




**StepUp Analytics**
VoiceLiveChat

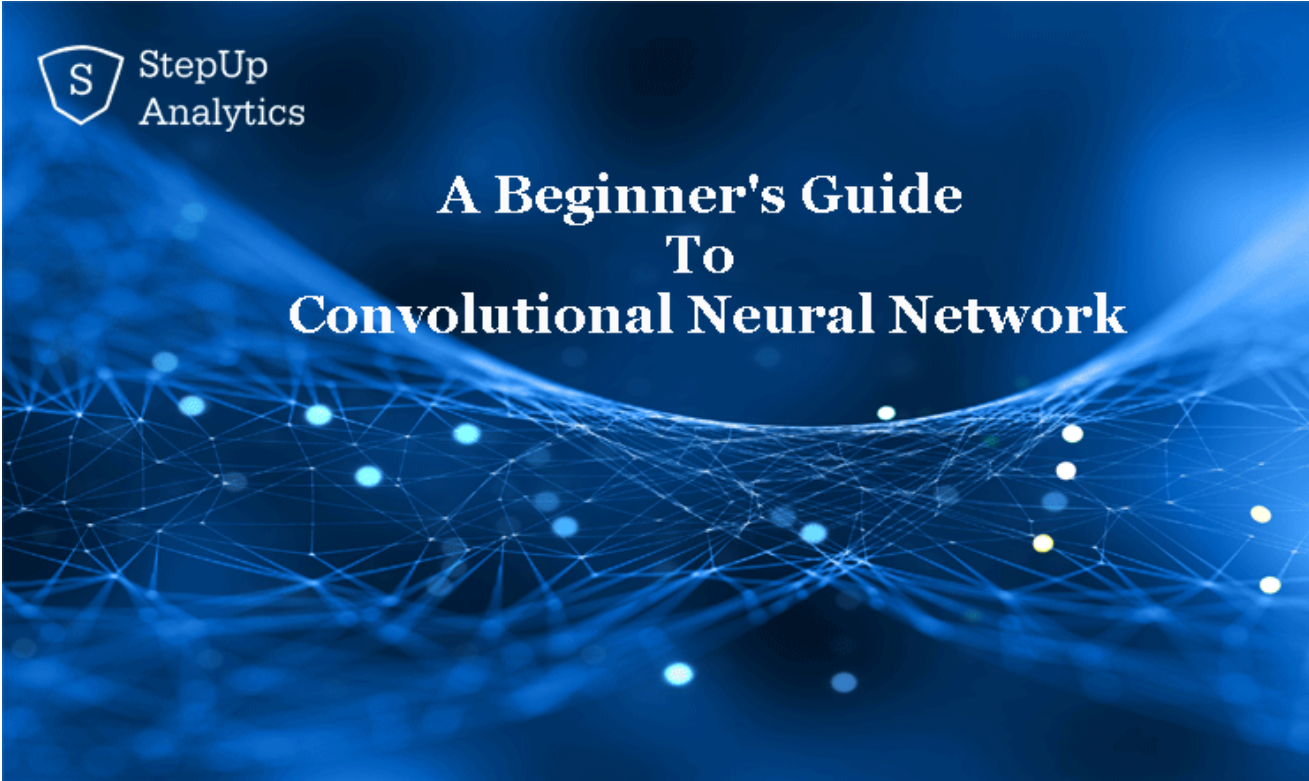
RECOMMENDER SYSTEM
Dr. Sarabjot Singh Anand
CO-FOUNDER, TATRAS DATA, INDIA
29th September '2018 at 5 PM (IST)








**TATRAS**
DATA | ALGORITHMS | INSIGHT



**StepUp Analytics**

A Beginner's Guide To Convolutional Neural Network



 Info@stepupanalytics.com  9902046528  www.stepupanalytics.com    

POPULAR CATEGORIES

Actuarial Science Discussion	45
Machine learning	44
Actuarial Science	42
Statistics	33
R Programming	25
Analytics	23

RECENT POSTS

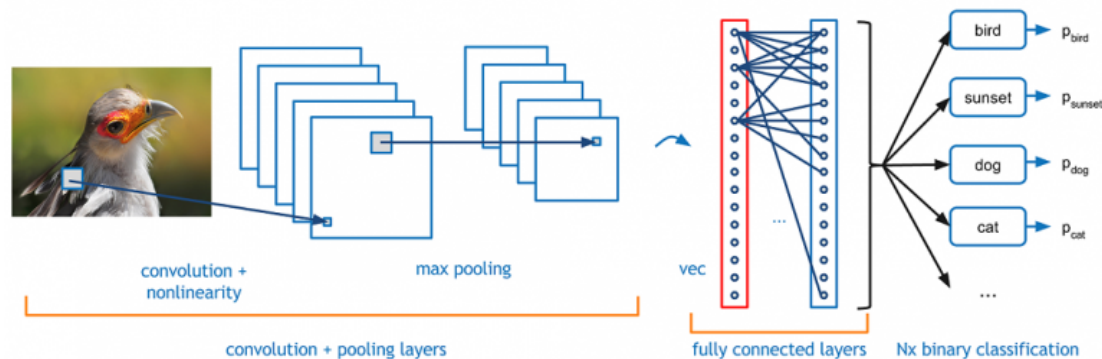


A Beginner's Guide To Convolutional Neural Network

ANKIT CHAKRABORTY • Sep 15, 2018 0



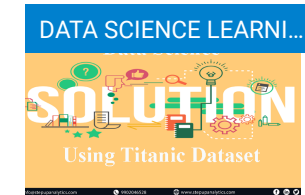
Convolutional neural networks or CNN. Sounds like a weird combination of biology and math with a little CS sprinkled in, but these networks have been some of the most influential innovations in the field of computer vision. 2012 was the first year that neural nets grew to prominence as Alex Krizhevsky used them to win that year's ImageNet competition (basically, the annual Olympics of computer vision), dropping the classification error record from 26% to 15%, an astounding improvement at the time.



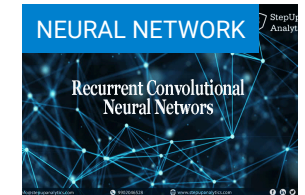
Ever since then, a host of companies have been using deep learning at the core of their services. Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.

Introduction To
Keras: Tutorial and
Implementation

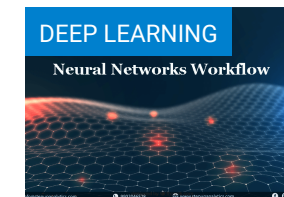
Data Talk With Dr.
Sarabjot Singh Anand



Data Science
Solution Using Titanic
Dataset



Recurrent
Convolutional Neural
Network: RCNN



Neural Networks
Workflow



Introduction To
Python For Data
Visualization With
Seaborn

POPULAR TOPICS

Actuarial Career

Actuarial Cheat Sheet

Actuarial Exam

Actuarial Science

Actuarial Science Career

However, the classic, and arguably most popular, use case of these networks is for image processing. Within image processing, let's take a look at how to use these CNNs for image classification.

The Problem Space

Image classification is the task of taking an input image and outputting a class (a cat, dog, etc) or a probability of classes that best describes the image. For humans, this task of recognition is one of the first skills we learn from the moment we are born and is one that comes naturally and effortlessly as adults. Without even thinking twice, we're able to quickly and seamlessly identify the environment we are in as well as the objects that surround us.

When we see an image or just when we look at the world around us, most of the time we are able to immediately characterize the scene and give each object a label, all without even consciously noticing. These skills of being able to quickly recognize patterns, generalize from prior knowledge, and adapt to different image environments are ones that we do not share with our fellow machines.

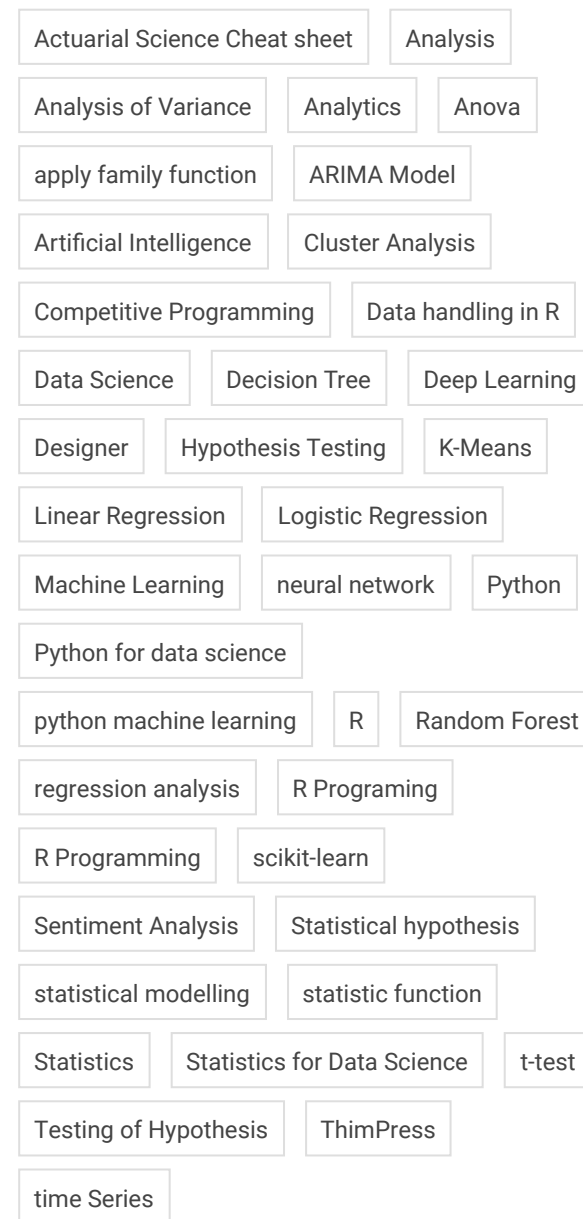


What We See

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 49 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 49 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 42 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 24 97 17 78 78 96 83 14 88 34 89 43 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 47 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 40 21 58 51 54 17 58
19 80 81 68 05 94 47 49 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 48 87 57 42 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 42 76 36
20 49 36 41 72 30 23 88 34 62 99 49 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 49 16 92 33 48 61 43 52 01 89 19 47 48
```

What Computers See

Inputs and Outputs



When a computer sees an image (takes an image as input), it will see an array of pixel values. Depending on the resolution and size of the image, it will see a 32 x 32 x 3 array of numbers (The 3 refers to RGB values). Just to drive home the point, let's say we have a color image in JPG form and its size is 480 x 480. The representative array will be 480 x 480 x 3.

Each of these numbers is given a value from 0 to 255 which describes the pixel intensity at that point. These numbers, while meaningless to us when we perform image classification, are the only inputs available to the computer. The idea is that you give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class (.80 for a cat, .15 for a dog, .05 for a bird, etc).

What We Want The Computer To Do

Now that we know the problem as well as the inputs and outputs, let's think about how to approach this. What we want the computer to do is to be able to differentiate between all the images it's given and figure out the unique features that make a dog a dog or that make a cat a cat. This is the process that goes on in our minds subconsciously as well.

When we look at a picture of a dog, we can classify it as such if the picture has identifiable features such as paws or 4 legs. In a similar way, the computer is able to perform image classification by looking for low-level features such as edges and curves and then building up to more abstract concepts through a series of convolutional layers. This is a general overview of what a CNN does. Let's get into the specifics.

Biological Connection

But first, a little background. When you first heard of the term convolutional neural networks, you may have thought of something related to neuroscience or biology, and you would be right. Sort of.

CNN's do take a biological inspiration from the visual cortex. The visual cortex has small regions of cells that are sensitive to specific regions of the visual field.

Structure

Back to the specifics. A more detailed overview of what CNNs do would be that you take the image, pass it through a series of convolutional, nonlinear, pooling (downsampling), and fully connected layers, and get an output. As we said earlier, the output can be a single class or a probability of classes that best describes the image. Now, the hard part is understanding what each of these layers do. So let's get into the most important one.

First Layer – Math Part

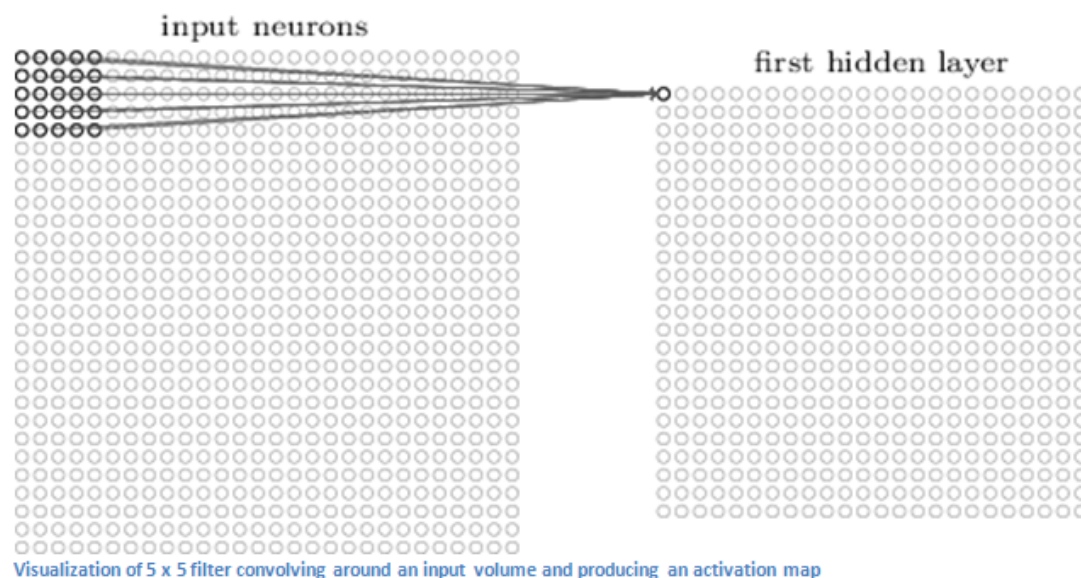
The first layer in a CNN is always a **Convolutional Layer**. First thing to make sure you remember is what the input to this conv (I'll be using that abbreviation a lot) layer is. Like we mentioned before, the input is a $32 \times 32 \times 3$ array of pixel values. Now, the best way to explain a conv layer is to imagine a flashlight that is shining over the top left of the image.

Let's say that the light this flashlight shines covers a 5×5 area. And now, let's imagine this flashlight sliding across all the areas of the input image. In machine learning terms, this flashlight is called a **filter** (or sometimes referred to as a **neuron** or a **kernel**) and the region that it is shining over is called the **receptive field**. Now this filter is also an array of numbers (the numbers are called **weights** or **parameters**). A very important note is that the depth of this filter has to be the same as the depth of the input (this makes sure that the math works out), so the dimensions of this filter is $5 \times 5 \times 3$.

Now, let's take the first position the filter is in for example. It would be the top left corner. As the filter is sliding, or **convolving**, around the input image, it is multiplying the values in the filter with the original pixel values of the image (aka computing **element wise multiplications**). These

multiplications are all summed up (mathematically speaking, this would be 75 multiplications in total). So now you have a single number.

Remember, this number is just representative of when the filter is at the top left of the image. Now, we repeat this process for every location on the input volume. (Next step would be moving the filter to the right by 1 unit, then right again by 1, and so on). Every unique location on the input volume produces a number. After sliding the filter over all the locations, you will find out that what you're left with is a $28 \times 28 \times 1$ array of numbers, which we call an **activation map** or **feature map**. The reason you get a 28×28 array is that there are 784 different locations that a 5×5 filter can fit on a 32×32 input image. These 784 numbers are mapped to a 28×28 array.



Let's say now we use two $5 \times 5 \times 3$ filters instead of one. Then our output volume would be $28 \times 28 \times 2$. By using more filters, we are able to preserve the spatial dimensions better. Mathematically, this is what's going on in a convolutional layer.

First Layer – High Level Perspective

However, let's talk about what this convolution is actually doing from a high level. Each of these filters can be thought of as **feature identifiers**. When I say features, I'm talking about things like straight edges, simple colors, and curves. Think about the simplest characteristics that all images have in common with each other.

Let's say our first filter is $7 \times 7 \times 3$ and is going to be a curve detector. (In this section, let's ignore the fact that the filter is 3 units deep and only consider the top depth slice of the filter and the image, for simplicity.) As a curve detector, the filter will have a pixel structure in which there will be higher numerical values along the area that is a shape of a curve (Remember, these filters that we're talking about as just numbers!).

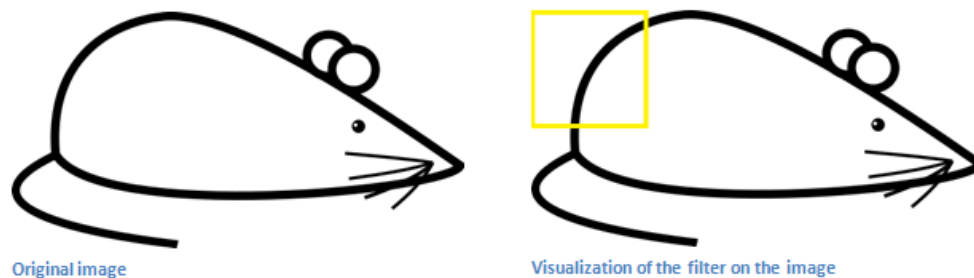
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Now, let's go back to visualizing this mathematically. When we have this filter at the top left corner of the input volume, it is computing multiplications between the filter and pixel values at that region. Now let's take an example of an image that we want to classify, and let's put our filter at the top left corner.



Remember, what we have to do is multiply the values in the filter with the original pixel values of the image.



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

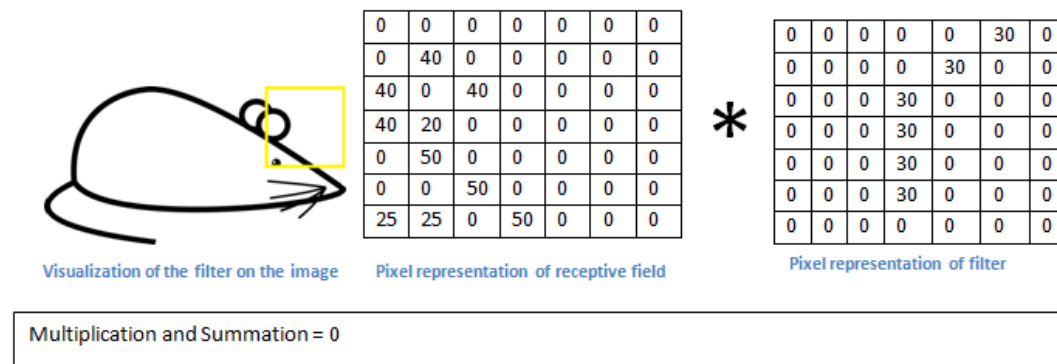
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = $(50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600$ (A large number!)

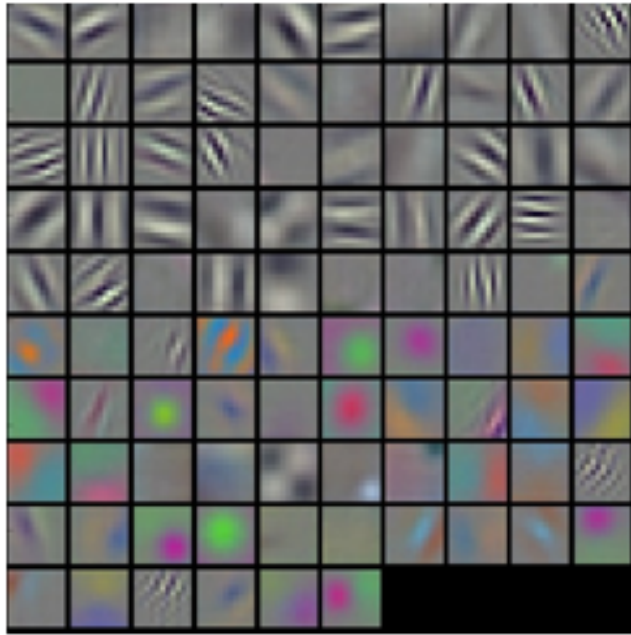
Basically, in the input image, if there is a shape that generally resembles the curve that this filter is representing, then all of the multiplications summed together will result in a large value! Now let's see what happens when we move our filter.



The value is much lower! This is because there wasn't anything in the image section that responded to the curve detector filter. Remember, the output of this conv layer is an activation map. So, in the simple case of a one filter convolution (and if that filter is a curve detector), the activation map will show the areas in which there are mostly likely to be curves in the picture.

In this example, the top left value of our 26 x 26 x 1 activation map (26 because of the 7x7 filter instead of 5x5) will be 6600. This high value means that it is likely that there is some sort of curve in the input volume that caused the filter to activate. The top right value in our activation map will be 0 because there wasn't anything in the input volume that caused the filter to activate (or more simply said, there wasn't a curve in that region of the original image).

Remember, this is just for one filter. This is just a filter that is going to detect lines that curve outward and to the right. We can have other filters for lines that curve to the left or for straight edges. The more filters, the greater the depth of the activation map, and the more information we have about the input volume.



Visualizations of filters



CNN

Convolutional neural networks

Deep Learning

AUTHOR

All Posts

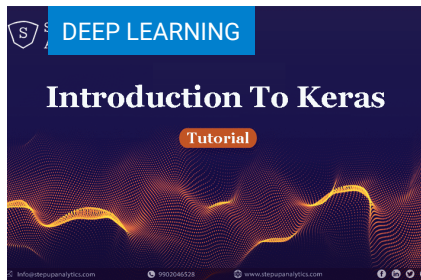


ANKIT CHAKRABORTY

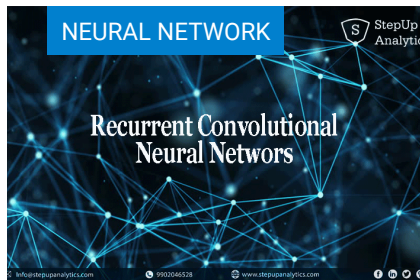
I am a passionate young engineering undergrad with a passion towards technology. I have been drawn towards Data Science and Machine Learning from my Final year days. My interest for Mathematics and Statistics sparked the interest for ML and DS. I strive for a wholesome personality based on the ideal of Swami Vivekananda.

YOU MIGHT ALSO LIKE

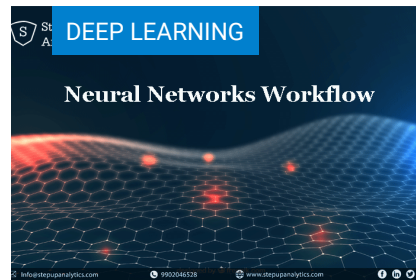
More From Author



Introduction To Keras: Tutorial And Implementation



Recurrent Convolutional Neural Network: RCNN



Neural Networks Workflow

◀ PREV

NEXT ▶

Sign in ▼

Newest ▼

Share



Hey, start typing...



[Terms](#)

[Privacy Policy](#)

metype



Twitter

Join us on Twitter



Google+

Join us on Google



Github

Join us on Github



Pinterest

Join us on Pinterest

WHATSAPP SHARE

[Write For Us](#) [About Us](#) [Privacy Policy](#) [Refund Policy](#) [FAQs](#) [Contact Us](#)

© 2018 - StepUp Analytics All Rights Reserved.

Website Design: Data Matters Knowledge Services Pvt Ltd