***Neural Network Graph Classifier of Probability Distributions***

Patrick Spohr

589909

20.02.2024

Master’s Program

Finanzmathematik, Aktuarwissenschaften und Risikomanagement

Department

Informatik, Kommunikation und Wirtschaft

Course

Deep Learning Seminar

ProfessorDr. Alla Petukhina

Table of Contents

[Motivation 1](#_Toc158018789)

[Graph Classifier 1](#_Toc158018790)

[Objectives 2](#_Toc158018791)

[Main Goal 2](#_Toc158018792)

[Literature Review 2](#_Toc158018793)

[Hurdles to Overcome 2](#_Toc158018794)

[Data 3](#_Toc158018795)

[CIFAR-10 Images 3](#_Toc158018796)

[Scraped Graphs 3](#_Toc158018797)

[Generated Graphs 3](#_Toc158018798)

[Methods 4](#_Toc158018799)

[Overview 4](#_Toc158018800)

[Feed-Forward Neural Networks 4](#_Toc158018801)

[Convolutional Neural Networks 4](#_Toc158018802)

[Rectified Linear Units 4](#_Toc158018803)

[Adam Optimizer 4](#_Toc158018804)

[Python, Keras, and Libraries 4](#_Toc158018805)

[Simple Graph Classifiers 4](#_Toc158018806)

[Distribution Graph Classifiers 5](#_Toc158018807)

[Results 6](#_Toc158018808)

[Simple Graph Classifiers 6](#_Toc158018809)

[Distribution Graph Classifiers 6](#_Toc158018810)

[Conclusions and Further Research 6](#_Toc158018811)

[References 7](#_Toc158018812)

# Motivation

## Graph Classifier

In the age of information, those that can successfully and quickly interpret large amounts of data will have an advantage. The field where converting data into visual aids is called descriptive statistics and the most common method to do this is with graphs. This follows the adage, “A picture says a thousand words.” With just a glance someone can understand and make connections about the data without needing to scan through a table of numbers. There are many different kinds of graphs: bar charts, pie, charts, line graphs, scatter plots, and radar charts to name a few. Within those charts they can be divided up into more types, like stacked bar charts or donut charts. From science to business charts are the main way key parts of the data are shown to other researchers, customers, or investors.

The McDonald’s 2022 Annual Report \cite{mcdonalds2022} has a total of 73 pages and 8 graphs. Each graph displays important information to the investor, like the figure which shows total revenues by segment on page 13. One can see that from 2020 to 2022 the revenue from international operated markets rose and fell, but overall, the revenue share from the different segments have remained static. Wouldn’t it be beneficial to have access to these graphs immediately without needing to browse through the whole report? The ability to identify and then classify graphs instantly in the report without needing to read through it and snip out the graphs manually would save time and effort. The overall flow of this system would have three parts: Identify, Locate, and Organize. Identify would simply see if it is a graph or not and classify it into a type of graph. Once that is finished then the graphs can be located on a page and extracted. The last part would organize the charts for easy viewing. My focus is on the Identify part of this system and will be limited to building a model that answers two questions: Is this image a graph? What kind of graph is it? Regarding the second question, I’ll limit the scope even further because of the multitude of different graphs. The focus will be on classifying the probability distribution in a graph. This was chosen because of the novelty of the problem – no similar projects were found – and because of my current study of stochastics.

## Literature Review

I’ll need to briefly mention the literature I tried to find and the literature that guided this project. As mentioned earlier, this project is quite novel in its end goal since I did not find any research that specifically tried to classify graphs into probability distributions. There were projects that classified graphs into bar charts, pie charts, etc., but even then, those projects were not part of published research. When one tries to search for a neural network graph classifier or a similar pattern the first results are about Graph Neural Networks or GNNs, which are unrelated. As for image classification itself, there are a plethora of papers and resources. I personally learned a lot from a Stanford University course \cite{cs231n2023} and \textit{An Introduction to Convolutional Neural Networks} from O’Shea and Nash \cite{oshea2015}.

# Objectives

## Main Goal

As stated previously, my focus is on the identification and classification of graph images. This can be accomplished in two types of models where the first one classifies a graph image from a natural image (non-graph) and the second one classifies a graph image into its probability distribution. There are many probability distributions to choose from, but I selected the normal distribution, log-normal distribution, exponential distribution, and uniform distribution. I primarily choose these distributions because they are common and relatively distinct from each other. The reasoning behind this two-model process is to be able to produce a baseline result where it is known that graphs can be distinguishable from natural images. I predicted early on that graphs would be quite distinguishable from natural images and the accuracy of this model would be high, but as it will be discussed later, there is a second component to this problem specific to my data. These hurdles will be expanded upon in the next section.

In summary, the workflow of the project was split into four steps: get data, create graph classifiers, create distribution classifiers, and evaluate the models. Getting data had the potential to be the hardest step of this project because of the novelty of its goals to classify graph images into their probability distributions and I was partly correct. The next major obstacle was choosing the right model to build the machine learning models and the method in which each image would be classified. In the “Methods” section, I will go in depth to explain the process of how the models were built and the images classified.

## Hurdles to Overcome

As image classification is a vibrant and bustling field, there is no shortage of natural images of various types of things – cars, airplanes, horses, cats, dogs, clothes, etc. One of the most common datasets for natural images is the CIFAR-10 \cite{ krizhevsky2009} as it is well labeled and contains various images. The task of finding graph images was much more difficult. However, there was a suitable dataset of graphs on Kaggle from SunEdition \cite{sunedition2021} which met my requirements to build the first model, which simply classifies an image as a graph or not a graph. Unfortunately, this dataset does not contain classes with specific probability distributions and cannot be used for the second model – classifying the four chosen distributions in graphs.

The biggest issue was getting a dataset with the normal distribution (norm), log-normal (lognorm) distribution, exponential distribution (exp), and uniform distribution (unif) as its four classes. As I saw it, this could be solved in two possible ways. First, I could have scraped the graphs from various online sources. This leads to a major issue of not only where to scrape the graphs but how to label them time efficiently. The dataset from SunEdition on Kaggle has some major drawbacks when wanting to classify the images into more specific classes. These would be apparent in my own scraped graphs as well. Many graphs have more than one graph type in the image. That makes the correct classification impossible. Additionally, the dataset includes ‘graph-like’ images which are depictions of graphs but have no numerical value and are closer to art or natural images. I could also mention that even in a graph type, like bar charts, there are subcategories – stacked bar charts, horizontal bar charts, vertical bar charts, etc. Were I try to scrape the graphs, then I would lose control of the variables and spend most of my time labeling each graph which would be impossible for the timeframe of the project. This leads to the second solution to my problem. I could generate the graphs.

Generating graphs with the aid of a graphing library was the most time effective solution to acquire thousands of graphs with the four distributions in already-labeled classes. This is not the ideal solution since it biases the dataset towards the design conventions of the chosen graphing library and it lacks the variety seen online, but it can be highly randomized to mimic the potential graphs in various sources and the variables can be controlled to ensure all graphs have the potential to be classified. The most important consideration in this method is the time saved by generating the graphs. The daunting task of scraping and labeling 10,000 graphs can now be minimized into writing a script using a graphing library. The specifics of the three datasets used will be further explained in the subsequent section.

# Data

## CIFAR-10 Images

The CIFAR in CIFAR-10 stands for the Canadian Institute for Advanced Research and the 10 represents the 10 classes in the dataset: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck \cite{ krizhevsky2009}. The data comes in six batches of 10,000 images with a total of 60,000 images. The 10,000 images in each batch are made up of 10 classes with 1,000 images per class. The images are colored and 32x32, which means they have a shape of (32,32,3). The first two values represent the dimensions and the 3 stands for the red, green, and blue channels. This data structure is known as a tensor. However, the original data type is a flat NumPy array of (3072,), so each image needed to be reshaped to fit in the model with the graphs.

The updated dataset has 10,000 images (first-batch) that are all 32x32 and in JPG format. Also, the 10 classes have been combined into one natural image class. I have only selected 10,000 because there are only 7,753 suitable graphs in the scraped graph dataset. Had I selected more that would have led to an imbalanced dataset in the model. In my project, the CIFAR-10 dataset will be labeled as CIFAR or natural.

## Scraped Graphs

Just like the CIFAR-10 dataset, the scraped dataset \cite{sunedition2021} needed to get updated to work with the simple graph classifier model as well. The original dataset has 15,786 images and 8 classes. The classes are just image, bar chart, diagram, flow chart, graph, growth chart, pie chart, table. As you can see ‘just image’ refers to non-graphs or natural images, so it needed to be removed. Other classes like table, flow chart, and growth chart were not necessary and were removed as well.

In the end, four classes were kept, bar chart, diagram, graph, and pie chart, so the total images now used were 7,753. The images came in various dimensions and were resized to the natural images 32x32 and converted to JPG format. I’ll note here that JPG was chosen because it only has 3 color channels and no alpha channel. If PNG is used then it could have a shape of (32,32,4) which adds unnecessary parameters for the scope of my objects.

## Generated Graphs

Lastly, 8000 graphs were generated in a 460x345 dimension and JPG format. The graphing library used was Matplotlib in Python, but I’ll discuss the programs and libraries in more detail in the “Methods” section. There were 4 classes – norm, lognorm, exp, and unif – with 2,000 images each. The only thing updated were the dimensions which included 32x32, 115x86, and 153x115. For clarity, these dimensions follow the Windows format of width by height.

As for the design, almost every part of the graph was randomized. The color of the histogram bars, line, figure color (outside the box), face color (inside the box where the plot is), and text labels were changed, but the figure color and face color were biased to white by 20 percent because most graphs tend to be white. There was also some changes to ensure that the histogram bars, line, and face don’t have the same color. Regarding the histogram and line, both were visible or only one was visible. The line style was solid or a dash. Each graph had a title, y label, and x label which consisted of lorum ipsum, which is Latin gibberish and is commonly used in web design. The idea behind this was to avoid a bias to English if someone used this model to predict graphs with different languages. The text also had randomized size. Lastly, the density function parameters were randomized to ensure many variations of each probability distribution.

Overall, the graphs looked like something that could be found in a report, book, or online. The biggest issue with the generated graphs were the so-called ‘duds’ which looked blank because the color of the face, histogram, and line were too similar. This wasn’t a large amount of images, so I decided to keep them instead of manually deleting all of them. I reasoned that keeping the original generated the same is better for reproducibility, rather than someone, me, deciding if a graph is a dud or not since some graphs are on the border of being a dud or not. Ultimately, the seed of the graph generator remains the same and the dataset can always be recreated and used in the model.

# Methods

## Overview

## Feed-Forward Neural Networks

## Convolutional Neural Networks

## Rectified Linear Units

## Adam Optimizer

## Python, Keras, and Libraries

## Simple Graph Classifiers

## Distribution Graph Classifiers

# Results

## Simple Graph Classifiers

## Distribution Graph Classifiers

## Conclusions and Further Research

# References