Presentation on "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," by Alec Radford, Luke Metz, and Soumith Chintala

John Hancock

Florida Atlantic University

jhancoc4@fau.edu

November 22, 2018

Overview

Introduction

- In the field of deep learning, we have crossed into an era where we are experimenting with designs that involve more than one neural network.
- Generative adversarial networks are a design pattern to employ two neural networks.
- "Unsupervised representational learning with deep convolutional generative adversarial networks" [1], is a milestone in the development of Generative Adversarial Networks where the authors report a reliable architecture that incorporates convolutional neural networks into the generative adversarial network design.
- The authors tout several applications of their design to prove its utility.
- For the remainder of this presentation, we will refer to the paper entitled , "Unsupervised representational learning with deep convolutional generative adversarial networks," as, "the DCGAN's paper," or by its reference number [1].

Background

A generative adversarial network (GAN) is a neural network with two components. Goodfellow *et. al* invent GAN's in [5]. To a first approximation, GAN's work as follows:

- The first component is a *generator* that learns to transform vectors of random numbers into output values that resemble instances from some dataset.
- The second component is a *discriminator* that classifies things into two categories:
  - the class of instances of the dataset, and
  - the class of generator outputs.
- "At convergence, the generators samples are indistinguishable from real data, and the discriminator outputs $\frac{1}{2}$ everywhere. The discriminator may then be discarded" [11]. from Deep learning , Goodfellow *et al.*

Background

- ... or not. The authors of the DCGAN paper find a use for the discriminator.
- In the context of this paper, the outputs are images. However, researchers use GAN's where the generators create other artifacts. We find an extensive list on Github [12] of over 500 research projects. Some examples from this list are:
  - imputing missing values in datasets,
  - generating music,
  - fraud detection, and
  - playing chess.

Contributions

The authors of the paper make several contributions they...

- invent an architecture for DCGAN's,
- use the convolutional layer filters of trained DCGAN's discriminators as feature extractors for doing classifications,
- demonstrate that after training the DCGAN, its filters learn how to represent images, and
- present a method of doing vector arithmetic using DCGAN inputs to do inferences *à la* Word2Vec [6].

Architecture

A high-level overview of the architecture:

- they use convolutions
- they use de-convolutions
- subsequent work such as the texts for this course [10] [11] suggests one should employ dropout, but we do not find that the authors of this paper use it when inspecting the code in [2].

Architecture

subsectionDeconvolution visualization The term deconvolution seems to have stuck because we see library functions in The authors of the paper prefer the term, “fractionally-strided.” We find another author who gives the name, “

Datasets

- The authors use three datasets for training:
  - Large Scale Scene Understanding (LSUN),
  - Imagenet 1-K, and
  - Faces.
- The authors two datasets for evaluating unsupervised learning:
  - Canadian Institute for Advanced Research (CIFAR) 10
  - StreetView House Numbers (SVHN)
- Note: the authors mention that they heuristically removed duplicate images from LSUN to prevent the DCGAN from memorizing images.

LSUN

- The authors used images of bedrooms from the LSUN dataset [3] as input to their model.

How did they do it?

The code for this paper, as well as many others in the references and that one may find in the course of research, is on Github in the dcgan_code project [2].
This code is a bit outdated, however the dcgan_code Github project has a link to the DCGAN-tensorflow [4] project that we find more accessible.

Fractionally-strided convolutions

A deep convolutional generative adversarial network (DCGAN) is a GAN that uses convolutional neural networks for the generator and discriminator.

The discriminator uses convolutional layers in the sense we are familiar with, such as the convolutional layers LeCun describes in LeNet-5 [13].

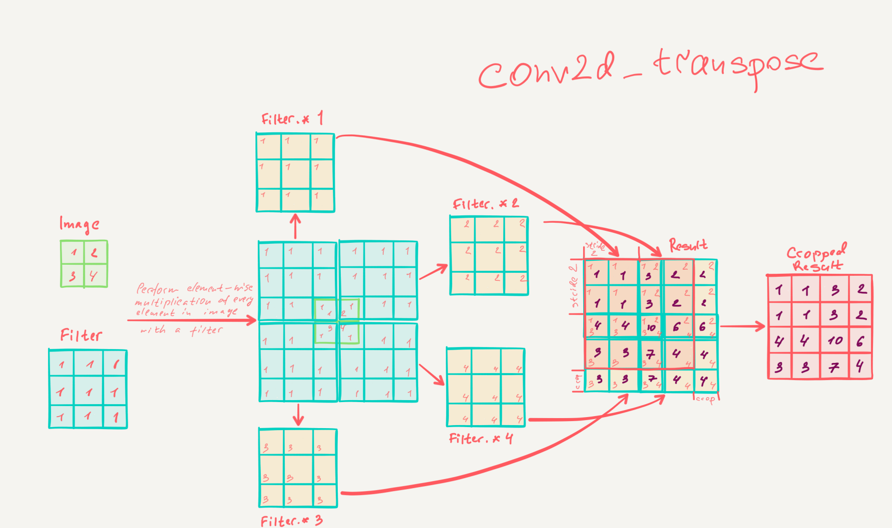
However, the generator uses convolutional layers that we find called deconvolutional layers in the source code that accompanies this paper [2], and elsewhere, but that in the paper the authors write that we should prefer the term "*fractionally-strided*." The computations that comprise fractionally-strided convolutions are not clear to us from the paper or the source code that accompanies it. We find the source code unclear because the authors implement fractionally-strided convolutions using library functions, the source code of which we run out of time to peruse. The paper lacks detail on how to compute a fractionally-strided convolutions.

Fractionally-strided convolutions

On the other hand, the github project [2] that accompanies the paper [1] links to a Tensorflow implementation of the same code: [4] where the author of this code implements fractionally-strided convolutions using Tensorflow's conv2d_transpose. We did some internet searching and found [14]. This reference plus using conv2d_transpose in a small example helps us understand precisely how the fractionally-strided convolution operation works. We feel confident to rely on conv2d_transpose because the authors of the paper [1] we review here provide a link to the code in [4] in their own code. We feel the authors of the DCGAN's paper's endorsement of the Tensorflow code means conv2d_transpose is a valid method for doing what the authors of [1] refer to as fractionally-strided convolutions, and that a good understanding of conv2d_transpose is a good understanding of fractionally-strided convolutions.

Fractionally-strided convolutions

We found some example code, and a great diagram from a StackExchange.com discussion [15]. That explains in detail how conv2d_transpose works. This is the diagram we found:



Fractionally-strided convolutions

- Since anyone might write anything in online discussion forums, we decided to confirm that Tensorflow's conv2d_transpose operation works as the digram implies. conv2d_transpose has three important parameters: input tensor, filter, and stride.
- One should be careful not to confuse the term filter we have for conv2d_transpose and the filters that the authors of the DCGAn's paper show on page 9.
- In the context of the DCGAN paper it is better to think of the filter parameter of conv2d_transpose as a kernel for the conv2d_transpose operation, and the filter is the result of applying conv2d_transpose to the input tensor.

Fractionally-strided convolutions

- The next slide shows a 4x4 input tensor, and the result of applying conv2d_transpose to that tensor, with stride of 1,4,4,1.
- Conv2d_transpose operates on 4 dimensional tensors, so we must embed the 4x4 matrix in a 4-dimensional tensor, and stride through it accordingly. Note on the next slide how most entries in the output tensor are copies of entries in the input tensor, except where the 5x5 kernels must overlap in order to achieve the 16x16 output.

Fractionally-strided convolutions

The input tensor:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

The output we can see the entries in the input matrix copied into 5x5 intermediate tensors and then added according to the stride of 4, and values are added when we have overlap. We show a screen shot to prove the code runs.

```
jhancock2010@minty ~/git/DCGAN-tensorflow
File  Edit  View  Search  Terminal  Help
result
[[ 1.  1.  1.  1.  3.  2.  2.  2.  5.  3.  3.  3.  7.  4.  4.  4.]
 [ 1.  1.  1.  1.  3.  2.  2.  2.  5.  3.  3.  3.  7.  4.  4.  4.]
 [ 1.  1.  1.  1.  3.  2.  2.  2.  5.  3.  3.  3.  7.  4.  4.  4.]
 [ 1.  1.  1.  1.  3.  2.  2.  2.  5.  3.  3.  3.  7.  4.  4.  4.]
 [ 6.  6.  6.  6. 14.  8.  8.  8. 18. 10. 10. 10. 22. 12. 12. 12.]
 [ 5.  5.  5.  5. 11.  6.  6.  6. 13.  7.  7.  7. 15.  8.  8.  8.]
 [ 5.  5.  5.  5. 11.  6.  6.  6. 13.  7.  7.  7. 15.  8.  8.  8.]
 [ 5.  5.  5.  5. 11.  6.  6.  6. 13.  7.  7.  7. 15.  8.  8.  8.]
 [14. 14. 14. 14. 30. 16. 16. 16. 34. 18. 18. 18. 38. 20. 20. 20.]
 [ 9.  9.  9.  9. 19. 10. 10. 10. 21. 11. 11. 11. 23. 12. 12. 12.]
 [ 9.  9.  9.  9. 19. 10. 10. 10. 21. 11. 11. 11. 23. 12. 12. 12.]
 [ 9.  9.  9.  9. 19. 10. 10. 10. 21. 11. 11. 11. 23. 12. 12. 12.]
 [22. 22. 22. 22. 46. 24. 24. 24. 50. 26. 26. 26. 54. 28. 28. 28.]
 [13. 13. 13. 13. 27. 14. 14. 14. 29. 15. 15. 15. 31. 16. 16. 16.]
 [13. 13. 13. 13. 27. 14. 14. 14. 29. 15. 15. 15. 31. 16. 16. 16.]
 [13. 13. 13. 13. 27. 14. 14. 14. 29. 15. 15. 15. 31. 16. 16. 16.]]
(2018-11-04-py3.6) jhancock2010@minty ~/git/DCGAN-tensorflow $
(2018-11-04-py3.6) jhancock2010@minty ~/git/DCGAN-tensorflow $
```

Results

The authors of the paper make several contributions: Here reference github code implementation How to run their mnist example:

- Use AWS Ubuntu Deep Learning Instance
-
  - Expensive $\approx$ \$0.65 per hour!
  - Configure an alarm to shut the instance down after 3 hours!
- Create virtual environment
-
  - Use pip to install libraries
  - force install of Theano 0.9.0 (pip install -I Theano 0.9.0)
-
  - Paper dcgan_code repository does not have MNIST data,
  - Download MNIST data from `https://github.com/Manuel4131/GoMNIST/tree/master/data`, and change location in lib/config.py

References I

- S. Chintala, L. Metz, and A. Radford, Unsupervised representation learning with deep convolutional generative adversarial networks. 2016. [Online]. Available: arXiv:1511.06434v2 [cs.LG]
- S. Chintala, L. Metz, and A. Radford, dcgan_code. (2016, May). Available: https://github.com/Newmu/dcgan_code. [Accessed Nov. 19, 2018].
- Large Scale Scene Understanding Challenge (2017, July). Available: http://lsun.cs.princeton.edu/2017/. [Accessed Nov. 19, 2018].
- T Kim. (2018, August). Available: https://github.com/carpedm20/DCGAN-tensorflow. [Accessed Nov. 19, 2018].
- Y. Bengio, A. Courville,I. Goodfellow, M. Mirza, S. Ozair, J. Pouget-Abadie, D. Warde-Farley, B. Xu, Generative adversarial nets. 2014. [Online]. Available: arXiv:1406.2661v1 [stat.ML]

References II

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," Advances in Neural Information Processing Systems 26 (NIPS 2013), 2013. [Online] Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

Creodocs Limited, "Beamer Presentation," *latextemplates.com*, 2018. [Online], Available: http://www.latextemplates.com/templates/presentations/1/presentation_1.zip. [Accessed Nov. 10, 2018].

IEEE, Piscataway, NJ, USA. *IEEE Editorial Style Manual*. 2016. [Online]. Available: http://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE_Style_Manual.pdf, [Accessed Nov. 11, 2018].

Y. LeCun, "Unsupervised Representation Learning," 2017. Accessed on: Nov 11, 2018. [online]. Available: https://www.youtube.com/watch?v=ceD736_Fknc

F. Chollet, and J.J. Allaire. *Deep Learning with R*. Manning Publications 2018. [E-Book] Available: Safari E-Book.

References III

Y. Bengio, A. Courville, I. Goodfellow. (2016). "Chapter 20 Deep Generative Models," 2016. [Online] Available: https://www.deeplearningbook.org/contents/generative_models.html. [Accessed: Nov. 20, 2018].

A. Hindupur, "A list of all named GANs!" github.com, Sep. 30, 2018. [Online]. Available: https://github.com/hindupuravinash/the-gan-zoo. [Accessed Nov. 21, 2018].

L. Bottou, P. Haffner, Y. Bengio, Y. LeCun, "Gradient-Based Learning Applied to Document Recognition," Proc of the IEEE, Nov., 1998. [Online]. Available: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf. [Accessed Nov. 21, 2018].

V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," Jan 2018. [Online]. Available: Github, https://github.com/vdumoulin/conv_arithmetic. [Accessed November 21, 2018].

References IV

Stack Exchange user Andriys, "What are deconvolutional layers," datascience.stackexchange.com, answer written July 14 2017, edited Nov. 19, 2017. [Online]. Available: https://datascience.stackexchange.com/questions/6107/what-are-deconvolutional-layers. [Accessed Nov. 22, 2018].

The End