

MAYFEST

2023

ĐANG DIỄN RA TRÊN

VIBLO PLATFORM

XEM THẺ LỆ

Nguyen Viet Hoai

@HoaiNV

Theo dõi

Đã đăng vào thg 10 19, 2020

1.1K

57

14

Sign in to viblo.asia with Google

A

Andy Singal

andysingal@gmail.com

A

Alpha singal

alphasingal@gmail.com


1 more account

1.3K

1

0

# Tự động điều chỉnh siêu tham số với Optuna và Pytorch

 Bài đăng này đã không được cập nhật trong 2 năm

## Giới thiệu

## Siêu tham số là gì ?

Đối với những bạn đã xây dựng các mô hình deep learning thì chắc hẳn không còn xa lạ gì với khái niệm siêu tham số này nữa.Trong học máy và học sâu, siêu tham số là những tham số mà giá trị của nó có thể điều khiển quá trình huấn luyện của mô hình. Nó không cố định, thay đổi theo từng bài toán, từng quá trình huấn luyện và có thể thay đổi trong quá trình huấn luyện đó. Đặc biệt, nó đóng vai trò quan trọng trong việc quyết định hiệu suất của mô hình. Ví dụ về siêu tham số như là số lượng các units và các layers trong neural network, hay trong network trainer có các siêu tham số như là batchsize, learning rate schedule, optimizer, momentum, adam alpha,...

## Optuna là gì ?

Optuna là 1 framwork hỗ trợ việc tự động điều chỉnh tham số mô hình để mô hình có thể đạt được hiệu năng tốt nhất ứng.

## Khó khăn gặp phải khi phải điều chỉnh tham số thủ công

Đặt vấn đề : các bạn cần tạo một mô hình MLP, làm sao để chọn số hidden layers, số units trong 1 layers một cách phù hợp và chính xác ?

Việc chọn siêu tham số khi thực hiện training mô hình dường như đều dựa vào cảm tính và kinh nghiệm của các bạn. Và điều nay không phải lúc nào cũng có thể làm cho mô hình đạt performance tốt nhất . Còn đối với những bạn chưa có kinh nghiệm lựa chọn siêu tham số thì đây thực sự là một thử thách.

## Cách sử dụng thư viện Optuna trong Pytorch

# Install

```
pip install optuna
```

# Định nghĩa objective function

```
def objective ( trial ):

    return
```

# Trial object definition

**Trial** là một đối tượng thể hiện của một class được thực thi trong Optuna.Nó được sử dụng để định nghĩa các siêu tham số được tối ưu.Giá trị được lấy trong phạm vi được bạn định nghĩa, thông tin của các tham số được tìm kiếm trong quá khứ vẫn được giữ lại và giá trị mới sẽ dựa trên những thông tin đó. **Trial** được định nghĩa như sau:

```
# Tham số thực hiện chọn loại
param1 = Trial.Suggest_categorical( Name , Choices )

# Định nghĩa tham số nguyên cho trial
param2 = trial.Suggest_int( name , low , high )

# Biểu diễn tham số có giá trị liên tục
param3 = trial.Suggest_uniform( name , low , high )

# Biểu diễn tham số có giá trị rời rạc
param4 = trial.Suggest_discrete_uniform( name , low , high , q )

# Biểu diễn các tham số logarithm
```

**name** là kiểu string và giá trị là tên tham số đó. **Choicesis** là dạng list và nó được biểu diễn là sự lựa chọn tên của nhiều loại. **Low** và **high** là giá trị của giá trị cực tiểu và cực đại của tham số Ví dụ sau:

```
def objective(trial):
    # Categorical parameter
    optimizer = trial.suggest_categorical('optimizer', ['MomentumSGD', 'Adam'])

    # Int parameter
    num_layers = trial.suggest_int('num_layers', 1, 3)

    # Uniform parameter
    dropout_rate = trial.suggest_uniform('dropout_rate', 0.0, 1.0)

    # Loguniform parameter
    learning_rate = trial.suggest_loguniform('learning_rate', 1e-5, 1e-2)
```

# Định nghĩa study object

Để tìm kiếm siêu tham số, bạn cần khởi tạo một đối tượng là **study** Đối tượng này lưu kết quả tối ưu của bạn.

```
Study = Optuna.Creat_study ()
```

```
Study.Optimize ( Objective , N_trials = 100 )
```

Trong đó, tham số thứ 1 là hàm **Objective** , tham số thứ 2 là số lượng thử nghiệm. Quá trình tối ưu này được thực hiện trong đối tượng Study và sẽ thực hiện tìm giá trị cực tiểu của các tham số trong hàm **Objective** bằng phương pháp tối ưu hóa. Khi kết thúc quá trình trên, giá trị tối ưu sẽ được lưu lại và bạn có thể xem nó bằng câu lệnh sau:

```
# Giá trị tối ưu của các tham số khởi tạo trong hàm **Objective**
Study.Best_params

# Kết quả mô hình tương ứng với các tham số tối ưu trên
Study.Best_value

# Xem tất cả trạng thái của các lần thử nghiệm
Study.Trials
```

# Thử nghiệm và so sánh giữa mô hình sử dụng Optuna và chọn cơm siêu tham số trên bộ Iris

## Install Optuna

Có thể install Optuna bằng pip or conda

```
!pip install --quiet optuna
```

Kiểm tra phiên bản

```
import optuna
optuna.__version__
```

## Tối ưu hóa siêu tham số

### Thử nghiệm với mô hình chọn tham số thử công

Mình sẽ thực hiện thử nghiệm mô hình random forest để thực hiện phân loại trên bộ dataset Iris và chọn tham số thử công như sau:

```
import sklearn.datasets
import sklearn.ensemble
import sklearn.model_selection
iris = sklearn.datasets.load_iris() # Prepare the data.
clf = sklearn.ensemble.RandomForestClassifier(
    n_estimators=5, max_depth=3) # Define the model.

sklearn.model_selection.cross_val_score(
    clf, iris.data, iris.target, n_jobs=-1, cv=3).mean()
```

Kết quả mà mô hình trên đạt được là : 0.966 . Thật may là kết quả đạt được khá tốt với n\_estimators = 5 và

# Thử nghiệm sử dụng Optuna để tự động điều chỉnh tham số

Siêu tham số của mô hình trên là `n_estimators` và `max_depth`. Sử dụng đối tượng **trial** để định nghĩa chúng. Sau đó tạo đối tượng **study** để tối ưu hóa siêu tham số này và cuối cùng lấy ra siêu tham số tốt nhất.

```
import optuna

def objective(trial):
    iris = sklearn.datasets.load_iris()

    n_estimators = trial.suggest_int('n_estimators', 2, 20)
    max_depth = int(trial.suggest_float('max_depth', 1, 32, log=True))

    clf = sklearn.ensemble.RandomForestClassifier(
        n_estimators=n_estimators, max_depth=max_depth)

    return sklearn.model_selection.cross_val_score(
        clf, iris.data, iris.target, n_jobs=-1, cv=3).mean()

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)
```

Sau khi xong quá trình tối ưu sau 100 lần thử nghiệm thì chúng ta có được thử nghiệm tốt nhất.

```
trial = study.best_trial

print('Accuracy: {}'.format(trial.value))
print("Best hyperparameters: {}".format(trial.params))
```

Kết quả như sau :

```
Accuracy: 0.9733333333333333
Best hyperparameters: {'n_estimators': 7, 'max_depth': 8.773682352088931}
```

Như vậy, accuracy tăng hơn 1%, và các siêu tham số cũng đã thay đổi và không giống với mình chọn thủ công ở trên.

## Thử nghiệm bộ dữ liệu trên các thuật toán khác nhau

Từ trước đến nay, việc thử nghiệm trên các thuật toán khác nhau để đưa ra được đánh giá khách quan nhất tốn rất nhiều thời gian code và huấn luyện thì nay đã có **Optuna** hỗ trợ cho bạn việc đó. Let's get it!

```
import sklearn.svm

def objective(trial):
    iris = sklearn.datasets.load_iris()

    classifier = trial.suggest_categorical('classifier', ['RandomForest', 'SVC'])

    if classifier == 'RandomForest':
        n_estimators = trial.suggest_int('n_estimators', 2, 20)
        max_depth = int(trial.suggest_float('max_depth', 1, 32, log=True))

        clf = sklearn.ensemble.RandomForestClassifier(
            n_estimators=n_estimators, max_depth=max_depth)
```

Cùng xem kết quả thôi :

```
trial = study.best_trial

print('Accuracy: {}'.format(trial.value))
print("Best hyperparameters: {}".format(trial.params))
```

Accuracy: 0.9866666666666667  
Best hyperparameters: {'classifier': 'SVC', 'svc\_c': 4.448968980739045}

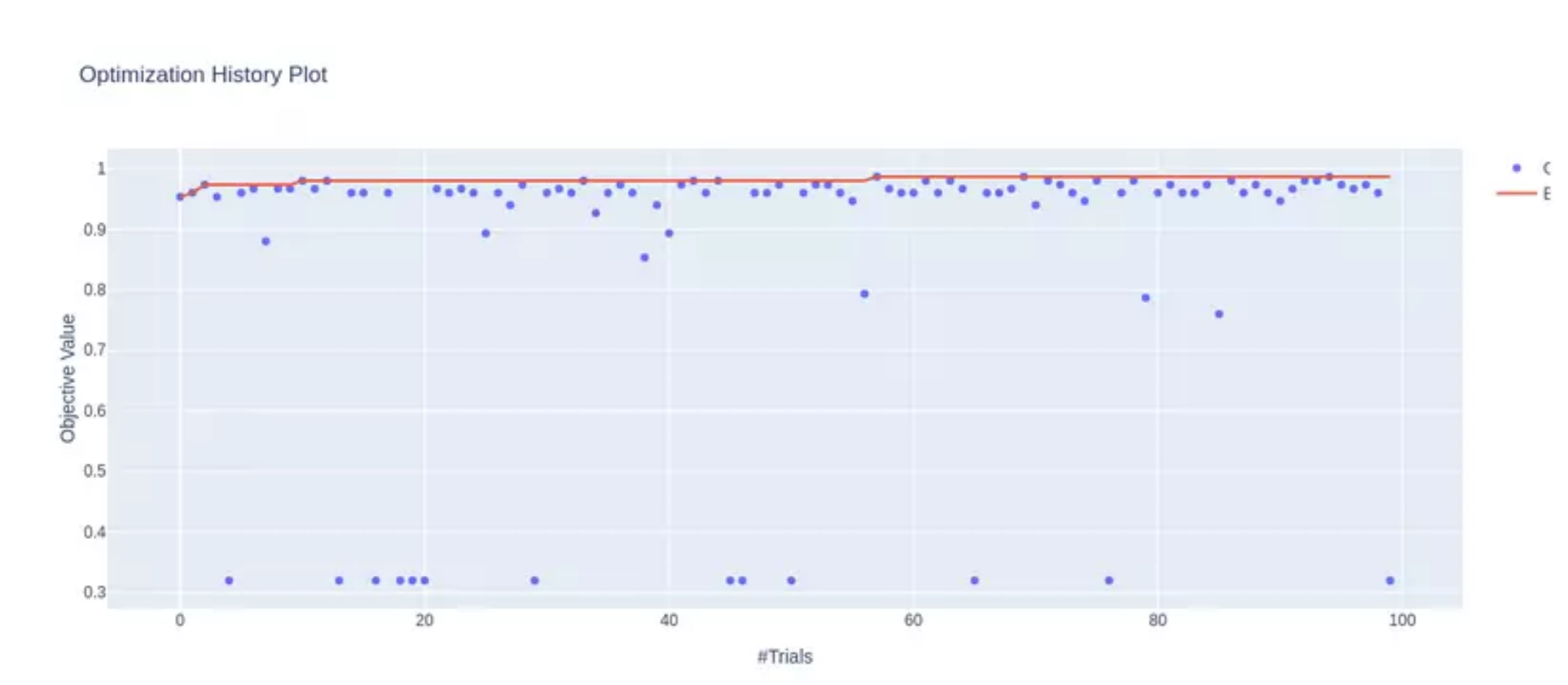
Sau chỉ với mấy dòng code sử dụng cô nàng **Optuna** thì độ chính xác đã đạt 0.987%, và như trên thì mô hình **SVC** đạt kết quả tốt hơn **Random forest**

## Visualization với Optuna

Không chỉ hỗ trợ việc tự động tối ưu các siêu tham số, cô nàng này còn hỗ trợ chúng ta việc visualize ra các trạng thái của các lần thử nghiệm

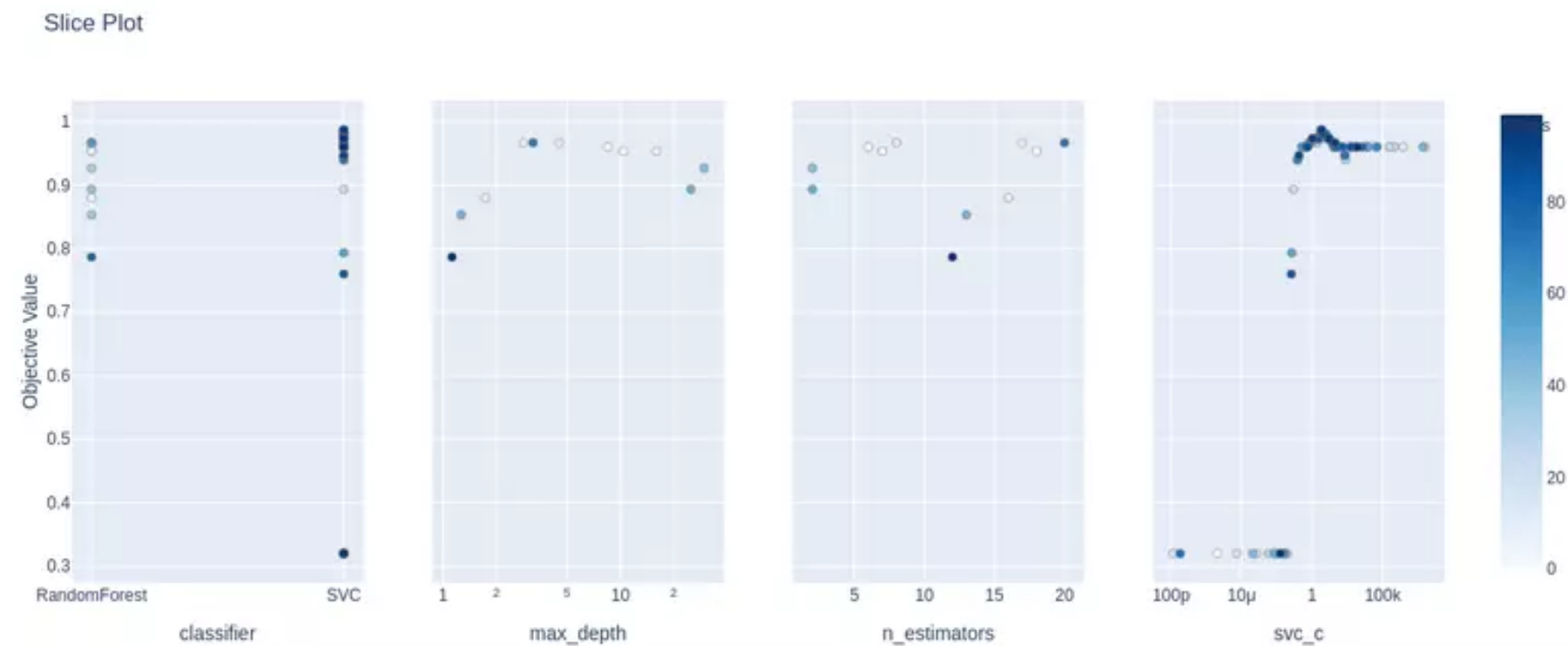
### 1. Visualize lịch sử của đối tượng study

```
optuna.visualization.plot_optimization_history(study)
```



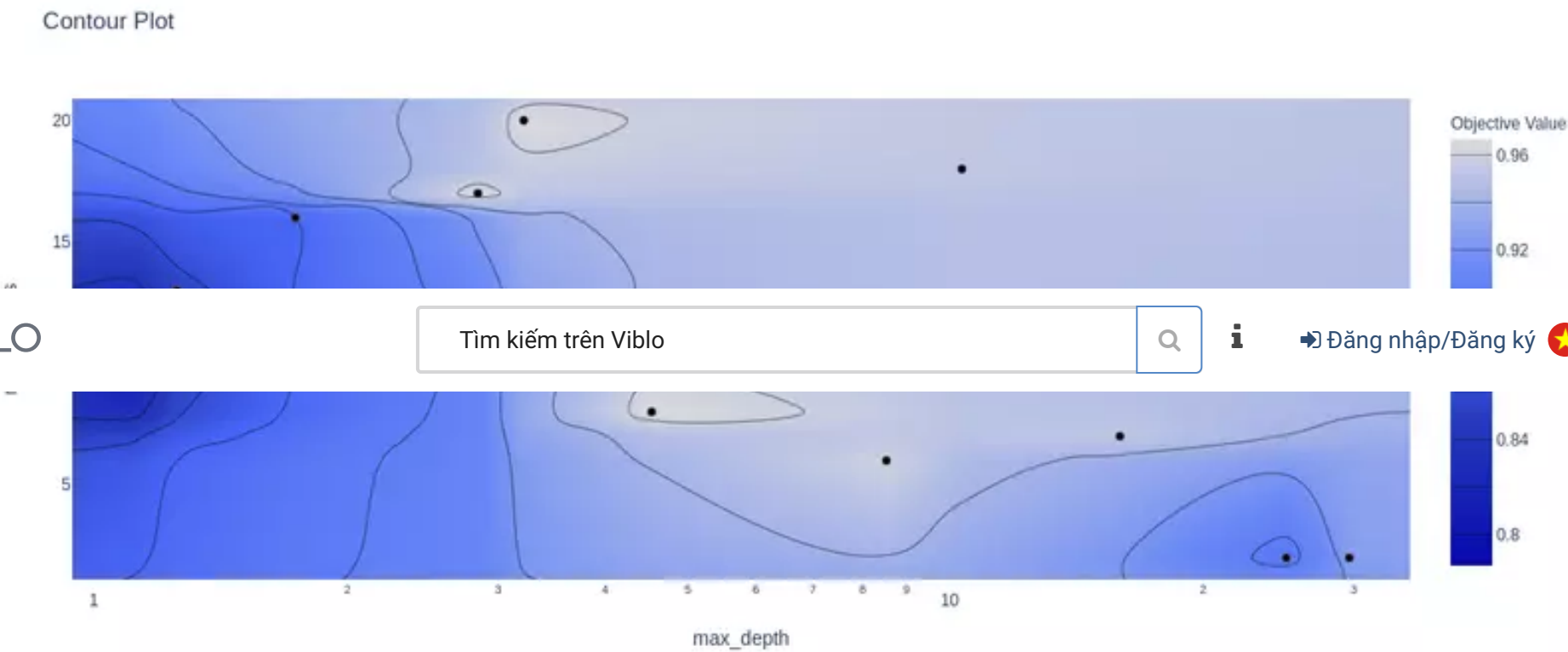
### 2. Visualize độ chính xác của một siêu tham số tại mỗi lần thử nghiệm





3. Visualize accuracy surface cho các siêu tham số của mô hình random forest

```
optuna.visualization.plot_contour(study, params=['n_estimators', 'max_depth'])
```



# Kết luận

Nếu bạn đang bận khoản hay mất thời gian cho việc lựa chọn tham số thì hãy thử dùng Optuna xem sao nhé. Nếu bài viết này được mọi người quan tâm thì mình sẽ cố gắng tìm hiểu và viết thêm bài viết về pruning model sử dụng Optuna. Cảm ơn các bạn đã theo dõi, hãy để lại 1 vote cho mình nhé.

# References

- <https://optuna.org/>
- <https://www.youtube.com/watch?v=P6NwZVI8ttc>

Hyperparameter Tuning Deep Learning PyTorch

Bài viết liên quan

[Shallow Neural Networks](#)

[Nguyễn Tiến Đạt](#)

9 phút đọc

👁 1.0K

🔖 1

💬 0

👍 3

[ChatGPT-4 sẽ có 100 Nghìn tỷ tham số](#)

[Hồ Ngọc Hải](#)

11 phút đọc

👁 314

🔖 0

💬 0

👍 1

[Các loại kiểm thử tự động và những ngộ nhận](#)

[Hồng Nguyễn](#)

9 phút đọc

👁 345

🔖 0

💬 0

👍 0

[Kiểm thử tự động với Selenium\\_P2](#)

[NgoThuyLien](#)

6 phút đọc

👁 488

🔖 0

💬 0

👍 0

[Tớ đã điều chỉnh Nginx để có hiệu suất tốt như thế nào](#)

[Nguyễn Văn Bách](#)

2 phút đọc

👁 2.2K

🔖 21

💬 1

👍 17

[\[Pytorch Tutorial\] - Deploy mô hình PyTorch lên web browser sử dụng ONNX.js](#)

[Phạm Văn Toàn](#)

15 phút đọc

👁 2.2K

🔖 11

💬 0

👍 22

[Kiểm thử tự động với Selenium\\_P3](#)

[NgoThuyLien](#)

3 phút đọc

👁 796

🔖 3

💬 0

👍 0

[Pytorch Fundamentals](#)

[Trần Quang Hiệp](#)

5 phút đọc

👁 424

🔖 3

💬 4

👍 6

Bài viết khác từ Nguyen Viet Hoai

[Hướng dẫn tất tần tật về Pytorch để làm các bài toán...](#)

[Nguyen Viet Hoai](#)

16 phút đọc

👁 15.8K

🔖 15

💬 2

👍 34

[Normalization and Normalization Techniques in Deep Learning](#)

[Nguyen Viet Hoai](#)

18 phút đọc

👁 8.3K

🔖 2

💬 9

👍 18

[Face Detection on Custom Dataset using Detectron2 in Pytorch](#)

[Nguyen Viet Hoai](#)

6 phút đọc

👁 2.7K

🔖 2

💬 0

👍 10

[Scene Text Recognition using Deep Learning](#)

[Nguyen Viet Hoai](#)

7 phút đọc

👁 1.9K

🔖 4

💬 0

👍 0

[XÂY DỰNG MÔ HÌNH 1 PHA PHÁT HIỆN VÀ NHẬN DẠNG VĂN BẢN NHIỀU DÒNG](#)

[Nguyen Viet Hoai](#)

32 phút đọc

👁 700

🔖 1

💬 0

👍 7

[Cơ chế mã hóa vị trí 2 chiều giải quyết bài toán nhận dạng nhiều dòng](#)

[Nguyen Viet Hoai](#)

10 phút đọc

👁 731

🔖 0

💬 2

👍 9

[Hướng dẫn lưu video từ luồng camera sử dụng tkinter và opencv](#)

[Nguyen Viet Hoai](#)

5 phút đọc

👁 1.5K

🔖 0

💬 0

👍 3

[\[Domain adaptation - P1\] Tổng quan về kỹ thuật transfer learning và domain adaptation](#)

[Nguyen Viet Hoai](#)

14 phút đọc

👁 1.7K

🔖 0

💬 1

👍 7

Bình luận

🗨 Đăng nhập để bình luận



**Van Lam Huynh** [@vesinhnhataisaigon](#)

<https://vesinhnhataisaigon.com/dich-vu-ve-sinh-cong-nghiep-quan-8/>

^ 0 v | [Trả lời](#) [Chia sẻ](#) ...

TÀI NGUYÊN

- [Bài viết](#)
- [Câu hỏi](#)
- [Videos](#)
- [Thảo luận](#)
- [Công cụ](#)
- [Trạng thái hệ thống](#)
- [Tổ chức](#)
- [Tags](#)
- [Tác giả](#)
- [Đề xuất hệ thống](#)
- [Machine Learning](#)

DỊCH VỤ

-  [Viblo Code](#)
-  [Viblo CV](#)
-  [Viblo CTF](#)
-  [Viblo Learning](#)
-  [Viblo Interview](#)

ỨNG DỤNG DI ĐỘNG



LIÊN KẾT

