VGG Neural Networks: The Next Step After AlexNet

Jerry Wei Jul 3, 2019 · 4 min read

AlexNet came out in 2012 and was a revolutionary advancement; it improved on traditional Convolutional Neural Networks (CNNs) and became one of the best models for image classification… until VGG came out.

. . .

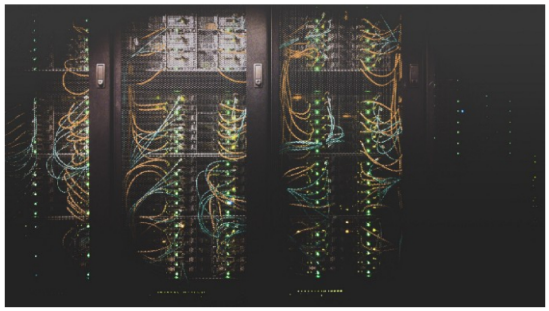

Photo by Taylor Vick on Unsplash

AlexNet. When AlexNet was published, it easily won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) and proved itself to be one of the most capable models for object-detection out there. Its key features include using ReLU instead of the tanh function, optimization for multiple GPUs, and overlapping pooling. It addressed overfitting by using data augmentation and dropout. So what was wrong with AlexNet? Well nothing was, say, particularly "wrong" with it. People just wanted even more accurate models.

The Dataset. The general baseline for image recognition is ImageNet, a dataset that consists of more than 15 million images labeled with more than 22 thousand classes. Made through web-scraping images and crowd-sourcing human labelers, ImageNet even hosts its own competition: the previously mentioned ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). Researchers from around the world are challenged to innovate methodology that yields the lowest top-1 and top-5 error rates (top-5 error rate would be the percent of images where the correct label is not one of the model's five most likely labels). The competition gives out a 1,000 class training set of 1.2 million images, a validation set of 50 thousand images, and a test set of 150 thousand images; data is plentiful. AlexNet won this competition in 2012, and models based off of its design won the competition in 2013.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					

FC-4096
FC-1000
soft-max

Configurations of VGG; depth increases from left to right and the added layers are bolded. The convolutional layer parameters are denoted as "conv<receptive field size> — <number of channels>". Image credits to Simonyan and Zisserman, the original authors of the VGG paper.

VGG Neural Networks. While previous derivatives of AlexNet focused on smaller window sizes and strides in the first convolutional layer, VGG addresses another very important aspect of CNNs: depth. Let's go over the architecture of VGG:

- **Input.** VGG takes in a 224x224 pixel RGB image. For the ImageNet competition, the authors cropped out the center 224x224 patch in each image to keep the input image size consistent.
- **Convolutional Layers.** The convolutional layers in VGG use a very small receptive field (3x3, the smallest possible size that still captures left/right and up/down). There are also 1x1 convolution filters which act as a linear transformation of the input, which is followed by a ReLU unit. The convolution stride is fixed to 1 pixel so that the spatial resolution is preserved after convolution.
- **Fully-Connected Layers.** VGG has three fully-connected layers: the first two have 4096 channels each and the third has 1000 channels, 1 for each class.
- **Hidden Layers**. All of VGG's hidden layers use ReLU (a huge innovation from AlexNet that cut training time). VGG does not generally use Local Response Normalization (LRN), as LRN increases memory consumption and training time with no particular increase in accuracy.

The Difference. VGG, while based off of AlexNet, has several differences that separates it from other competing models:

- Instead of using large receptive fields like AlexNet (11x11 with a stride of 4), VGG uses very small receptive fields (3x3 with a stride of 1). Because there are now three ReLU units instead of just one, the decision function is more discriminative. There are also fewer parameters (27 times the number of channels instead of AlexNet's 49 times the number of channels).
- VGG incorporates 1x1 convolutional layers to make the decision function more non-linear without changing the receptive fields.
- The small-size convolution filters allows VGG to have a large number of weight layers; of course, more layers leads to improved performance. This isn't an uncommon feature, though. GoogLeNet, another model that uses deep CNNs and small convolution filters, was also showed up in the 2014 ImageNet competition.

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**

Performance of VGG at multiple test scales. Image credits to Simonyan and Zisserman, the original authors of the VGG paper.

Results. On a single test scale, VGG achieved a top-1 error of 25.5% and a top-5 error of 8.0%. At multiple test scales, VGG got a top-1 error of 24.8% and a top-5 error of 7.5%. VGG also achieved second place in the 2014 ImageNet competition with its top-5 error of 7.3%, which they decreased to 6.8% after the submission.

Now what? VGG is an innovative object-recognition model that supports up to 19 layers. Built as a deep CNN, VGG also outperforms baselines on many tasks and datasets outside of ImageNet. VGG is now still one of the most used image-recognition architectures.

I've attached some further resources below that may be interesting

- Original Paper
- Blog Post on AlexNet
- Wikipedia Article on CNNs

miss. Take a look.

Your email

Get this newsletter

By signing up, you will create a Medium account if you don't already have one. Review our Privacy Policy for more information about our privacy practices.

209

Machine Learning | Deep Learning | Artificial Intelligence | Data Science | Coding

More from Towards Data Science

Follow

Your home for data science. A Medium publication sharing concepts, ideas and codes.

Salim Chemlal · Jul 3, 2019 ★

A Comprehensive State-Of-The-Art Image Recognition Tutorial

From the basics, a fast multi-class image recognition using fastai and PyTorch

Photo by Matt Noble on Unsplash

This tutorial was adapted from Fastai DL 2019 Lessons with many of my additions and clarifications. I hope you find it helpful.

Following this tutorial, you will be able to build and train an Image Recognizer on any image dataset of your choice, with a good understanding of the underlying model architecture and training process.

This tutorial covers:
1. Data Extraction
2. Data Visualization
3. Model Training: CNNs, ResNets, transfer learning
4. Results Interpretation
5. Freezing & Unfreezing of model layers
6. …

Read more · 16 min read

152

Lukas Frei · Jul 3, 2019 ★

Speed Up Your Python Code with Cython

And spend less time waiting in front of your screen

Picture from Unsplash

Introduction

If you have ever coded in Python, you have probably spent more time waiting for certain code blocks to execute than you would like. While there are ways to make your code more efficient, it will most likely still be slower than C code, for instance. This boils down mainly to the fact that Python is a dynamic programming language and moves many things to runtime that C takes care of during compilation.

Nevertheless, if you, like me, enjoy coding in Python and still want to speed up your code you could consider using **Cython**. While Cython itself is a…

Read more · 5 min read

382 1

Rene Draschwandtner · Jul 3, 2019 ★

JupyterLab — A Next Gen Python Data Science IDE

An article to nudge Python Data Scientists towards JupyterLab.

While working on data science projects with Python, you probably asked yourself which IDE serves best your needs on data exploration, cleaning, preparing, modeling, evaluation and deployment. You'd probably also have done some research on Google, scanned through various pages titled like 'Top Python Data Science IDE' and started desperately realizing that none of the mentioned products combines a seamless look and feel for all necessary implementation steps. Ultimately you turned back to your well known, but yet separated set of tools. The good news is, there is a very promising project waiting just to be released as 1.0 version…

Read more · 7 min read

283 2

Yan Gobeil · Jul 3, 2019 ★

Generating Pokémon names using RNNs

An example of character level word generation

A while ago when I was finishing the Deep Learning specializaion on Coursera I decided to try to implement some things that I learned, As everyone who took theses courses know the theory is incredible but the practice is pretty much just a pre-made code with some blanks to fill. I think it is not a very good way to learn so I wanted to code it all from scratch. The example that attracted me the most was generating words that look like the data the model is given. The course used dinosaur names and I decided to use Pokémon…

Read more · 7 min read

76 1

Puneet Sharma · Jul 3, 2019 ★

Beginning with R — The uncharted territory

Coming from a non-programming background and python being the first exposure to programming and data analysis, trying to get my hands dirty in R seemed pretty daunting at first. R at times can feel a bit peculiar and unique since it is based on the premise of doing data analysis and statistics rather than software programming which is the case with python. But as I push myself and try to learn the many quirks and leverages of R over python, it sort of gives a different perspective of doing data analysis. …

Read more · 7 min read

2

Read more from Towards Data Science

More From Medium

Getting to know probability distributions
Cassie Kozyrkov in Towards Data Science

Semi-Automated Exploratory Data Analysis (EDA) in Python
Destin Gong in Towards Data Science

Jupyter: Get ready to ditch the IPython kernel
Dimitris Poulopoulos in Towards Data Science

Import all Python libraries in one line of code
Satyam Kumar in Towards Data Science

Four Deep Learning Papers to Read in March 2021
Robert Lange in Towards Data Science

The flawless pipes of Python/ Pandas
Dr. Gregor Scheithauer in Towards Data Science

How to Boost Pandas Functions with Python Dictionaries
Soner Yıldırım in Towards Data Science

Read this before you write your next SQL query.
Robert Yi in Towards Data Science