Bedtime Story: Manipulating and Replicating Deep Dreams
Esti mese: Deep Dream manipulációk és replikák

Pap Ervin, Horváth Lehel, Gyöngyössy Natabara Máté
Budapest University of Technology and Economics
4-6 Bertalan Lajos Street, 1111 Budapest, Hungary
e-mail: papervin1998@gmail.com, horilehel5@gmail.com, natabara@gyongyossy.hu

***Abstract*—This paper is a summary of the group's deep learning project, about Deep Dreams. Included algorithms utilize standard techniques of neural style-transfer, Deep Dream and GANs to create new artistical or representationally informative images. A fusion of neural style-transfer and Deep Dream is presented. These "Guided Dreams" use the activation patterns of guide images to reduce or modify the element set from which the network works while dreaming. Tests include various popular image processing networks.**

Jelen mü csapatunk deep learning projektjének összefoglalása a Deep Dream témakörében. Az itt ismertett algoritmusok a klasszikus neurális stílusátvitel, a Deep Dream és a GAN hálózatok kihasználására törekszenek, hogy új müvészi vagy belső reprezentációt bemutató képeket generáljunk. Bemutatásra kerül a neurális stílusátvitel és a Deep Dream összeolvasztása is. Ezek az "Irányított álmok" segédképek által kiváltott aktivációk segítségével módosítják, illetve csökkentik az "álmodó" háló elemkészletét. A teszteket több népszerü képfeldolgozó hálózaton hajtottuk végre.

***Index Terms*—Deep Dream, Neural Style-Transfer, Generative Adversarial Network, Representation Analysis**

I. Introduction

Making artwork with neural networks is a relatively new field in deep learning. Within this, image generation and image distortion techniques are the most widespread. With the help of a few neural networks, we can easily create realistic images from just some noise and transform existing ones into something dreamlike. Some of these methods are discussed in this document. First of all, we focus on the existing Deep Dream method, but then we will introduce another with GAN architectures.

II. Problem Statement

This chapter presents the theoretical background of the problem.

A. Inner representations

Inner representations of neural networks is a well studied, yet still not completely discovered field. This short section aims to give a logical (not chronological) overview of how these methods, like content recreation, neural style-transfer, and Deep Dreams. The basic idea behind this kind of optimization is the fact, that every input activates each layer in an unique way. This pattern of activations is the network's representation of the given input image. Using dimension reduction (like pooling layers) of course results in some loss of details, but the representation becomes more general.

The main idea behind content reproduction is the inverse of the above mentioned process. Saving the given activations of an image in some, or all of the layers should be enough to reproduce the original input image at least partially. Of course weights of networks undergoing these processes should be constant matrices. We start the optimization with a white-noise input and use the gradient of input image pixels for activations to perform optimization steps on the brightness values of each pixel. A loss function can be defined as the mean square error of the saved activations and the network's response to the input image. Steps like this will gradually lead to a reconstructed image. Deeper layers will result in blurry images, as they focus on higher level objects on the image, and lose pixelscale information [1].

Style reconstruction takes this idea to another level, by looking for correlations between different layer activations. By calculating the Gram-matrix of these layers and using them to provide a loss value a gradient descent can also be performed in order to get an image with the style of the image we got the activations from. Mixing the content and style reconstruction as the goal of optimization results in the neural style-transfer method which tries to reconstruct an image with the recorded style of another image. [1] Tinkering around with this method superresolution as style [2] can also be applied for enhancing resolution of images. Some solutions suggest to start with the content-image as an initalizer input, and other variations of loss definitions are also available.

The above mentioned methods both aim to observe patterns of activations, but we get barely no idea about the key stimuli which would make a map, a layer or even a neuron reach a high activation. This is where Deep Dreams can make some difference.

DeepDream is a computer vision program created by Google engineer Alexander Mordvintsev which uses a convolutional neural network to find and enhance patterns in images via algorithmic pareidolia, thus creating a dream-like hallucinogenic appearance in the deliberately over-processed images. [3] Google's program popularized the term (deep) "dreaming" to refer to the generation of images that produce

desired activations in a trained deep network, and the term now refers to a collection of related approaches. [4] The software is designed to detect faces and other patterns in images, with the aim of automatically classifying images. However, once trained, the network can also be run in reverse, being asked to adjust the original image slightly so that a given output neuron (e.g. the one for faces or certain animals) yields a higher confidence score. This can be used for visualizations to understand the emergent structure of the neural network better, and is the basis for the DeepDream concept. However, after enough reiterations, even imagery initially devoid of the sought features will be adjusted enough that a form of pareidolia results, by which psychedelic and surreal images are generated algorithmically. The optimization resembles Backpropagation, however instead of adjusting the network weights, the weights are held fixed and the input is adjusted. For example, an existing image can be altered so that it is ”more cat-like”, and the resulting enhanced image can be again input to the procedure. This usage resembles the activity of looking for animals or other patterns in clouds. Applying gradient descent independently to each pixel of the input produces images in which adjacent pixels have little relation and thus the image has too much high frequency information. The generated images can be greatly improved by including a prior or regularizer that prefers inputs that have natural image statistics (without a preference for any particular image), or are simply smooth. [5] For example, Mahendran et al. [5] used the total variation regularizer that prefers images that are piecewise constant. Various regularizers are discussed further in. [6] An in-depth, visual exploration of feature visualization and regularization techniques was published more recently. [3]

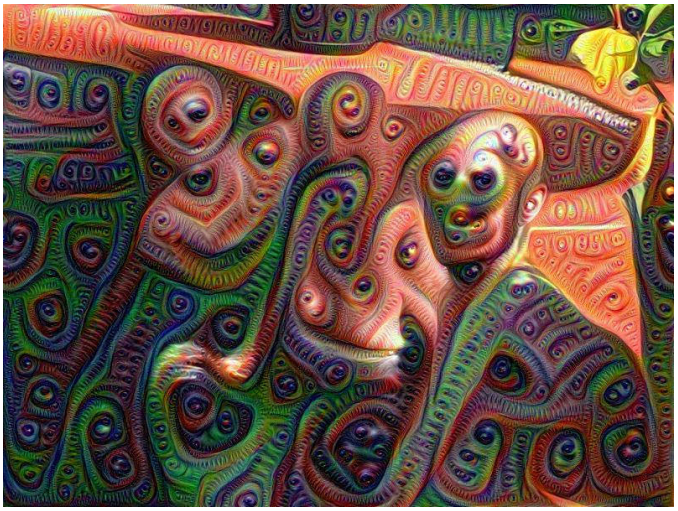

Fig. 1. An image generated by a DeepDream software

Tensorflow Lucid, a project ran by the Tensorflow crew offers a wide range of other ideas. An idea of this homework comes from their work related to negative activation representations, where, after running through the network a batch of images, they select a group of maximally and minimally activated neurons, and optimizing for the minimalization or maximalization of those selected neurons. This results in interesting images somewhat similar or almost totally opposing (as a neural network's point of view) to the original batch of images. The proposed Deep Dream of this work pulls this idea on the skeleton of a classical Deep Dream.

B. GAN

Generative adversarial networks (GANs) are an exciting recent innovation in machine learning [7]. For example, GANs can create images that look like photographs of human faces, even though the faces don't belong to any real person (Figure 2).

Fig. 2. Fake faces generated by GANs

GANs achieve this level of realism by pairing a generator, which learns to produce the target output, with a discriminator, which learns to distinguish true data from the output of the generator. The generator tries to fool the discriminator, and the discriminator tries to keep from being fooled.

- The generator learns to generate plausible data. The generated instances become negative training examples for the discriminator.
- The discriminator learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.

When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake. As training progresses, the generator gets closer to producing output that can fool the discriminator, Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases. Here's a picture of the whole system:

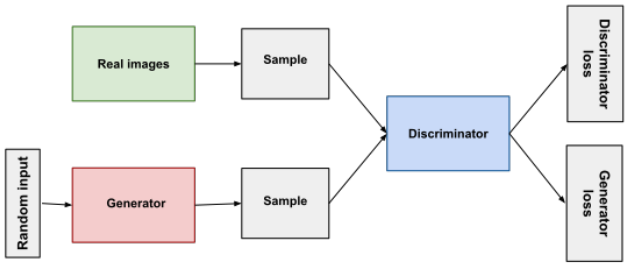

Fig. 3. The structure of GANs

Both the generator and the discriminator are neural networks. The generator output is connected directly to the discriminator input. Through backpropagation, the discriminator's classification provides a signal that the generator uses to update its weights [8].

III. Proposed Algorithm

This section is an introduction to the work done as the group's project work, it consists of three parts. Exploring basic Deep Dream and deep style solutions, their necessary improvements, in order to create HD dreams, a new approach, and mixture of deep style and Deep Dream, Guided Dreams, and GANs which can replicate Deep Dreams based on a small dataset of ours.

After playing around with a style-transfer and content reconstruction application made for practice reasons we started to study Deep Dream solutions. Ideas and programming hints were taken from two online tutorials [9] [10]. We are working with a recursive gradient ascent optimizer, which starts out from a downscaled image, and upscales it step by step, until the original size is reached. Of course networks work well with smaller images close to their default input size. To reduce big input image is rolled randomly in order to get randomized smaller parts of it and run it through the dreamer network. These random rolls are essential to have dreams equally distributed around the image without harsh lines and borders. On top of this architecture a maplist was implemented, which lists maps of each layer to use, thus handpicking a neural map is possible.

As mentioned in section II, the creators of Tensorflow Lucid came up with an idea to select maps with highest or lowest activation and run optimization based on them. We tried the same method fused together with Deep Dream, and the results somewhat changed. This idea was a good starter point, to make something better and more general, Guided Dreams.

The idea behind Guided Dreams is pretty simple. Usual Deep Dreams are really psychedelic, sometimes even scarry. They are a mixture of elements which the layers used for gradient ascent recognize. Why not choose some elements we would want to include? Connecting this artistic desire and the puzzle of finding out exactly what kind of elements does the network recognize on an input image, Deep Dreams work based on activations caused by a guide image. The normalized sum of activation of each map in a layer is used as a weight when calculating loss. This emphasises the recognition of those patterns which are present in the guide image as well. Every other step of the optimization is the same, and the results are quite promising. This method is similar to a style-transfer or content reconstruction, because it uses relations between another image's activations to tune the input of the network, while not exactly looks for stlye, nor for content, it rather tries to reproduct patterns it saw before dreaming. The main difference is the unbounded gradient ascent type of optimization, which gives the network enough freedom to create whatever is the most likely in the image with the likelyhood of choosing an element acquired from the guide image.

Fig. 4. Original image

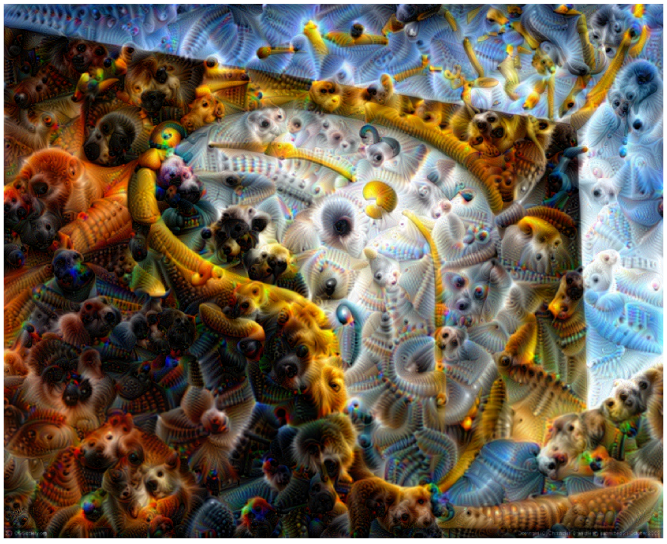

Fig. 5. Standard Deep Dream without guide

IV. Experimental Results

Besides creating a working, customizable Deep Dream generator, our group was able to develop a new technique which combines the base ideas of style-transfer and Deep Dream. In this section examples of the achievements realized by GANs and Guided Dreams are presented. These images were generated using Inception V3 [11], VGG16 [12] and MobileNet V2 [13], all three pretrained on ImageNet [14] dataset. Transfertraining networks to specific tasks (e.g.: subsets of ImageNet) did not make significant difference in dream generation.

Fig. 6. Hungarian folk patterns (Matyó) used as guide image for Fig. 7

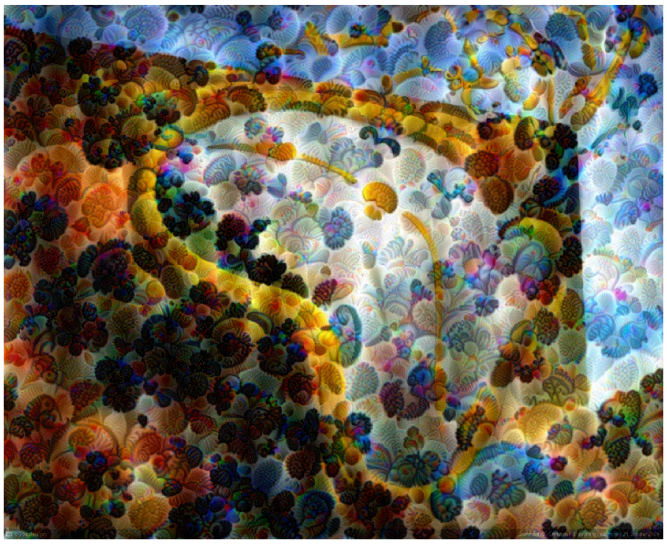

Fig. 7. Guided Dream with floral guide image

As a demonstration of how this Guided Dream works, please take a closer look at the figures of this section, and look for the similarities of the guide and dreamed images. To generate these images we used Inception v3 network with concatenate (mixed in keras) layers 3, 5, 6, 7. It is clear that these layers prefer furry animal parts, but they have some hidden features like perfect floral patterns and even box or plate-like elements, and shiny globes. To process the style of text however was a too hard objective for this kind of layer setup.

Fig. 8. Arcade games as guide image for Fig. 9

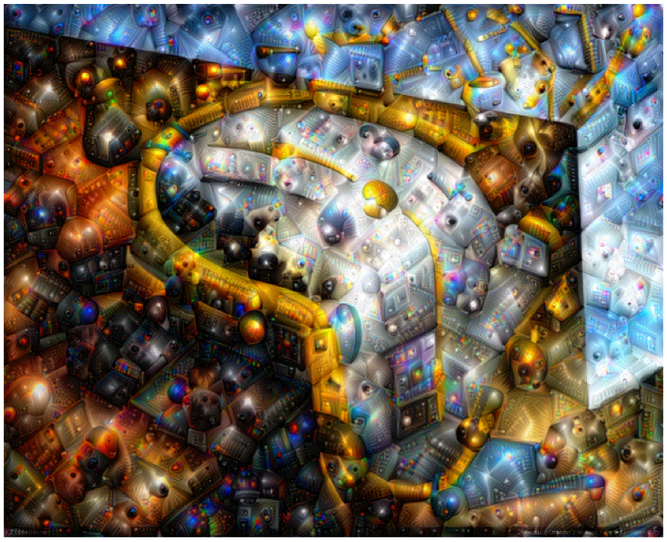

Fig. 9. Guided Dream with box-like patterns

An interesting observation was, that MobileNet v2 had seemingly different active spots, and it memorized more meaningless pattern-like information. Another interesting thing is, that parts of bigger items are stored in a short range of maps, and they are often close to eachother, while this was totally different when tested with VGG or Inception.

In the first part of our project, we created input images for GAN training. We got images from kaggle natural images dataset [15]. These images needed some preprocessing steps. For example, we resized them to 299x299 pixel size, this is the standard VGG size and rescaled with ImageDataGenerator. After these preprocessing steps we could use the Deep Dream network to make images with dreamed effects. The most popular area of Generative Adversarial Network usage is face

Fig. 10. A textual guide image for Fig. 11

Fig. 11. Guided Dream with text-like guide

generation. But using GANs in Deep Dream generation is not a bad idea too. The base idea was that if we generate the same style of dreams with just some selected objects, we might be able to get similar images. The GAN starts to generate images from random noise and learns the main features, so if every image has same dreamed objects on it, the network learns the dream itself. In the first step the images were scaled to a smaller size. After this begins the image generation. In this flow we printed the generated images after a few epochs of training. With this function we were able to prove that learning occurs in these networks. The result of this training is not too bad, because after some thousand epochs most images show Deep Dream effects, but the network started to learn the main object of the original images too, so many more source categories should be involved, to avoid fitting to the original images.

V. Conclusions

This project has two main contributions, one of them is the utilization of GANs to generate Deep Dream-like images, the second one is generating Deep Dreams similar to a guide image. GANs and Guided Dreams both provide a relatively fine precision when it comes to image generation, however further optimizations are needed.

GANs could be improved to enable full scale image-to-image transformations, with more types of dreams in their repertoire.

Guided Dreams are promising not just as some form of art, but as a representation analysis as well. By creating local or scale-dependent weights of the dream temporal information could also be processed. Generating key images for specific neuron activations and teaching a separate network, to recognise the existence of these keys or clues could in theory lead to quick position estimation, when using local groups. Deeper research should definitely be conducted here.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[2] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[3] C. S. et al., "Going deeper with convolutions," *arXiv*, 09 2014.

[4] "Deepdream," https://en.wikipedia.org/wiki/DeepDream, 2019, [visited: 2019.12.12].

[5] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," *arXiv*, 11 2014.

[6] J. Y. et al., "Understanding neural networks through deep visualization," *arXiv*, 06 2015.

[7] I. J. G. et al., "Generative adversarial nets," *arXiv*, 06 2014.

[8] "Gan," https://developers.google.com/machine-learning/gan, 2019, [visited: 2019.12.10].

[9] (2018) Deep dream with tensorflow: A practical guide to build your first deep dream experience. [visited: 2019.12.12]. [Online]. Available: https://hackernoon.com/deep-dream-with-tensorflow-a-practical-guide-to-build-your-first-deep-dream-experience-f91df601f479

[10] (2019) Deepdream. [visited: 2019.12.12]. [Online]. Available: https://www.tensorflow.org/tutorials/generative/deepdream

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[15] P. Roy, S. Ghosh, S. Bhattacharya, and U. Pal, "Effects of degradations on deep neural network architectures," *arXiv preprint arXiv:1807.10108*, 2018.