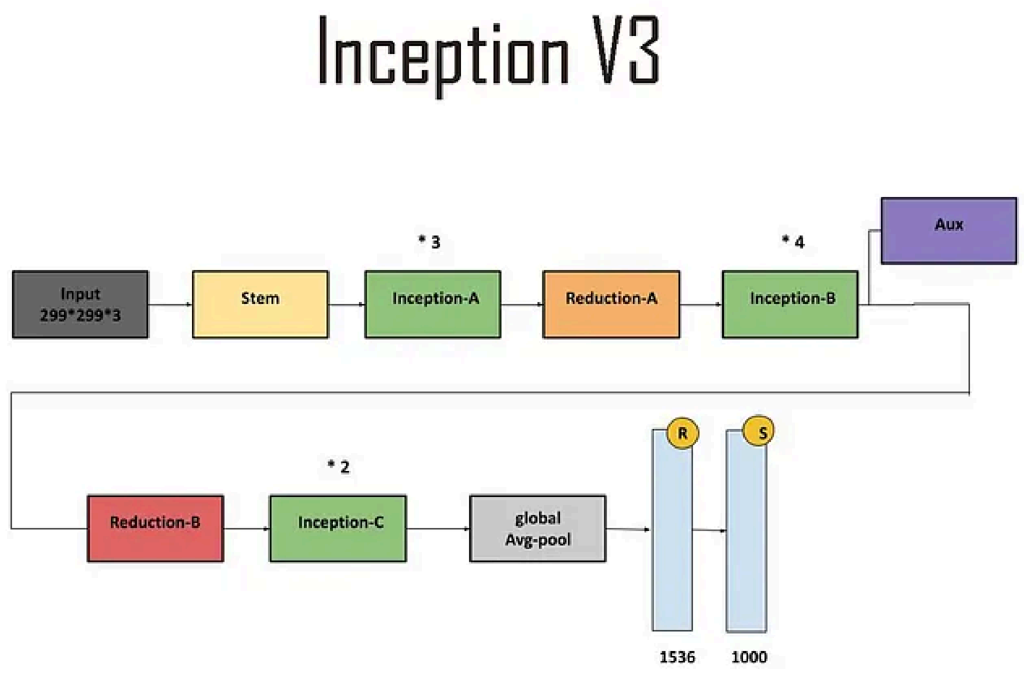
Inception V3 CNN Architecture Explained .

Anas BRITAL · Follow
4 min read · Oct 24, 2021

Listen Share More

Inception-V3 CNN Architecture illustrated and Implemented in both Keras and PyTorch .



In This Article i will try to explain to you Inception V3 Architecture , and we will see together how can we implement it Using **Keras** and **PyTorch** .

Inception V3 :

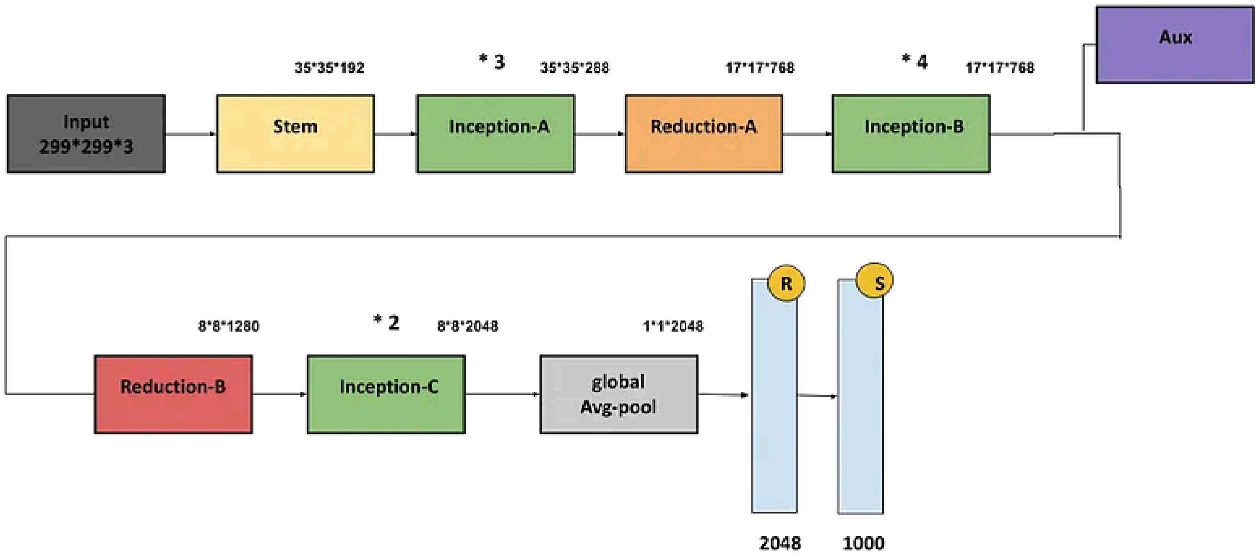
Paper : Rethinking the Inception Architecture for Computer Vision .

Authors : Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi , Google Inc .

Published in : Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence .

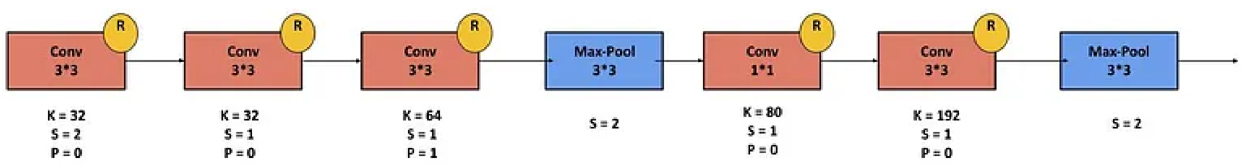
Inception V3 Architecture was published in the same paper as Inception V2 in 2015, and we can consider it as an improvement over the previous Inception Architectures.

The Main Architecture :



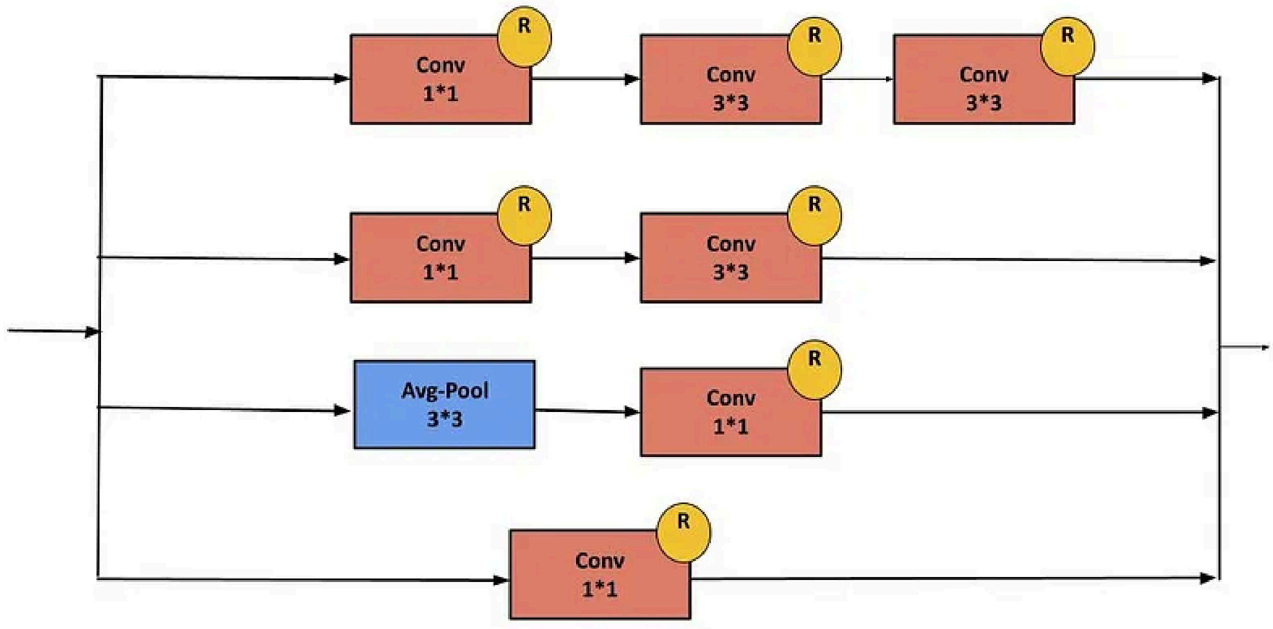
Inception V3

Stem Block :



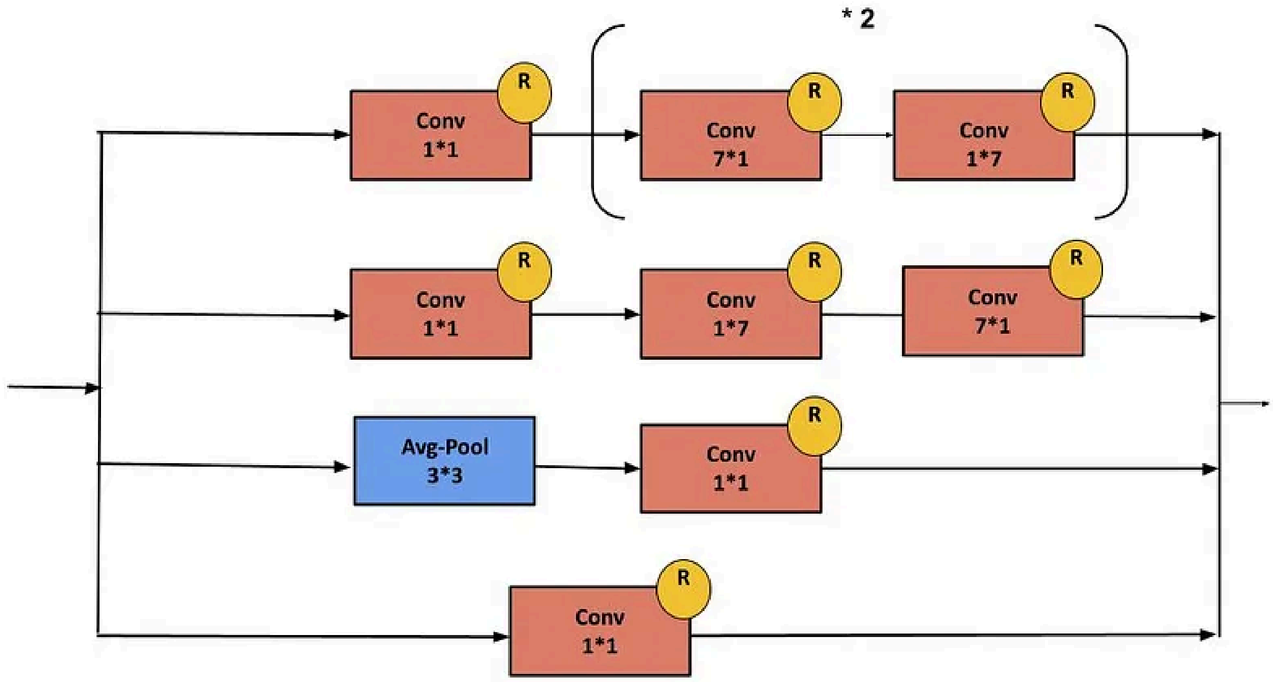
Stem Block

Inception-A Block :



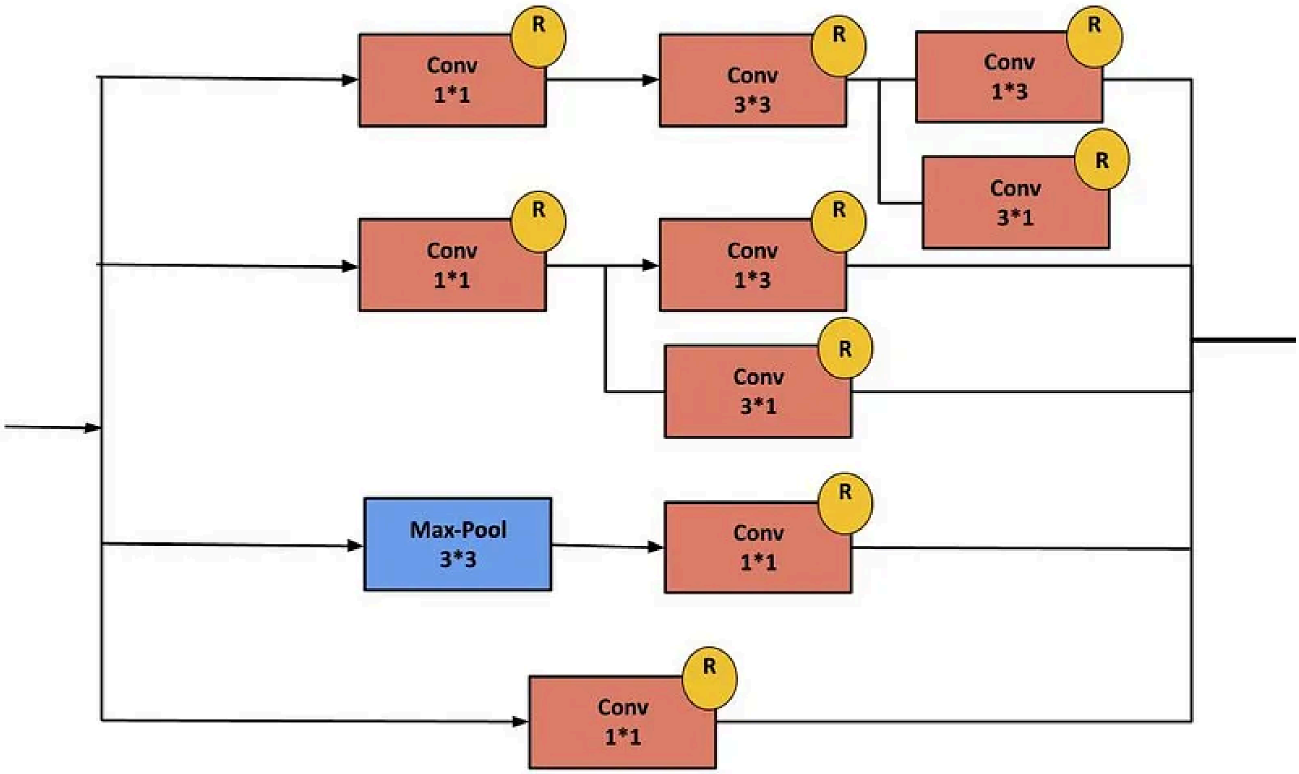
Inception A Block

Inception-B Block :



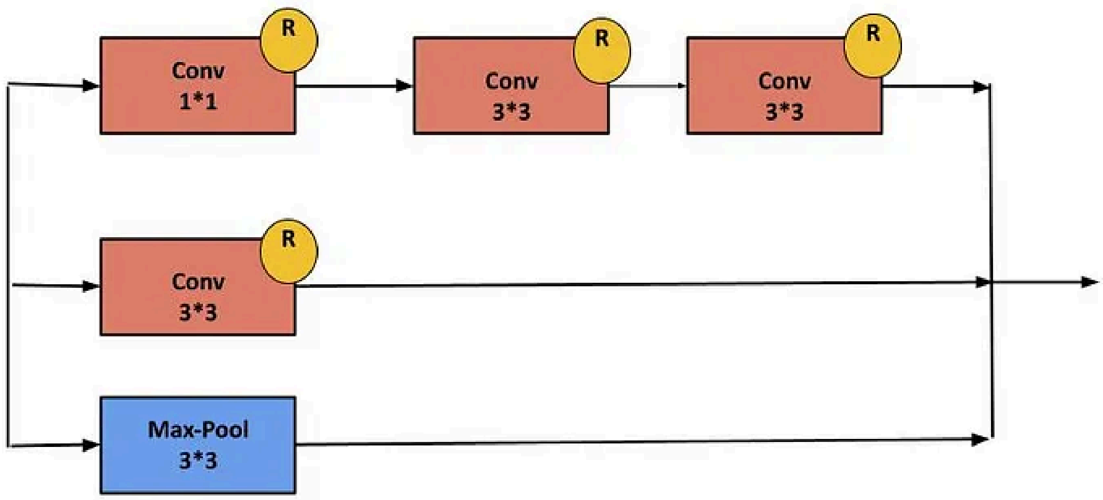
Inception B Block

Inception-C Block :



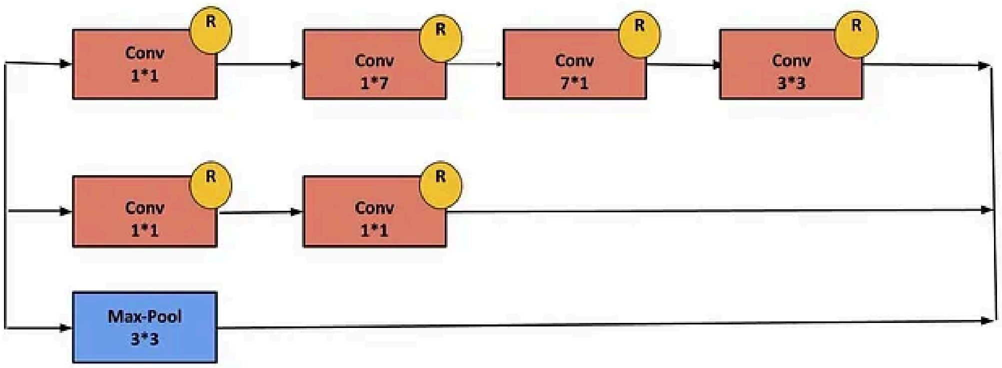
Inception C Block

Reduction-A Block :



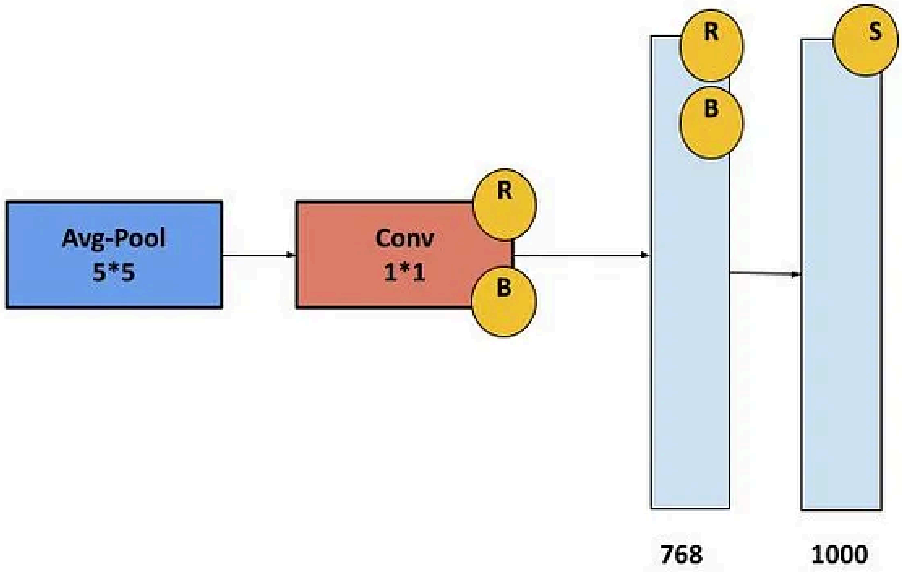
Reduction A Block

Reduction-B Block :



Reduction B Block

Auxiliary Classifier Block :



Aux Classifier Block

Implementation :

1. Inception-V3 Implemented Using Keras :

To Implement This Architecture in Keras we need :

- Convolution Layer in Keras .

```
tf.keras.layers.Conv2D(
    filters, #Number Of Filters
```

```
    kernel_size, # filter of kernel size
    strides=(1, 1),# by default the stride value is 1 .
    padding="valid",#valid means no padding,same means with padding.
    data_format=None,
    dilation_rate=(1, 1),
    groups=1,
    activation=None, # The Activation Function Used .
    use_bias=True, # Using bias or not .
    kernel_initializer="glorot_uniform",#init kernels method .
    bias_initializer="zeros", # init bayes method .
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

- Pooling Layer (Max and Avg) in Keras :

```
# Max Pooling :
```



```
tf.keras.layers.AveragePooling2D(pool_size=(2, 2), strides=None,
padding="valid", data_format=None, **kwargs)
```

we can use The Activation Function embedded with Convolution Layer or Pooling Layer or we can use it separately like this .

```
tf.keras.activations.relu(x, alpha=0.0, max_value=None, threshold=0)
```

- Fully Connected Layer in Keras .

```
tf.keras.layers.Dense(
    units, # The Number of neurons
    activation=None, # The Activation Function Used .
    use_bias=True, # Using Bias or not
    kernel_initializer="glorot_uniform", # init method
    bias_initializer="zeros", # init method .
    kernel_regularizer=None,
    bias_regularizer=None,
```

```
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

- Dropout Layer in Keras.

```
tf.keras.layers.Dropout(rate, noise_shape=None, seed=None, **kwargs)
```

- Concatenation Methods In Keras .

```
#Method 1 :

from tensorflow.keras.layers import Concatenate

layer1 = tf.keras.layers.Dense(...)
layer2 = tf.keras.layers.Dense(...)#By Default The axis is -1
output = Concatenate(axis = 1)[(layer1 , layer2)]

#Method 2 :

from tensorflow.keras.layers import Add

layer1 = tf.keras.layers.Dense(...)
layer2 = tf.keras.layers.Dense(...)
output = Add()([layer1 , layer2])

#Difference Between Add and Concatenate :

x1 = tf.constant([1,2,3])
x2 = tf.constant([1,2,3])

output1 = Add()([x1,x2]) # => Output [2 , 4 , 6]
output2 = Concatenate(axis = 1)([x1,x2]) # => Output [1,2,3,1,2,3]
```

- Methods to Build a Model In Keras .

```
#Method 1 : Using Sequential Model

model = keras.Sequential()
model.add(layers.Dense(2, activation="relu"))
model.add(layers.Dense(3, activation="relu"))
model.add(layers.Dense(4))

#Method 2 : Using Keras functional API
```

```
inputs = keras.Input(shape=(784,))
layer1 = layers.Dense(...)(inputs)
layer2 = layers.Dense(...)(layer1)
.
.
.
layer_n = layers.Dense(...)(layer[n-1])

model = keras.Model(inputs=inputs, outputs=layer_n, name="MyModel")
```

Now i think we've everything we need , give it a try and see if we have the same result .

2. Inception-V3 Implemented Using PyTorch :

To Implement This Architecture In PyTorch we need :

- Convolution Layer In PyTorch :

```
torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1,
padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros',
device=None, dtype=None)
```

- Activation Layer :

```
torch.nn.ReLU(inplace=False)
```

- Pooling Layer :

```
# Max Pooling :

torch.nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1,
return_indices=False, ceil_mode=False)

# Avg Pooling :

torch.nn.AvgPool2d(kernel_size, stride=None, padding=0,
ceil_mode=False, count_include_pad=True, divisor_override=None)
```

- DropOut Layer :

```
torch.nn.Dropout(p=0.5, inplace=False)
```

- Fully Connected Layer :

```
torch.nn.Linear(in_features, out_features, bias=True, device=None,
dtype=None)
```

- Concatenation In PyTorch :

```
layer1 = torch.nn.Linear(...)
layer2 = torch.nn.Linear(...)
```

```
output = torch.cat([layer1 , layer2] , axis = 1)
```

- Building a Model Using PyTorch :

```
# Your Model should have a structure like this
class NeuralNetwork(nn.Module):

    def __init__(self):
        super(NeuralNetwork, self).__init__()

        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10))

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)

        return logits
```

Now i think we've everything we need , give it a try and see if we have the same result .

References :

- If you want to see other architectures implemented in both PyTorch and Keras you can check this repo and you can also find high quality images (svg format) of the illustrations above architectures .
- This article was inspired by Illustrations: 10 CNN Architectures Article Written by Remy Karim, I saw his work and really liked what he did, then i decided to make some illustrations with more details, and implement those architectures using both Keras and PyTorch .
- PyTorch documentation .
- Keras documentation .

Inception V3 | Cnn | Keras | Pytorch | Computer Vision

Follow

Written by Anas BRITAL

69 Followers · 0 Following

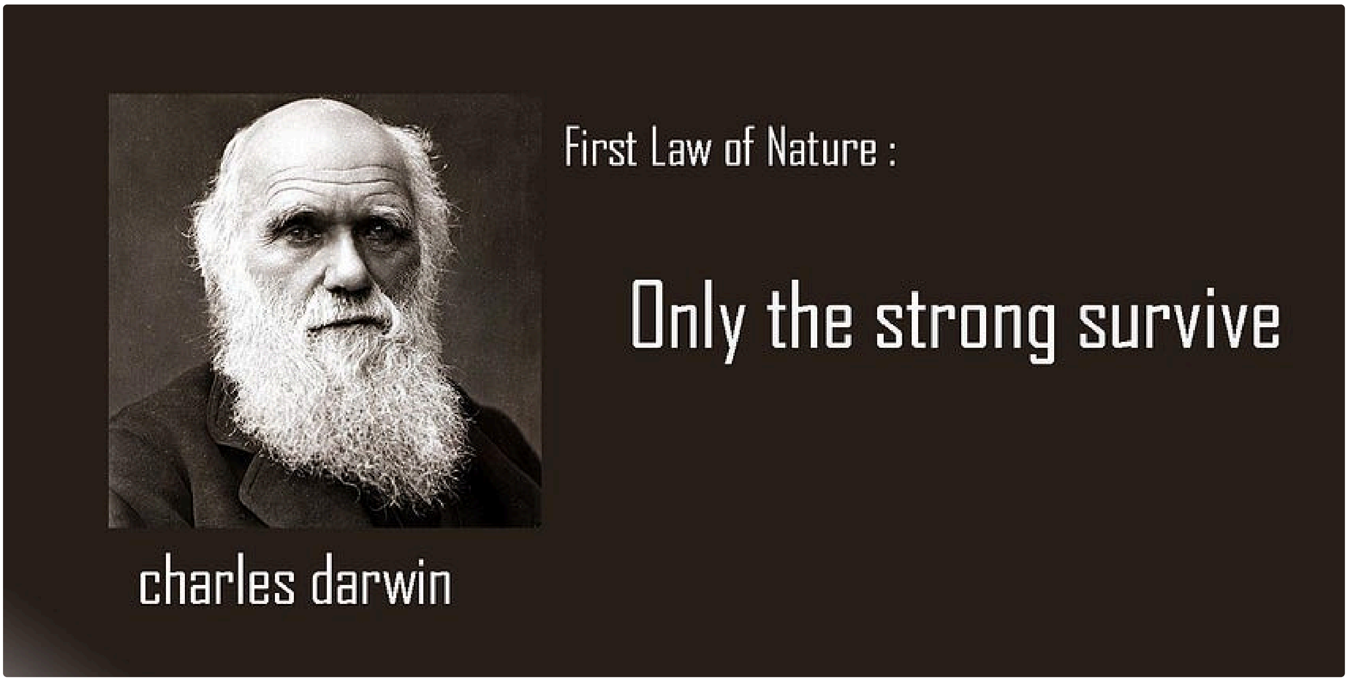
AI and Math Enthusiast (Personal Blog : anasbrital98.github.io/) .

No responses yet

What are your thoughts?

Respond

More from Anas BRITAL

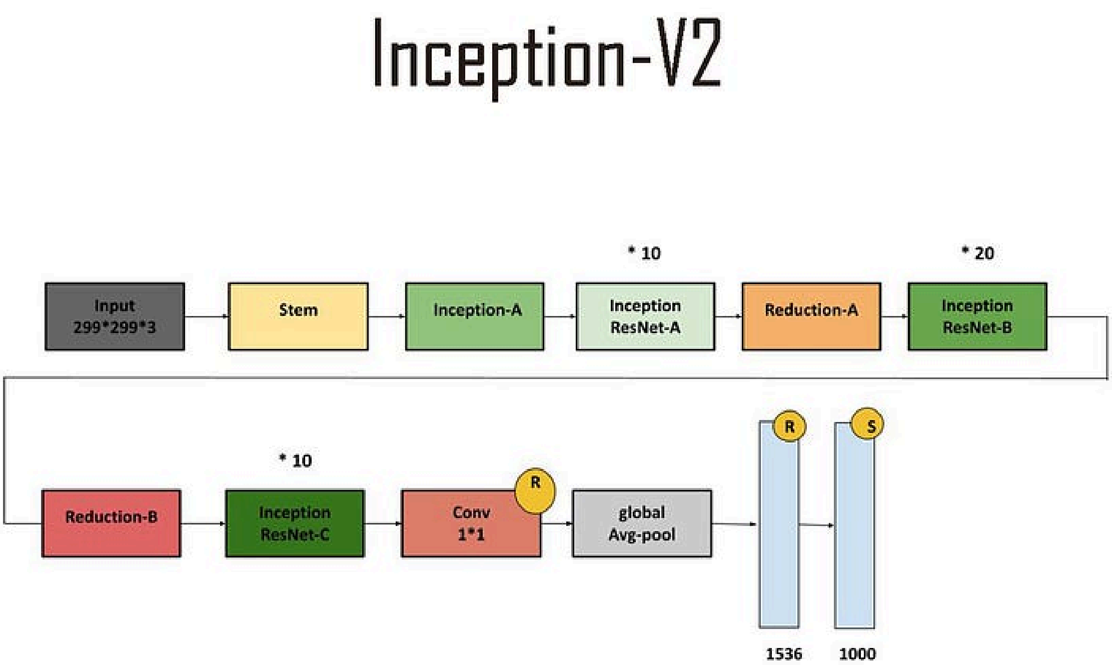


Anas BRITAL

Genetic Algorithm Explained :

Everything you need to know About Genetic Algorithm .

Oct 16, 2021 116 2

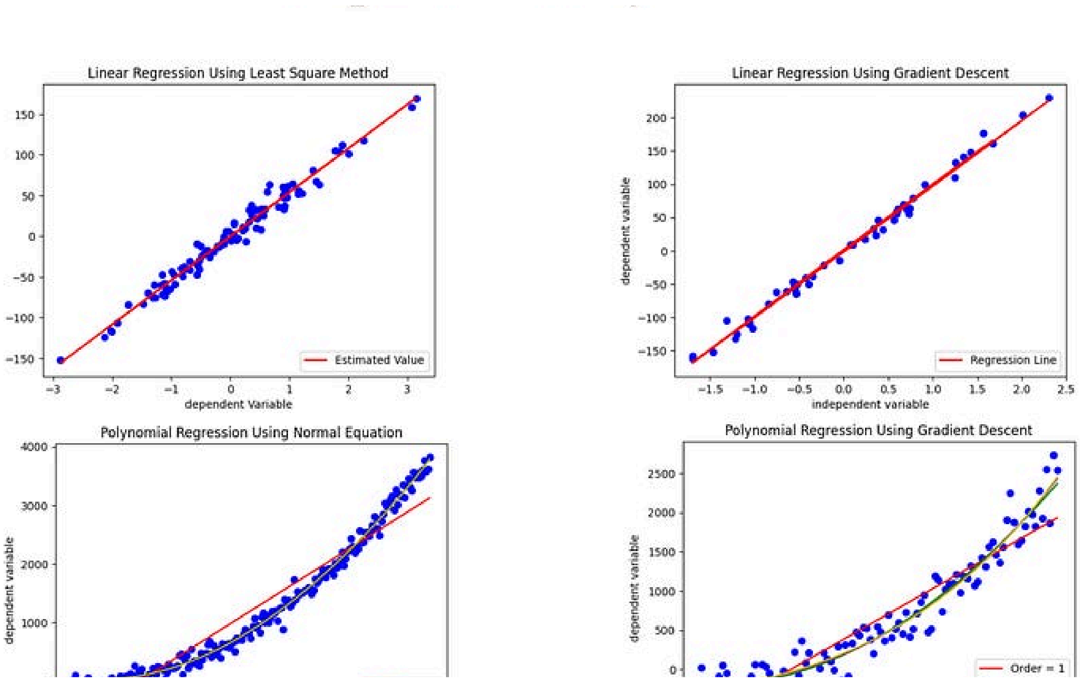


Anas BRITAL

Inception V2 CNN Architecture Explained .

Inception-V2 CNN Architecture illustrated and Implemented in both Keras and PyTorch .

Oct 24, 2021 5

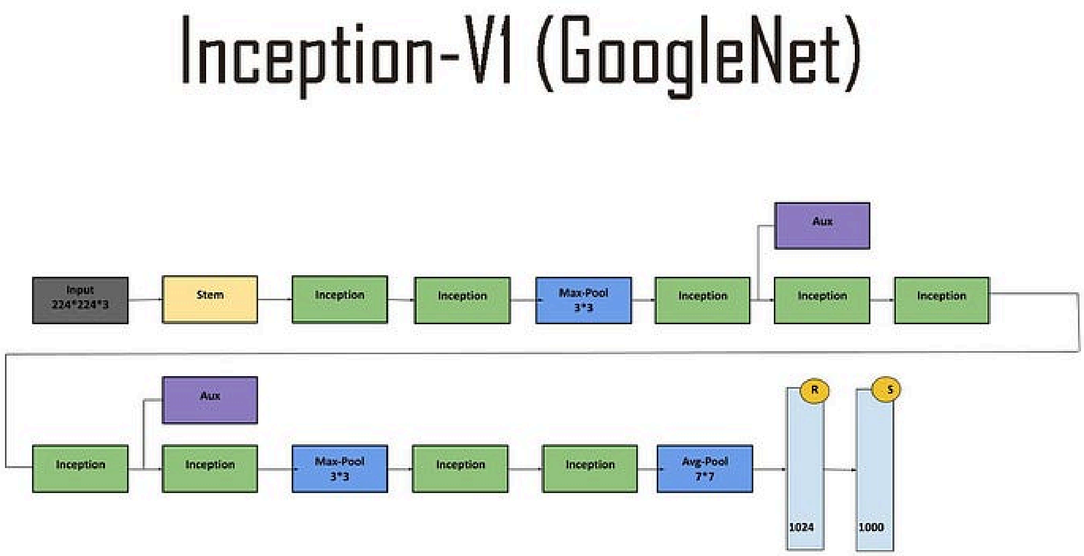


Anas BRITAL

Regression Analysis

Regression analysis models Explained and Implemented Using Python

Nov 7, 2021 55



Anas BRITAL

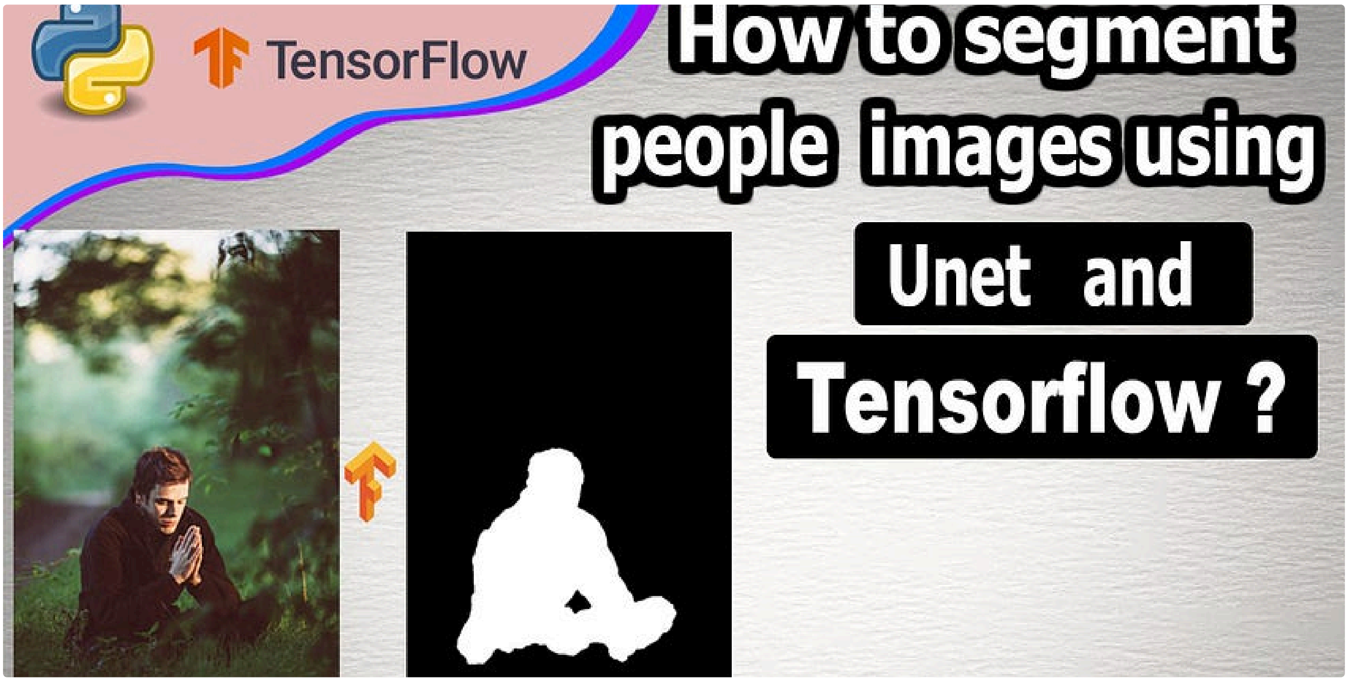
GoogLeNet CNN Architecture Explained (Inception V1).

GoogleNet Architecture illustrated and Implemented in both Keras and PyTorch .

Oct 23, 2021 40

See all from Anas BRITAL

Recommended from Medium

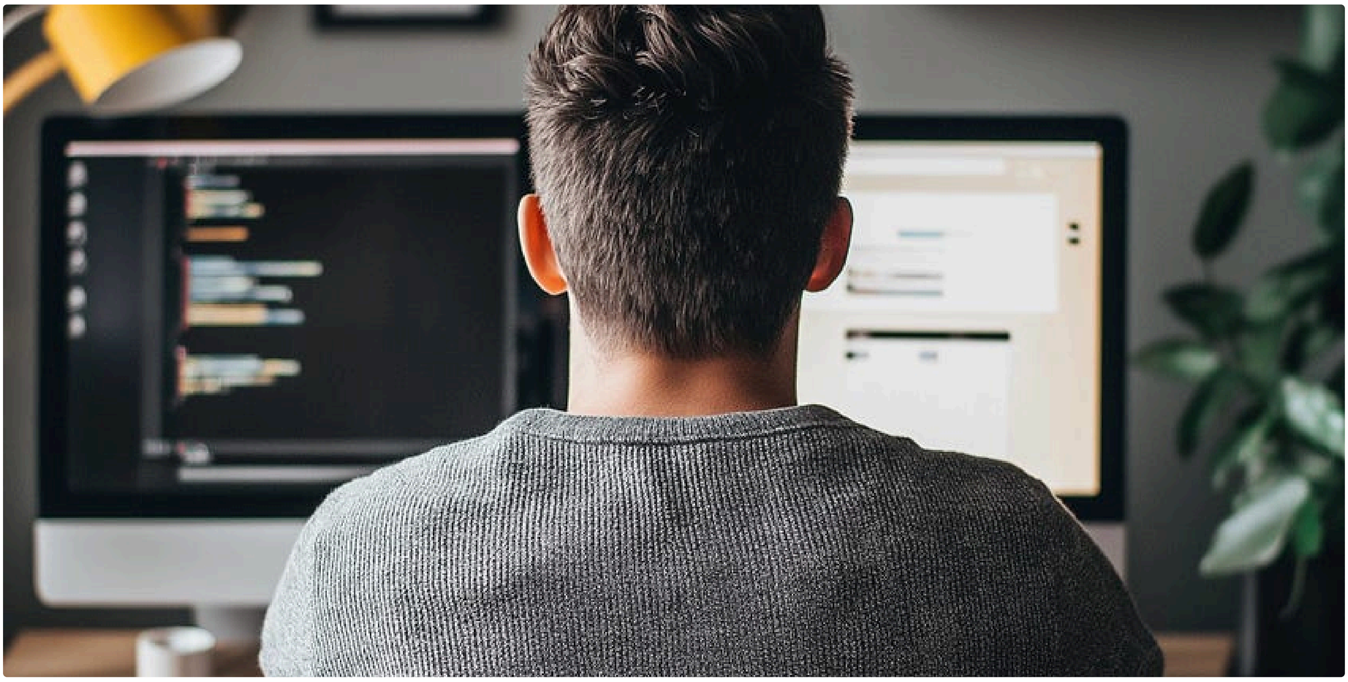


Eran Feit

U-net Image Segmentation - How to segment persons in images 👤

Summary :

Jan 3 10



YasinShafiei

Residual Networks (ResNets) with implementation from scratch

Having a deep understanding about different concepts of deep learning is a crucial. In This article I'll fully

Aug 15, 2024 66

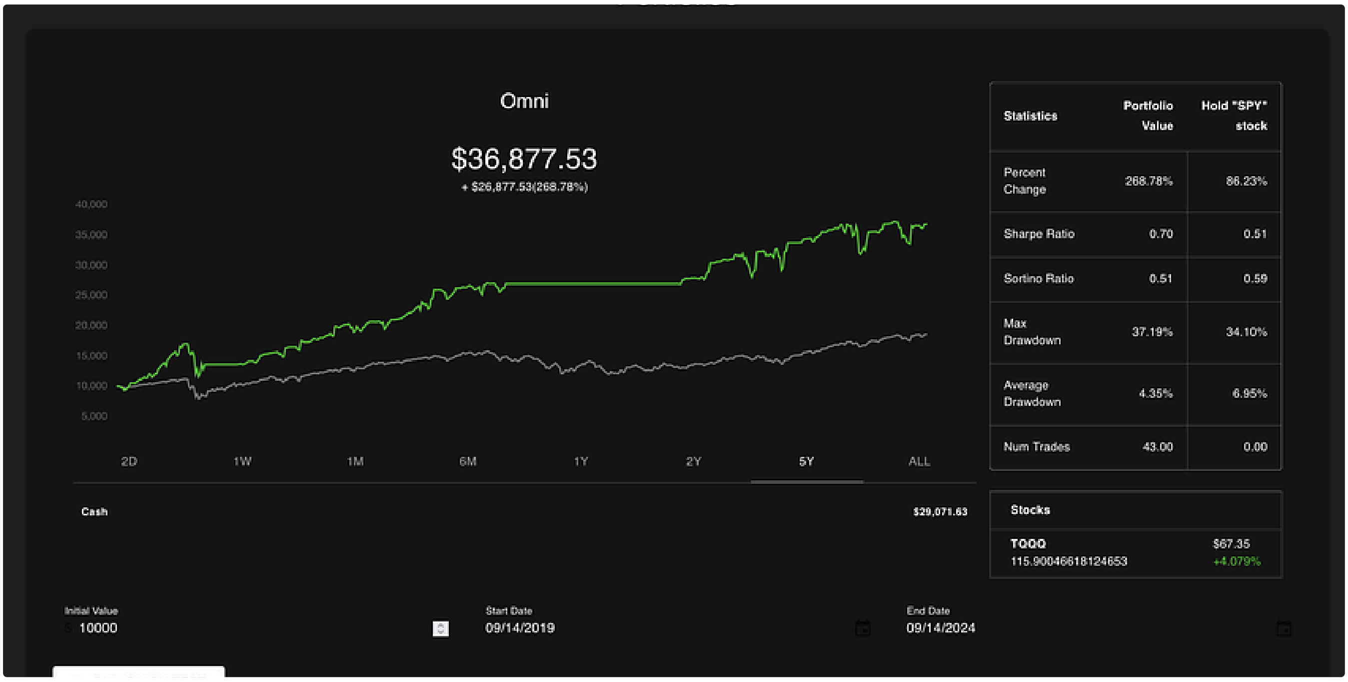
Lists

Practical Guides to Machine Learning

10 stories · 2142 saves

Natural Language Processing

1887 stories · 1536 saves

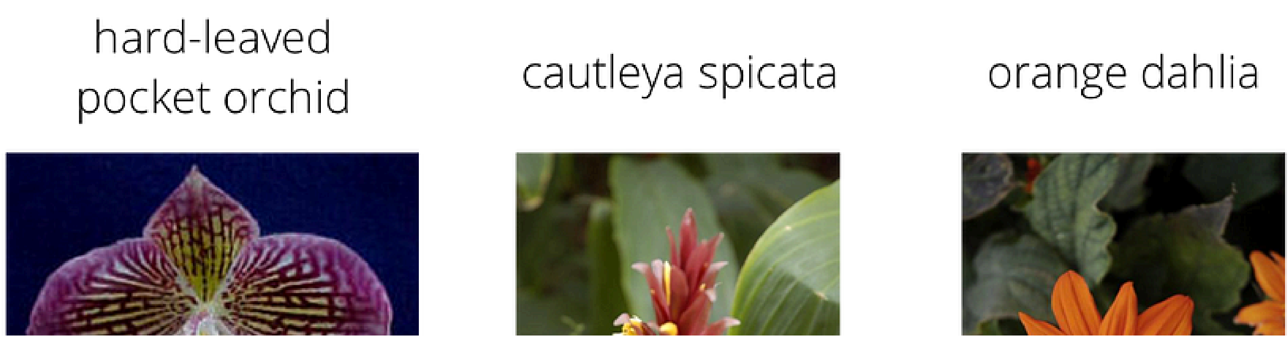


In DataDrivenInvestor by Austin Starks

I used OpenAI's o1 model to develop a trading strategy. It is DESTROYING the market

It literally took one try. I was shocked.

Sep 16, 2024 8.3K 207

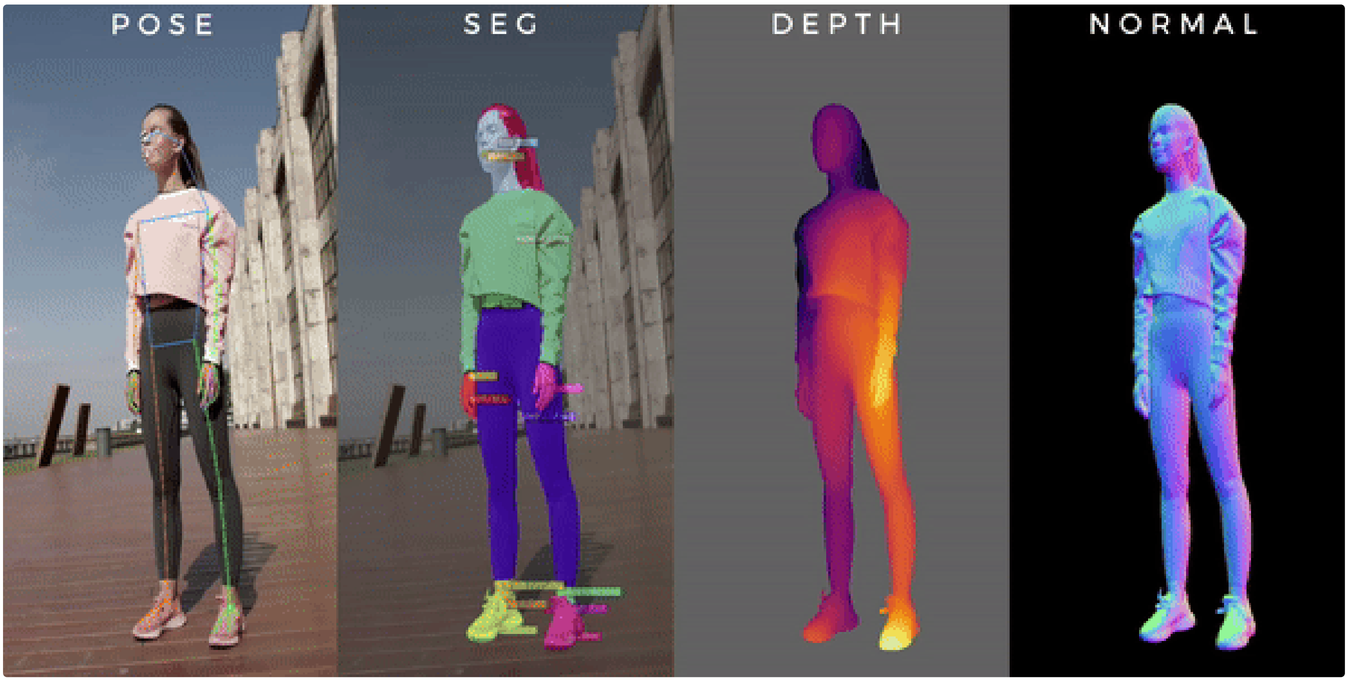


Hadeel Bkhaitan

Mastering Image Classification: Creating a Flower Classifier with MobileNetV2 and TensorFlow

1. Introduction

Oct 6, 2024 47 2



AI Papers Academy

Sapiens by Meta AI: Foundation for Human Vision Models

In this post we dive into Sapiens, a new family of computer vision models by Meta AI that show remarkable advancement in human-centric...

Aug 26, 2024 130

In RareSkills by RareSkills

The interest rate model of AAVE V3 and Compound V2

Interest rates in TradFi (traditional finance) are largely determined by central banks and influenced by market factors. In contrast, DeFi...

Jul 27, 2024 17

See more recommendations