Universität Passau
Fakultät für Informatik und Mathematik

Classification Of Visualization In Scientific Literature

Masterarbeit zur Erlangung des akademischen Grades
Master of Science (M.Sc.)

Lehrstuhl für Intelligent Systems und Lehrstuhl für Data Science
der Fakultät für Informatik und Mathematik
der Universität Passau

Name:	Arnold Azeem
Matrikelnummer:	79176
Fachbereich:	Informatik
Studiengang:	Master Informatik
Erstprüfer:	Prof. Dr. Christin Siefert
Zweitprüfer:	Prof. Dr. Michael Granitzer
Date:	June 11, 2018

0.1 Abstract

Distinct visualization techniques are used in scientific research publications to summarize large amount of data and also represent a variety of data. These visualizations help to communicate complex information and support the arguments being presented in the publication in a way that is easy to understand and follow. These figures tend to reveal trends, patterns or relations that might have otherwise been difficult to grasp using only text. It is therefore relevant that we extract the data from these visualizations since the extracted data can be used for validating the publication or presenting the data in another form for a different audience. In this context, classifying these visualizations is the initial step since, there is a variety of visualizations and each one is processed in a specific way. It is only after classification that extracting of raw data from these visualizations can be acquired for other tasks. This thesis presents an approach whereby real world data is used to create four types of plots (scatter plots, bar charts, line charts, and box-plots) and random plots also of the same kind from the Internet are added together and used to train and evaluate a CovNet model to be able to classify these plots.

Contents

List of Figures

List of Tables

1 Introduction

"A picture is worth a thousand words" even though a widely used phrase stands to be very true especially when complex data is visualized and presented in scientific research publications. Data is ever growing and sometimes complex, using figures and diagrams to interpret and represent this data cannot be undermined since, they provide a way to easily give insight into the research findings, which would have otherwise been more complex relying on only textual data. For this reason, there has been a growing interest in the chart analysis area and quite a number of techniques have been developed. I spite of this growing interest, there has been little groundbreaking results achieved due to different variations in appearance of plots [3]. For example, Manollis Savva, et a [4] proposed a model to classify charts using extracted low-level features and textual features. After extracting the features, a Support Vector Machines (SVMs) classifier is used for the classification step. This method was limited since most charts contain the same type of features like axes, grid lines, and legends. In V. Shiv Naga Prasad's work [5] classification was based on using features based on the shape and spatial relationships of their primitives. This work was limited due to the inconstancy in which data in most charts can be depicted. The process of extracting data already visualized as figures can be done relatively easier manually but becomes more complicated if done automatically. This process can be divided into two main steps [4]. The first step which our work focuses on, classifying the chart and the second step which involves extracting the data from the classified chart. To achieve the classification step, this paper presents an approach where charts are created with real-world datasets, different plotting programs (Python, Matlab, R, and Java) and different libraries supported by these plotting programs were used together with downloaded chart images from the Internet. We then use these images as input to Convolutional Neural Network (CovNet). CovNet was used instead of primitive approaches because it has achieved ground breaking results in the area of image classification [6]. Our model can identify four classes of plots namely Box-plots, Line Charts, Scatter-plots and, Bar-charts. The other parts of this paper are organized in the follows: In the next Sections, we present the motivation behind this work. Other works related to this, our pro-

posed method is described, Experimental evaluation and results are reported, and finally, the conclusion and the way to approach this work in the future.

1.1 MOTIVATION

Complex data is better explained in scientific papers with the aid of visualizations. These plots present complex data in an easy to understand format as compared to the textual representation. The data which these visualizations contain when extracted play an important role in events where another researcher wants to verify the work of the publisher. The extracted data can also be used to develop other visualizations in situations where the paper needs to be presented to a different audience with a distinct background as opposed to the audience for which the visualizations were created initially. Another reason for extracting this data is that, when comparing two plots, looking at the raw data, gives us more insight and consequently helps in better decision making as compared to just looking at the figures. Since each plot will be processed differently to extract the raw data, it is very relevant that we can distinguish one plot from another and this is the main aim of this thesis.

1.2 OBJECTIVE

The purpose of this thesis is to answer the question:

> How Well Can We Classify the Four Different Types of Plots (Line-Charts, Bar-Charts, Scatter-plots, and Box-plots) in Scientific Literature?

In this work we focus on only four plots. These plots are scatter-plots, bar-charts, line-charts, and box-plots. Figure 1.1 shows the vision of this work, The first part of the diagram involves, extracting or obtaining the four different types of plots mentioned earlier, after which we then label our plots and train a neural network model to be able to classify with high accuracy any of the four plots if shown to our model, then finally the raw data can be extracted from the detected plot. But this thesis mainly focuses on the red dotted lines shown below in the diagram which involves, gathering a dataset of chart images, labeling them, training the model and classifying the plots.

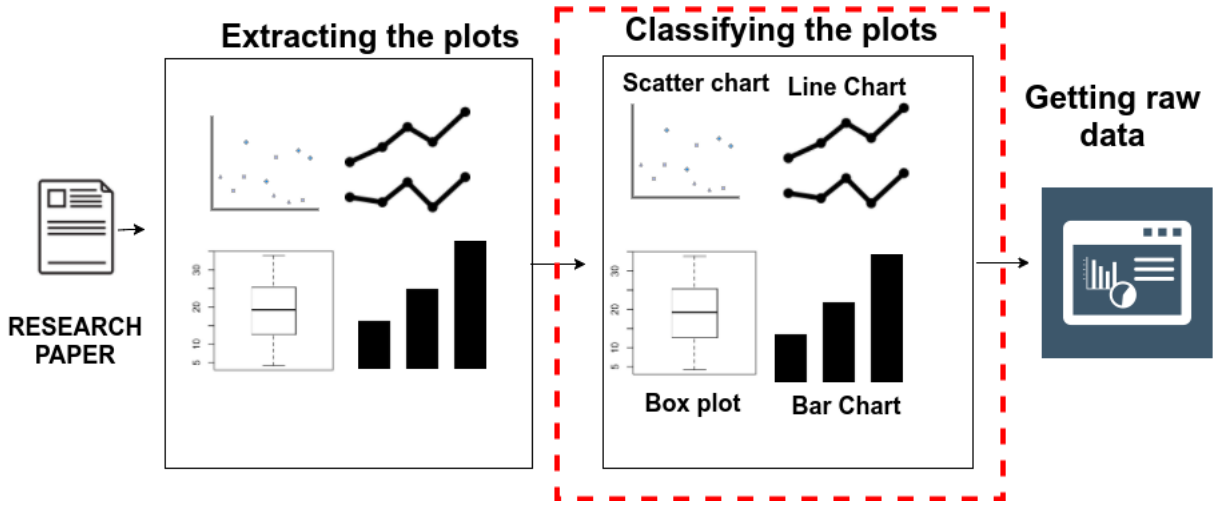


Figure 1.1: What we hope to achieve in the thesis

2 Background and Related Work

Classification involving charts images has been a topic of huge interest in recent times. This is due to the fact that, most scientific publications if not all, are embedded with these charts as a way of conveying complex research findings in more visualizable and easy to understand format. As a result of this popularity, chart images have gained, different techniques have over time been used for classification of charts and these techniques have evolved over the years. The techniques have been inspired by techniques from image processing, raster to vector conversion and layout analysis. These techniques can be put into two categories; Model-Based Approaches and Machine Learning Based Approaches [6]. In the next section, the methods and techniques used in this thesis, and techniques that inspired us to choose these techniques are explained in details.

2.1 Model Based Approaches

The following is written with inspiration from Boyle et al. work [7]. Numerous chart images even of the same type, are very different in sense of their context structure. This is as a result of the variability of positions of the context structure of a chart. There is exists no real standard for all charts to follow, content like; legends, axes, grids etc., have no fixed positions and sometimes are not even present in some charts. As a result of this non standardization of charts, classifying becomes a very difficult task. As a results this approach tries to uses the structure of the charts. The model-based approach is divided into two steps. Firstly, a number of predefined object classes are created with abstract models and then an image is matched with the created model. For example, a model for a 2D/3D pie chart consists of line segments (radii and circular/elliptical arcs), and these line segments are used to create the model, however, just using these line segments wont be enough, so some constraints are introduced into the model, example of constraints of a pie chart include; the center of the pie chart is where all radii meet; all 2D pie charts have radii of equal length; arcs mainly form parts of the same 2D pie chart circle, or if its a 3D then the arcs form part of the ellipse. After this a goodness-of-fit is introduced to help measure the discrepancy between the image and the condi-

tions for the model. For example, a goodness-of-fit of a pie chart is as follows: difference in length of radii; difference in how the arcs are curved; distance between a center of a circle and that of a radii. When a chart is presented for classification, the edges are detected, thinned, linked, vectorized, extracted and compared to the various edge models created and the best match is selected. After this, the good-to-fit criteria is used to measure the difference between the edges of the model and the image. Finally, a voting is performed base on the value of the good-to-fit results. The model-based approach has some drawbacks though. The main drawback is that human intervention is needed to develop the various models. This could be a cumbersome and time consuming procedure. Due to the limitation of this approach a machine learning approach was introduced to handle some of the limited of this approach.

2.2 Machine Learning Based Approaches

This method was developed in recent times, and most works in area of text mining and image processing employ this approach. This method involves using handcrafted features extracted from the charts for classification task. In Zhou and Tan's [8] work, features like legend, x-y-axis title, chart title, and values of the bar of a bar-chart were extracted for purposes of classifying a bar-chart. In Shao and Futrelle's [9] work, graphical elements of the charts like colors,tick marks on an axis and data point markers are the handcrafted feature used for classification. Another instance where handcrafted features were used was in Inokuchi et al. work [10]. In this work regularly appearing substructures of chart are extracted and used as features for classification. After the feature acquisition step, the obtained features are then represented in vector form for the classification step. The features are stored in a matrix vector-like structure. Algorithm 1 shows an example of the structure in which the features are stored. This structure has multiple columns containing the features and one column indicating the target or label of the chart. This vector-like structure is then fed as input into a machine learning technique for the model training step. In Karthikeyani and Nagarajan's paper [2] after the features were obtained SVM Classification [1], MLP Classification [2], and KNN Classification[3] were used to create a function that maps the set of features to a predefined label. This technique achieves very good results but, the drawback is the over reliance on handcrafted features. This drawback makes it difficult when large amount of data consisting of a variable context structure is involved, and this situation is evident in most cases. Recently, however CovNets

[1]http://www.statsoft.com/Textbook/Support-Vector-Machines
[2]http://www.iro.umontreal.ca/ bengioy/ift6266/H12/html/mlp_en.html
[3]https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

(this abbreviation stands for Convolutional Neural Networks and will be used a lot through this work) [4], a machine learning technique which removes the drawback of using handcrafted features by learning important features anywhere in the image by itself. CovNets performance on image classification keeps on improving and in recent times, results are being compared to human performance levels.
In the rest of this chapter, three related works that performed chart image classification, with very unique methods and also achieved good results are described in details.

Machine Learning Classification Algorithms to Recognize Chart Types in Portable Document Format(PDF) Files

Karthikeyani and Nagarajan's work [2] focuses on classifying charts found in pdf files. The steps employed for the classification task involves extracting texture features from the charts and then, using a machine learning technique for the classifying phase. The dataset consists of images extracted from various pdf's. The dataset is made up of 155, 256*256 RGB images in total. Table 2.1 shows the details of the dataset used. The technique used involved extracting handcrafted features from the chart image and then, using these features as input for a model to be trained and tested. For the feature extraction phase, Gray Level Co-Occurrence Matrix (GLCM) is employed. GLCM is a technique which uses co-occurrence matrix to extract texture features of an image with the use of statistical equations. GLCM was used to extract eleven features, these included: area, median, minimum and maximum intensity, contrast, homogeneity, energy, entropy, mean, variance, standard deviation and correlation. These extracted features are correlated to the pixels of the image. These extracted features are then stored using a 2-dimensional matrix vector data structure. This structure has thirteen (13) columns and 'x' rows, where x is the size of the dataset. The features are stored in twelve (12) columns and the label is stored in the thirteenth column. The features and labels were stored in the below structure 1.

Chart Type	No of Charts	Chart Type	No of Charts
2D Bar chart	40	Doughnut 2D	7
3D Bar chart	16	Doughnut 3D	11
2D Pie chart	13	Line	35
3D Pie Chart	20	Mixed Chart	13

Table 2.1: Details on Dataset from [2]

[4]https://deeplearning4j.org/convolutionalnetwork

Algorithm 1 This is the structure used to store the features

```
Struct FeatureVector {
float feature1; float feature2;
float feature3; float feature4;
float feature5; float feature6;
float feature7; float feature8;
float feature9; float feature10;
float feature11; float feature12;
int target;
}
```

This vector like structure is then fed as input into three classifiers SVM, MLP neural network, and K-Nearest Neighbor. These classifiers are then trained and a classification model is formed for the recognition step. After this, to see if the model works well a test set consisting of new records is fed into the model for the model to predict their labels. Three metrics were used to check the performance of the model, these metrics are; error rate, classification accuracy and speed of classification. The error rates obtained were as followes; MLP 0.30, K-NN 0.22 and SVM 0.23. For the accuracy metrics were as follows; KNN (78.06%), MLP (69.68%) and SVM (76.77%) and finally the speed metric. The speed metric which sum of training and test time results are; MPL was the slowest with 8.38,followed by SVM with 0.31secs and KNN with 0.26secs. Even though these results were very good, the paper proposed extracting features which are related to shape and curves since these features will carry more unique information to distinguish charts.

Architecture proposal for data extraction of chart images using Convolutional Neural Network

Our work is inspired by De Freitas et al. [11], proposed a way to extract the wealth of information contained in different visualization techniques. The paper talks about two main stages of accomplishing this task. Firstly, classification of the charts is done since it allows a different variety of chart to be detected automatically allowing the next step, which is the extraction of data from the classified plots. The paper, however, focuses on the first step, classification of charts. In this paper, a Convolutional Neural Network is used for the classification task. The Convolutional neural network encapsulates the characterization and classification processes during its learning process, unlike other techniques. The dataset used for this task were searched for and downloaded from Google image search. Table 2.2 shows the chart types which were collected and the number of train and test sets which the respective charts were divided into.

For the classification, a variant of CovNet called LeNet-based CovNet

Chart Type	Test	Train
Area Chart	50	555
Bar Chart	50	657
Line Chart	50	489
Map	50	476
Pareto Chart	50	261
Pie Chart	50	361
Radar Chart	50	454
Scatter Chart	50	552
Table	44	236
Venn Diagram	48	304
Total	498	4345

Table 2.2: Number of Train and Test Dataset collected

model is used. The model was implemented using Tensorflow[5], LeNet-based CovNet has an architecture which is comprised of 3 convolutional layers, followed by a fully connected layer. The model is trained in a way that the dataset is divided into mini-batches, samples of fixed sizes(100) are selected and fed into the CovNet, as a result of this process the model becomes robust since it learns to generalize from the different min-batches which are fed into the model. Also, all the images are converted to JPG and resized to 224x224x3, that is, 224 pixels of height, 224 of width and 3 layers of output. The other parameters used were 1000 epochs and a learning rate of 0.003. The accuracy at the end of the training process was 70%.

Chart classification by combining deep convolutional networks and deep belief networks

In another paper by Liu et al. [3], a new approach was proposed for the process of chart classification. The process involves using CovNet to extract deep hidden features of charts and then deep belief networks then use the extracted features to predict the labels of the charts. Due to a difficulty in acquiring a large number of charts as training data, natural images where first used to train the model and later the model was fined tuned with just over 5,000 collected charts. The types of charts collected were pie charts, scatter charts, line charts, bar charts, and flowcharts. The architecture of the CovNet is made up of five Convolutional layers and two fully-connected layers and then an output layer.

[5]https://www.tensorflow.org/

The preprocessing steps for the images involve down-sampling them to 256 x 256 x 3, after which each is cropped to a size 227 x 227 from the center and its horizontal flip are extracted as the input of the CovNet, other parameters used for the CovNet include a learning rate that starts with 0.01 initially and is then decreased by a factor of 0.1 after every lOOk iterations, the weight decay parameter was set at 0.0005 and a dropout rate of 0.5. This results in an output of a 5-way softmax which produces the distribution over the 5 class labels and this is used as input for the deep belief network. The deep belief network architecture has three hidden layers, whose dimensions are 5000, 500 and 2000. This results in a softmax predicting the probability distribution over the 5 categories of charts as output. The training process was done with 4000 randomly selected images and the rest were used as test set. The accuracy of the model after the evaluation was 75.4%. Table 2.3 show the results after the training was done without deep belief networks but pre-trained with the natural images and finally the training done with only the chart dataset but with deep belief networks.

Chart	CovNets	CovNets+DBN without pre-training	CovNets+DBN
Bar Chart	75.6%	45.6%	74.2%
Flow Chart	88.3%	56.8%	91.3%
Line Chart	71.2%	22.3%	67.9%
Scatter Chart	69.8%	44.5%	84.2%
Pie Chart	58.1%	50.1%	59.4%
Ave. Accuracy	72.6%	43.9%	75.4%

Table 2.3: Comparing Results of Proposed Framework from (Liu et al. [3])

3 Approach

As seen in the previous chapter, different approaches and methods have been described for chart image recognition. Due to the complexity of this task, a CovNet approach is employed for our work. Reasons for selecting this approach include; automatic capturing, learning and extraction of features; parameter sharing which allows the model to learn a single set of weights once, rather than a different set of weights every time [12] and more importantly a performance accuracy nearing human capability. In spite of all the advantages of CovNets, they are very computationally expensive to apply on high-resolution images. Fortunately, current GPU capabilities with the help of good optimization techniques can handle this issue [1].

The specific contributions of this thesis are as follows: to create a dataset of chart images (scatter-plot, bar-chart,line-chart and box-plot) using real-world data. This chart image dataset tries to capture all variety of structure in the various chart images to be handled. The scripts, libraries, and parameters used in creating the chart images are readily available for the recreation of the dataset. The second contribution of this thesis involves training a CovNet model for recognizing these types of chart images.
The rest of this chapter describes into details how the dataset was created, it also describes the CovNet architecture that was used for the classification and evaluation tasks.

3.1 Dataset

The following paragraph is inspired by a blog written on what to look for in training data [13]. The saying 'Garbage in Garbage out' is a valid statement when it comes to creating a dataset for machine learning. The machine learning technique will learn from whatever data fed to it. So if a dataset of good quality is fed into the algorithm, then the model created will also be of good quality. The dataset creation stage is therefore a very important stage. In most approaches that worked on chart images classification, the dataset used consisted of chart images downloaded from Google and a few others were ob-

tained from extracting chart images from pdf's. This approach we believe is limited since, we don't know the programming languages, the libraries used and parameters used in creating these charts. This messing information is very relevant since it tells us how diverse our resultant dataset is. For example, how are we sure that all the charts that were downloaded from Google, were not only created in Python or Java?, and in such a case how well will a new chart created with Matlab or R be classified. For this reason the dataset comprised of image charts created with different libraries and in different programming languages. In the next sections, the various datasets and the languages used in plotting are described. To make the dataset as diverse as possible, charts created in each programing language used a different set of CSV files.

3.1.1 Dataset for Matlab

The Data used for creating the plots in Matlab were randomly chosen from Project Dataset [14], a free CSV data repository, DatPlot [15] and Plotly CSV repository in github [16]. The datasets are multidimensional and compiled from normal day to day activities like dating, what makes people happy etc, and objects like cameras and cars. On the average the datasets used contain about 500 instance and 5 different columns. The biggest dataset is called Speed dating data. It is made up of over 8,000 observations of answers to survey questions about how people rate themselves and how they rate others on several dimensions. The smallest dataset used has 33 instances and 12 columns. It contains information about cars. The number of gears and speed, just to name a few attributes.

3.1.2 Dataset for R

For the plots in R, 13 random CSV files where downloaded from an archive of datasets distributed with R called Rdatasets [17]. Rdatasets is a collection of dataset distributed with R. On the average there are 80 instances and 5 columns in each dataset. The biggest CSV file is the Australian athletes dataset. It's made of 203 instances and 14 columns and contains attributes like sex,height,weight and sports. The smallest dataset is the Canadian Women's Labour-Force Participation. This dataset has 30 rows and 7 columns. It contains information like average wages of women, percent of adult women in the workforce etc.

3.1.3 Dataset for Python

The data used for creating the plots in Python were 15 randomly seleted csv files also from Rdatasets [17]. The biggest dataset among the 15 is the Mono-

clonal gammapothy data, it contains natural history patients with monoclonal gammapothy of undetermined significance. The dataset is made up of 1384 observations with 10 columns, it has attributes like age, sex, time of death and last contact in months. On the average each dataset contains about 200 instances and 7 columns of multi-dimensional data. The smallest dataset however contains only 33 instances with 11 columns and is called the Nuclear Power Station Construction Data.The data relate to the construction of 32 light water reactor (LWR) plants constructed in the U.S.A in the late 1960's and early 1970's.

3.1.4 Dataset for Java

For the plots created in java, I used the dataset made available by Plotly [16], a github repository of CSV datasets used in the Plotly API examples. 14 random CSV files were downloaded, the biggest file has 1002 instances and 9 columns, and on the average each file contains about 100 instances and 9 columns. The smallest file however is made of 33 instances and 12 columns called the mtcars file. It contain information about a variety of different car models like the number of gears, speed etc. The table 3.1 contains the names of all CSV files that were used in the different languages with the different plotting programs.

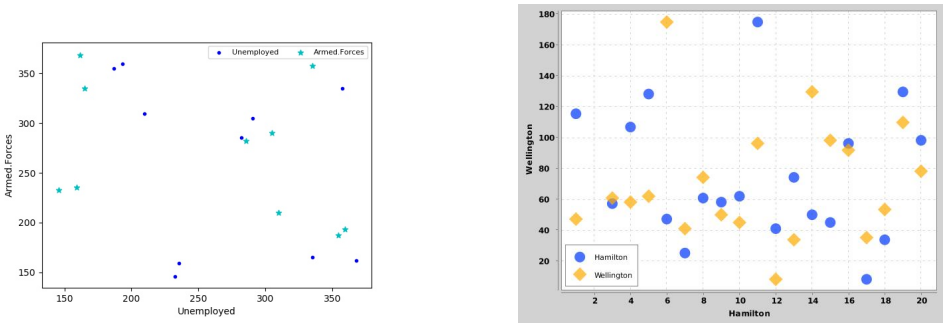
Datasets			
Python	Matlab	R	Java
3d_line_sample_data.csv LightFordwardFlapStall.csv line_3d_dataset.csv longley.csv loti.csv lung.csv nuclear.csv timeseries.csv USJudgeRatings WVSCulturalMap.csv wind_rose.csv volcano.csv uspop2.csvm tips	Camera.csv Cars.csv speedDating.csv Cereal.csv happiness.csv TestData1.csv TestData2.csv mpg.csv okcupid- religion.csv spectral.csv stockdata.csv subplots.csv	ais.csv Angell.csv Baumann.csv Bfox.csv cane.csv carprice.csv Chirot.csv Davis.csv Ericksen.csv Florida.csv Highway1.csv Pottery.csv Prestige.csv salin- ity.csv urine.csv	3d-line-plot.csv 3d-scatter.csv 2011_flight_paths.csv 2011_us_exports.csv auto-mpg.csv candlestick_dataset.csv finance-charts-apple.csv globe_contours.csv hobbs-pearson- trials.csv motor_trend_tests.csv nz_weather.csv volcano.csv iris.csv mtcars.csv

Table 3.1: Names of datasets used in each plotting program

3.2 Creating Plots

The inspiration for creating a variety of plots to capture all type of plots used in scientific papers was gotten by inspecting the dataset of Architecture proposal for data extraction of chart images using CovNet paper [11] and Viziometrics: Analyzing visual information in the scientific literature [18] dataset. Scripts in various languages were written to handle the plotting and labeling process automatically. All datasets for a particular plot (example scatter plot for python) are put into one folder. The scripts reads each CSV file column by column while creating the plots.

Table 3.2 describe how the plots where created in each language. The type column describes the different variety of a particular plot, for example bar charts can be of type stacked, grouped, vertical and horizontal bar charts, also scatter plots types can be a scatter plot consisting of one type of marker, one scatter plot with multiple markers and finally a scatter plot with a line showing the correlation between the plots. Figure 3.1 shows two different types of bar charts. Figure 3.1a is a stacked bar chart and Figure 3.1b a normal vertical bar chart. The Library column shows the different plotting libraries used, the parameter column describes parameters that were changed and finally the number of plots created were also added. The images below the tables are sample images that exist in our dataset of created plots for each language.



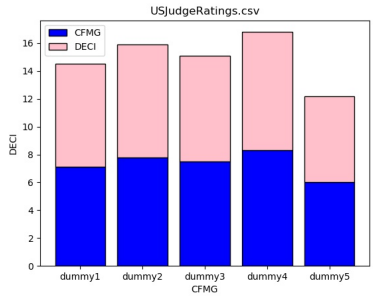
(a) Matplotlib scatter plot with star and circular markers

(b) Java scatter plot with circular and diamond markers

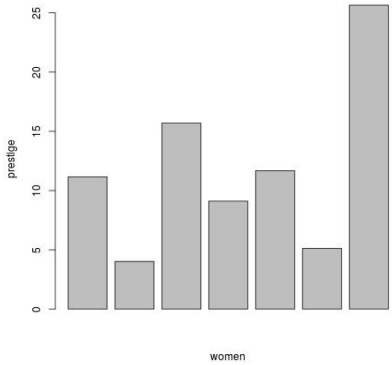
Figure 3.1: Scatter plots

Table 3.2: Overview of the varied parameters for creating plots in the different plotting programs

SCATTER PLOTS				
Language	Library	Parameters	Number of plots	Type
Python	Matplotlib v2.1.2 Plotly v2.5.1 Seaborn v0.8.1	MarkerStyle ['o', '*', '.', '+','x']	1020	Unique markers, With legends, multiple markers regplot
MATLAB	Default Plotly	MarkerStyle ['o', '*','+','x','s']	1044	
R	Plotly Lattice Ggplot2	MarkerStyle ['o', '*', '+','x','s']	1644	
JAVA	XChart 3.5.1 jfreechart 1.0.1	MarkerSize (15 - 18)	1644	
BAR CHARTS				
Language	Library	Parameters	Number of plots	Types(bar)
Python	Matplotlib v2.1.2 Plotly v2.5.1 Seaborn v0.8.1		1000	Horizontal and Vertical, Stacked, Grouped bar charts
MATLAB	Default	Width of bar(14-16)	1000	
R	Default,Plotly R Library ggplot2	space (0-3)	1144	
JAVA	XChart 3.5.1 jfreechart:1.0.192 javafx.scene	PlotOrientation (vertical or horizontal) with error bars	1144	
LINE CHARTS				
Language	Library	Parameters	Number of plots	Types(Line with)
Python	Matplotlib v2.1.2 Plotly v2.5.1 Seaborn v0.8.1	Linestyle ['-', '–', '-.', ':']	1000	Markers, Multiple Lines
MATLAB	Default Plotly	MarkerStyle ['o', '*', '.', '+','x','s'] markersize [8-10]	1000	
R	Default,Plotly R Library ggplot2		1644	
JAVA	XChart 3.5.1 javafx JFreeChart	MarkerSize (12-16)	1644	
Box Plots				
Language	Library	Parameters	Number of plots	Types(Box with)
Python	Matplotlib v2.1.2 Plotly v2.5.1 Seaborn v0.8.1		1000	Notches, Multiple Boxes
MATLAB	Default		1000	
R	Default,Plotly R Library ggplot2		1644	
JAVA	XChart 3.5.1	LegendPosition	1644	

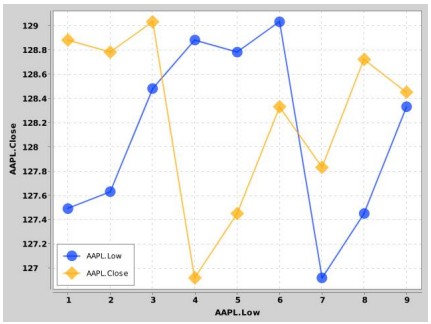


(a) Matlab stacked bar chart (bar width 16)

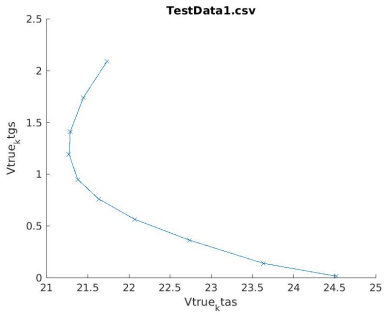


(b) R horizontal bar chart (bar width 16)

Figure 3.2: Example Bar Charts

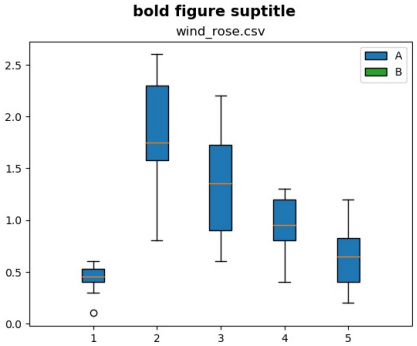


(a) Java line chart with diamond and circular markers

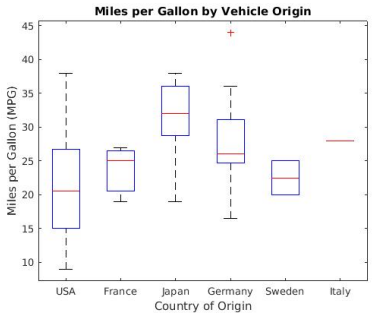


(b) simple Matlab line chart with Asterix marker

Figure 3.3: Example Line Charts



(a) Vertical multiple boxplots in python



(b) Vertical multiple boxplots in Matlab

Figure 3.4: Example Box Plots

3.3 Training, Validation and Test set

Figure 3.5 summaries how our dataset was split into various activities. In machine learning, a model is an artifact created during the training phase, this model can be likened to a function that maps specific features to their respective labels, the model is trained with a portion of the dataset called the train set. As the model is trained, the validation set is used to decide and pick which metric out of hyper-parameters, early stopping and architecture considerations yields the best performance. This helps to adjust and optimize the model [19]. Finally, the test-set in machine learning, is the other portion of the dataset which was not used during the training phase. The test-set checks how well the trained model performs on unseen data by giving an unbiased assessment of the model. For this work, the dataset is divided into train, validation and test set. The validation set is not shown in the figure since it forms part of the train data.

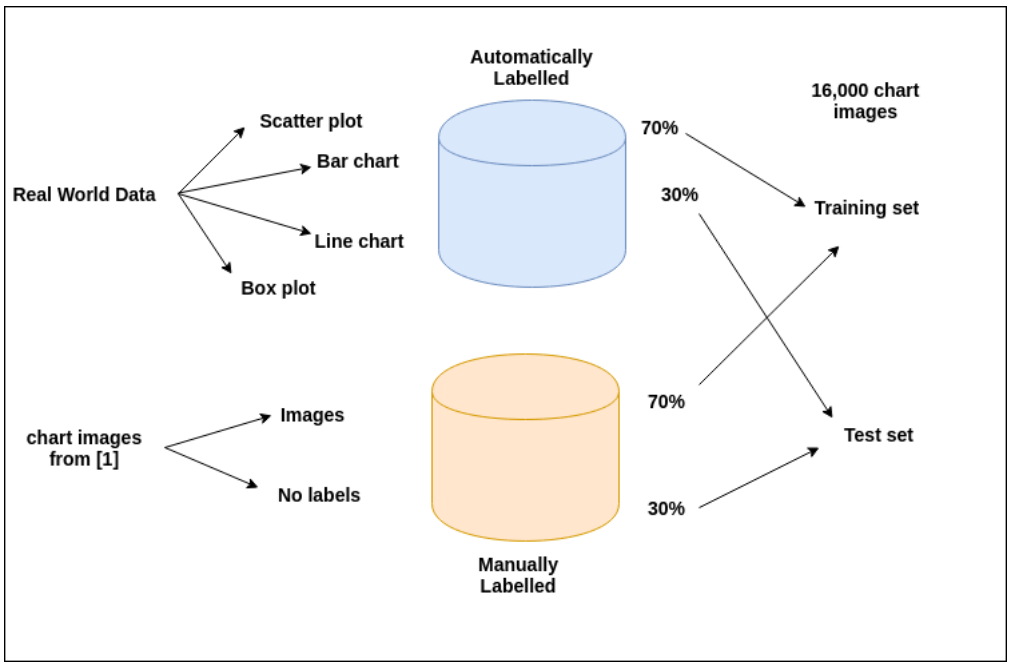


Figure 3.5: Steps involved in building the Classification model

3.4 The Architecture

As mentioned in previous sections, our model is trained with a CovNet. This architecture is inspired by the famous AlexNet CovNet architecture [1]. The AlexNet architecture was used in the famous ImageNet LSVRC-2010 contest to classify 1.2 million into 1000 classes, the results obtained on the test-set had a top-5 error rate of and 17.0%, which was better than the previous state of the art. Our CovNet architecture just like the AlexNet architecture contains 8 learned layers, these 8 layers are made up of five Convolutional layers and 3 fully connected layers. A Relu is used after some Convolutional layers and fully connected layers. Another parameter employed in our architecture is a dropout, a dropout is applied in the first and second fully connected layers of our network and finally, max pooling is applied in some of the layers. A summary of our CovNet architecture used to train our model is shown in Figure 3.6. The parameters used and why they were employed are explained into details in the rest of this chapter.

Convolutional Layer

The convolutional layer will be explained using the following example, let consider the image as a magical cake of height and width 48x48, and there is a cake cutter of size say 5x5. The cutter is used to cut the cake from the top left. The cutter is referred to as a filter in machine learning. The cutter in this case is also an array of numbers called weights and for the maths to work the cutter must have the same depth as our cake (5x5x3). Lets first consider the

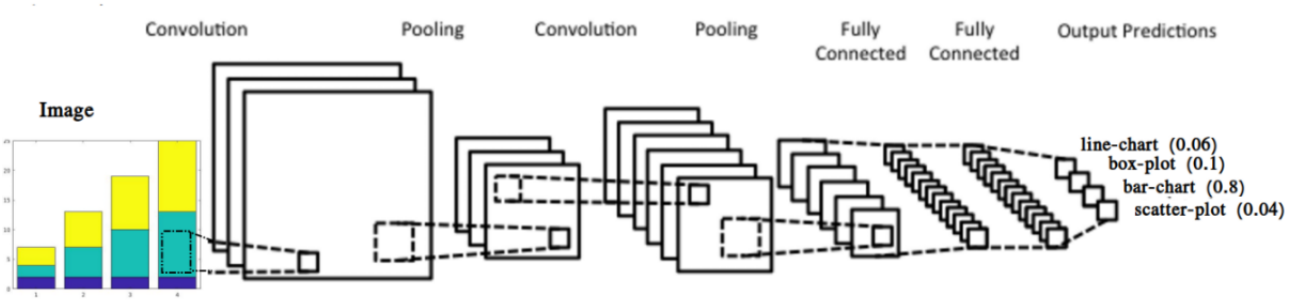


Figure 3.6: Architecure of CovNet used to train our model

cutter being in the first position ie. the top left of the cake. The values in the cutter are multiplied with the values of the cake (element wise multiplications). These multiplications are all summed up and result in a single number. This single number is only for the first part and this process is repeated through out the whole cake (Next step would be moving the cutter to the right by 1 unit, then to the right again by 1, and so on) keep in mind its a magical cake so you can cut a part more than once. After using the cutter on the whole of the magical cake we result in a new cake of size 28x28x1, the results is called a feature map. The filters are low level feature identifiers (straight edges, simple colors, and curves). As mentioned above 5 of this layers are used in our architecture, and these are responsible for extracting the features of the chart images. Our input image size used is 224 x 224 x 3 since its an RGB image, this is fed into our first Convolutional layer, this filters the image with a stride of 4 pixels. The second Convolutional layer takes as its input the output of the first Convolutional layer, between this two layers max-pooling is performed to reduce the size of the image. The third and fourth Convolutional layers, unlike the previous two have no max-pooling applied between them. The final Convolutional layer then takes as input the output of the previous layer and performs a max-pooling.

ReLU Layer

The Rectified Linear Units (ReLU) is found in all layers during the Convolution phase, this layer implements element-wise nonlinearity to our Convolutional layer, this just means it helps to handle situations where the relation between the input values and the CovNet output is non-linear. The ReLU has a function $f(x) = max(0, x)$ which means if you give it a value x, it will return 0 if x is negative and will return the value itself if its positive. There are also other functions which can used in place of the ReLU, that is the tanh or the sigmoid. We used ReLU mainly because it speeds up the training process significantly and it does not saturate(the gradient is very small, if the input

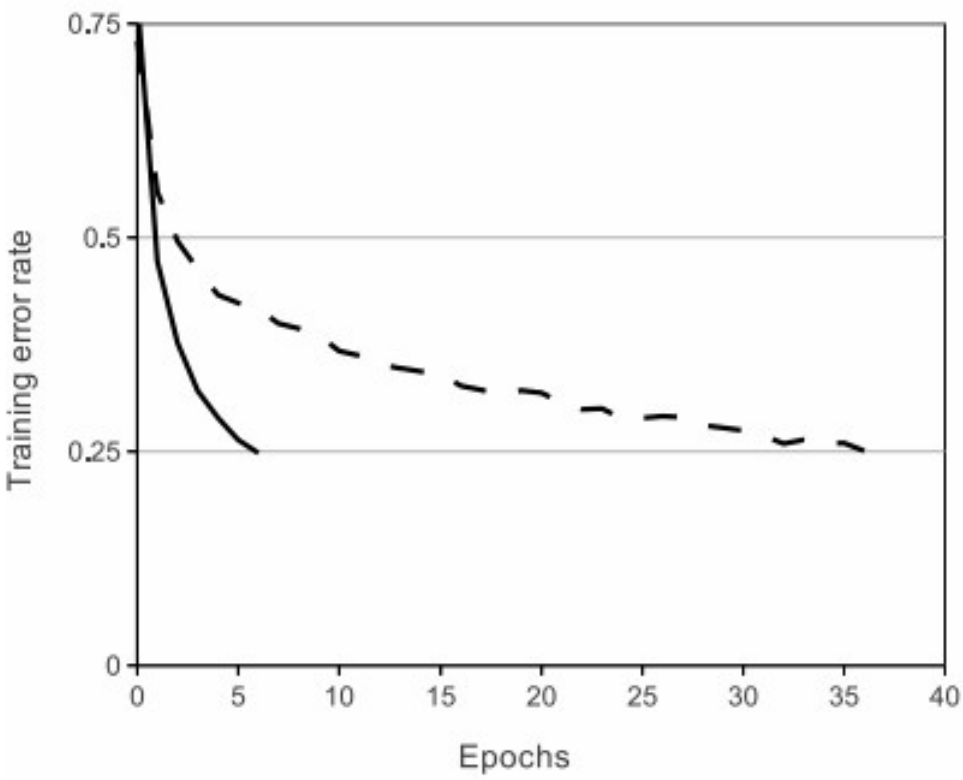


Figure 3.7: ReLU unit compared to the tanh unit from Krizhevsky et al. paper [1]

is elevated or small) unlike the tanh and sigmoid [20]. Figure 3.7 shows a plot from Krizhevsky et al. [1] paper shows that there is a 6x advancement in convergence with the ReLU unit compared to the tanh unit. The bold line represents the ReLU and tanh is the dashed line.

Pooling Layer

The pooling layers basically have two main functions, first one is that it helps the CovNet to locate features regardless of which part of the image it is located. This results in the model being robust against small changes in position of the features of the images. The second function is that it also helps to reduce the size of the feature map. Therefore computations in the futures layers are relatively less complex. One way of performing pooling is using max pooling technique, thats is sliding a window through the feature map, the windows fills a number of arrays in the feature map therefore, you pick the largest among all the numbers and disregard the rest of the numbers. We employ this technique in the first, second and last Convolutional layers of our network.

Dropout Layers

The following is inspired by an article on Medium [21]. A dropout was used in the first and second Fully connected layers in our architecture. The dropout is

used in our CovNet to prevent over-fitting. The Fully connected layers eventually handles more parameters and therefore neurons become co-dependent on one another, and this leads to over-fitting during the training phase. When dropout is implemented in a network, it does not use all neurons during a particular forward or backward pass during training but, choose random neurons at a specified probability for each pass. We set dropout probability to 0.5 in our network, this means nodes are dropped out with a probability of 1-05 or or maintained at a probability of 0.5.

Fully Connected Layer

The last set of layers used are the Fully connected layers. The first layer takes as input the 1D feature vector generated by a flatten layer in our network. The flatten layer converts all our 2D arrays from our previous Convolutional layers into a 1D resultant vector. The output of this layer is then fed as input into the second fully connected layer. The output of the second layer is then fed into the last fully connected layer. The last layer in this set of 3 layers uses the softmax activation to give our output prediction. The output is a separate probability of our four classes and the choice with the highest probability is our chosen prediction. We used a sofmax activation as opposed to other activation functions because, it handles low invigoration in images (say blurry charts) of your neural net with rather uniform distribution and to high invigoration in images (say large numbers, like sharp images) with prediction probabilities between 1 and 0.

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \tag{3.1}$$

The equation 3.1 is a softmax function, (z) represents a vector, so the function takes (in z) and crushes it to a probability value between zero and one [22].

3.5 Image Preprocesing

Image preprocessing is a very important step in any image based application. This process involves taking the image and improving it in a way that enables easier machine understanding. This ultimately makes it easier for the machine to extract important features for other operations.

4 Evaluation

5 Summary

6 Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] V Karthikeyani and S Nagarajan. Machine learning classification algorithms to recognize chart types in portable document format (pdf) files. *International Journal of Computer Applications*, 39(2), 2012.

[3] Xiao Liu, Binbin Tang, Zhenyang Wang, Xianghua Xu, Shiliang Pu, Dapeng Tao, and Mingli Song. Chart classification by combining deep convolutional networks and deep belief networks. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 801–805. IEEE, 2015.

[4] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. Revision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 393–402. ACM, 2011.

[5] V Shiv Naga Prasad, Behjat Siddiquie, Jennifer Golbeck, and Larry S Davis. Classifying computer generated charts. In *Content-Based Multimedia Indexing, 2007. CBMI'07. International Workshop on*, pages 85–92. IEEE, 2007.

[6] Jihen Amara, Pawandeep Kaur, Michael Owonibi, and Bassem Bouaziz. Convolutional neural network based chart image classification. 2017.

[7] Richard Boyle, Bahram Parvin, Darko Koracin, Nikos Paragios, and Syeda-Mahmood Tanveer. *Advances in visual computing*. Springer, 2007.

[8] Yan Ping Zhou and Chew Lim Tan. Bar charts recognition using hough based syntactic segmentation. In *International Conference on Theory and Application of Diagrams*, pages 494–497. Springer, 2000.

[9] Mingyan Shao and Robert P Futrelle. Recognition and classification of figures in pdf documents. In *International Workshop on Graphics Recognition*, pages 231–242. Springer, 2005.

[10] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23. Springer, 2000.

[11] Paulo Roberto Silva Chagas Junior, Alexandre Abreu De Freitas, Rafael Daisuke Akiyama, Brunelli Pinto Miranda, Tiago Davi Oliveira De Araújo, Carlos Gustavo Resque Dos Santos, Bianchi Serique Meiguins, and Jefferson Magalhães De Morais. Architecture proposal for data extraction of chart images using convolutional neural network. In *Information Visualisation (IV), 2017 21st International Conference*, pages 318–323. IEEE, 2017.

[12] Rodriguez, J. Convolutional neural networks for the rest of us part iii: Benefits and motivation. https://medium.com/@jrodthoughts/convolutional-neural-networks-for-the-rest-of-us-part-iii-benefits-and-motiv 2011. [Online; accessed 5-Jun-2018].

[13] Editor. what-to-look-for-in-training-dataset. https://medium.com/@jrodthoughts/convolutional-neural-networks-for-the-rest-of-us-part-iii-bene 2018. [Online; accessed 5-Jun-2018].

[14] James Eagan. Project datasets. https://perso.telecom-paristech.fr/eagan/class/igr204/datasets. [Online; accessed 20-April-2018].

[15] Michael Vogt. Datplot. https://vincentarelbundock.github.io/Rdatasets/datasets.html, 2011. [Online; accessed 20-April-2018].

[16] plotly/datasets. Latex — Wikipedia, the free encyclopedia. https://github.com/plotly/datasets, 2011. [Online; accessed 20-April-2018].

[17] vincentarel bundock. Rdatasets. https://vincentarelbundock.github.io/Rdatasets/datasets.html, 2011. [Online; accessed 20-April-2018].

[18] Po-shen Lee, Jevin D West, and Bill Howe. Viziometrics: Analyzing visual information in the scientific literature. *IEEE Transactions on Big Data*, 4(1):117–129, 2018.

[19] Akshay Balwally,. What is the difference between validation set and test set? https://www.quora.com/

What-is-the-difference-between-validation-set-and-test-set, 2016. [Online; accessed 7-Jun-2018].

[20] Balwally, A. What is the role of rectified li. https://www.quora.com/What-is-the-role-of-rectified-linear-ReLU-activation-function-in-CNN, 2018. [Online; accessed 9-Jun-2018].

[21] Rodriguez, J. Convolutional neural networks for the rest of us part iii: Benefits and motivation. https://medium.com/@jrodthoughts/convolutional-neural-networks-for-the-rest-of-us-part-iii-benefits-and-motiv 2018. [Online; accessed 10-Jun-2018].

[22] karpathy. Cs231n convolutional neural networks for visual recognition. http://cs231n.github.io/linear-classify/#softmax, 2018. [Online; accessed 11-Jun-2018].