Deep Inceptionism learning performance analysis using TensorFlow with GPU – Deep Dream Algorithm

T. Tritva Jyothi Kiran
PhD Scholar, A.P.J Abdul Kalam University, Indore, India.

ABSTRACT: *Deep Inceptionism learning is a computer vision algorithm also called Deep Dream algorithm which is the freakiest A.I. algorithm to date. When you feed in an image to a trained ANN, the neurons fire and generate activations. The deep dream algorithm work by trying to change the input image in a way that would make some of these neuron's fire more (boost the neurons firing or activations). You can select which neurons in which layer you are interested in making them fire more prominently. The process is continuously repeated until the input image now contains all features that a specific layer was originally looking for and these kinds of trippy effects it gives you a kind of a sense of how close a eye is to the actual biological neurons that our human level intelligence. We can see weird features in the actual image. I have tested the Deep Inceptionism learning vision performance on GPU of Intel® Core™ i3-7100U CPU and I got more lot smoothen image with a kind of trippy effect and the algorithm works by creating dream Like Effects.*

KEYWORDS*: Deep Dream algorithm, Inceptionism, CNN (Convolution Neural Networks), AI (Artificial Intelligence), Activations, Features, Loss.*

I. INTRODUCTION

In this paper I would present an amazing algorithm known as Deep Inceptionism learning algorithm the Deep dream algorithm. So Deep Inceptionism is the freakiest A.I. algorithm to date. trained this AI to perform image classification. Actually, go inside the network and open some of neurons open some of hidden layers and try to understand what do they actually like interpret, that is tons of features. There are different features to create kind of art using AI algorithms or the Deep Dream algorithms specifically [1][2][3].

So, what is the dream, the dream is a computer vision algorithm that has been created by Alexander.
It's basically like giving humans an extremely powerful drug. let's say acid for example they can actually see these kinds of trippy effects because it gives you kind of a sense of how close an eye is to the actual biological neurons that our human level intelligence.

Figure (1) showing different images of actually you know like what people imagine to be when they try the acid. when people take these powerful drugs, you can actually see these extremely powerful colours. An idea of how close artificial intelligence is to our natural intelligence or human level intelligence.

Figure (1): Game of Thrones on acid

So, what we do is that we keep feeding in these images so we increasingly feed the image to the network and the more you feed it to the network the more you will be able to extract or you know see all these weird features in the actual image [1][2][3].

So that's kind of a quick overview of the dream outline which is the similarity that we have between the dream and humans in general. When you were a kid and looking at the clouds and trying to interpret the shapes in the sky and trying to kind of interpret different shapes of the clouds. For example, that's maybe a horse maybe that's a dog and so on. Deep Dream does the exact same thing by boosting the

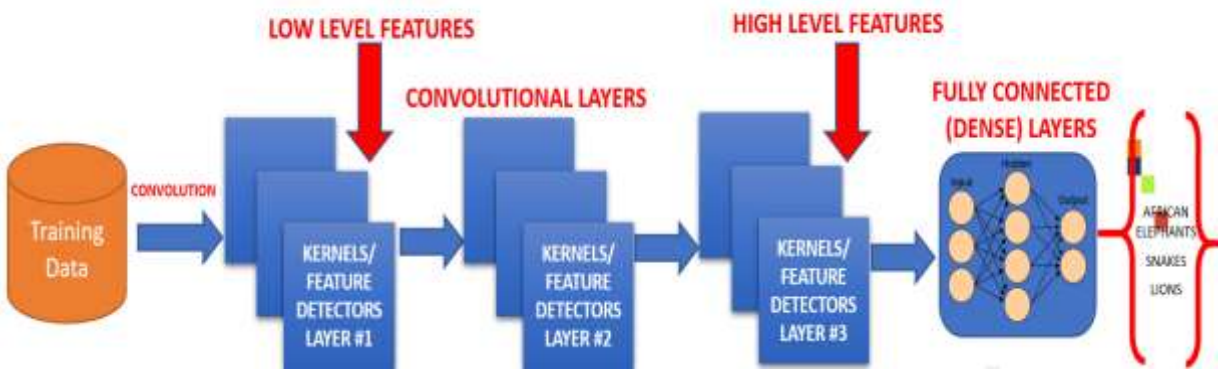

Figure (2): Work Flow of Deep Dream Algorithm

patterns it sees in a given image based on what it has been trained to see in the past during training. So, what actually deep dream we're going to do from a mathematical standpoint, take a network, this network has been trained to see you know different features of animals or like you know food or whatever. And try to boost some of the out of the activations coming out of these layers of certain layers just to try to maximize it. All these features
pop up when we feed in images to the network. When we apply the Deep Dream algorithm, we'll be able to magnify what's actually the network has been trained upon beforehand. Let's say animal images to try to make it pop up or make it like increased activations [1][2][3].

A. How does deep dream algorithm work?

The algorithm works by creating dream-like effect. As you increasingly feed the image to the network, the weirder features will start to pop up. If you feed an image to a CNN, the first layers generally detect low-level features such as edges. As you go deeper in the network, higher level features are then detected such as faces, trees, and cars. “The last rare layers collect those into whole clarifications—these neurons activate in retort to very composite effects such as whole structures or trees.” [1][2][3].

When you feed in an image to a trained ANN, the neurons fire and generate activations. The deep dream algorithm work by trying to change the input image in a way that would make some of these neuron’s fire more (boost the neurons firing or activations). You can select which neurons in which layer you are interested in making them fire more prominently. The process is continuously repeated until the input image now contains all features that a specific layer was originally looking for Example: if a certain layer was expert in recognizing dog faces and you feed in an image of a blue sky, the deep dream algorithm will continuously change the input

image and start creating images of dogs faces on top of the blue sky. The process keeps repeating until the layer of interest is happy with the results [1][2][3].

B. Deep Dream Algorithm Steps:

1. Forward an image through a trained ANN, CNN, ResNet. Etc

2. Select a layer of choice (first layer’s capture edges, deep layers capture full shapes such as faces)
3. Calculate the activations (output) coming out from the layer of interest.
4. Compute the gradient of the activations with respect to the input image.
5. Modify the image to increase these activations, and thus enhance the patterns seen by the network resulting in trippy hallucinated image!
6. Iterate and repeat over multiple scales

As shown in figure (2) Look at the standard convolution all neural network you will find that training data. And then here we have certainly multiple layers of feature maps or feature detectors, here we have the kernels or feature detectors. This is layer 1. We have layer to layer 3. And these are call convolution layers. Then we do Max pooling for example which is a way of kind of compressing the image, Now, we have rectified linear units and we do afterwards we apply these convolution layers we feed in the actual pixels the flattened pixels will feed it within them to a fully connected dense artificial neural network and the network will be able to classify images based on the tuning data for let's say African elephant’s snakes lions whatever the network has been trained to perform. So, if you feed in an image to CNN, it's convolution and you're a network in general the first layers, these layers generally detect low level features such as edges. So, from a very high level if you check out these early layers, they check kind of very low-level features such as edges curves and stuff like that in the image.

However, as you go deeper in the network as you go inside deeper in the network you will find that higher
level features are being detected such as faces trees and cars. The final layers or the final few layers assemble those into complete interpretations. These neurons activate in response to a very complex things such as in power buildings or trees. And we could do

maximize the activation. [1][2][3] Train artificial network the neurons fire and will generate activations the Deep Dream algorithm work by trying to change the input image in a way that it would make some of these neuron's fire more. It's kind of boost the neurons firing or activations. You can select whatever neurons in which layer you are interested in making them fire more prominently to resonate or residual net or Inception net. Then we calculate the gradient of the image with respect to the activations of the selected layer to change the original image and my desired output with lost function and try to minimize the error.

II. IMPLEMENTATION

The dream is a computer vision algorithm. The algorithm works by creating dream Like Effects.

Import the libraries apply our image pass our image to it to generate our activations. Import our image simple image and feed to network to generate activations. Next, import our model with weights then take the trained network that has already been trained on image net which is basically a repository of millions and millions of images of tons of classes.
Next, load our base model you can see we have around 22 million parameters in it. There is activation function and another convolution. Another batch normalization and other activation there is we call a mixed layer. So, mixed zero is basically a concatenation of multiple activations or multiple kind of outputs. You're mixing together and here throughout the inception it there are multiples of them. So, there is mixed zero mixed one two three four Up until mixed tens mixed one and then there is mixed two. To specify an input, we have a 3D model and we need to specify which layer which activation
we select our specific layer of interest and try to
attempt at maximizing the loss which is the activations generated by the layers of interest and run that and specify my target size which is 2
to 5 by 3 7 5. Afterwards take image divided by two hundred and fifty-five let's run that it expands dimensions

The Deep Dream algorithm network able to capture a lot more kind of complex features such as faces eyes. Now, calculate loss and attempt at maximizing the loss which is simply the activations generated by the layer of interest and if we decided to select deeper layers within the network, we'd be able to generate more complex features such as face car and tree and so on.

And Inception network mainly has multiple concatenated layers. Up until mixed 10 so we will calculate the loss which simply represents the sum of activations for a given layer. That's the definition basically of the loss. And most of the time in general when we train artificial neural network, we wanted to obtain the gradient descent. So, we actually wanted to minimize the loss or minimize the error.
However indeed dream it's a little bit different. It's actually a little bit tricky. What we wanted to do that we wanted to actually maximize the loss. We want to perform gradient ascent. We want to go up not down. We want to see these activations. And that's why we did dream on going to perform gradient descent. And to do that First, we're going to build a function. This function is called calculate loss and this function simply takes my image and takes my model.

The function simply works by feeding in or feed forwarding the input image through the entire network and generate activations and afterwards after using these activations we are going to calculate the average end summation as well of those outputs. Again, the function we are going to send that function arguments. So, we're going to send that function my image and the model we're going to see the image and we're going to expand dimensions. So, we're going to CTF got expand dimensions. we can take the image and convert it into a batch format. Then we do DBS remodel pass along here in my image batch to it. And that will generate layer activations that will be the output coming out from my model. And here we're going to simply visualize or just plot activation values. And we're going to show as well the shape to which is simply the dimensions of activations afterwards.

Let's create an accumulator which is an empty vector on an empty array and we're going to call it losses. And we're going to do it again to create a for loop and this full loop will go for activation in layer activations which is simply here what we got once we'd end the model. Every time we're going to take all these activations coming out from the from these specific layers and we're going to calculate the mean or simply the average of all of them. And that would be the loss per layer. Remember that here they have two layers had mixed three and mixed five and that means in losses we have two values because now we have the mean or the average values per layer or less or four mixed three. Now we have a loss and four mixed five.

And again, we have another loss and that's why what we have done here is that we took losses and we used
append simply kind of add to the accumulated every time you calculate one loss from a layer you appended to losses and you keep doing that and you finish all your activations afterwards you are going to print losses so we can print for multiple activation layers you can pick the losses and show as well loss of shape and afterwards. That's very critical is that now we have multiple kind of losses coming out from multiple layers or what I need out of this function is to come up with one single number. And I wanted to maximize that specific single number.

So, to do that we are going to sum up all the losses coming out from all the layers and pass all the losses and that would be the exact same parameter that is plotting here or printing and return it back from the function. Simply put this function to send it image centre with my model and the function will give me back one single number the present that represents the summation of all the losses coming out from all the layer those that have selected the basic.

Now, maximize by performing gradient ascent and run our dream algorithm. That's the core of the algorithm from a very high level. And here creates call octave scale. And do the image resize it to let's say one point three one point six and so on and run for 400 steps. We are running at the actual algorithm at various sizes repeat that five times. So, that was the first granular level. And then we're going to change it again. There'll be the next level next level next level. And you can see it all becomes a lot smoother a lot more.

III. RESULTS AND COMPARISIONS

By running my model and you can see it all becomes a lot smoother a lot more.

The parameters the model used totally after simulation are given below:

Total params: 21,802,784
Trainable params: 21,768,352
Non-trainable params: 34,432

With the above parameters the image is smoothen, the result of output you can see in below figure (3).

And for mix values of (225, 375, 3) the result of output image smoothen is shown below figure (4).

Figure (3)

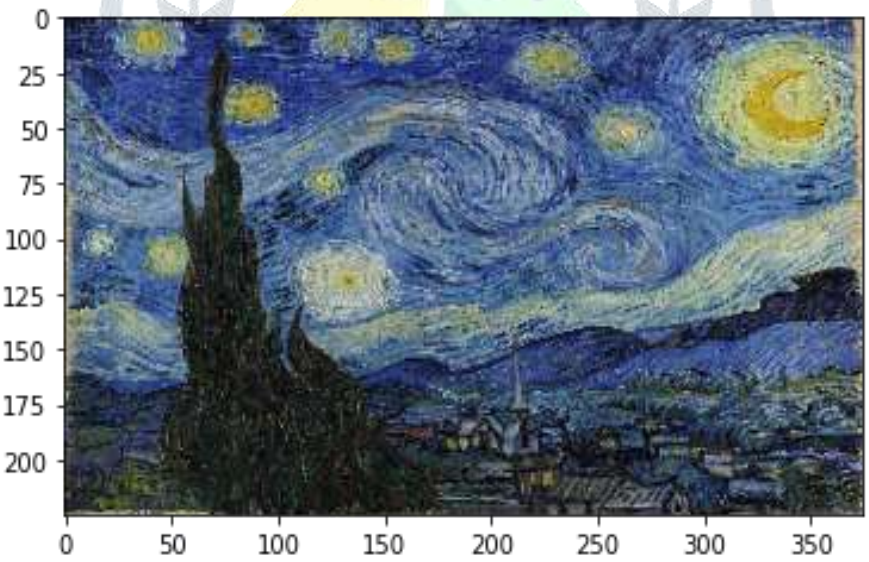

Figure (4): with mix values (225, 375, 3)

Below I presented the output of losses and Activation's output resulted by the Deep Dream Inceptionism learning model.

LOSSES (FROM MULTIPLE ACTIVATION LAYERS) = [<tf. Tensor: id=34133, shape= (), dtype=float32, numpy=0.2634555>, <tf. Tensor: id=34135, shape= (), dtype=float32, numpy=0.17727219>]

LOSSES SHAPE (FROM MULTIPLE ACTIVATION LAYERS) = (2,)
SUM OF ALL LOSSES (FROM ALL SELECTED LAYERS) = tf. Tensor (0.4407277, shape= (), dtype=float32)

With the above losses below, I am showing the result of smoothen of input image with the deep dream algorithm for step 0, step 100, step 200 and step 300 with corresponding loss values in Figure (5), Figure (6), Figure (7) and Figure (8) accordingly.

Figure (5): Step 0, loss 0.6168044805526733

Figure (6): Step 100, loss 1.4076865911483765

Figure (7): step 200, loss 1.7209874391555786

Figure (8): Step 300, loss 1.90653395652771

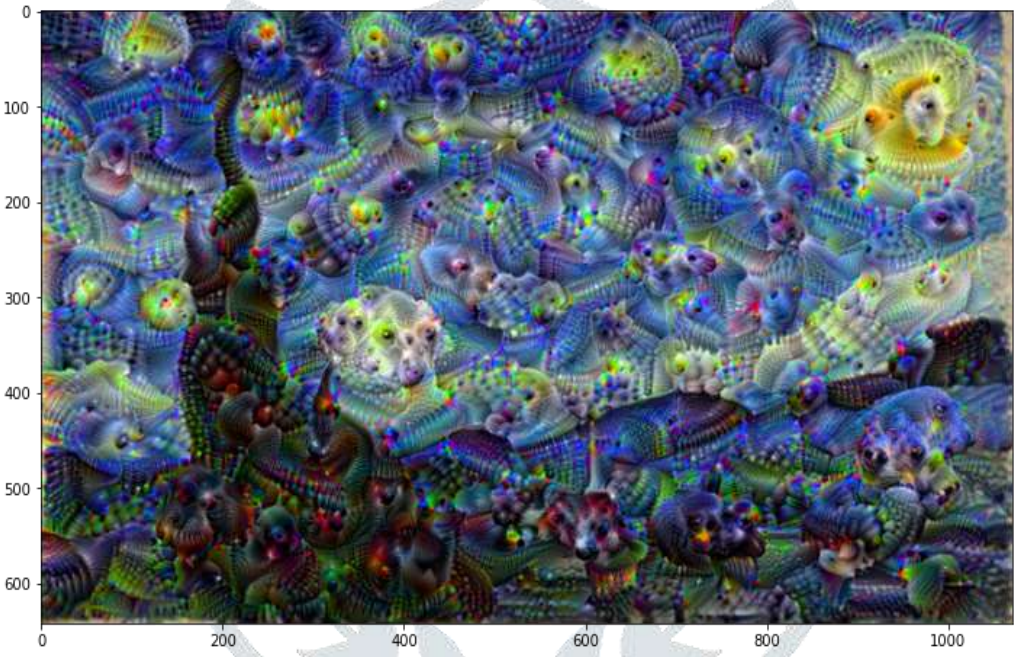

Figure (9): Step 0, loss 0.9181942939758301

And the above figure (9) is showing the final lot more smoothen image output of Deep Inceptionism (Dream) algorithm. Actually, see these kinds of trippy effects because it gives you kind of a sense of how close a eye is to the actual biological neurons that our human level intelligence.

IV. CONCLUSION

The dream is a computer vision algorithm. The algorithm works by creating dream Like Effects. By running my model and you can see it all becomes a lot smoother a lot more. It's basically like giving humans an extremely powerful drug. let's say acid for example they can actually see these kinds of trippy effects because it gives you kind of a sense of how close a eye is to the actual biological neurons that our human level intelligence. I have tested the Deep Inceptionism learning vision performance on GPU of Intel® Core™ i3-7100U CPU and I got more lot smoothen image with a kind of trippy effect.

Future Work: Much better when you run it at the octave and you can see here becomes you know like amazing artists.

V. REFERENCES

[1] Keisuke Suzuki, Warrick Roseboom, David J. Schwartzman & Anil K. Seth, A Deep-Dream Virtual Reality Platform for Studying Altered Perceptual Phenomenology, SCIENTIFIC REPORTS, 2017.

[2] S. Chien, S. Choo, M. A. Schnabel, W. Nakapan, M. J. Kim, S. Roudavski, ARTIFICIAL IMAGINATION OF ARCHITECTURE WITH DEEP CONVOLUTIONAL NEURAL NETWORK, 21st International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA, 2016.

[3] L. Spratt, Dream Formulations: Humanistic Themes in the Iconology of the Machine-Learned Image, art historical journal Kunsttexte.de.

Author:

I am Mrs. T. Tritva Jyothi Kiran with 10 years of work as Assistant Professor in Computer Science Department. Previously I have completed AICTE funding Project on IEEE802.11e in JNTUH and published in IEEE. I have been awarded Two times for the National Award for Excellence “Adarsh Vidya Saraswathi Rastriya Puraskar” from Glacier Global Management in 2020. And Received “Women Researcher” Award from 9th international conference organized by VDGOOD Professional Association. Extant I am working in the research domain Deep Learning using TensorFlow and Swarm Intelligence. You can find my Lectures during COVID in my Blog is tritvajyothikiran.blogspot.com and in Tritva Jyothi YouTube Channel.