Phạm Duy Long

NHẬN DẠNG CAPTCHA

Mục lục

CAPTCHA là gì? Tại sao cần nhận dạng?

CAPTCHA là gì?

Định nghĩa: Completely Automated Public Turing test to tell Computers and Humans Apart.

Mục đích: Phân biệt người dùng thật với bot, chống spam, bảo vệ tài khoản.

Tại sao cần nhận dạng CAPTCHA tự động?

Phân tích bảo mật: Thử nghiệm độ mạnh của các hệ thống CAPTCHA.

Nghiên cứu học máy: Một bài toán challenging cho Computer Vision và NLP.

Tổng quan về CRNN (Convolutional Recurrent Neural Network)

Ý tưởng chính: Kết hợp sức mạnh của CNN và RNN (LSTM/GRU) trong xử lý chuỗi.

- CNN (Convolutional Neural Network): Trích xuất đặc trưng không gian từ ảnh (cạnh, góc, hình dạng ký tự).
- RNN (Recurrent Neural Network - Bidirectional LSTM): Xử lý chuỗi đặc trưng, học mối quan hệ giữa các ký tự.

Sơ đồ kiến trúc tổng quát

Input Image -> CNN Layers -> Feature Sequence -> RNN (Bi-LSTM) Layers -> Output Sequence.

CTC Loss

Huấn luyện mô hình dự đoán chuỗi ký tự từ ảnh mà không cần biết chính xác vị trí/rìa của từng ký tự.

Giải pháp của CTC Loss:

- Cho phép mô hình dự đoán chuỗi đầu ra dài hơn nhãn thực tế.
- Sử dụng ký tự "blank" (-) đại diện cho khoảng trống hoặc dự đoán lặp lại.
- Gộp các ký tự giống nhau liền kề (ví dụ: HH_ELL_LLO -> HELLO).
- Tính toán xác suất của tất cả các "path" dẫn đến nhãn đúng.

Minh họa:

Input: HH_EL_L_O -> Output: HELLO

Thiết kế và Triển khai hệ thống (Sử dụng MLTU)

Tổng quan Pipeline:

- Thu thập và tiền xử lý dữ liệu CAPTCHA.
- Xây dựng mô hình CRNN.
- Huấn luyện mô hình với CTC Loss.
- Đánh giá và Demo.

Tải và Tiền xử lý dữ liệu:

- Bước tải dataset.
- Sử dụng mltu.DataProvider và Transformers (ImageResizer, LabelIndexer, LabelPadding).
- Áp dụng Augmentors (RandomBrightness, RandomRotate) cho dữ liệu huấn luyện.

Cấu hình mô hình (configs.py):

- Giới thiệu về mltu.configs.BaseModelConfigs.
- Các tham số chính: height, width, max_text_length, vocab, batch_size, learning_rate.

Kiến trúc Mô hình chi tiết

Giải thích chi tiết các khối:

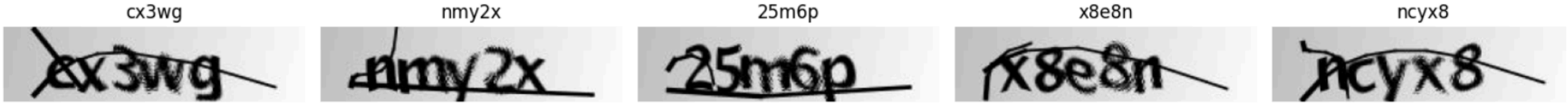
- CNN (Residual Blocks): Trích xuất đặc trưng phức tạp, giảm kích thước không gian.
- Reshape: Chuyển đổi đầu ra 2D của CNN thành chuỗi 1D cho RNN.
- Bi-LSTM: Xử lý chuỗi đặc trưng, học mối quan hệ ngữ cảnh hai chiều.
- Dense + Softmax: Lớp đầu ra tạo phân phối xác suất cho từng ký tự (bao gồm "blank").

Thiết kế và Triển khai hệ thống (Sử dụng MLTU)

Tải Tệp Dataset

- Tải tệp captcha_images_v2.zip, bao gồm 1000 ảnh.
- Lưu đường dẫn của các file ảnh vào Drive.
- Xuất ảnh minh họa cho từng file.

Ảnh minh họa trong dataset



Thiết kế và Triển khai hệ thống (Sử dụng MLTU)

```
%%writefile configs.py
# configs.py
import os
from datetime import datetime
from mltu.configs import BaseModelConfigs

class ModelConfigs(BaseModelConfigs):
    def __init__(self):
        super().__init__()
        self.model_path = os.path.join('/content', 'drive', 'MyDrive', 'captcha_model')
        self.vocab = ''
        self.height = 50
        self.width = 200
        self.max_text_length = 0
        self.batch_size = 64
        self.learning_rate = 1e-3
        self.train_epochs = 1000
        self.train_workers = 20

configs = ModelConfigs()
```

Vai Trò của `configs.py`

- Quản lý tập trung các siêu tham số (hyperparameters).
- Giúp dễ dàng điều chỉnh và thử nghiệm các cấu hình khác nhau.
- Đảm bảo tính nhất quán trong quá trình huấn luyện và đánh giá.

Thiết kế và Triển khai hệ thống (Sử dụng MLTU)

Tiền xử lý dữ liệu và khởi tạo DataProvider

```
# Khởi tạo DataProvider của mltu.tensorflow
data_provider = DataProvider(
    dataset=dataset,
    skip_validation=True,
    batch_size=configs.batch_size,
    data_preprocessors=[ImageReader(CVImage)], # Sử dụng CVImage của mltu cho việc đọc ảnh
    transformers=[
        ImageResizer(configs.width, configs.height),
        LabelIndexer(configs.vocab),
        LabelPadding(max_word_length=configs.max_text_length, padding_value=len(configs.vocab))
    ],
)
print("DataProvider đã được khởi tạo thành công!")
```

Cụ thể, đoạn code này sẽ:

- Tải và thu thập dữ liệu: Duyệt qua các tệp ảnh CAPTCHA, trích xuất nhãn từ tên tệp, và xây dựng bộ từ vựng (vocab) chứa tất cả các ký tự duy nhất có trong nhãn, đồng thời xác định độ dài tối đa của một CAPTCHA (max_text_length).
- Khởi tạo DataProvider: Tạo một đối tượng DataProvider từ mltu.tensorflow, đây là một công cụ hiệu quả để quản lý và cung cấp dữ liệu theo từng lô (batch) cho mô hình.

Huấn luyện Mô hình

mô hình ResNet (Residual Network)

```
x2 = residual_block(x1, 16, activation=activation, skip_conv=True, strides=2, dropout=dropout)
x3 = residual_block(x2, 16, activation=activation, skip_conv=False, strides=1, dropout=dropout)

x4 = residual_block(x3, 32, activation=activation, skip_conv=True, strides=2, dropout=dropout)
x5 = residual_block(x4, 32, activation=activation, skip_conv=False, strides=1, dropout=dropout)

x6 = residual_block(x5, 64, activation=activation, skip_conv=True, strides=2, dropout=dropout)
x7 = residual_block(x6, 32, activation=activation, skip_conv=True, strides=1, dropout=dropout)

x8 = residual_block(x7, 64, activation=activation, skip_conv=True, strides=2, dropout=dropout)
x9 = residual_block(x8, 64, activation=activation, skip_conv=False, strides=1, dropout=dropout)

squeezed = layers.Reshape((x9.shape[-3] * x9.shape[-2], x9.shape[-1]))(x9)

blstm = layers.Bidirectional(layers.LSTM(128, return_sequences=True))(squeezed)
blstm = layers.Dropout(dropout)(blstm)

output = layers.Dense(output_dim + 1, activation="softmax", name="output")(blstm)
```

Các mạng sâu hơn và được huấn luyện ổn định hơn với residual_block thường có khả năng tổng quát hóa tốt hơn trên dữ liệu mới, điều này rất quan trọng đối với một hệ thống nhận dạng CAPTCHA thực tế.

Thiết kế và Triển khai hệ thống (Sử dụng MLTU)

RNN (Recurrent Neural Network) sẽ nhận chuỗi đặc trưng từ CNN và xử lý chúng theo trình tự để học mối quan hệ giữa các ký tự và dự đoán chuỗi cuối cùng. Bi-LSTM hiệu quả vì nó xem xét ngữ cảnh từ cả hai phía của một ký tự.

```
blstm = layers.Bidirectional(layers.LSTM(128, return_sequences=True))(squeezed)
blstm = layers.Dropout(dropout)(blstm)
```

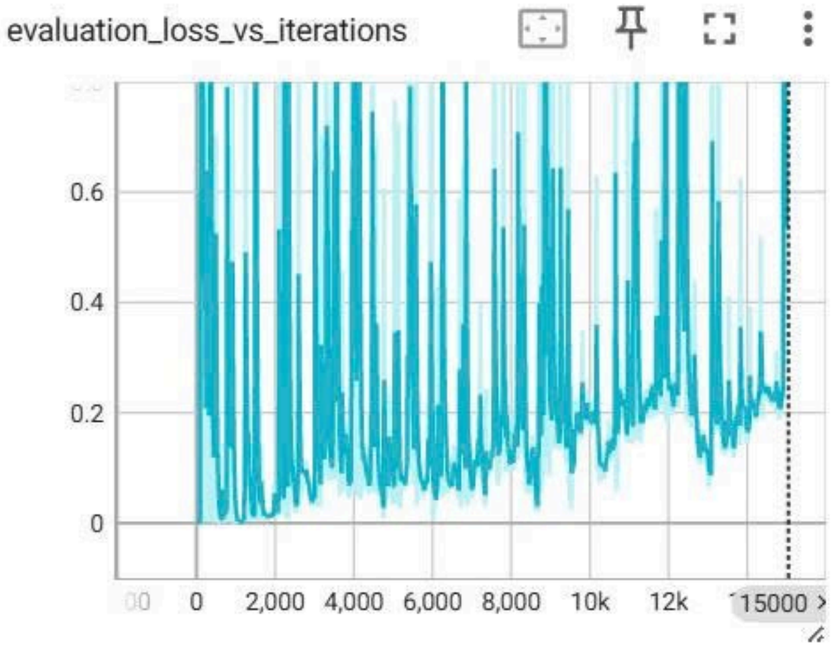
Đoạn code `blstm = layers.Bidirectional(layers.LSTM(128, return_sequences=True))(squeezed)` tạo ra lớp RNN chính của mô hình. Nó nhận squeezed (chuỗi đặc trưng từ CNN) làm đầu vào và học các mối quan hệ tuần tự giữa các đặc trưng đó để nhận dạng các ký tự

Thiết kế và Triển khai hệ thống (Sử dụng MLTU)

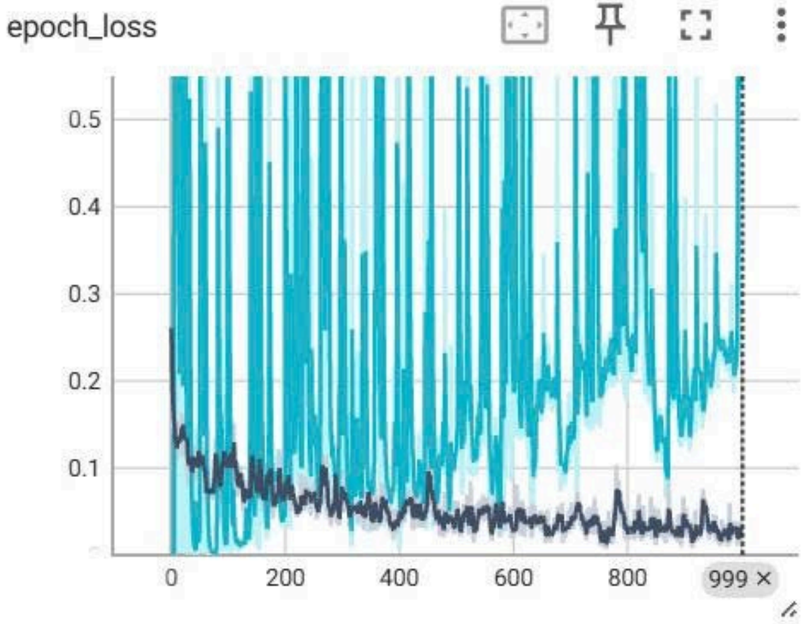
```
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=configs.learning_rate),
    loss=CTCloss(), # Dùng CTCloss của mltu.tensorflow
    metrics=[CWERMetric(padding_token=len(configs.vocab))],
    run_eagerly=False
)
```

loss=CTCloss() cho TensorFlow biết rằng hàm CTCloss sẽ được sử dụng để tính toán giá trị lỗi trong mỗi bước huấn luyện. Thuật toán tối ưu (Optimizer, ví dụ Adam) sau đó sẽ sử dụng giá trị lỗi này để điều chỉnh trọng số của toàn bộ mô hình (cả CNN và RNN) theo hướng giảm thiểu lỗi, giúp mô hình học cách nhận dạng CAPTCHA chính xác hơn.

Đánh giá và Kết quả Demo



Run ↑	Smoothed	Value	Step	Relative
validation	0.539	0.2842	15,000	1.202 hr

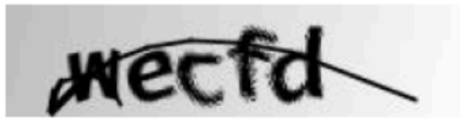


Run ↑	Smoothed	Value	Step	Relative
train	0.0258	0.0162	999	1.202 hr
validation	0.539	0.2842	999	1.202 hr

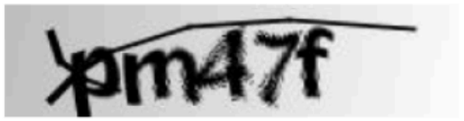
Đánh giá và Kết quả Demo

```
--- Hiển thị ảnh mẫu và kết quả dự đoán ---
```

Label: wecfd
Pred: 272
CER: 1.00



Label: pm47f
Pred: 272
CER: 0.80



Label: nbwnn
Pred: 272
CER: 1.00

Label: 7dyww
Pred: 272
CER: 1.00

Label: n6xc5
Pred: 272
CER: 1.00