Using Convolutional Neural Networks to Analyze Function Properties from Images

Yoad Lewenberg
The Hebrew University of Jerusalem, Israel

Yoram Bachrach
Microsoft Research Cambridge, UK

Ian Kash
Microsoft Research Cambridge, UK

Peter Key
Microsoft Research Cambridge, UK

Abstract

We propose a system for determining properties of mathematical functions given an image of their graph representation. We demonstrate our approach for two-dimensional graphs (curves of single variable functions) and three-dimensional graphs (surfaces of two variable functions), studying the properties of convexity and symmetry. Our method uses a Convolutional Neural Network which classifies functions according to these properties, without using any hand-crafted features. We propose algorithms for randomly constructing functions with convexity or symmetry properties, and use the images generated by these algorithms to train our network. Our system achieves a high accuracy on this task, even for functions where humans find it difficult to determine the function's properties from its image.

Introduction

Consider the case when an experiment has been performed for determining the influence of a single or multiple parameters on an outcome of interest. For example, we wish to determine the impact of parameters of an algorithm on its performance, or the influence of various parameters of a simulation model on the result. In many cases one wants to determine whether the relation between the parameters and outcome exhibits various properties, such as convexity or symmetry. One may present the results of the experiment in a figure, however any additional parameter value would result in a different figure. It would be very difficult, costly and time consuming to manually examine each figure to determine whether it exhibits the desired properties, so an automated method of achieving this is very desired.

The recent surge in research on neural networks has uncovered many domains where they have excellent performance. Approaches based on Deep Convolutional Neural Networks (CNNs) (LeCun et al. 1998; Simard, Steinkraus, and Platt 2003) do well on many image and video classification tasks (Krizhevsky, Sutskever, and Hinton 2012; Ciresan, Meier, and Schmidhuber 2012; Karpathy et al. 2014), without relying on hand-crafted features.

In contrast to the above research, which focuses on images of real-world objects, we demonstrate a CNN based approach to classifying *functions*, abstract mathematical objects, given their representation in the form on an image.

We provide algorithms for generating random functions with properties of convexity and generalized symmetry, and generate a large image dataset of function images, consisting of 4,000 single-variable functions and 10,000 two-variable functions. We then use the dataset to train a CNN-based classifier and show it achieves a high performance.

To generate **two-dimensional graphs**, we first draw n random numbers from some distribution, which are used as the values of the function at n points. Once we set the values of the function at these points, we use a linear interpolation to define the function on the rest of a interval $[a, b]$. To make the function smoother, we integrate the obtained function twice.

We consider two mathematical properties:

Convexity: A twice-differentiable function $f : [a, b] \rightarrow \mathbb{R}$ is convex iff for all $x \in [a, b]$ it holds that $f''(x) \geq 0$. By drawing the values of the second derivative from a positive support distribution (e.g. $\mathcal{U}(\alpha, \beta)$ where $0 < \alpha < \beta$), we are guarantied that f is convex.

Generalized Symmetry: We consider symmetric functions which are stretching or translated. Let f_{even} and f_{odd} be random generated even and odd functions. For some $\theta \in (-\pi, \pi)$ and a function f, the *stretching* of f by θ is $\overline{f_\theta}(x) = f(x) + x \cdot \tan(\theta)$. Finally, given an even function f_{even} we define $f^*_{even,\theta}$ as

$$f^*_{even,\theta}(x) = \begin{cases} f_{even}(x) & x \leq 0 \\ \overline{f_{even}}_\theta(x) & x \geq 0 \end{cases}$$

$f^*_{odd,\theta}$ is defined similarly.

Our process for generating **two-variable functions** $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is more elaborate. We cannot simply randomly draw values multiple points and perform linear interpolation and integration, as the numerical integration is sensitive to the *direction of integration*, resulting in "unnatural" functions. Instead, we randomly draw values for n^2 points from a distribution $\mathcal{D}$. Next, we repeatedly smooth the function by applying Gaussian kernel to achieve a more natural function, and add noise to the function to increase the variety among the returned functions.

We say that a function $f : [-a, a] \times [-a, a] \rightarrow \mathbb{R}$ is x-*even* if $\forall x, y \in [-a, a] : f(x, y) = f(-x, y)$, and x-*odd*

Figure 1: Convex (left) and random (right) functions

Figure 2: Stretched symmetric (left) and random (right) functions

if $\forall x, y \in [-a, a] : f(x, y) = -f(-x, y) + 2 \cdot f(0, y)$. In contrast to the single variable case, only requiring that $f(x, y) = -f(-x, y)$ implies that $\forall y \in [-a, a] : f(0, y) = 0$; such functions are trivial to identify, so we use a more general definition. Generating x-even and x-odd function is done by generating the values of the function at $[-a, 0] \times [-a, a]$ and setting the values at $[0, a] \times [-a, a]$ accordingly. Given an angle θ and a function f, the rotation of f by θ is:

$$f_\theta(x, y) = f(x\cos(\theta) - y\sin(\theta), x\sin(\theta) + y\cos(\theta))$$

CNN Based Function Classification

Deep convolutional neural networks achieve good results for real-world images, but how well do they perform on function graphs, which are image representation of abstract mathematical objects? We used a prominent CNN structure (Krizhevsky, Sutskever, and Hinton 2012), sometimes referred to as AlexNet, in our experiments. For training our CNN, we used the Caffe framework (Jia et al. 2014), a popular framework for inference with CNNs.

Empirical Results: We used an 80%-20% train-test partitioning on an image subset to determine the performance of the CNN in classifying the images. We maintained a balanced partitioning in creating the data, where half the generated images exhibit a property (convexity or generalized-symmetry) and the other half does not exhibit the property.

For **single-variable functions**, we generated 2,000 images of convex functions and 2,000 images of non-convex functions. The convex functions were generated by drawing the second derivative from either $\mathcal{U}(0, 1)$ or $\mathcal{N}(\frac{1}{2}, \frac{1}{4}) \cdot \in [0, 1]$. Examples of convex and non-convex function are given in Figure 1. For this simple case the network achieved a very high accuracy of 99.36%. We also examined generalized symmetry where given an image of $f : [-a, a] :\to \mathbb{R}$, we check whether there exists $g : [0, a] \to \mathbb{R}$ and $\theta \in (-\pi, \pi)$ such that $f = g^*_{even,\theta}$ or $f = g^*_{odd,\theta}$. We generated a dataset of 2,000 function images (with exactly 1,000 of them stretched symmetric ones). Figure 2 shows examples of stretched-symmetric and random functions. In this case, the CNN achieved an accuracy of 93.76%, which is high considering that this problem is also difficult for humans.

For **two-variable functions** we used a heatmap image representation, and trained a network to identify generalized symmetry. The classifier checks whether for the given

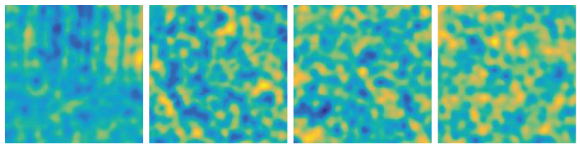

Figure 3: Rotated odd and even function (left) and random (right) function

Trait	Accuracy
Convexity (single variable)	99.36%
Generalized Symmetry (single variable)	93.76%
Generalized Symmetry (two variable)	81.84%

Table 1: Accuracy of CNN based function classification

$f : [-a', a'] \times [-a', a'] :\to \mathbb{R}$, there exist $g : [-a, a] \times [-a, a] \to \mathbb{R}$ and $\theta \in (-\pi, \pi)$ such that f is either x-even or x-odd and $f = g_\theta$. We generated 10,000 two-variable function images (with exacly 5,000 generalized-symmetric ones). Examples of rotated odd and even functions and random (non-symmetric) functions are given in Figure 3. The CNN classifier in achieved an accuracy of 81.84%. This accuracy is high, but this clearly indicates that this is a more difficult for the CNN to handle than the single-variable case.

Discussion and Conclusions

We presented a CNN based architecture to identify properties of mathematical functions. The CNN is trained on a dataset of images constructed using our algorithms for generating random functions exhibiting the studied properties. Our empirical analysis, summarized in Table 1, shows great promise for CNNs in studying abstract objects, but also indicates that in some cases the predictions are imperfect.

References

Ciresan, D.; Meier, U.; and Schmidhuber, J. 2012. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 3642–3649. IEEE.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 675–678. ACM.

Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; and Fei-Fei, L. 2014. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 1725–1732. IEEE.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Simard, P. Y.; Steinkraus, D.; and Platt, J. C. 2003. Best practices for convolutional neural networks applied to visual document analysis. In *null*, 958. IEEE.