bản

+ **Output:** Một đoạn text chứa nội dung trong ảnh

## 2. Tập dữ liệu

Tập dữ liệu bao gồm hơn 113,000 ảnh màu chứa captcha văn bản có 5 kí tự. Bộ dữ được lấy từ bộ dữ liệu Captcha trên Kaggle.

Mỗi ví dụ (example) là một ảnh màu RGB, kích thước là 200x50 (width=200, height=50)

Nhãn tương ứng là một chuỗi (string) chứa **5 kí tự** là nội dung của captcha. Các kí tự này có thể là một trong số các loại kí tự sau:

* Các chữ số 0 – 9
* Các chữ cái La tinh có phân biệt hoa thường a – z và A – Z

A picture containing text, screenshot, colorfulness, magenta

Description automatically generated

*Hình. Bộ dữ liệu captcha văn bản hơn 113 000 ảnh*

Dataset có các nhiễu (noise) thường thấy ở Captcha văn bản nhằm gây khó với các chương trình máy tính:

* Chữ liền nhau, viết đè lên nhau
* Chữ bị cong, lệch, không thẳng hàng
* Các gạch cong chèn lên chữ
* Chữ viết hoa hay thường khó phân biệt.

Dataset nhìn chung cũng gây khó khăn với con người. Vì lý do đặc thù là chống lại các chương trình máy tính không phải là con người nên đây cũng là một thách thức rất lớn của bài toán này.

# III. Các nghiên cứu liên quan

## 1. Bài toán Captcha regconition với Keras

Bài toán captcha regconition với keras xây dựng với cấu trúc chính là :

**(CNN → BiLSTM → CTC)**

A diagram of a diagram

Description automatically generated

*Hình. Kiến trúc mô hình keras cho bài toán Captcha Regconition*

Bộ dữ liệu bao gồm 1040 ảnh captcha .png ảnh là ảnh màu RGB, kích thước width là 200, height là 50. Nhãn là văn bản chứa nội dung trong ảnh captcha đó ở dạng string.

A picture containing text, font, aircraft

Description automatically generated

*Hình. Ảnh dữ liệu 1040 ảnh captcha .png*

Bài toán đã xây dựng một mô hình có khả năng nhận diện tương đối tốt bài toán Nhận diện captcha.

Tuy nhiên, bài toán chỉ mới xây dựng với bộ dữ liệu còn nhỏ ( chỉ hơn 1000 ảnh .png) với nhãn chỉ bao gồm khoảng 20 loại kí tự khiến cho bài toán chưa được tiếp cận với những captcha văn bản chứa nhiều loại kí tự hơn.

# IV. Các thuật toán

Kiến trúc mô hình bao gồm 3 nhóm lớp chính: **(CNN → BiLSTM → CTC)**

Bài này sẽ đánh giá thực nghiệm một số cách rút trích đặc trưng khác nhau (tức thay đổi kiến trúc CNN) dựa trên AlexNet, ResNet,… và sử dụng một bộ dữ liệu lớn hơn so với mô hình mà Keras đã dùng.

## 1. Trích xuất đặc trưng

### 1.1. Mạng nơ-ron tích chập (CNN)

CNN (convolutional neural network) hay mạng nơ-ron tích chập là một phương pháp rút trích đặc trưng chuyên dùng cho phân tích hình ảnh thông qua phép tích chập (convolution) thay cho phép nhân ma trận.

Mạng nơ-ron tích chập (CNN) là một tập hợp các lớp tích chập chồng lên nhau và sử dụng các hàm kích hoạt (activation function) như ReLU hay tanh để tính toán kết quả các node.

Một lớp tích chập là thực hiện các phép nhân tích chập giữa một ảnh với một ma trận kernel. Ma trận kernel sẽ trượt để tính toán ra kết quả cuối cùng. Kết quả này được gọi là đặc trưng tích chập (convolutional feature).

*A screenshot of a grid

Description automatically generated*

*Hình. Nhân tích chập ma trận*

A screenshot of a computer

Description automatically generated

*Hình. Thực hiện nhân tích chập ma trận*

### 1.2. Kiến trúc 2 lớp CNN của mô hình xây dựng bởi Keras

Kiến trúc 2 lớp CNN của mô hình này được xây dựng gồm 2 lớp tích chập (convolutional layer) và 2 lớp tổng hợp giá trị lớn nhất (max pooling) xen kẽ nhau.

Các thông số cụ thể của từng lớp được ghi chi tiết trong hình bên dưới.

*A screenshot of a computer program

Description automatically generated*

*Hình. Đoạn kiến trúc 2 lớp CNN của mô hình xây dựng bởi Keras*

### 1.3. Kiến trúc xây dựng dựa trên AlexNet

AlexNet là tên của một kiểu kiến trúc mạng nơ-ron tích chập (CNN) được thiết kế bởi Alex Krizhevsky với sự cộng tác của Ilya Sutskever và Geoffrey Hinton.

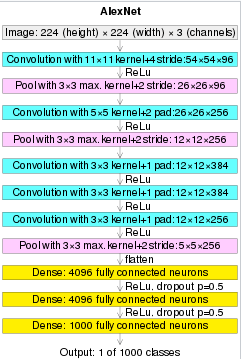
Kiến trúc AlexNet đã thi đấu trong Thử thách nhận dạng hình ảnh quy mô lớn ImageNet vào ngày 30 tháng 9 năm 2012.

Kiến trúc AlexNet sẽ bao gồm 5 lớp chính là CNN đi cùng với Max Pooling, sau đó là 3 lớp Dense (Fully connected layer).

Toàn bộ cấu trúc của AlexNet được thiết kế như hình bên dưới.

A diagram of a function

Description automatically generated



*Hình. Kiến trúc AlexNet*

Trong phạm vi đồ án này, nhóm sẽ dựa trên kiến trúc của các lớp CNN mà xây dựng ra một cấu trúc khác để thực nghiệm cho mô hình gốc được xây dựng bởi Keras.

Kiến trúc xây dựng dựa trên AlexNet, là giữ nguyên số lượng lớp CNN và Maxpooling, cũng như thứ tự của nó để thực hiện trích xuất đặc trưng. Tuy nhiên, có điều chỉnh các ma trận kernel, padding và stride cho phù hợp và bỏ đi xử lí chuẩn hóa (local response normalization).

Kiến trúc xây dựng dựa trên AlexNet cụ thể như sau:

*A screenshot of a computer

Description automatically generated*

*Hình. Kiến trúc xây dựng dựa trên AlexNet*

Kiến trúc xây dựng lại dựa trên AlexNet, được điều chỉnh các ma trận kernel để phù hợp với đoạn kiến trúc BiLSTM và CTC tiếp theo.

Khi thực hiện Input ảnh đầu vào là (200, 50, 3):

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Input** | **Output** | **Reshape** |
| AlexNet | (200, 50, 3) | (2, 1, 256) | (2, 256) |
| Kiến trúc dựa trên AlexNet | (200, 50, 3) | (12, 3, 128) | (12, 384) |

*Bảng. Output giữa AlexNet và Kiến trúc dựa trên AlexNet*

Kiến trúc CTC cần số lượng step >= 5 để tính toán, nhưng số step quá nhỏ của AlexNet ban đầu không đáp ứng được. Do đó, giữ nguyên kiến trúc của AlexNet là hoàn toàn không phù hợp.

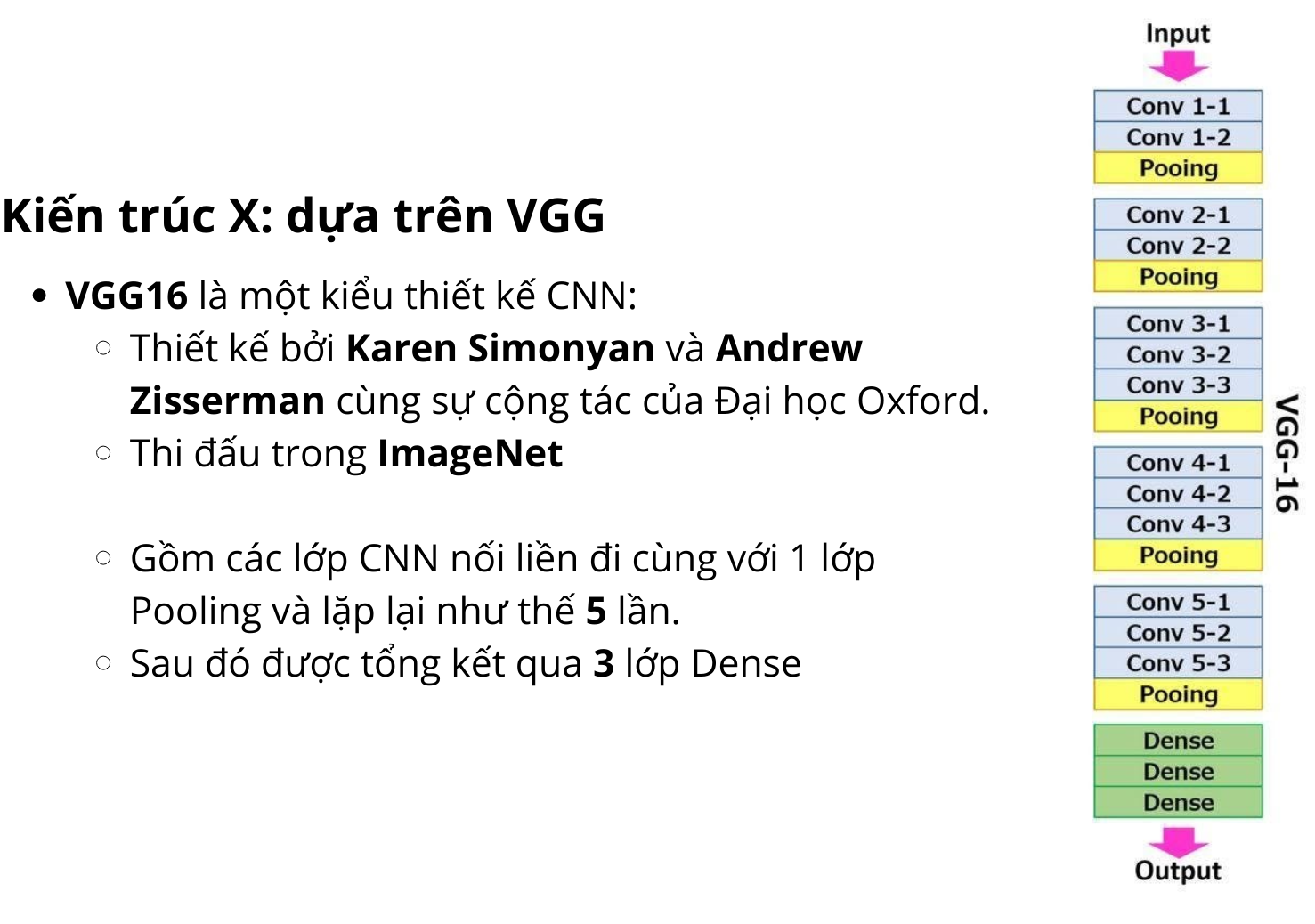
### 1.4. Kiến trúc xây dựng dựa trên VGG16

[Tác giả: Karen Simonyan, Andrew Zisserman. University of Oxford, UK](https://arxiv.org/abs/1409.1556)

A diagram of a diagram

Description automatically generated

*Hình. Kiến trúc VGG16*

**

*Hình. Kiến trúc VGG16*

Với VGG-16, quan điểm về một mạng nơ ron sâu hơn sẽ giúp ích cho cải thiện độ chính xác của mô hình tốt hơn. Về kiến trúc thì VGG-16 vẫn dữ các đặc điểm của AlexNet nhưng có những cải tiến:

* Kiến trúc VGG-16 sâu hơn, bao gồm 13 layers tích chập 2 chiều (thay vì 5 so với AlexNet) và 3 layers fully connected.
* Lần đầu tiên trong VGG-16 chúng ta xuất hiện khái niệm về khối tích chập (block). Đây là những kiến trúc gồm một tập hợp các layers CNN được lặp lại giống nhau. Kiến trúc khối đã khởi nguồn cho một dạng kiến trúc hình mẫu rất thường gặp ở các mạng CNN kể từ đó.
* VGG-16 cũng kế thừa lại hàm activation ReLU ở AlexNet.
* VGG-16 cũng là kiến trúc đầu tiên thay đổi thứ tự của các block khi xếp nhiều layers CNN + max pooling thay vì xen kẽ chỉ một layer CNN + max pooling.
* VGG-16 chỉ sử dụng các bộ lọc kích thước nhỏ 3x3 thay vì nhiều kích thước bộ lọc như AlexNet. Kích thước bộ lọc nhỏ sẽ giúp giảm số lượng tham số cho mô hình và mang lại hiệu quả tính toán hơn

Mạng VGG-16 sâu hơn so với AlexNet và số lượng tham số của nó lên tới 138 triệu tham số. Đây là một trong những mạng mà có số lượng tham số lớn nhất.

Bắt đầu từ VGG-16, một hình mẫu chung cho các mạng CNN trong các tác vụ học có giám sát trong xử lý ảnh đã bắt đầu hình thành đó là các mạng trở nên sâu hơn và sử dụng các block dạng [Conv2D\*n + Max Pooling].

Trong báo cáo này nhóm em đã cắt một phần của VGG16 để làm bộ trích xuất đặc trưng. Cụ thể là cắt từ đầu đến block3\_conv3 (Conv2D) kiến trúc model được cắt ra như hình dưới:

*A screenshot of a computer

Description automatically generated*

*Hình. Kiến trúc cắt ra từ VGG16*

Feature map đầu ra của model cắt ra từ VGG16 có shape=(50,12,256) lí do nhóm em chọn cắt tại block này là tạo ra số lượng time-step phù hợp, đầu ra này sẽ làm đầu vào cho lớp biLSTM sau đó, nó sẽ được reshape về dạng (50,12x256) như vậy sẽ có 50 time-step, nếu số lượng này quá nhỏ thì mỗi time-step sẽ chứa đặc trưng của nhiều kí tự, nếu quá lớn mỗi time-step sẽ không chứa hết đặc trưng của một kí tự gây ảnh hưởng tới độ chính xác.

### 1.5. Kiến trúc xây dựng dựa trên ResNet50V2

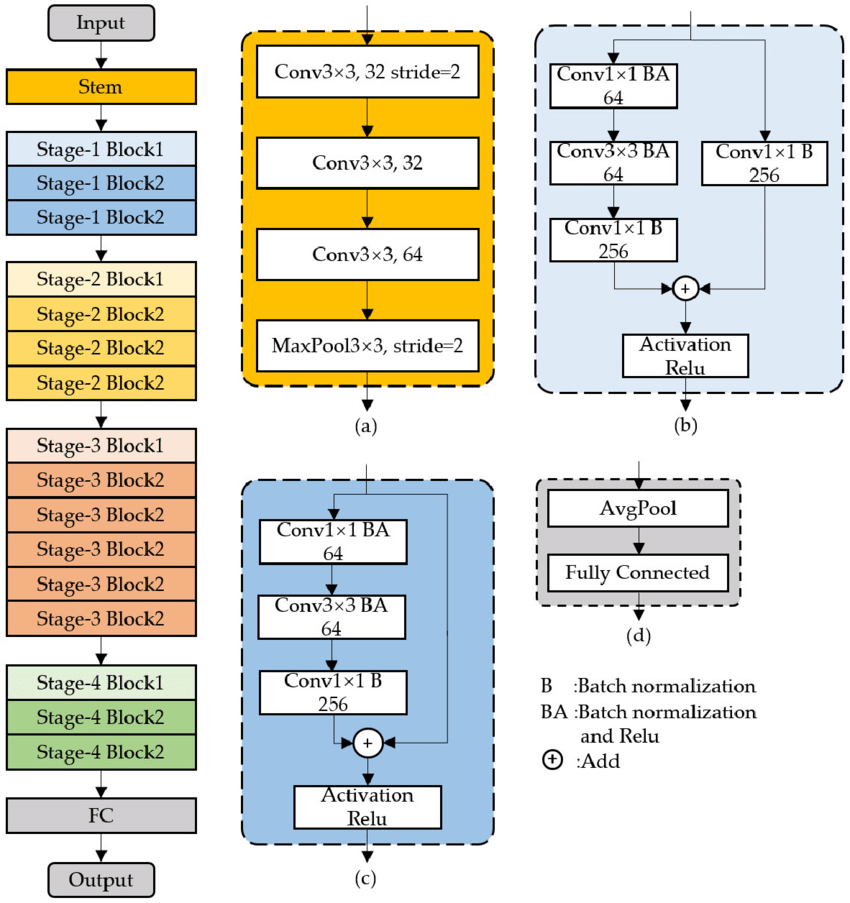
ResNet là một mạng nơ-ron sâu tiên tiến trong xử lý ảnh và nhận dạng đối tượng, đã mang lại nhiều cải tiến quan trọng và đạt được độ chính xác ấn tượng trong các tác vụ học có giám sát.

Một trong những ưu điểm nổi bật của ResNet là khả năng xây dựng mạng sâu hơn so với các mạng trước đó. Kiến trúc này cho phép mạng học được các đặc trưng phức tạp và biểu diễn dữ liệu chi tiết hơn, dẫn đến độ chính xác cải thiện đáng kể.

ResNet sử dụng các khối *residual block* để giải quyết vấn đề *gradient vanishing* và mất mát thông tin trong quá trình huấn luyện. Các khối này cho phép truyền thông tin trực tiếp từ layer trước đến layer sau, giúp mạng học được các đặc trưng sâu hơn và độ chính xác được nâng cao.

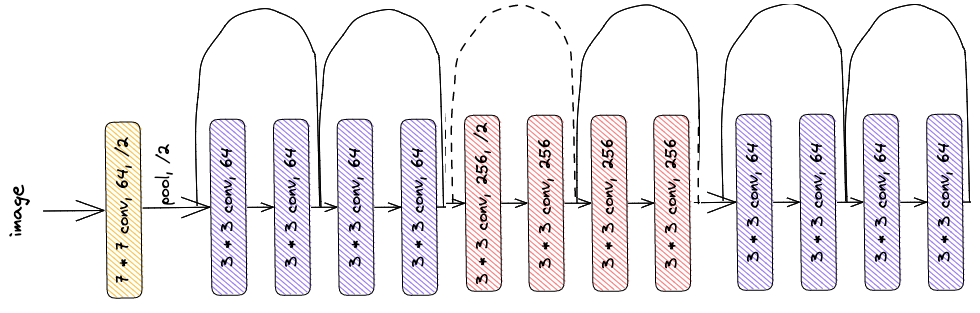
Trong mạng ResNet, hàm activation ReLU được sử dụng để tăng tính phi tuyến và đơn giản hóa tính toán. Điều này đóng góp vào việc mạng học nhanh chóng và cải thiện độ chính xác.

Mạng ResNet cũng sử dụng bộ lọc kích thước nhỏ, chủ yếu là 3x3, trong các *convolutional layer*. Việc sử dụng bộ lọc nhỏ giúp giảm số lượng tham số và tăng hiệu quả tính toán của mạng, đồng thời giữ được độ chính xác cao.



*Hình. Kiến trúc ResNet*

Trong bài toán này, chúng ta sử dụng các block đầu của ResNet để xử lý và trích xuất đặc trưng từ ảnh. Kích thước đầu vào của ảnh là 200x50x3. Các block đầu của ResNet giúp tạo ra một biểu diễn đặc trưng tốt từ ảnh, sử dụng khối *residual block* và *skip connection* để cải thiện độ chính xác và học được các đặc trưng sâu hơn.

*Hình.Minh* *họa kiến trúc ResNet được cắt ra*

Qua các block đầu, quá trình xử lý dừng lại ở kích thước 50x13x64, mục đích của việc dừng này là để giữ cho độ dài chuỗi thời gian của dữ liệu trong bài toán, đồng thời tạo ra một biểu diễn đặc trưng tốt để đưa vào mạng các mạng RNN bổ sung sau đó, (LSTM, BiLSTM, GRU), để xử lý dữ liệu chuỗi thời gian và đưa ra dự đoán cuối cùng.

A screenshot of a computer program

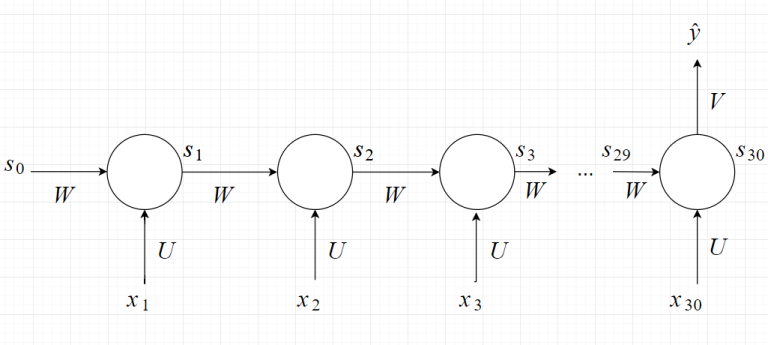
Description automatically generated A screenshot of a computer code

Description automatically generated

*Hình. Kiến trúc ResNet được cắt ra*

## 2. Kiến trúc BiLSTM­­

RNN là một mạng nơ-ron nhân tạo có khả năng xử lý các dữ liệu có tính tuần tự (sequence/series), mang những thông tin có ở các node trước về các node sau và tông hợp kết quả ở đó.

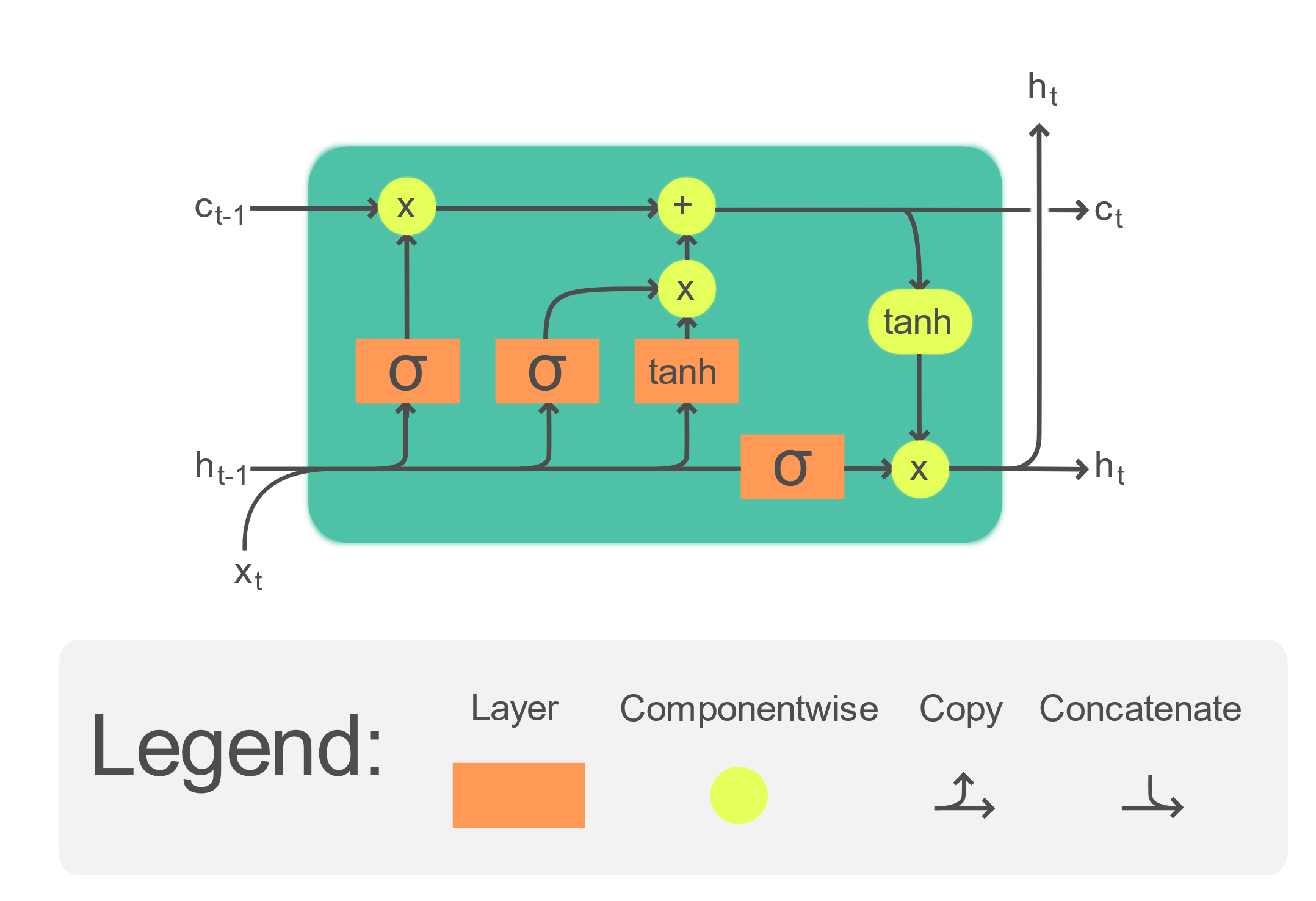
**

*Hình. Kiến trúc của RNN*

Kiến trúc LSTM (Long short-term memory) là một kiểu kiến trúc mạng nơ-ron nhân tạo (ANN) cải tiến dựa trên RNN. LSTM thuộc kiểu mạng nơ-ron lan truyền thẳng (feedforward neural network) và có những kết nối phản hồi, tượng tự như RNN (Recurrent neural network).

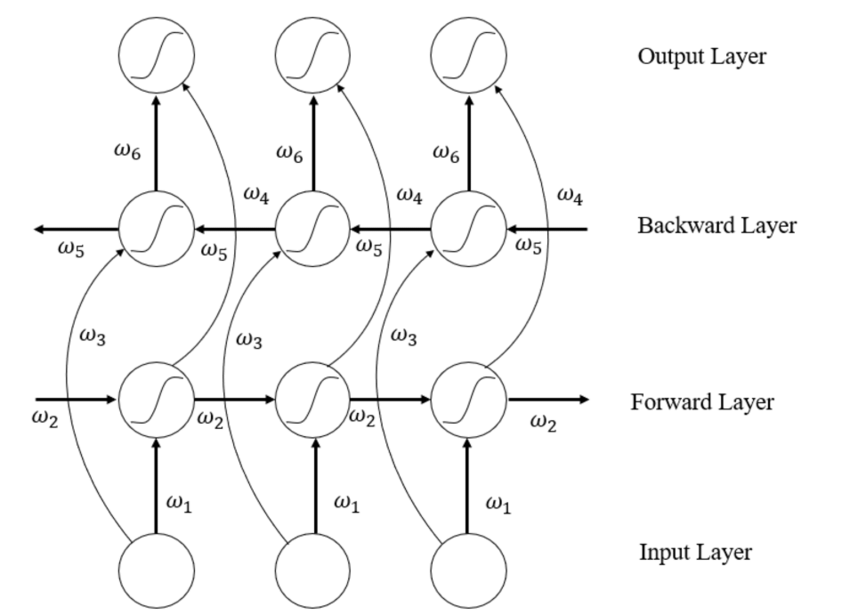
LSTM hơn RNN ở chỗ, ngoài bộ nhớ ngắn hạn (short-term memory) thì LSTM còn lưu trữ bộ nhớ dài hạn, nhằm phục vụ cho việc lan truyền thông tin đi xa hơn nhiều so với RNN. Khả năng này nhờ việc xử lý vấn đề triệt tiêu đạo hàm (vanishing gradient) của các node khi thông tin được truyền đi xa trên các node của một mạng RNN.

Cấu trúc chính của LSTM thường bao gồm cell, input gate, output gate và forget gate.

**

*Hình. Kiến trúc một node trong LSTM*

Kiến trúc BiLSTM là kiến trúc xử lí tuần tự hai chiều bao gồm 2 LSTM, trong đó một LTSM lấy input theo chiều thuận (forward direction) và một LSTM lấy input theo chiều ngược lại (backward direction).



*Hình. Kiến trúc BiLSTM*

## 3. Kiến trúc CTC

CTC (Conectionest temporal classification) là một mạng lưới phân loại có output theo thời gian, nhằm mục đích căn chỉnh các kết quả cho phù hợp.

Đầu vào (input) của nó là từ một mô hình phân loại theo thời gian (temporal classifier). Cụ thể trong đồ án môn học này là BiLSTM.

Gọi *S* là tập huấn luyện gồm các mẫu từ một phân phối cố định *DX*×*Z*​, trong đó:

* *X*=(R*m*)\* là một tập gồm tất cả các chuỗi (sequence) của các vectors số thực có độ dài là *m*.
* *Z*=*L*\* là một tập gồm tất cả các chuỗi chỉ có các ký tự alphabet có hạn *L* của các labels. Các phần tử của *L*\* còn được gọi là label sequences hay labellings.

Mỗi mẫu trong  *S*  bao gồm một cặp chuỗi (*x*, *z*). Chuỗi mục tiêu (target sequence)  *z* = (*z*1​, *z*2​, ..., *zU*​) có độ dài gần như chuỗi đầu vào (input sequence) *x* = (*x*1​, *x*2​, ..., *xT*​), tức là *U* ≤ *T*.

Xác suất của một labelling là tổng xác suất của tất cả các alignment cho ra labelling đó. Trong đó *π* là path hay còn gọi là alignment

*A mathematical equation with a number of symbols

Description automatically generated*

*Hình. Xác suất của một alignment*

Ta định nghĩa một many-to-one map *B*: *L*′*T* ↦ *L*≤*T* bằng cách loại bỏ các ký tự blank và ký tự lặp lại *B*(*a*−*b*−*b*)=*B*(*aa*−*bb*−*b*)=*abb*) và dùng *B* để tính xác suất của một labelling *l* trong *Z*:

*A mathematical equation with numbers and symbols

Description automatically generated*

*Hình. Xác suất của một labelling l*

Từ công thức bên trên, đầu ra của bộ phân loại sẽ là labelling có vẻ gần đúng nhất:

*A close up of a logo

Description automatically generated*

*Hình. Bộ tham số cần đi tìm*

## 4. Đánh giá

### 4.1. Phương pháp đánh giá

Bộ dữ liệu ban đầu gồm hơn 113 000 ảnh .jpg

Bộ dữ liệu được chia thành 3 phần gồm Train dataset, Test dataset, Validation dataset theo tỷ lệ 8:1:1

* Bộ train dataset bao gồm 80% số ảnh khoảng 90 400 ảnh
* Bộ test dataset bao gồm 10% số ảnh, khoảng 11 300 ảnh
* Bộ validation dataset bao gồm 10% số ảnh, khoảng 11 300 ảnh

Bộ dữ liệu train dataset và validation dataset sẽ được dùng trong quá trình huấn luyện mô hình. Các mô hình đều được kiểm thử (test) trên bộ test dataset.

### 4.2. Hàm mất mát (Loss function)

Ta sẽ xây dựng hàm mất mát để có thể train bằng gradient descent. Hàm mất mát được lấy theo maximum likelihood. Nghĩa là khi cực đại hóa nó thì sẽ cực đại hóa log likelihood.

Như vậy hàm mất mát (hàm mục tiêu) sẽ là **negative log likelihood**:

*A mathematical equation with a number and a symbol

Description automatically generated*

*Hình. Hàm mất mát với CTC*

### 4.3. Độ đo

Sử dụng độ đo Accuracy cho kí tự và cho từ:

* **Character Accuracy**: Tỉ lệ giữa số lượng kí tự dự đoán đúng với số lượng kí tự cần dự đoán. Một kí tự được gọi là chính xác khi loại kí tự và vị trí kí tự đúng với nhãn thực (label) của nó.
* **Word Accuracy:** Tỉ lệ giữa số lượng từ dự đoán đúng với số lượng từ cần dự đoán. Một từ được gọi là chính xác nếu nó có tất cả các kí tự đều chính xác.

Ý nghĩa độ đo:

* **Character Accuracy** cho biết độ chính xác dựa trên các kí tự, điều này có ý nghĩa khi xác định được mức độ sai của các nhãn dự đoán, thay vì chỉ xét xem kết quả dự đoán là đúng hay sai.
* **Word Accuracy** cho biết độ chính xác so với nhãn thực, hay mức chính xác của mô hình dự đoán.

# V. Kết quả thực nghiệm và phân tích

## 1. Kết quả

Kiến trúc các mô hình đều theo dạng **(X → BiLSTM → CTC).**

Trong đó **X** là kiến trúc nhằm **trích xuất đặc trưng** dựa trên CNN, có thể là:

* 2 lớp CNN của mô hình xây dựng bởi Keras
* Kiến trúc dựa trên AlexNet
* Kiến trúc dựa trên VGG
* Kiến trúc dựa trên ResNet50V2

Giải thích thuật ngữ:

* Feature Extractor (**FE**): kiến trúc sử dụng model Pre-trained với trọng số bị đóng băng chỉ để trích xuất đặc trưng.
* Transfer Learning (**TL**): kiến trúc sử dụng model Pre-trained làm điểm để bắt đầu thay vì khởi tạo ngẫu nhiên trọng số. Kiến trúc huấn luyện lại toàn bộ bộ trọng số.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X | | **2 lớp CNN của Keras** | **Kiến trúc dựa trên AlexNet** | **Kiến trúc dựa trên VGG**  **(FE)** | **Kiến trúc dựa trên ResNet50V2**  **(FE)** | **Kiến trúc dựa trên ResNet50V2**  **(TL)** |
| Character Accuracy | Val | 89.36% | 85.21% | 80.74% | 74.73% | 89.32% |
| Test | 89.36% | 84.92% | 79.87% | 74.60% | 89.04% |
| Word  Accuracy | Val | 69.61% | 59.00% | 45.76% | 34.42% | 70.26% |
| Test | 69.41% | 58.52% | 44.45% | 34.27% | 70.04% |
| Train Loss | | 1.4299 | 2.0872 | 3.5267 | 4.7796 | 0.8199 |
| Validation Loss | | 1.0632 | 2.2726 | 2.6772 | 4.1457 | 1.2719 |

*Bảng. Kết quả thực nghiệm*

Kết quả chi tiết quá trình huấn luyện mô hình được biểu diễn theo sơ đồ bên dưới (Loss trên tập train và validation):

*A graph of training and validation

Description automatically generated*

*Hình. Quá trình train với kiến trúc dựa trên AlexNet*

A graph of a graph

Description automatically generated

*Hình. Quá trình train với kiến trúc dựa trên VGG16*

A graph of training loss

Description automatically generated

*Hình. Quá trình train với kiến trúc dựa trên ResNet50V2 (FE)*

A graph of a train loss

Description automatically generated

*Hình. Quá trình train với kiến trúc dựa trên ResNet50V2 (TL)*

*A screenshot of a computer

Description automatically generated*

*Hình. Kết quả dự đoán với AlexNet*

## 2. Phân tích

Trong đồ án này độ chính xác của các thực nghiệm đạt được ở mức khá ổn trên tập test. Phân hóa trên Word Accuracy đối với tập test rất rõ ràng.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2 lớp CNN của Keras** | **Kiến trúc dựa trên AlexNet** | **Kiến trúc dựa trên VGG**  **(FE)** | **Kiến trúc dựa trên ResNet50V2**  **(FE)** | **Kiến trúc dựa trên ResNet50V2**  **(TL)** |
| 69.41% | 58.52% | 44.45% | 34.27% | 70.04% |

*Bảng. Kết quả Word Accuracy trên tập test*

Độ đo trên Character Accuray đối với tập test thì lại có phần tương đương nhau.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2 lớp CNN của Keras** | **Kiến trúc dựa trên AlexNet** | **Kiến trúc dựa trên VGG**  **(FE)** | **Kiến trúc dựa trên ResNet50V2**  **(FE)** | **Kiến trúc dựa trên ResNet50V2**  **(TL)** |
| 89.36% | 84.92% | 79.87% | 74.60% | 89.04% |

*Bảng. Kết quả Character Accuracy trên tập test*

Từ hai nhận xét trên ta nhận ra rằng, các mô hình đều làm khá tốt trong việc nhận diện được một kí tự riêng lẻ, tuy nhiên độ chính xác của mô hình không được cao là do những từ chỉ được nhận diện đúng một lượng nhất định kí tự chứ không phải toàn bộ. Một từ (nhãn dự đoán) gồm 5 kí tự nhưng nếu chỉ được nhận diện đúng 4/5 kí tự thì vẫn bị đánh giá là từ đó sai.

Từ những sơ đồ kết quả hàm mất mát (Loss) trong khi huấn luyện ở tập train và validation thì nhận thấy hai kết quả này gần như giảm đồng thời với nhau, không hề bị lệch nhau quá nhiều, không cho thấy dấu hiệu bị overfitting.

So sánh giữa các kiến trúc với nhau thì mô hình cắt từ kiến trúc gốc (sử dụng lại bộ trọng số) cho kết quả kém hơn so với những kiến trúc còn lại bao gồm những mô hình dữ nguyên kiến trúc hoàn chỉnh và huấn luyện lại từ đầu cho những bộ trọng số mới.

Các mô hình còn nhầm lẫn giữa các kí tự khó phân biệt, chữ in hoa in thường, chữ nhiều nét tương đồng nhau như các cặp: I i L l 1, B 8, …

*A close up of a logo

Description automatically generated*

*Hình. Ảnh predict bị nhầm lẫn*

# VI. Kết luận và hướng phát triển

## 1. Kết luận

Độ chính xác của mô hình vẫn chưa đạt được như kỳ vọng, chỉ dừng ở mức Khá, cụ thể là với các captcha đơn giản mô hình có thể nhận diện chính xác, với những trường hợp khó, con người cũng có thể sai sót mô hình đã không đưa ra được kết qủa chính xác, đòi hỏi phải cải tiến thêm.

**Ưu điểm:**

* + Giải quyết được những vấn đề cơ bản là nhận diện captcha văn bản đối với những đối tượng dễ (con người nhận biết được dễ dàng, ít nhiễu).
  + Quá trình huấn luyện với thời gian không quá nhiều.
  + Nhiều trường hợp sai, tuy nhiên vẫn có triển vọng khắc phục được

**Nhược điểm:**

* + Chưa dự đoán được những kết quả khó
  + Kiến trúc các nhóm lớp gồm nhiều lớp và tương đối phức tạp

**Nguyên nhân:**

* + Bộ dữ liệu sau khi nhóm kiểm tra lại vẫn phát hiện còn sai sót. Nhóm kiểm tra với hơn 1000 ảnh thì phát hiện có hơn 50 ảnh bị gắn sai nhãn.
  + Kiến trúc mô hình mang tính thử nghiệm có thể chưa thực sự được tối ưu hóa các bộ tham số hyperparameter cho phù hợp nên có thể kết quả chưa thực sự khả quan (hyperparameter mang tính thử nhiều lần để tìm ra).
  + Sai sót trong khả năng nhận diện ở một số kí tự chứ cũng dẫn đến nhãn đó bị đánh giá là sai trong Độ đo Word Accuracy.
  + Đặc thù bộ dữ liệu là xử lý với dữ liệu chống robot nên đây là trở ngại rất lớn cho những bài toán máy học.

*A close up of a logo

Description automatically generated*

*Hình. Một số ảnh bị gán sai nhãn thực*

## 2. Hướng phát triển

Tăng số lượng eporch, do hạn chế về mặt tài nguyên nên nhóm chưa có cơ hội train với số eporch cao hơn. Việc train với số Eporch cao hơn cũng là một hướng phát triển cực kì tiềm năng.

Cái thiện kiến trúc chứa CNN. Những hyperparameter của kiến trúc đã được nhóm sử dụng còn mang tính thử nghiệm, chưa phải là tối ưu nhất, chưa có cơ hội phát huy hết tiềm năng của kiến trúc đó. Tìm bằng cách thử nhiều hyperparameter khác nhau và chọn cái ổn nhất.

Thử nghiệm với những kiến trúc chứa CNN được dùng để trích xuất đặc trưng khác. Những kiến trúc đang dùng có thể không thực sự phù hợp cho bài toán đang quan tâm, do dó khả năng của những kiến trúc này đóng góp được là có giới hạn, muốn nâng cao giới hạn này thì cần phải thử và tìm ra kiến trúc được cho là hợp lý nhất có thể.

Thử nghiệm những kiến trúc khác biệt cho nhóm lớp Recurrent và Transcription. Thay thể những kiến trúc BiLSTM và CTC thành những loại kiến trúc khác cũng có cơ hội để nâng cao khả năng hiệu quả của mô hình.

Cải thiện kĩ thuật huấn luyện: hiện tại nhóm chưa áp dụng nhiều kĩ thuật huấn luyện mà chỉ đang sài early-stopping. Nhóm định thử nghiệm với LambdaCallback, Learning Rate Scheduler, Tuning các tham số của kĩ thuật huấn luyện, model,...Bằng việc sử dụng Callback + loop, nhóm muốn tăng thêm số lượng epoch cũng như giảm việc bị overfitting thay vì sử dụng mỗi EarlyStopping

# VII. Ứng dụng

## 1. Thực tế hóa các phương pháp rút trích đặc trưng khác nhau

Thử nghiệm nhiều mô hình khác nhau có cấu trúc tương tự**.** Nhằm phục vụ cho nhu cầu học hỏi và tìm hiểu của những sinh viên có nhu cầu hiểu biết thêm về Thị giác máy tính (Computer Vision) và các bài toán có sử dụng các cấu trúc trong máy học (machine learning).

Bài toán này là một dạng thực hành để áp dụng những kiến thức đã học vào xây dựng một mô hình tương đối hoàn chỉnh.

Thực hiện so sánh kết quả giữa những phương pháp khác nhau, từ đó đánh giá được những ưu nhược điểm trong từng trường hợp cụ thể.

## 2. Phục vụ cho lĩnh vực an toàn thông tin

Mục tiêu chính của bài toán là tập trung vào lĩnh vực an toàn thông tin.

Nhằm xác định khả năng có thể giải quyết bài toán captcha regconition từ đó có thể xác minh được mức độ an toàn của việc sử dụng hình thức bảo mật thông qua captcha chứa văn bản.

Thông qua đó có thể kịp thời đưa ra những giải pháp cải tiến hoặc thậm chí thay đổi loại hình bảo mật mới để tăng cường mức độ an toàn thông tin cho hệ thống.

Những hình thức bảo mật tốt hơn có thể được sinh ra và thay thế chỗ cho captcha, cuối cùng là nâng cao mức độ an toàn cho hệ thống, cũng như nâng cao mức độ an toàn cho người sử dụng.

# VIII. Demo

Demo của nhóm được thực hiện ở link google colab bên dưới:

Demo:<https://colab.research.google.com/drive/1YAYRUe2VqV22po1fhVHvoWZjkHFuyAjk?usp=sharing>

# IX. Tài liệu tham khảo

[1]. <https://en.wikipedia.org/wiki/CAPTCHA>

[2]. <https://keras.io/examples/vision/captcha_ocr/>

[3]. [https://www.kaggle.com/datasets/parsasam/captcha-dataset](https://www.kaggle.com/datasets/parsasam/captcha-dataset?fbclid=IwAR33mwEQzRg5U878b7SR4x9SHQXu_1ZfukSkn_d0uD6C-2kFq4mgWFGxZqA)

[4]. <https://en.wikipedia.org>

[5]. <https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/>

[6]. <https://www.tensorflow.org/guide/keras?hl=vi>

[7]. <https://paperswithcode.com/method/bilstm>

[8]. <https://en.wikipedia.org/wiki/Long_short-term_memory>

[9]. <https://viblo.asia/p/gioi-thieu-ve-connectionist-temporal-classification-ctc-phan-1-ORNZqj08l0n>

[10] <https://arxiv.org/abs/1409.1556>