Contents

[ABSTRACT 2](#_Toc37942108)

[1. INTRODUCTION 3](#_Toc37942109)

[1.1 Introduction & Objectives 3](#_Toc37942110)

[1.2 PURPOSE OF THE SYSTEM 4](#_Toc37942111)

[1.3 Existing system 4](#_Toc37942112)

[1.4 Proposed system 4](#_Toc37942113)

[2. Literature Survey 5](#_Toc37942114)

[3. SYSTEM SPECIFICATION 8](#_Toc37942115)

[3.3 FUNCTIONAL REQUIREMENTS 8](#_Toc37942116)

[3.5 FEASIBILITY STUDY 11](#_Toc37942117)

[3.5.1 ECONOMIC FEASIBILITY 11](#_Toc37942118)

[3.5.2 OPERATIONAL FEASIBILITY 11](#_Toc37942119)

[3.5.3 TECHNICAL FEASIBILITY 12](#_Toc37942120)

[4 SOFTWARE SPECIFICATIONS 12](#_Toc37942121)

[4.1 PYTHON 12](#_Toc37942122)

[3.6.1 Python Features 13](#_Toc37942123)

[3.6.2 Standard Data Types 16](#_Toc37942124)

[3.6.3Python Strings 16](#_Toc37942125)

[3.6.4 Python Lists 17](#_Toc37942126)

[3.6.5 Python Tuples 17](#_Toc37942127)

[3.6.6 Python Dictionary 17](#_Toc37942128)

[3.6.7 Python libraries 18](#_Toc37942129)

[3.6.8 Functions in Python 21](#_Toc37942130)

[3.6.9 Python Namespace 22](#_Toc37942131)

[3.6.10 Python-Data Base Communication 22](#_Toc37942132)

[4.2 Convolutional Neural Networks 25](#_Toc37942133)

[4.3 2-D Convolution Neural Network 26](#_Toc37942134)

[4.4 3D Convolutions 26](#_Toc37942135)

[5. SOFTWARE & HARDWARE REQUIREMENTS 36](#_Toc37942136)

[5.1 HARDWARE REQUIREMENTS: 36](#_Toc37942137)

[5.2 SOFTWARE REQUIREMENTS: 36](#_Toc37942138)

[6. CODING AND IMPLEMENTATION 37](#_Toc37942139)

[7. OUTPUT SCREEN: 49](#_Toc37942140)

# ABSTRACT

There is a great growing interest in the domain of deep learning techniques for identifying and classifying images with various datasets. An enormous availability of datasets (e.g. ChestX-Ray14 dataset) has developed a keen interest in deep learning. Pneumonia is a disease that is caused by various bacteria, virus etc. X-ray is one of the major diagnosis tools for diagnosing pneumonia. This research work mainly proposes a convolutional neural system (CNN) model prepared without any preparation to group and identify the occurrence of pneumonia disease from a given assortment of chest X-ray image tests. Dissimilar to different strategies that depend exclusively on more learning draws near or conventional carefully assembled systems to accomplish an amazing grouping execution, and developed a convolutional neural arrange model without any preparation to separate and character the images to decide whether an individual is suffering with pneumonia. This model could help alleviate the dependability and difficult challenges frequently confronted to manage therapeutic problems. In this paper, CNN algorithm has been used along with different data augmentation techniques for improving the classification accuracies which has been discussed to increase the performance which will help in improving the validation and training accuracies and characterization of exactness of the CNN model and accomplished various results. This experiment was carried out using python language and has shown improved outcomes.

# 1. INTRODUCTION

## 1.1 Introduction & Objectives

The risk of pneumonia is immense for many, especially in developing nations where billions face energy poverty and rely on polluting forms of energy. The WHO estimates that over 4 million premature deaths occur annually from household air pollution-related diseases including pneumonia. Over 150 million people get infected with pneumonia on an annual basis especially children under 5 years old. In such regions, the problem can be further aggravated due to the dearth of medical resources and personnel. For example, in Africa’s 57 nations, a gap of 2.3 million doctors and nurses exists. For these populations, accurate and fast diagnosis means everything. It can guarantee timely access to treatment and save much needed time and money for those already experiencing poverty.

Deep neural network models have conventionally been designed, and experiments were performed upon them by human experts in a continuing trial-and-error method. This process demands enormous time, know-how, and resources. To overcome this problem, a novel but simple model is introduced to automatically perform optimal classification tasks with deep neural network architecture. The neural network architecture was specifically designed for pneumonia image classification tasks. The proposed technique is based on the convolutional neural network algorithm, utilizing a set of neurons to convolve on a given image and extract relevant features from them.

In recent times, CNN-motivated deep learning algorithms have become the standard choice for medical image classifications although the state-of-the-art CNN-based classification techniques pose similar fixated network architectures of the trial-and-error system which have been their designing principle.

CNNs have an edge over DNNs by possessing a visual processing scheme that is equivalent to that of humans and extremely optimized structure for handling images and 2D and 3D shapes, as well as ability to extract abstract 2D features through learning. The max-pooling layer of the convolutional neural network is effective in variant shape absorptions and comprises sparse connections in conjunction with tied weights. When compared with fully connected (FC) networks of equivalent size, CNNs have a considerably smaller amount of parameters. Most importantly, gradient-based learning algorithms are employed in training CNNs and they are less prone to diminishing gradient problem. Since the gradient-based algorithm is responsible for training the whole network in order to directly diminish an error criterion, highly optimized weights can be produced by CNNs.

## 1.2 PURPOSE OF THE SYSTEM

We propose below methodology for solving the problem. Raw data collected would be pre-processed for missing data, anomalies and outliers. Then an algorithm would be trained on this data to create a model. This model would be used for forecasting the final results. ETL stands for Extract, Transform and load. It is a tool which is a combination of three functions. It is used to get data from one database and transform it into a suitable format. Data pre-processing is a data mining technique used to transform sample raw data into an understandable format. Real world collected data may be inconsistent, incomplete or contains an error and hence data pre-processing is required.

## 1.3 Existing system

The no of patients are growing day by day through various diseases, it’s difficult to determine the patients suffering from diseases as we are having a handful of doctors and limited equipment’s. India is one of the largest populated country, we need a huge medical assistance to support. We can’t recruit doctors in overnight so, we need a better diagnosing method to treat the patients.

Disadvantages:

* Not all doctors have the proper diagnosing tools to diagnose patients.
* Or there diagnosing may not be accurate
* It involves high cost and time.
* Detecting pneumonia in the critical stage of diagnosis can be life threatening.

## 1.4 Proposed system

This section deals with the detailed description of the applied methodology. The proposed pneumonia detection system using the ’Densely Connected Convolutional Neural Network’ (DenseNet-169) is described in Figure 2. The architecture of the proposed model has been divided into three different stages - the preprocessing stage, the feature-extraction stage and the classification stage.

# 2. Literature Survey

Deep learning has been termed as among the ten breakthrough technologies of 2013. It has become a leading machine-learning tool in medical imaging domain, including computer aided detection in chest radiography. In particular, deep convolutional neural networks (CNNs)(a type of deep learning approach) have been found well suited for analysing images. Although CNNs have been around for more than two decades, such networks have only become practical relatively recently. This is mainly due to the vast acceleration (approximately 40 times) enabled by Graphic Processing Unit (GPU).

CNNs were first applied in chest radiography for lung nodule detection in 1993. In recent years, there is a growing trend for the application of CNNs in detecting chest pathologies. The results of exploiting CNNs in chest disease detection are very promising. There are few studies that have reported the performance of CNNs in pneumonia detection. In Kermany et al. have exploited a pre-trained Inception v3 model to classify paediatric chest X-rays in to two classes: rmal’ and ‘pneumonia’. Classification into bacterial and viral to note that Kermany et al. used the same dataset [7] as the research presented in this paper, so performance comparison can be easily made between the two studies. In, Wang et al. used a localization algorithm to detect and locate pneumonia in chest X-rays. Wang et al. have reported area under the curve (AUC) of 0.633 for pneumonia detection. They released the largest publicly available Chest X-ray dataset with 112,120 frontal chest X-rays. The released dataset is research works. In, Rajpurkar et al. have used a 121-layer convolutional neural network for detecting pneumonia with AUC of 0.768. They call their model ‘CheXNet’ and it is trained and evaluated on the ‘ChestX-ray14’ dataset. In, Guan et al. placed special emphasis on two points:

1) a chest disease normally happens in small localized areas. Therefore, training CNNs on global images may be affected by irrelevant noisy areas.

2) Some chest X-rays are not properly aligned and this also affects the performance of CNN. For these

reasons, Guan et al. proposed an attention guided two- branch CNN for chest disease classification. During training, both the local and global cues are considered. The proposed system uses attention heat maps to mask the disease affected regions which are then used to train the network’s local branch. The proposed CNN produces high accuracy rates on the ‘ChestX-ray14’ dataset.

III. CNN ARCHITECTURE

CNNs basically center on the premise that the info will be included pictures. Such architectures would help in managing different types of data using various datasets. Fig 1 shows the flow diagram of all the layers that how each process works step by step. The major key contrasts is that the neurons present inside the CNN model are involved neurons composed into three measurements, the spatial dimensionality of the info (stature and the width) and the profundity. The profundity doesn't allude to the all out number of layers inside the ANN, yet the third element of an initiation volume. Not at all like standard ANNS, the neurons inside some random layer will just associate with a little area of the layer going before it. CNNs are contained three sorts of layers. These are convolutional layers pooling layers and completely associated layers. At the point when these layers are stacked, a CNN technique has been framed. The working of the CNN model has been categorized into four main functions as given below: 1. Firstly, there is an input layer which is used for holding the pixel values of the image. 2.Then, the convolution layer is there which helps in determining the output of several neurons and these neurons are being connected to the local regions. Then, the further calculation is being done by scalar product between their weights and with the regions which is connected to the input volume. After this the Rectified Linear Unit (ReLu) is there which has a function of applying an activation function which is done element wise like sigmoid function to the output which is produced by the activation of the previous layer. 3.Then, the pooling layer is there which is used to down sample the spatial dimensionality of the input and then it reduces the various parameters and shorten the image sometimes to its half within that activation. 4.The fully connected layers help in producing the various scores obtained from the activations. The main aim of this layer is that it takes the results from the convolution or pooling layer and then us that result to classify the image into a form of label. After this they pass the obtained result to the output layer, where each neuron will represent a classification label.

# 3. SYSTEM SPECIFICATION

## 3.3 FUNCTIONAL REQUIREMENTS

OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

• External Outputs, whose destination is outside the organization

• Internal Outputs whose destination is within organization and they are the

• User’s main interface with the computer.

• Operational outputs whose use is purely within the computer department.

• Interface outputs, which involve the user in communicating directly.

OUTPUT DEFINITION

The outputs should be defined in terms of the following points:

Type of the output

* Content of the output
* Format of the output
* Location of the output
* Frequency of the output
* Volume of the output
* Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

INPUT STAGES:

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

INPUT TYPES:

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

INPUT MEDIA:

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to:

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

ERROR AVOIDANCE

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

ERROR DETECTION

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

DATA VALIDATION

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

## 3.5 FEASIBILITY STUDY

An important outcome of the preliminary investigation is the determination that system requested is feasible. This is to identify the objectives of a new system. Before solving a problem one must know what the problem is. The study is carried out by a small group of people who are familiar with system analysis and design process. Fact finding techniques are used to gather the required information.

The three major areas consider while determining the feasibility of the project are

* Economic Feasibility
* Operational Feasibility
* Technical Feasibility

### 3.5.1 ECONOMIC FEASIBILITY

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

### 3.5.2 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.

Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.

Have the user been involved in the planning and development of the project?

Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project. Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

### 3.5.3 TECHNICAL FEASIBILITY

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many-detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc.

A number of issues have to be considered while doing a technical analysis.

i) Understand the different technologies involved in the proposed system:

Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system.

ii) Find out whether the organization currently possesses the required technologies:

Is the required technology available with the organization?

If so is the capacity sufficient?

For instance -

“Will the current printer be able to handle the new reports and forms required for the new system?”

# 4 SOFTWARE SPECIFICATIONS

## 4.1 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, SmallTalk, and 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.

### 3.6.1 Python Features

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the 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 a 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 enable 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 supports 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 data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Dynamic vs Static**

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

**Variables**

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

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

### 3.6.2 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 various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

### 3.6.3Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:] ) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

### 3.6.4 Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

### 3.6.5 Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ( [ ] ) and their elements and size can be changed, while tuples are enclosed in parentheses ( ( ) ) and cannot be updated. Tuples can be thought of as **read-only** lists.

### 3.6.6 Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

**Different modes in python**

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

### 3.6.7 Python libraries

**1.** Requests. The most famous http library written by kenneth reitz. It’s a must have for every python developer.

**2.** Scrapy. If you are involved in webscraping then this is a must have library for you. After using this library you won’t use any other.

**3.** wxPython. A gui toolkit for python. I have primarily used it in place of tkinter. You will really love it.

**4.** Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

**5.** SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

**6.** BeautifulSoup. I know it’s slow but this xml and html parsing library is very useful for beginners.

**7.** Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.

**8.** NumPy. How can we leave this very important library ? It provides some advance math functionalities to python.

**9.** SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

**10.** matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

**11.** Pygame. Which developer does not like to play games and develop them ? This library will help you achieve your goal of 2d game development.

**12.** Pyglet. A 3d animation and game creation engine. This is the engine in which the famous [python port](https://github.com/fogleman/Minecraft) of minecraft was made

**13.** pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI’s for my python scripts.

**14.** pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.

**15.** Scapy. A packet sniffer and analyzer for python made in python.

**16.** pywin32. A python library which provides some useful methods and classes for interacting with windows.

**17.** nltk. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But it’s capacity is beyond that. Do check it out.

**18.** nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

**19.** SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

**20.** IPython. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

**Numpy**

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.**

**• Offers Matlab**-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

**matplotlib**

• High quality plotting library.

**Python modules**

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

**Testing code**

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

he entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

### 3.6.8 Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

To carry out that specific task, the function might or might not need multiple inputs. When the task is carred out, the function can or can not return one or more values.There are three types of functions in python:

help() ,min() ,print().

### 3.6.9 Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e. the naming of people in firstname and familiy name (surname).

Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

* **global names** of a module
* **local names** in a function or method invocation
* **built-in names**: this namespace contains built-in functions (e.g. abs(), cmp(), ...) and built-in exception names

**Garbage Collection**

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

### 3.6.10 Python-Data Base Communication

Connector/Python provides a connect() call used to establish connections to the MySQL server. The following sections describe the permitted arguments for connect() and describe how to use option files that supply additional arguments.

A database is an organized collection of data. The data are typically organized to model aspects of reality in a way that supports processes requiring this information.

The term "database" can both refer to the data themselves or to the database management system. The Database management system is a software application for the interaction between users database itself.

Databases are popular for many applications, especially for use with web applications or customer-oriented programs

Users don't have to be human users. They can be other programs and applications as well. We will learn how Python or better a Python program can interact as a user of an SQLdatabase.   
  
This is an introduction into using SQLite and MySQL from Python.

The Python standard for database interfaces is the Python DB-API, which is used by Python's database interfaces.

The DB-API has been defined as a common interface, which can be used to access relational databases.

In other words, the code in Python for communicating with a database should be the same, regardless of the database and the database module used. Even though we use lots of SQL examples, this is not an introduction into SQL but a tutorial on the Python interface.

SQLite is a simple relational database system, which saves its data in regular data files or even in the internal memory of the computer, i.e. the RAM.

It was developped for embedded applications, like Mozilla-Firefox (Bookmarks), Symbian OS or Android.

SQLITE is "quite" fast, even though it uses a simple file. It can be used for large databases as well.

If you want to use SQLite, you have to import the module sqlite3. To use a database, you have to create first a Connection object.

The connection object will represent the database. The argument of connection - in the following example "companys.db" - functions both as the name of the file, where the data will be stored, and as the name of the database. If a file with this name exists, it will be opened.

It has to be a SQLite database file of course! In the following example, we will open a database called company.

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library.

For notes detailing the changes in each release of Connector/Python, see MySQL Connector/Python Release Notes.

MySQL Connector/Python includes support for:

Almost all features provided by MySQL Server up to and including MySQL Server version 5.7.

Converting parameter values back and forth between Python and MySQL data types, for example Python datetime and MySQL DATETIME. You can turn automatic conversion on for convenience, or off for optimal performance.

All MySQL extensions to standard SQL syntax.

Protocol compression, which enables compressing the data stream between the client and server.

Connections using TCP/IP sockets and on Unix using Unix sockets.

Secure TCP/IP connections using SSL.

Self-contained driver. Connector/Python does not require the MySQL client library or any Python modules outside the standard library

**Jupyter notebook:**

The **Jupyter** Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning

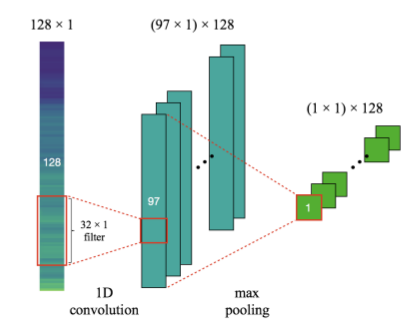
## 4.2 Convolutional Neural Networks

[Convolutional Neural Network (CNN)](https://missinglink.ai/guides/neural-network-concepts/convolutional-neural-network-build-one-keras-pytorch/)  models were developed for image classification, in which the model accepts a two-dimensional input representing an image’s pixels and color channels, in a process called feature learning.

This same process can be applied to one-dimensional sequences of data. The model extracts features from sequences data and maps the internal features of the sequence. A 1D CNN is very effective for deriving features from a fixed-length segment of the overall dataset, where it is not so important where the feature is located in the segment.

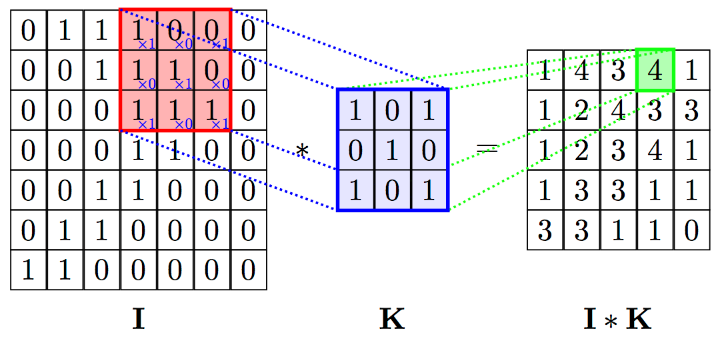
1D Convolutional Neural Networks work well for:

 Analysis of a time series of sensor data.

* Analysis of signal data over a fixed-length period, for example, an audio recording.
* Natural Language Processing (NLP), although [Recurrent Neural Networks](https://missinglink.ai/guides/neural-network-concepts/recurrent-neural-network-glossary-uses-types-basic-structure/) which leverage [Long Short Term Memory (LSTM)](https://missinglink.ai/guides/neural-network-concepts/deep-learning-long-short-term-memory-lstm-networks-remember/) cells are more promising than CNN as they take into account the proximity of words to create trainable patterns.
* 

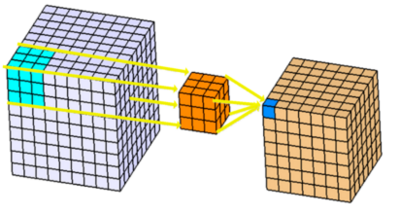
## 4.3 2-D Convolution Neural Network

This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. If use\_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.



## 4.4 3D Convolutions

3D convolutions applies a 3 dimentional filter to the dataset and the filter moves 3-direction (x, y, z) to calcuate the low level feature representations. Their output shape is a 3 dimentional volume space such as cube or cuboid. They are helpful in event detection in videos, 3D medical images etc. They are not limited to 3d space but can also be applied to 2d space inputs such as images.



In a convolution, small areas of an image are scanned and the probability that they belong to a filter class is assigned and translated to an activation map, a representation of the image layers. In a 3D CNN, the kernels move through three dimensions of data (height, length, and depth) and produce 3D activation maps.

3D Convolutional Neural Networks for Human Action Recognition. Convolutional neural networks (CNNs) are a type of deep model that can act directly on the raw inputs. However, such models are currently limited to handling 2D inputs. In this paper, we develop a novel 3D CNN model for action recognition.

In a convolution, small areas of an image are scanned and the probability that they belong to a filter class is assigned and translated to an activation map, a representation of the image layers. In a 3D CNN, the kernels move through three dimensions of data (height, length, and depth) and produce 3D activation maps

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.

This is done to in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

Max pooling is done by applying a *max filter* to (usually) non-overlapping subregions of the initial representation.

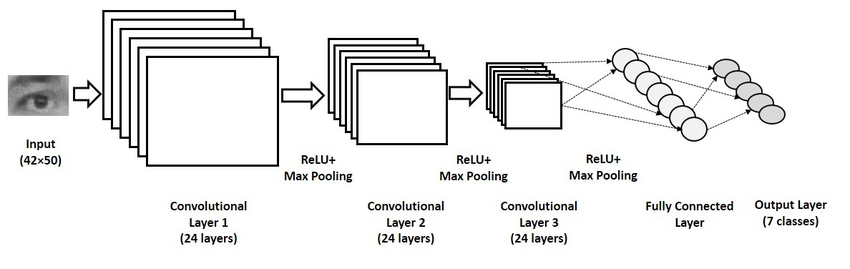
Let's say we have a 4x4 matrix representing our initial input.  
Let's say, as well, that we have a 2x2 filter that we'll run over our input. We'll have a stride of 2 (meaning the (dx, dy) for stepping over our input will be (2, 2)) and won't overlap regions.

For each of the regions represented by the filter, we will take the *max*of that region and create a new, output matrix where each element is the max of a region in the original input.

In order to make this super easy, with a nice pictorial representation - I give you this:



**Flattening** is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer. In other words, we put all the pixel data in one line and make connections with the final layer. And once again. What is the final layer for? The classification of ‘the cats and dogs.’



Adding multiple convolutional layers and pooling layers, the image will be processed for feature extraction. And there will be fully connected layers heading to the layer for softmax (for a multi-class case) or sigmoid (for a binary case) function. I didn’t mention the ReLu activation step, but there’s no difference with the activation step in ANN.

As the layers go deeper and deeper, the features that the model deals with become more complex. For example, at the early stage of ConvNet, it looks up for oriented line patterns and then finds some simple figures. At the deep stage, it can catch the specific forms of objects and finally able to detect the object of an input image.

**A dense layer** is a classic fully connected neural network layer: each input node is connected to each output node. A dropout layer is similar except that when the layer is used, the activations are set to zero for some random nodes. This is a way to prevent overfitting.

The activation function is a node that is put at the end of or in between Neural Networks. They help to decide if the neuron would fire or not. “The activation function is the nonlinear transformation that we do over the input signal

In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs. A standard computer chip circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on input.

**Use of Optimizer?**

Optimization algorithms helps us to minimize (or maximize) an Objective function (another name for Error function) E(x) which is simply a mathematical function dependent on the Model's internal learnable parameters which are used in computing the target values(Y) from the set of predictors(X) used in the model.

Role of an optimizer Optimizers update the weight parameters to minimize the loss function. Loss function acts as guides to the terrain telling optimizer if it is moving in the right direction to reach the bottom of the valley, the global minimum.

The softmax activation is normally applied to the very last layer in a neural net, instead of using ReLU, sigmoid, tanh, or another activation function. The reason why softmax is useful is because it converts the output of the last layer in your neural network into what is essentially a probability distribution.

The softmax activation is normally applied to the very last layer in a neural net, instead of using ReLU, sigmoid, tanh, or another activation function. The reason why softmax is useful is because it converts the output of the last layer in your neural network into what is essentially a probability distribution.

Softmax layer

A softmax layer, allows the neural network to run a multi-class function. In short, the neural network will now be able to determine the probability that the dog is in the image, as well as the probability that additional objects are included as well.

**Why do we use Softmax?**

The main advantage of using Softmax is the output probabilities range. The range will 0 to 1, and the sum of all the probabilities will be equal to one. If the softmax function used for multi-classification model it returns the probabilities of each class and the target class will have the high probability.

**What is Softmax classifier?**

The Softmax classifier uses the cross-entropy loss. The Softmax classifier gets its name from the softmax function, which is used to squash the raw class scores into normalized positive values that sum to one, so that the cross-entropy loss can be applied

**Dropout**

Dropout is a popular regularization technique for neural networks. Deep neural networks are particularly prone to overfitting.

Let us now see what dropout is and how it works.

In the words of Geoffrey Hinton, one of the pioneers of Deep Learning, ‘If you have a deep neural net and it's not overfitting, you should probably be using a bigger one and using dropout’.

Dropout is a technique where during each iteration of gradient descent, we drop a set of randomly selected nodes. This means that we ignore some nodes randomly as if they do not exist.

Each neuron is kept with a probability of q and dropped randomly with probability 1-q. The value q may be different for each layer in the neural network. A value of 0.5 for the hidden layers, and 0 for input layer works well on a wide range of tasks.

During evaluation and prediction, no dropout is used. The output of each neuron is multiplied by q so that the input to the next layer has the same expected value.

The idea behind Dropout is as follows − In a neural network without dropout regularization, neurons develop co-dependency amongst each other that leads to overfitting.

**Implementation trick**

Dropout is implemented in libraries such as TensorFlow and Pytorch by keeping the output of the randomly selected neurons as 0. That is, though the neuron exists, its output is overwritten as 0.

**Loss function**

Machines learn by means of a loss function. It’s a method of evaluating how well specific algorithm models the given data. If predictions deviates too much from actual results, loss function would cough up a very large number. Gradually, with the help of some optimization function, loss function learns to reduce the error in prediction. In this article we will go through several loss functions and their applications in the domain of machine/deep learning.

loss functions can be classified into two major categories depending upon the type of learning task we are dealing with — **Regression losses** and **Classification losses**.

 In classification, we are trying to predict output from set of finite categorical values i.e Given large data set of images of hand written digits, categorizing them into one of 0–9 digits.

Regression, on the other hand, deals with predicting a continuous value for example given floor area, number of rooms, size of rooms, predict the price of room.

**Binary cross entropy** measures how far away from the true value (which is either 0 or 1) the prediction is for each of the class to r to reach the bottom of the valley, the global minimum. each the bottom of the valley, the global minimum. es and then averages these class-wise errors to obtain the final loss

**Note**  
The block before the **Target** block must use Sigmoid as activation function.

METRICS

The **loss function** is used to optimize your model. This is the **function** that will get minimized by the optimizer. A **metric** is used to judge the performance of your model. This is only for you to look at and has nothing to do with the optimization process.

Categorical crossentropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only.

**Note**  
The block before the **Target** block must use the activation function ​Softmax.

Use categorical crossentropy​​

Use categorical crossentropy in classification problems where only one result can be correct.

​​Example:​ In the ​MNIST​​ problem where you have images of the numbers 0,1, 2, 3, 4, 5, 6, 7, 8, and 9. Categorical crossentropy gives the probability that an image of a number is, for example, a 4 or a 9.

Usage of metrics

A metric is a function that is used to judge the performance of your model. Metric functions are to be supplied in the metrics parameter when a model is compiled.

ImageDataGenerator Class for Pixel Scaling

The ImageDataGenerator class in Keras provides a suite of techniques for scaling pixel values in your image dataset prior to modeling.

The class will wrap your image dataset, then when requested, it will return images in batches to the algorithm during training, validation, or evaluation and apply the scaling operations just-in-time. This provides an efficient and convenient approach to scaling image data when modeling with neural networks.

The usage of the ImageDataGenerator class is as follows.

* 1. Load your dataset.
* 2. Configure the ImageDataGenerator (e.g. construct an instance).
* 3. Calculate image statistics (e.g. call the *fit()* function).
* 4. Use the generator to fit the model (e.g. pass the instance to the *fit\_generator()* function).
* 5. Use the generator to evaluate the model (e.g. pass the instance to the *evaluate\_generator()* function).

The ImageDataGenerator class supports a number of pixel scaling methods, as well as a range of data augmentation techniques. We will focus on the pixel scaling techniques and leave the data augmentation methods to a later discussion.

The three main types of pixel scaling techniques supported by the ImageDataGenerator class are as follows:

* **Pixel Normalization**: scale pixel values to the range 0-1.
* **Pixel Centering**: scale pixel values to have a zero mean.
* **Pixel Standardization**: scale pixel values to have a zero mean and unit variance.

**Normalize Images With ImageDataGenerator**

The ImageDataGenerator class can be used to rescale pixel values from the range of 0-255 to the range 0-1 preferred for neural network models.

Scaling data to the range of 0-1 is traditionally referred to as normalization.

This can be achieved by setting the rescale argument to a ratio by which each pixel can be multiplied to achieve the desired range.

In this case, the ratio is 1/255 or about 0.0039. For example:

# create generator (1.0/255.0 = 0.003921568627451)

datagen = ImageDataGenerator(rescale=1.0/255.0)

Target

**Target represents the desired output that we want our model to learn. In the case of a classification problem, the targets would be the labels of each of the examples in the training set. target\_size**: tuple of integers (height, width), default: (256, 256). The dimensions to which all images found will be resized.

**Batch Size:** Total number of training examples present in a single batch.

**Note:***Batch size and number of batches are two different things.*

**Batches**

You can’t pass the entire dataset into the neural net at once. So, you **divide dataset into Number of Batches or sets or parts.**

**Epoch**

An epoch represents a full pass over the entire training set, meaning that the model has seen each example once. An epoch is thus the total number of examples / batch size number of training iterations.

What is epoch and batch size?

one **epoch** = one forward pass and one backward pass of all the training examples. **batch size** = the number of training examples in one forward/backward pass.

What is steps per epoch?

An **epoch** usually means one iteration over all of the training data. For instance if you have 20,000 images and **a** batch size of 100 then the **epoch** should contain 20,000 / 100 = 200 **steps**. However I usually just set **a** fixed number of **steps** like 1000 **per epoch** even though I have **a** much larger data set

**Samples per epoch**

One **epoch** means that **each sample** in the training dataset has had an opportunity to update the internal model parameters. An **epoch** is comprised of one or more batches.

**Validation data**

A validation set is a subset of your dataset which contains examples available to a neural network to adjust the hyper parameters or the model architecture based on the validation loss.

The validation set is used during training to run validation examples through the model after each epoch

# 5. SOFTWARE & HARDWARE REQUIREMENTS

## 5.1 HARDWARE REQUIREMENTS:

* System : Intel I-3, 5, 7 Processor.
* Hard Disk : 500 GB.
* Floppy Drive : 1.44 Mb.
* Monitor : 14’ Colour Monitor.
* Mouse : Optical Mouse.
* RAM : 2Gb.

## 5.2 SOFTWARE REQUIREMENTS:

* Operating system : Windows 7,8,10 Ultimate, Linux, Mac.
* Front-End : Python.
* Coding Language : Python.
* Software Environment : Anaconda.

# 6. CODING AND IMPLEMENTATION

#conda install -c anaconda keras (which install tensorflow version 1.12.0)

import tensorflow as tf

print(tf.\_\_version\_\_)

# Building the CNN

# Importing the Keras libraries and packages'''

from tensorflow.keras.models import Sequential

#from tensorflow.keras.layers import Conv2D

#from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

# Initialising the CNN

classifier = Sequential()

# Step 1 - Convolution

# apply a 3x3 convolution with 32 output filters on a 64x64 image:

# input: 64x64 images with 3 channels -> (3, 64, 64) tensors.

# this applies 32 convolution filters of size 3x3 each.

classifier.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu'))

# now model.output\_shape == (None, 32, 64, 64)

# Step 2 - Pooling

#pool\_size: tuple of 2 integers, factors by which to downscale (vertical, horizontal).

#(2, 2) will halve the image in each dimension.

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Adding a second convolutional layer

# add a 3x3 convolution on top, with 32 output filters:

classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))

# now model.output\_shape == (None, 32, 64, 64)

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Step 3 - Flattening

# this converts our 3D feature maps to 1D feature vectors

classifier.add(Flatten())

# Step 4 - Full connection

classifier.add(Dense(output\_dim = 128, activation = 'relu'))

# Dense(64) is a fully-connected layer with 64 hidden units.

# in the first layer, you must specify the expected input data shape:

# here, 20-dimensional vectors.

#model.add(Dense(64, input\_dim=20, init='uniform'))

# first, let's define an image model that

# will encode pictures into 128-dimensional vectors.

# it should be initialized with pre-trained weights.

classifier.summary()

"""

Dense(32,input\_dim=16)

model.add(Dense(input\_layer\_neurons=16, hidden\_layer\_neurons=32,

kernel\_initializer='normal', activation='relu'))

model.add(Dense(12, input\_dim=8, init='uniform', activation='relu'))

It means 8 input parameters, with 12 neurons in the FIRST hidden layer.

interpolation: Interpolation method used to resample the image if the

target size is different from that of the loaded image.

"""

classifier.add(Dense(output\_dim = 1, activation = 'sigmoid'))

# Compiling the CNN

classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

#Fitting the CNN to the images

from keras.preprocessing.image import ImageDataGenerator

# this is the augmentation configuration we will use for training

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

"""

rotation\_range: is a value in degrees (0-180), a range within which to randomly rotate pictures

width\_shift and height\_shift are ranges (as a fraction of total width or height)

within which to randomly translate pictures vertically or horizontally

rescale: is a value by which we will multiply the data before any other processing.

Our original images consist in RGB coefficients in the 0-255, but such values would

be too high for our models to process (given a typical learning rate), so we target values

between 0 and 1 instead by scaling with a 1/255. factor.

shear\_range: is for randomly applying shearing transformations

zoom\_range: is for randomly zooming inside pictures

horizontal\_flip: is for randomly flipping half of the images horizontally --relevant

when there are no assumptions of horizontal assymetry (e.g. real-world pictures).

"""

# this is the augmentation configuration we will use for testing:

# only rescaling

test\_datagen = ImageDataGenerator(rescale = 1./255)

import os

os.chdir("E:/INFLAMMATION OF AIRSPACE MEDICAL ANALYSIS USING CNN/IMAGES")

# this is a generator that will read pictures found in

# subfolers of 'data/train', and indefinitely generate

# batches of augmented image data

training\_set = train\_datagen.flow\_from\_directory('train', # this is the target directory

target\_size = (64, 64),# all images will be resized to 64x64

batch\_size = 32,#when size of data should be less batch size should be less

class\_mode = 'binary')# since we use binary\_crossentropy loss, we need binary labels

# this is a similar generator, for validation data

test\_set = test\_datagen.flow\_from\_directory('test',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

# this will take long time

classifier.fit\_generator(training\_set,

samples\_per\_epoch = 5216,

nb\_epoch = 1, # nb\_epoch = 25,each epoch will take sample of 8000

validation\_data = test\_set,

nb\_val\_samples = 624)

"""

model.fit\_generator(generator(features, labels, batch\_size), samples\_per\_epoch=50, nb\_epoch=10)

Breaking it down:

generator(features, labels, batch\_size): generates batches of samples indefinitely

sample\_per\_epoch: number of samples you want to train in each epoch

nb\_epoch: number of epochs

As you can manually define sample\_per\_epoch and nb\_epoch , you have to provide

codes for generator . Here is an example:

Assume features is an array of data with shape (100,64,64,3) and

labels is an array of data with shape (100,1).

We use data from features and labels to train our model.

With the generator above, if we define batch\_size = 10 , that means it will randomly

taking out 10 samples from features and labels to feed into each epoch until an epoch hits 50

sample limit. Then fit\_generator() destroys the used data and move on repeating the same

process in new epoch.

One great advantage about fit\_generator() besides saving memory is user can integrate

random augmentation inside the generator, so it will always provide model with new data

to train on the fly.

"""

classifier.save\_weights('weights.h5') # always save your weights after training or during training

# In[ ]:

import numpy as np

from keras.preprocessing import image

test\_image = image.load\_img('val/NORMAL/NORMAL2-IM-1440-0001.JPEG', target\_size = (64, 64))

# This is test img

# first arg is the path

# img is 64x64 dims this is what v hv used in training so wee need to use exactly the same dims

# here also

test\_image

# In[ ]:

test\_image = image.img\_to\_array(test\_image)

# Also in our first layer below it is a 3D array

# Step 1 - Convolution

# classifier.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu'))

# this will convert from a 3D img to 3D array

test\_image # shld gv us (64,64,3)

test\_image = np.expand\_dims(test\_image, axis = 0)

# axis specifies the position of indx of the dimnsn v r addng

# v need to add the dim in the first position

test\_image # now it shld show (1,64,64,3)

result = classifier.predict(test\_image)

# v r trying to predict

result # gv us 1

training\_set.class\_indices

# In[ ]:

print(training\_set.class\_indices)

if result[0][0] == 1:

prediction = 'PNEUMONIA'

else:

prediction = 'NORMAL'

print('Model prediction is : ',prediction)

def predict(training\_set,classifier,loc):

import numpy as np

from keras.preprocessing import image

test\_image = image.load\_img(loc, target\_size = (64, 64))

test\_image.show()

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image,axis=0)

result = classifier.predict\_classes(test\_image)

print(training\_set.class\_indices)

if result[0][0] == 1:

prediction = 'PNEUMONIA'

else:

prediction = 'NORMAL'

print('Model prediction is : ',prediction)

predict(training\_set,classifier,'val/PNEUMONIA/person1950\_bacteria\_4881.JPEG')

# In[ ]:

#import numpy as np

#from keras.preprocessing import image

test\_image = image.load\_img('val/PNEUMONIA/person1950\_bacteria\_4881.JPEG', target\_size = (64, 64))

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

result = classifier.predict(test\_image)

training\_set.class\_indices

if result[0][0] == 1:

prediction = 'PNEUMONIA'

else:

prediction = 'NORMAL'

print(prediction)

# 7. OUTPUT SCREEN:

Epoch = 1 :

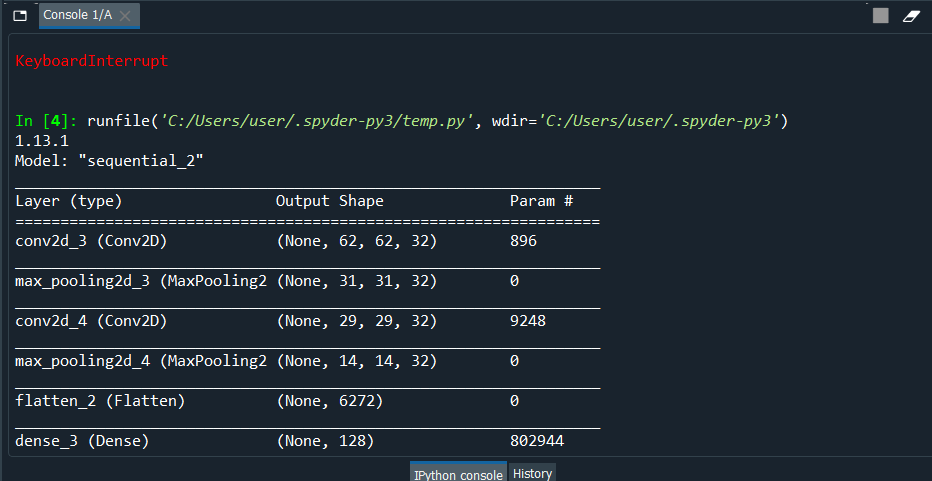


Fig i(i)

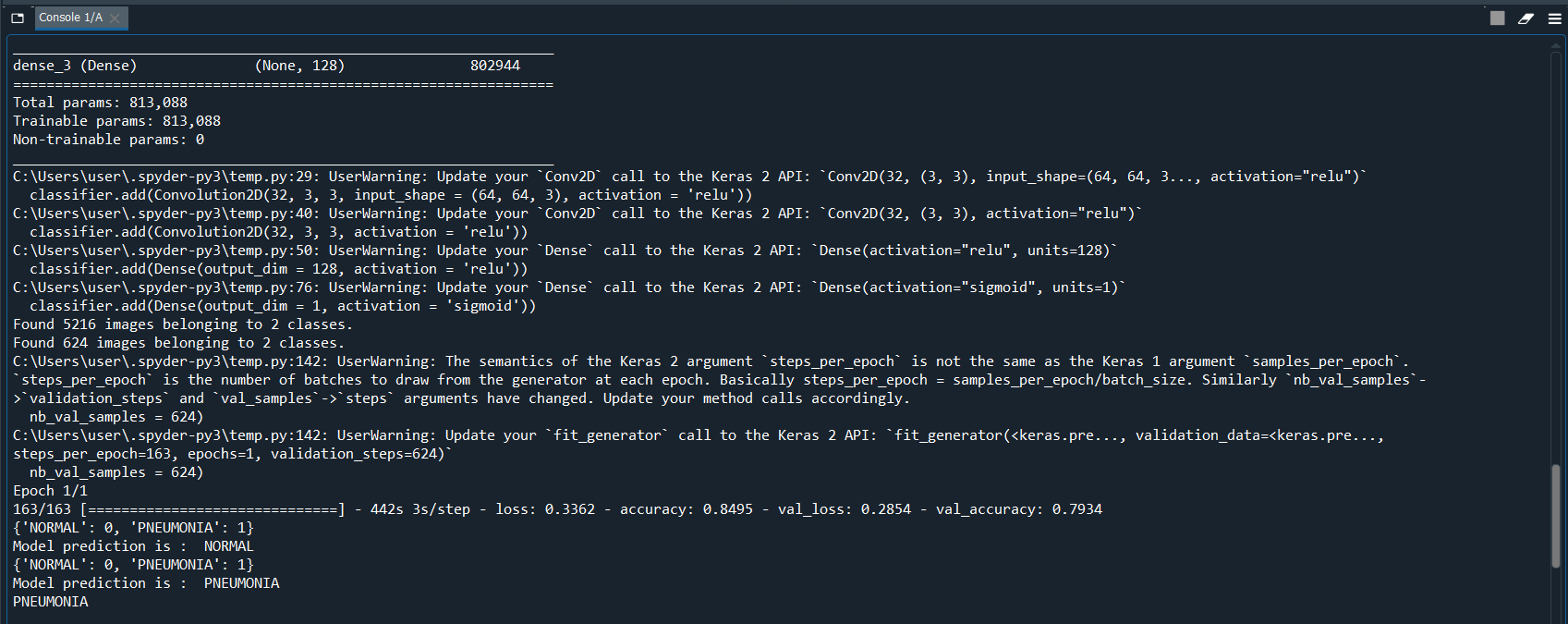


Fig i(ii)

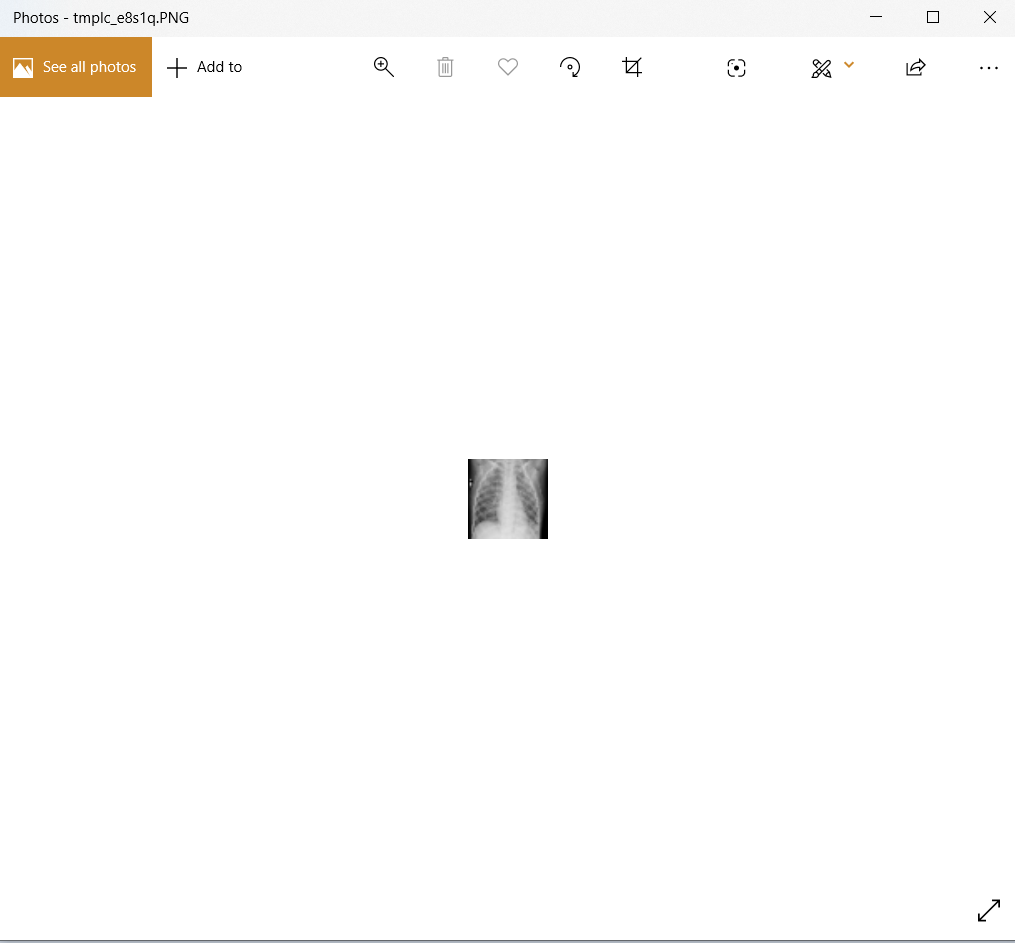


Fig i(iii)

Epoch = 5:

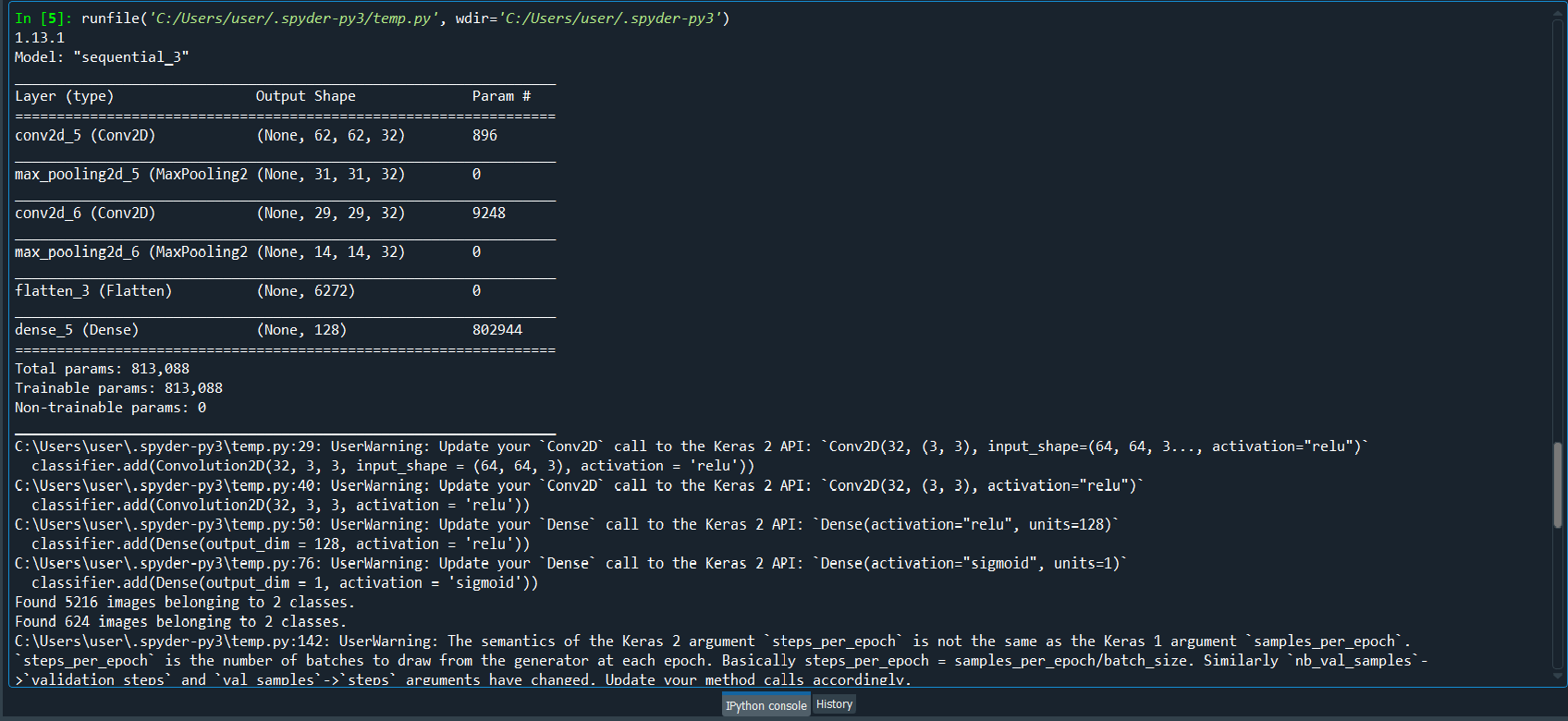


Fig ii(i)

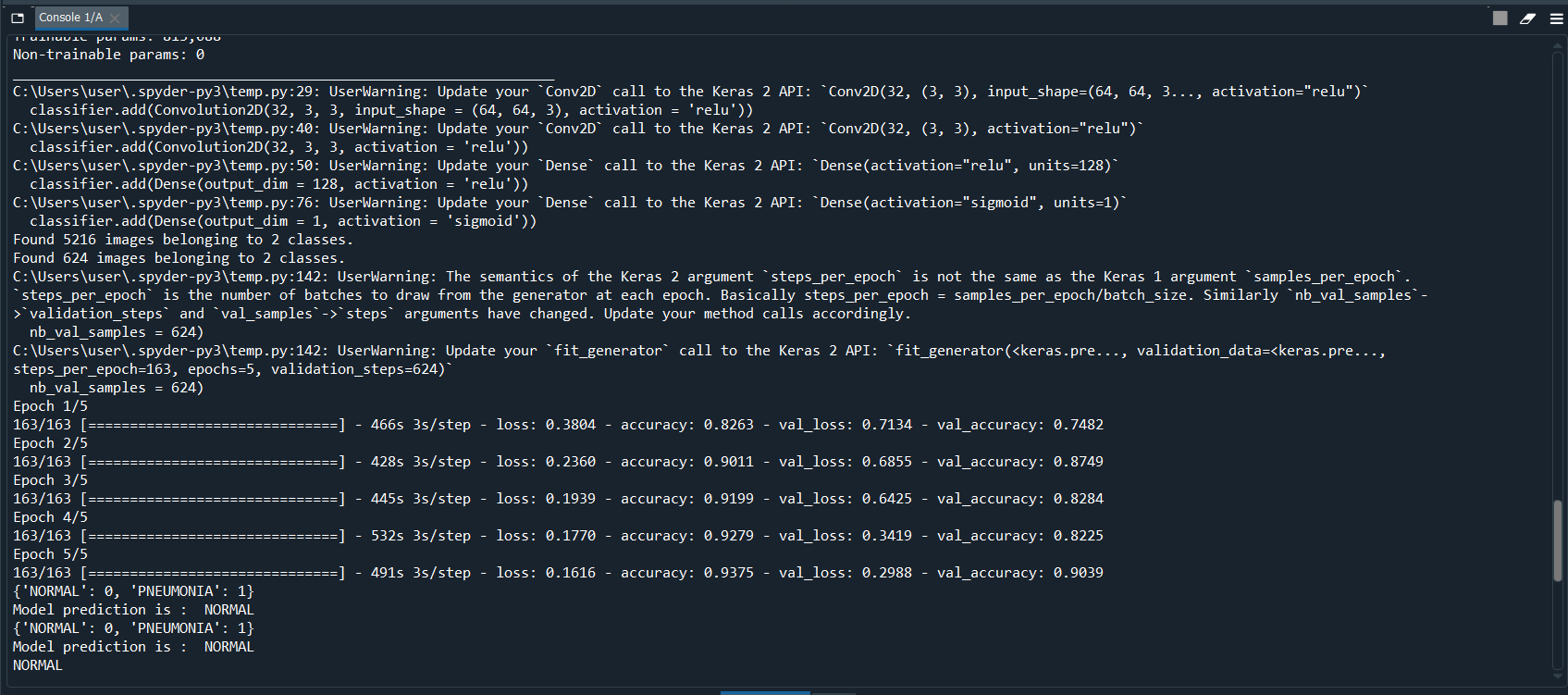


Fig ii(ii)

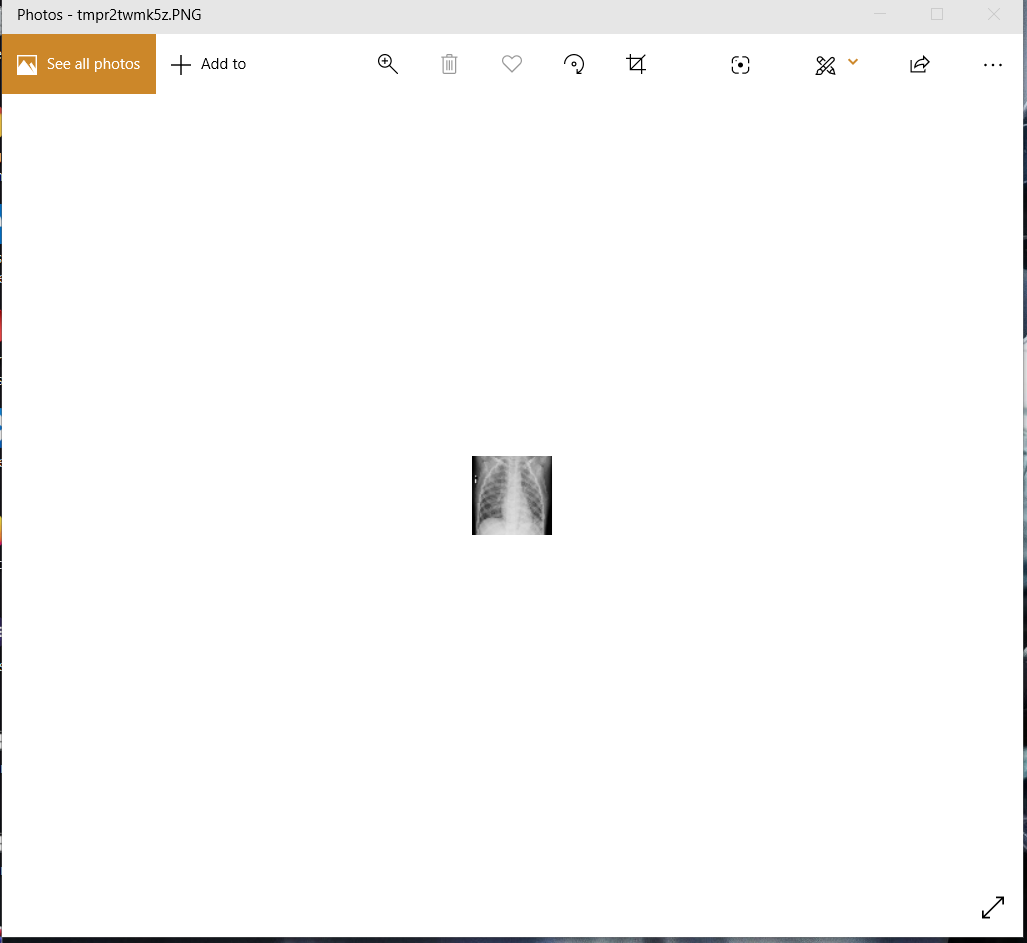


Fig ii(iii)