

CHESS PIECE IDENTIFICATION

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the guidance of

MAHENDRA DATTA

BY

DRIPTO BHATTACHARYYA (14800121098)

ASIM MANDAL (14800121079)



In association with



(ISO9001:2015)

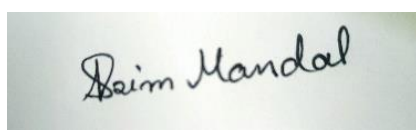
SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

1. Title of the Project: **CHESS PIECE IDENTIFICATION**
2. Project Member: **ASIM MANDAL**
3. Name of the guide: **Mr. MAHENDRA DUTTA**
4. Address: **Ardent Computech Pvt. Ltd**
(An ISO 9001:2015 Certified)
SDF Building, Module #132, Ground Floor, Salt
Lake City, GP Block, Sector V, Kolkata, West
Bengal, 700091

Project Version Control History

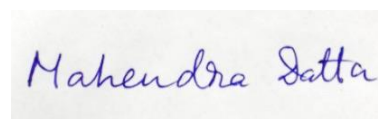
Version	Primary Authors	Description of Version	Date Completed
Final	DRIPTO BHATTACHARYYA	Project Report	13 TH MARCH,2024
Final	ASIM MANDAL	Project Report	13 TH MARCH,2024



Signature of Team Member

Date: 18.03.2024

For Office Use Only



Signature of Approver

Date: 18.03.2024

MR.MAHENDRA DUTTA

Project Proposal Evaluator

Approved

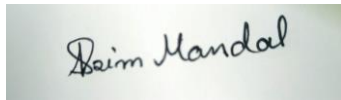
Not Approved

DECLARATION

I hereby declare that the project work being presented in the project proposal entitled “**CHESS PIECE IDENTIFICATION**” in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** at **ARDENT COMPUTECH PVT. LTD, SALT LAKE, KOLKATA, WEST BENGAL**, is an authentic work carried out under the guidance of **MR. MAHENDRA DUTTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 25.03.2024

Name of the Student: ASIM MANDAL



Signature of the student



Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

CERTIFICATE

This is to certify that this proposal of minor project entitled “**CHESS PIECE IDENTIFICATION**” is a record of bona fide work, carried out by ***DRIPTO BHATTACHARYYA*** under my guidance at **ARDENT COMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT®**. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor

MR. MAHENDRA DUTTA

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to ***Mr. MAHENDRA DUTTA***, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS

DECLARATION	3
CERTIFICATE	4
ACKNOWLEDGEMENT	5
OVERVIEW OF PYTHON	8
HISTORY OF PYTHON	8
FEATURES OF PYTHON	9
ENVIRONMENT SETUP	10
BASIC SYNTAX OF PYTHON PROGRAM	10
Python Identifiers	11
Python Keywords	11
Lines & Indentation	11
Command Line Arguments	12
VARIABLE TYPES	12
Assigning Values to Variables	12
Multiple Assignment	13
Standard Data Types	13
Data Type Conversion	14
FUNCTIONS	15
Defining a Function	15
Global vs. Local variables	16
MODULES	17
PACKAGES	18
MACHINE LEARNING	19
INTRODUCTION TO MACHINE LEARNING	19
SUPERVISED LEARNING	20
UNSUPERVISED LEARNING	20
NUMPY	20
SCIPY	23
The SciPy Library/Package	23

Data Structures	24
SCIKIT-LEARN	24
MATPLOTLIB	25
PANDAS.....	26
CLASSIFICATION	27
CONVOLUTION NEURAL NETWORK	28
CNN ARCHITECTURE.....	29
ADVANTAGES OF CNN.....	30
DISADVANTAGES OF CNN	30
ALGORITHM.....	30
CHESS PIECE IDENTIFICATION	32
INTRODUCTION	32
PROBLEM STATEMENT	32
OBJECTIVE	32
CODE.....	33
Importing Dataset and Libraries	33
EDA and Preprocessing	34
Creating Model	36
Evaluation	38
CONCLUSION	39
FUTURE SCOPE.....	41
BIBLIOGRAPHY	41

OVERVIEW OF PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

FEATURES OF PYTHON

Easy-to-learn: Python has few Keywords, simple structure and clearly defined syntax. This allows a student to pick up the language quickly.

Easy-to-Read: Python code is more clearly defined and visible to the eyes.

Easy -to-Maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building
- large applications.
- It provides very high level dynamic datatypes and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**.

However in Python version 2.4.3, this produces the following result –

Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

And, exec, not

Assert, finally, or

Break, for, pass

Class, from, print

continue, global, raise

def, if, return

del, import, try

elif, in, while

else, is, with

except, lambda, yield

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
```

```
    print "True"
```

```
else:  
    print "False"
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h –

```
$ python-h
```

```
usage: python [option]...[-c cmd|-m mod | file |-][arg]...
```

Options and arguments (and corresponding environment variables):

- c cmd: program passed in as string(terminates option list)
- d : debug output from parser (also PYTHONDEBUG=x)
- E : ignore environment variables (such as PYTHONPATH)
- h : print this help message and exit [etc.]

VARIABLE TYPES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10          # An integer assignment  
weight=10.60        # A floating point
```

```
name="Ardent"    # A string
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example –

```
a = b = c = 1
```

```
a,b,c = 1,2,"hello"
```

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types –

- ☐ String
- ☐ List
- ☐ Tuple
- ☐ Dictionary
- ☐ Number

Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

Sr.No.	Function & Description
1	int(x [,base]) Converts x to an integer. base specifies the base if x is a string
2	long(x [,base]) Converts x to a long integer. base specifies the base if x is a string.
3	float(x) Converts x to a floating-point number.
4	complex(real [,imag]) Creates a complex number.
5	str(x) Converts object x to a string representation.
6	repr(x) Converts object x to an expression string.
7	eval(str) Evaluates a string and returns an object.
8	tuple(s) Converts s to a tuple.
9	list(s) Converts s to a list.

FUNCTIONS

Defining a Function

- `def function name(parameters):`

`"function_docstring"`

`function suite`

`return [expression]`

Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

Function definition is here

`def change me(mylist):`

`"This changes a passed list into this function"`

`mylist.append([1,2,3,4]);`

`print"Values inside the function: ",mylist`

`return`

Now you can call changeme function

`mylist=[10,20,30];`

`change me(mylist);`

print" Values outside the function: ",mylist

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result –

Values inside the function: [10, 20, 30, [1, 2, 3, 4]]

Values outside the function: [10, 20, 30, [1, 2, 3, 4]]

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;           # This is global variable.
```

```
# Function definition is here
```

```
def sum( arg1, arg2 ):
```

```
# Add both the parameters and return them."
```

```
total= arg1 + arg2;    # Here total is local variable.
```

```
print"Inside the function local total: ", total
```

```
return total;
```

```
# Now you can call sum function
```

```
sum(10,20);
```



```
Print"Outside the function global total: ", total
```

When the above code is executed, it produces the following result –

```
Inside the function local total: 30
```

```
Outside the function global total: 0
```

MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, *support.py*

```
def print_func( par ):  
    print"Hello : ", par  
    return
```

The *import* Statement

You can use any Python source file as a module by executing an *import* statement in some other Python source file. The *import* has the following syntax –

```
Import module1 [, module2 [... moduleN]
```

PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-packages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code –

```
def Pots ():  
    print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- *Phone/Isdn.py* file having function *Isdn ()*
- *Phone/G3.py* file having function *G3 ()*

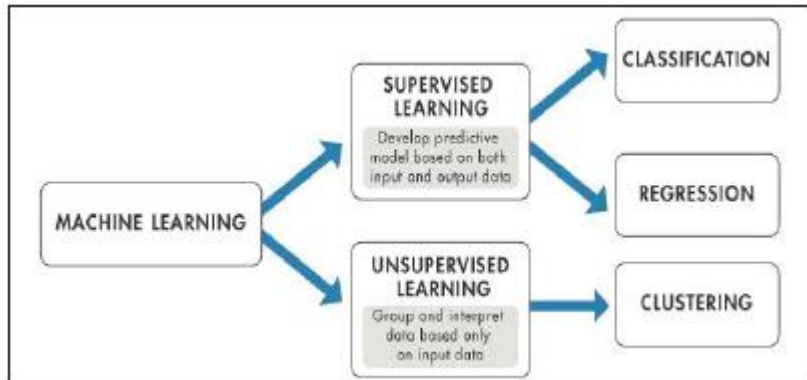
Now, create one more file `__init__.py` in *Phone* directory –

- *Phone/__init__.py*

To make all of your functions available when you've imported *Phone*, you need to put explicit import statements in `__init__.py` as follows –

```
from Pots import Pots  
from Isdn import Isdn  
from G3 import
```

MACHINE LEARNING



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from *labelled training data*.^[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[1., 0., 0.],  
 [ 0., 1., 2.]]
```

NumPy's array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

Import numpy as np

```
a = np.array ([[1, 2, 3],[3,4,5],[4,5,6]])
```

```
print 'Our array is:'
```

Print a

```
print '\n'
```

```
print 'The items in the second column are:'
```

```
print a[... ,1]
```

```
print '\n'
```

```
print 'The items in the second row are:'
```

```
print a[1...]
```

```
print '\n'
```

```
print 'The items columns 1 onwards are:'
```

```
print a [... ,1:]
```

OUTPUT

Our array is:

```
[[1 2 3]
```

```
[3 4 5]
```

```
[4 5 6]]
```

The items in the second column are:

```
[2 4 5]
```

The items in the second row are:

```
[3 4 5]
```

The items column 1 onwards are:

```
[[2 3]
```

```
[4 5]
```

```
[5 6]]
```

SCIPY

Scipy modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The SciPy Library/Package

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

- **constants:** physical constants and conversion factors (since version 0.7.0)
- **cluster:** hierarchical clustering, vector quantization, K-means
- **fftpack:** Discrete Fourier Transform algorithms
- **integrate:** numerical integration routines
- **interpolate:** interpolation tools
- **io:** data input and output
- **lib:** Python wrappers to external libraries
- **linalg:** linear algebra routines
- **misc:** miscellaneous utilities (e.g. image reading/writing)
- **ndimage:** various functions for multi-dimensional image processing
- **optimize:** optimization algorithms including linear programming
- **signal:** signal processing tools
- **sparse:** sparse matrix and related algorithms
- **spatial:** KD-trees, nearest neighbours, distance functions
- **special:** special functions
- **stats:** statistical functions
- **weave:** tool for writing C/C++ code as Python multiline strings

Data Structures

The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multi-dimensional container of data with arbitrary data-types. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favour of the newer NumPy array code.

SCIKIT-LEARN

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

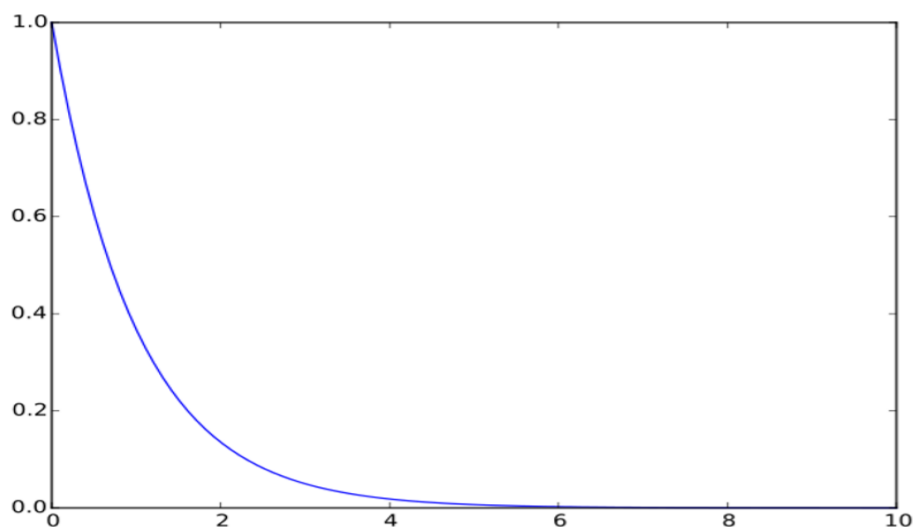
MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

EXAMPLE

➤ LINE PLOT

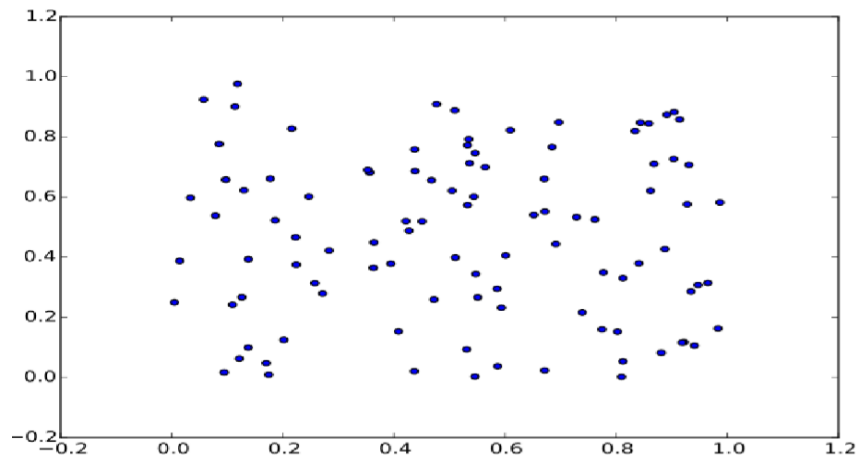
```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>>a=np.linspace(0,10,100)
>>>b=np.exp(-a)
>>>plt.plot(a,b)
>>>plt.show()
```



➤ SCATTER PLOT

```
>>>importmatplotlib.pyplotasplt
```

```
>>>fromnumpy.randomimport rand
>>> a =rand(100)
>>> b =rand(100)
>>>plt.scatter(a, b)
>>>plt.show ()
```



PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

CLASSIFICATION

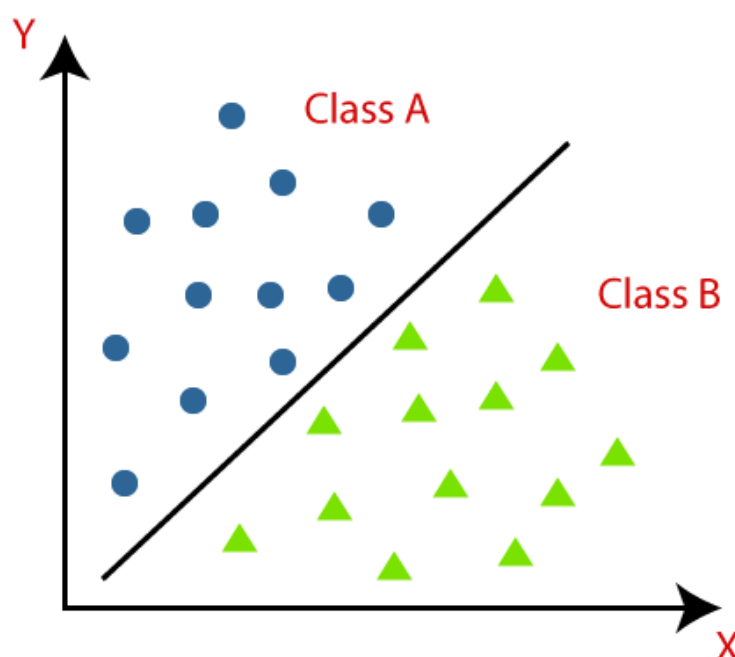
The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc. Classes can be called as targets/labels or categories. Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a Supervised learning technique, hence it takes labelled input data, which means it contains input with the corresponding output.

In classification algorithm, a discrete output function(y) is mapped to input variable(x).

$y=f(x)$, where y = categorical output

The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data.

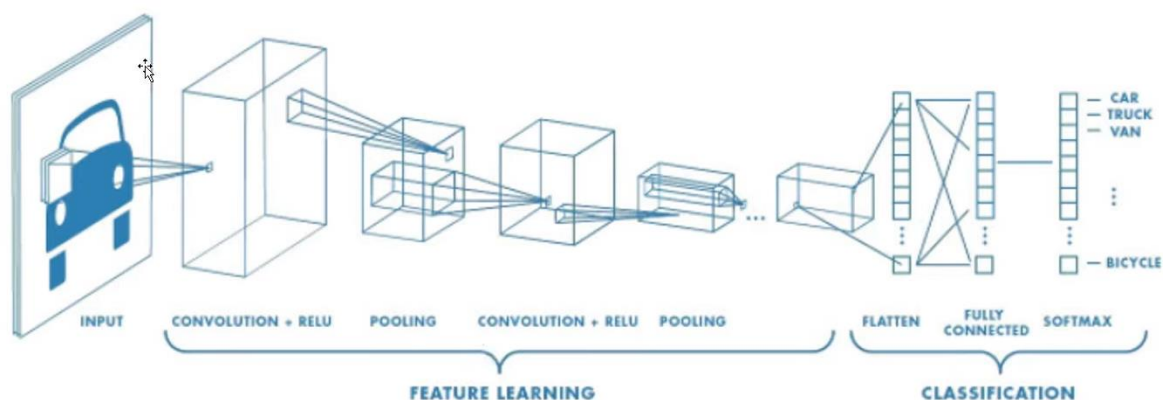
Classification algorithms can be better understood using the below diagram. In the below diagram, there are two classes, class A and Class B. These classes have features that are similar to each other and dissimilar to other classes.



The algorithm which implements the classification on a dataset is known as a classifier. There are two types of Classifications:

- **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier. **Examples:** YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
- **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called as Multi-class Classifier. **Example:** Classifications of types of crops, Classification of types of music

CONVOLUTION NEURAL NETWORK



A **Convolutional Neural Network (CNN)** is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use **Recurrent Neural Networks** more precisely an LSTM, similarly for image classification we use Convolution

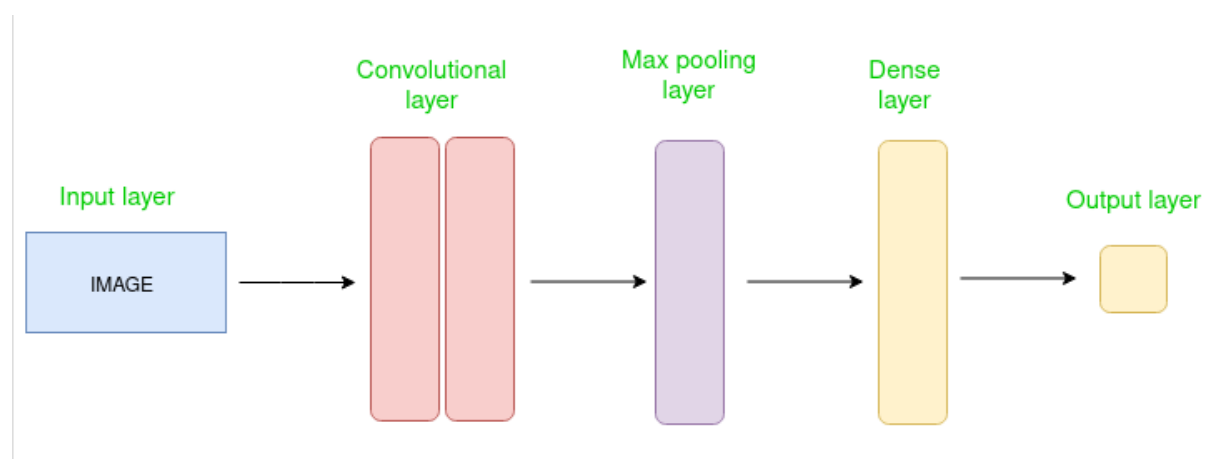
Neural networks. In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network there are three types of layers:

1. **Input Layers:** It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or SoftMax which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called **feedforward**, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, we backpropagate into the model by calculating the derivatives. This step is called **Backpropagation** which basically is used to minimize the loss.

CNN ARCHITECTURE



Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

ADVANTAGES OF CNN

1. Good at detecting patterns and features in images, videos, and audio signals.
2. Robust to translation, rotation, and scaling invariance.
3. End-to-end training, no need for manual feature extraction.
4. Can handle large amounts of data and achieve high accuracy.

DISADVANTAGES OF CNN

1. Computationally expensive to train and require a lot of memory.
2. Can be prone to overfitting if not enough data or proper regularization is used.
3. Requires large amounts of labelled data.
4. Interpretability is limited, it's hard to understand what the network has learned.

ALGORITHM

- Data Collection
- Padding
- Image Filtering
- Convolution
- Max Pooling
- Training
- Testing

Data Collection: I have collected image sets of weather from Kaggle in a zip folder.

Padding: Creating a padding with 0s around an image.

Image Filtering: Essentially doing dot product

Convolution: This filtering is done continuously, which essentially makes the convolution.

Max Pooling: Picks the maximum value and makes a more condensed form

Training: The data set was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.

Testing: The model was tested with y_train and stored in y_predict. Both y_train and y_predict was compared.

CHESS PIECE IDENTIFICATION

INTRODUCTION

Chess, a timeless game of intellect and strategy, has intrigued enthusiasts for centuries. In this project, we embark on a journey to harness the power of deep learning to address the challenge of identifying chess pieces from images. By leveraging the simplicity and effectiveness of Keras, a renowned deep learning library, we aim to develop a robust model capable of accurately recognizing various types of chess pieces with high precision.

PROBLEM STATEMENT

The task of identifying chess pieces from images presents a unique set of challenges. The intricate shapes and patterns of the pieces, coupled with variations in lighting, angle, and image quality, make accurate classification a formidable endeavor. Traditional computer vision techniques often struggle to achieve satisfactory results in this domain, necessitating the adoption of deep learning methodologies for improved accuracy and reliability.

OBJECTIVE

Our primary objective is to develop an efficient image classification model using Keras specifically tailored for identifying different types of chess pieces. By training the model on a diverse dataset comprising images of various chess pieces, captured under different conditions, we aim to achieve high accuracy and efficiency in classification. The ultimate goal is to empower applications such as automated chess game analysis, virtual chess assistants, and educational tools for chess enthusiasts.

CODE

Importing Dataset and Libraries

✓ Importing Dataset and Libraries

✓
1s [1] ! unzip archive.zip

```
Archive:  archive.zip
  inflating: Chessman-image-dataset/Chess/Bishop/00000000.JPG
  inflating: Chessman-image-dataset/Chess/Bishop/00000001.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000002.JPG
  inflating: Chessman-image-dataset/Chess/Bishop/00000003.png
  inflating: Chessman-image-dataset/Chess/Bishop/00000004.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000006.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000007.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000008.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000009.png
  inflating: Chessman-image-dataset/Chess/Bishop/00000010.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000011.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000012.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000013.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000015.png
  inflating: Chessman-image-dataset/Chess/Bishop/00000016.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000017.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000018.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000019.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000020.jpg
  inflating: Chessman-image-dataset/Chess/Bishop/00000024.jpg
```

✓ [5]

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import *
from sklearn.metrics import classification_report
```

EDA and Preprocessing

✓ EDA and Preprocessing

```
✓ 1s [6] image_size = (48, 48)
      batch = 32

      train = keras.preprocessing.image_dataset_from_directory(
          'Chessman-image-dataset/Chess/',
          validation_split=.2,
          subset='training',
          seed=42,
          image_size=image_size,
          batch_size=batch,
          label_mode='categorical'
      )
```

Found 552 files belonging to 6 classes.
Using 442 files for training.

```
✓ 0s [7] validation = keras.preprocessing.image_dataset_from_directory(
      'Chessman-image-dataset/Chess/',
      validation_split=.2,
      subset='validation',
      seed=42,
      image_size=image_size,
      batch_size=batch,
      label_mode='categorical'
  )
```

Found 552 files belonging to 6 classes.
Using 110 files for validation.

First, we are setting the size of each image to be 48 x 48. This would help in training.

Then, the data is split into Training Set and Validation Set, with a validation split of 0.2.

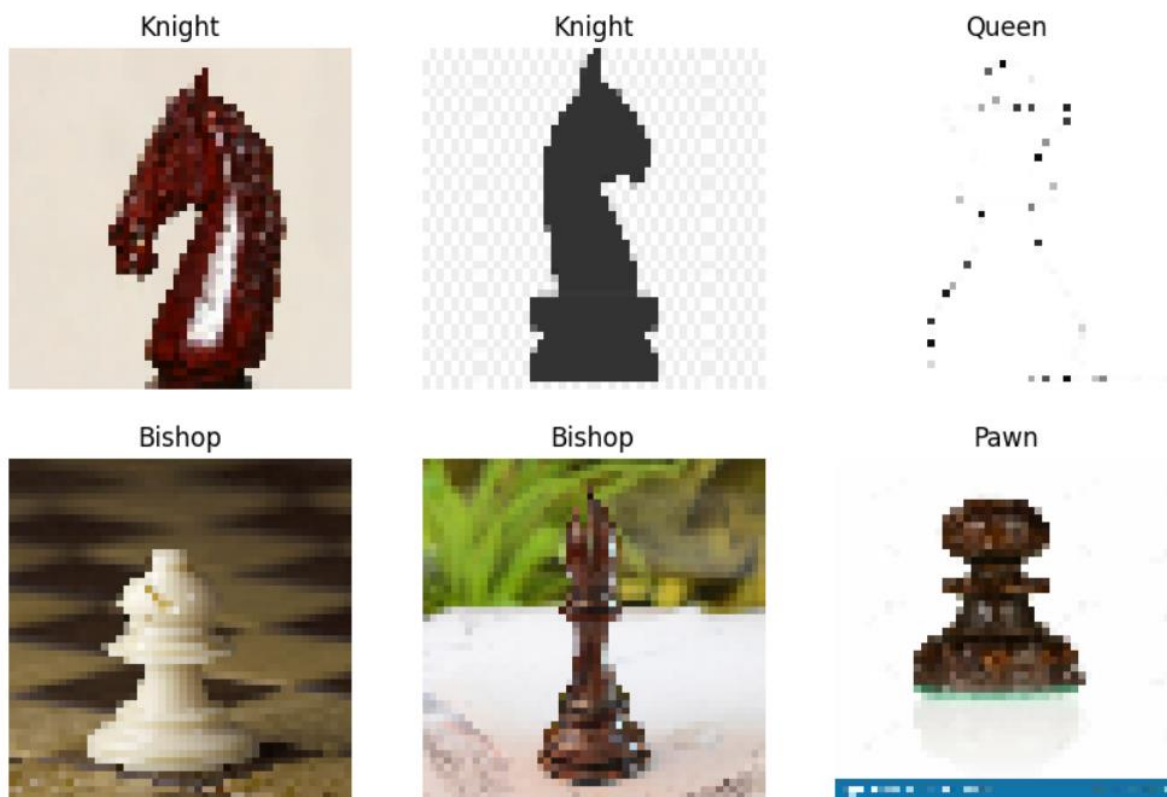
Next, we will display a few sample image, randomly, along with the labels.

Displaying random samples

```
[5] def display_samples(dataset, n_samples, classes_name):  
    plt.figure(figsize=(10, 10))  
    for images, labels in dataset.take(1):  
        for i in range(n_samples):  
            ax = plt.subplot(3, 3, i + 1)  
            plt.imshow(images[i].numpy().astype("uint8"))  
            plt.title(classes_name[np.argmax(labels[i])])  
            plt.axis("off")
```

```
[6] display_samples(train, 9, train.class_names)  
for images, labels in train.take(1):  
    for i in range(1):  
        print(images[i].shape)
```

(48, 48, 3)



These images were selected randomly from the dataset. Re-running the code block would show different images every time.

Finally, the number of classes under each label is counted.

```
✓ 4s [7] class_names = train.class_names
      labels = np.array([])
      for _, label in train:
          labels = np.concatenate((labels, np.argmax(label, axis=-1)))
      _, counts = np.unique(labels, return_counts=True)

✓ 0s [9] counts

      array([71, 62, 85, 83, 64, 77])
```

Creating Model

```
✓ 0s [8] Creating a Sequential model

      input_shape = (image_size[0], image_size[1], 3)
      reg = keras.regularizers.l2(0.0005)

      model = keras.Sequential()
      model.add(Conv2D(32, (3, 3), padding="same", activation="relu", input_shape=image_size + (3,), kernel_regularizer=reg))
      model.add(MaxPooling2D(pool_size=(2, 2)))

      model.add(Conv2D(64, (3, 3), padding="same", activation="relu", kernel_regularizer=reg))
      model.add(MaxPooling2D(pool_size=(2, 2)))

      model.add(Conv2D(128, (3, 3), padding="same", activation="relu", kernel_regularizer=reg))
      model.add(MaxPooling2D(pool_size=(2, 2)))

      model.add(Dropout(0.25))

      model.add(Flatten())
      model.add(Dense(128, activation='relu'))
      model.add(BatchNormalization())
      model.add(Dropout(0.5))
      model.add(Dense(len(train.class_names), activation='softmax'))
```

This is the main step of the project. We create a Keras Sequential model, into which we add steps like using convolutions, max pooling, dropout, flattening, normalization, etc. Each step has a complex set of parameters, which when adjusted properly, gives a decent CNN model, as can be seen in the `model.summary()`.

```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 32)	896
max_pooling2d (MaxPooling2D)	(None, 24, 24, 32)	0
conv2d_1 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 128)	0
dropout (Dropout)	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 128)	589952
batch_normalization (Batch Normalization)	(None, 128)	512
dropout_1 (Dropout)	(None, 128)	0

dense_1 (Dense)	(None, 6)	774
-----------------	-----------	-----

```
=====  
Total params: 684486 (2.61 MB)  
Trainable params: 684230 (2.61 MB)  
Non-trainable params: 256 (1.00 KB)  
=====
```

After this, the data is fitted into this model. I have used only 8 epochs for the sake of faster compilation, but this can be easily increased if needed.

```
model.compile(
    loss="categorical_crossentropy",
    optimizer="adam",
    metrics=["accuracy"]
)

epochs = 8
model.fit(
    train,
    epochs=epochs,
    validation_data=validation
);
```

Epoch 1/8
14/14 [=====] - 7s 229ms/step - loss: 2.6454 - accuracy: 0.1787 - val_loss: 5.3103 - val_accuracy: 0.2182
Epoch 2/8
14/14 [=====] - 6s 295ms/step - loss: 2.2835 - accuracy: 0.1810 - val_loss: 6.7845 - val_accuracy: 0.1182
Epoch 3/8
14/14 [=====] - 5s 204ms/step - loss: 2.1849 - accuracy: 0.2195 - val_loss: 2.4609 - val_accuracy: 0.1091
Epoch 4/8
14/14 [=====] - 6s 282ms/step - loss: 2.1823 - accuracy: 0.2240 - val_loss: 1.9794 - val_accuracy: 0.1636
Epoch 5/8
14/14 [=====] - 5s 207ms/step - loss: 2.1399 - accuracy: 0.2534 - val_loss: 2.2561 - val_accuracy: 0.1545
Epoch 6/8
14/14 [=====] - 6s 212ms/step - loss: 2.0531 - accuracy: 0.2353 - val_loss: 1.9447 - val_accuracy: 0.2455
Epoch 7/8
14/14 [=====] - 5s 212ms/step - loss: 2.0664 - accuracy: 0.2195 - val_loss: 2.1396 - val_accuracy: 0.2091
Epoch 8/8
14/14 [=====] - 6s 296ms/step - loss: 1.9154 - accuracy: 0.2964 - val_loss: 1.9109 - val_accuracy: 0.2909

The model is now trained with the image sets. We can see the accuracy improve with every epoch.

Evaluation

Now, we evaluate the model. We do so by comparing the performance of the model on the training set and the validation set.

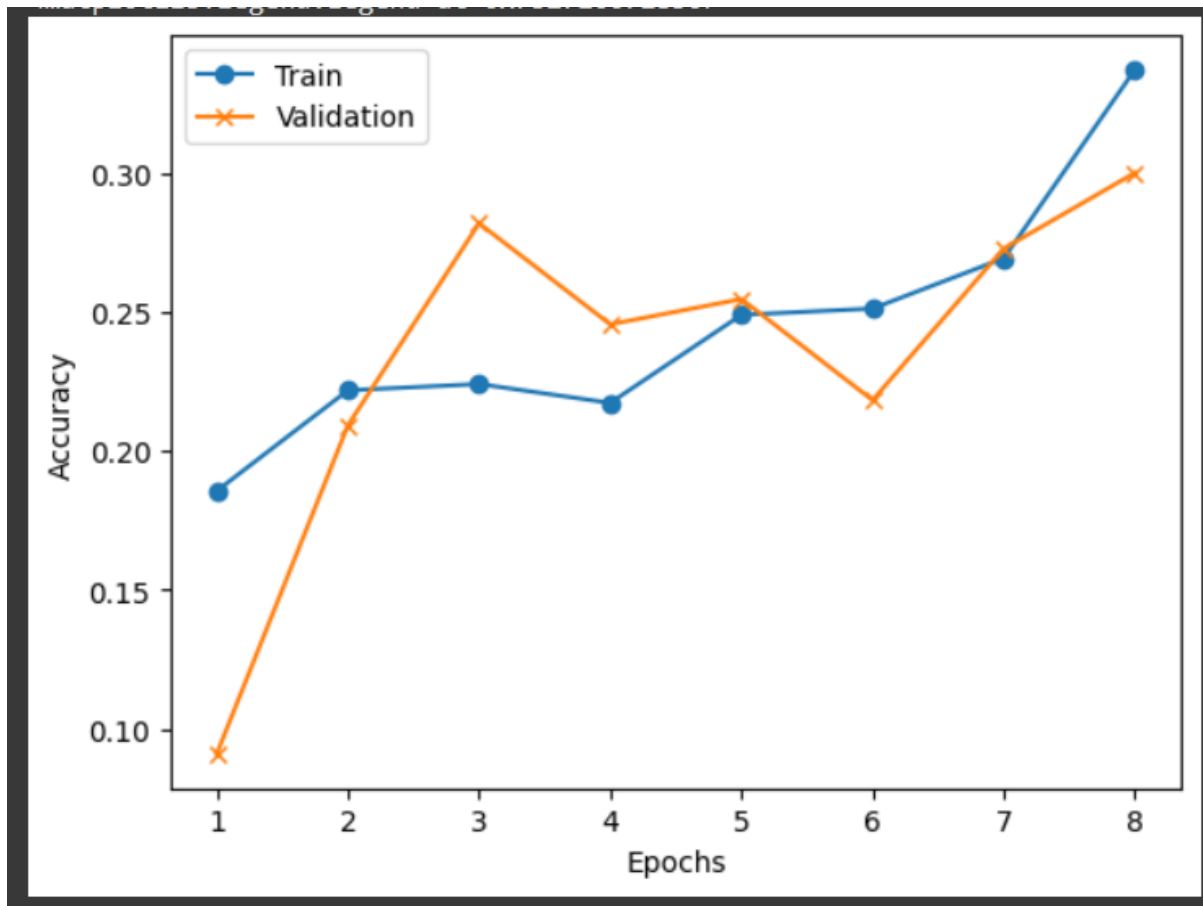
And finally, we visualize how well the CNN model has recognized each chess piece. We do so by using a heat map.

```
✓ 1s Evaluation of the model

epochs_range = [i+1 for i in range(epochs)]
plt.plot(epochs_range, model.history.history['accuracy'], '-o', label='Train')
plt.plot(epochs_range, model.history.history['val_accuracy'], '-x', label='Validation')

plt.ylabel('Accuracy')
plt.xlabel('Epochs')

plt.legend()
```



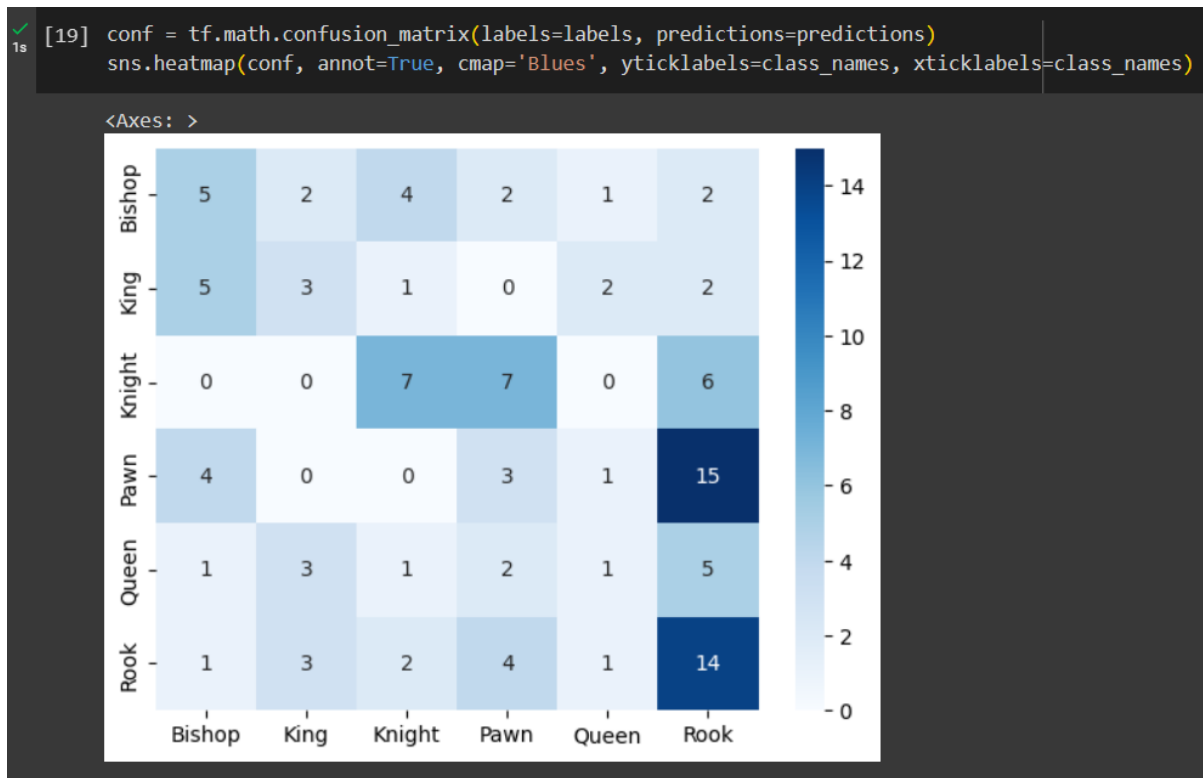
The Accuracy graph indicates that there is a scope for improvement in the model.

And now, we will concatenate the predications and labels, to properly display in the heat map.

```
✓ [18] y_pred = np.argmax(model.predict(validation), axis=-1)
2s

predictions = np.array([])
labels = np.array([])
for x, y in validation:
    predictions = np.concatenate([predictions, np.argmax(model.predict(x), axis=-1)])
    labels = np.concatenate([labels, np.argmax(y.numpy(), axis=-1)])

4/4 [=====] - 1s 32ms/step
1/1 [=====] - 0s 134ms/step
1/1 [=====] - 0s 54ms/step
1/1 [=====] - 0s 49ms/step
1/1 [=====] - 0s 118ms/step
```



This heatmap reflects curious results:

- I. The Rook was correctly identified by the model with high confidence.
- II. The model misclassified the Pawn as the Rook with the highest confidence.
- III. The Knight and Queen were also misclassified as Queen with high confidence.
- IV. The Knight was classified as a Pawn almost as much as it correctly classified it.
- V. The model also misclassified a Pawn as a Bishop with a confidence higher than the confidence of correctly identifying a Pawn.
- VI. The King was misclassified as a Bishop with higher confidence, than being correctly classified as the King itself.

CONCLUSION

Our endeavors in developing a chess piece identification model using Keras have yielded promising results. We have successfully constructed a robust model capable of accurately recognizing different types of chess pieces with high precision. This achievement holds significant implications for enhancing the chess-playing experience, enabling applications such as automated game analysis and educational tools for players of all skill levels. The validation set produces slightly lower accuracy score, but within acceptable range, and it improves with increase in epochs. In the heat map, we see that some pieces are recognized well, while others are not, which indicates that there is a scope of improvement.

FUTURE SCOPE

Looking ahead, there exist ample opportunities for further refinement and enhancement of our chess piece identification model. Future endeavors may include augmenting the dataset with additional variations, exploring advanced CNN architectures such as residual networks (ResNets) and attention mechanisms, and integrating real-time image recognition capabilities for interactive chess applications. Additionally, the deployment of the model in web or mobile applications could democratize access to intelligent chess assistance tools, fostering a deeper appreciation for the game among enthusiasts worldwide.

BIBLIOGRAPHY

1. [Chessman image dataset \(kaggle.com\)](https://kaggle.com/datasets/ashleykrumholz/chessman-image-dataset)
2. [Keras Documentation](https://keras.io/)
3. Deep Learning with Python by François Chollet

THANK YOU