**IMAGE CAPTION GENERATOR USING CNN & LSTM**

**PROBLEM STATEMENT:**

Based on the content of an Image, this application will generate a caption for any natural image. Such an application might help blind people to see the world full with images. This model automatically generates natural language captions which then can be utilized for indexing and searching of images, tagging in social media, helping the visually impaired etc. The uses of such an application is immense. In this paper, we will build such an application using Convolution Neural Networks (CNN) for feature extraction and Long Short Term Memory (LSTM) for generating the captions.

**ABSTRACT:**

When we see an image, we can quickly recognize what is going on in the image, what objects are present and what they are doing. With the progress in Artificial Intelligence (AI), we are trying to do the same automatically by our computers. The need for such a system is increasing especially due to the advent of autonomous vehicles / semi-autonomous vehicles which involves reading and understanding millions of images. Automatically generating captions for any given image requires the use of Natural Language Processing (NLP) techniques and Neural Networks to classify the images. The ability for a computer to generate captions to an image has various business and individual benefits.

**INTRODUCTION**

Each and every day, we visually encounter thousands of images though our eyes and on the internet as well. Human brain is good enough to understand most of the images even when they do not have any descriptions assigned to them. On the other hand, for machines it is a difficult task to caption an image. Forming natural English captions to any image automatically is a challenging task. But, once it is done, it could have a significant impact in our society. An image caption should not only capture the objects/elements in them, but it should also be able to relate them with respect to their activities etc.

Automatically generating captions could help the visually impaired to understand and feel the situation in an image. It helps in enormous indexing of images done by Google LLC and others to create an image search engine. Social media platforms can use it for tagging and image captioning purposes. Military around the world might use them for reconnaissance purposes.

Traditional machine learning methods uses Local Binary Patterns (LBP) etc, are used. The features extracted are the passed on to any classification algorithm like Support Vector Machines (SVM) to classify the images. Such a model depends upon hand selected features and hence their use is not feasible for large datasets especially with large features. Manually designing features for a complex task can require enormous amounts of human labor and time.

Modern image captioning methods uses deep learning techniques does not requires us to specify all the features that we want. They automatically captures the features and hence such methods are highly scalable. Deep learning solves the problem of feature extraction and selection by representing the representation in the form in simpler representations in multiple layer (hence the name Deep Learning).

Convolution Neural Network (CNN) is widely used for automatic feature extraction from images which is generally followed by Long Short Term Memory (LSTM) to predict the captions one by one in sequence of words.

Convolution Neural Networks (CNN), unlike Multi Layer Perceptron (MLP) which are also know as vanilla neural networks do not have a fully-connected property. In MLP, the fully-connected property implies that each neuron is connected to every other neuron which makes the model prone to overfitting.

Convolution networks uses the mathematical operation convolution instead of matrix multiplication which the usual neural networks uses.

Long Short Term Memory (LSTM) cannot just process individual images, but can also process sequence of images or videos. It is a type of RNN model. An RNN is a type of neural network which is repeated again and again and the newer repetitions are only dependent on the previous one computation enabling the processing of a sequence of data. But, RNN poses the problem of vanishing gradient for multi layered model. It can make the model harder to train. To prevent the problem to vanishing gradients, we use Long Short Term Memory (LSTM) which is a modified version of RNN.

We merge both CNN and LSTM model to produce results using softmax. Softmax takes in a vector of numbers and converts them to probabilities which are then used for image generating results.

**LITERATURE REVIEW**

**[1] Sherstinsky, Alex. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network." *Physica D: Nonlinear Phenomena* 404 (2020): 132306. Print.**

This paper provides a mathematical understanding and differentiation between Recurring Neural Networks (RNN) and Long Short Term Memory (LSTM). An RNN model can be defined using Delay Differential Equations. An RNN can be transformed to an LSTM. This paper explains RNN and LSTM fundamental concepts.

**Summary:** This article is used to develop an LSTM model for our project which will be used for generating captions as a sequence processor.

**[2] Liu, Yu Han. "Feature Extraction and Image Recognition with Convolutional Neural Networks." *Journal of Physics: Conference Series* 1087 (2018): 062032. Print.**

In the age of Machine Learning and Artificial Intelligence, Image recognition is one of the most important tool. Its uses ranges from searching of images, to tagging objects in social media or vehicle driving assistant systems. The basic problem is to determine if an image has any object or not, and if it does have then which category does it belongs to i.e., feature extraction. That is where Convolution Neural Networks (CNN) comes into use. It is a type of feed-forward Artificial Neural Network which is widely used for image processing.

**Summary:** This article helps us in understanding and using the Convolution Neural Networks which we will use for feature extraction from the images.

**[3] Aung, San & Pa, Win & nwe, tin. (2020). "Automatic Myanmar Image Captioning using CNN and LSTM-Based Language Model." Proceedings of the 1st Joint SLTU and CCURL Workshop (SLTU-CCURL 2020), pages 139–143. Print.**

A natural scene image contains objects, colors, activities, attributes etc. Humans can differentiate between these elements of an image instantaneously. But, for a machine, it is a difficult task. Image captioning requires 2 components: Identifying objects and attributes; and understanding the relationships between those objects. We use a merged model with CNN and LSTM to generate the captions for Burmese language captions.

**Summary:** In this paper, we learn the importance and the need of a merged model to generate image captions for Burmese language which can be generalized for English as well.

**[4] Chollet, Francois. "Xception: Deep Learning with Depthwise Separable Convolutions." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). Print.**

An architecture in which Inception modules are replaced with depthwise seperable convolutions is known as an Xception model. A depthwise seperable convolution is an inception modeule but with a very large number of towers. Xception architecture is proven to outperform Inception(v3) architecture on small dataset but the performance improvements are very high on large datasets. They both have same number of parameters. Hence, the performance gain is due to higher efficiency.

**Summary:** In this paper, we learned the importance of Xception model. How Xception and Inception(v3) models differ from each other.

**SCOPE:**

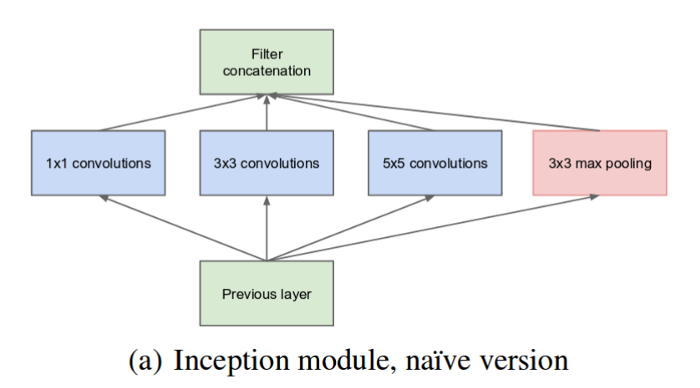
* Image searching using context.
* Visually impaired can listen to the caption of an Image.
* Classifying different photos in our phone gallery automatically for better management.
* Self driving vehicles like Tesla, Waymo etc. can use image captioning to better understand the images.
* Social media platforms like Facebook uses image captioning to understand the users and their behaviors.

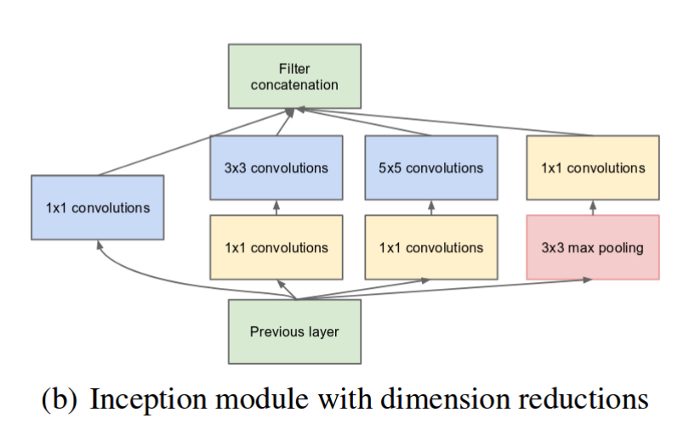
**EXISTING METHOD**

In most of the currently available systems, the model used for extracting features from the image is Inception deep learning model. But, Google in 2017 proposed a new model known as Xception which is a modified version of the Inception v3 model (also known as extreme inception). Xception, when compared to inception v3 is proven to provide a slightly better classification model for an image. It is trained on ImageNet dataset and it performs slightly better for small datasets but it outperforms Inception v3 by a significant margin when trained on large datasets.

Important features in an image can have high variations in area which makes it difficult to select the correct kernel size for the convolution layer. Going deeper to solve the problem can be computationally expensive as well. In 2014, a new deep convolutional neural network was proposed known as Inception to tackle the above mentioned problems. Rather than just going deeper, this architecture also went towards being more wider (in the form of levels) to increase efficiency. In doing so, it also reduces the need to go deeper as deeper models are prone to overfitting.

The architecture of the Inception model are as shown below:



****

As we can observed, in Fig (a), each Inception module consists of multiple parallel convolution which are then pooled together and passed to the next layer. In Fig (b), 1x1 convolution is performed on every convolution so as the improve computing efficiency.

**DISADVANTAGES:**

* Inception v3 has slightly lower accuracy when used for image classification compare to the proposed system.
* Computational complexity is higher than the proposed system.

**PROPOSED METHOD**

We propose a modified image caption generator which uses Xception model from keras library instead of Inception v3. Since, we are using Flickr8k dataset (which is a considerably small dataset), the difference between our proposed system and the existing system is minimal. But, if we use large datasets like flickr30k, MSCOCO etc., the increase in computational efficiency will be highly significant. We will generate captions by merging Xception and LSTM model.

**Advantages:**

* Higher Accuracy especially for large datasets.
* Low computational complexities.

**APPLICATIONS:**

* Visually Impaired people.
* Social media companies like Facebook, Twitter etc.
* Search engine for image search like Google, Yahoo, Bing, DuckDuckGo etc.
* Autonomous vehicle developing companies like Tesla, Google, Uber, Baidu etc.

**HARDWARE & SOFTWARE REQUIREMENTS**

# H/W Configuration:

# Processor - I3/Intel Processor

* RAM - 4GB (min)
* Hard Disk - 160GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**S/W Configuration:**

* Operating System : Windows 7/8/10
* Server side Script : Python, Anaconda
* IDE : PyCharm, Jupyter Notebook
* Libraries Used : Pandas, Numpy, Keras, Tensorflow, Pillow, OpenCV, Flask, Pickle, os.
* Dataset : Flickr8k dataset.

**ARCHITECTURE**

**Feature Extractor**

softmax

Output Caption

Merging using Dense layer

LSTM Layer

Embedding Layer (Reducing size to 256 nodes)

List of words

Reduction to 256 nodes using Dense

Feature Vector (2048 features)

CNN model Xception

Input Image

**Sequence Processer**

**SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS:**

Installing Python:

1. To download and install Python visit the official website of Python <https://www.python.org/downloads/> and choose your version.



1. Once the download is complete, run the exe for install Python. Now click on Install Now.
2. You can see Python installing at this point.
3. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

Installing PyCharm:

1. To download PyCharm visit the website <https://www.jetbrains.com/pycharm/download/> and Click the "DOWNLOAD" link under the Community Section.



1. Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
2. On the next screen, Change the installation path if required. Click “Next”.
3. On the next screen, you can create a desktop shortcut if you want and click on “Next”.
4. Choose the start menu folder. Keep selected JetBrains and click on “Install”.
5. Wait for the installation to finish.
6. Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.
7. After you click on "Finish," the Following screen will appear.



9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. The prompt will get open, with specified path, type “pip install package name” which you want to install (like numpy, pandas, seaborn, scikit-learn, matplotlib.pyplot)

Ex: pip install numpy



**MODULES**

**Upload:**

Upload the dataset of images (either .jpg or .png) to be read using OpenCV library.

**View:**

Uploaded dataset can be viewed.

**Preprocessing:**

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. Cleaning the data refers to removing the null values, filling the null values with meaningful value, removing duplicate values, removing outliers, removing unwanted attributes. If dataset contains any categorical records means convert those categorical variables to numerical values.

**Identifying Features:**

Identify the Independent variables by Feature Extraction, on which the captions are dependent.

**Train and Test Split:**

Our data is already separated in 3 parts, training data with 6000 images. Testing data with 1000 images and Validation data with 1000 images.

**Building the model:**

* To understand an image and generate captions, we are proposing a Deep learning based method.
* Deep learning can provide increased accuracy and decrease in computational power.
* We will use Convolution Neural Networks (CNN) and Long Short Term Memory (LSTM) to do so.
* Convolution Neural Network (CNN) is widely used for automatic feature extraction from images. It uses the mathematical operator convolution.
* LSTM is used to handle sequences of images at a time.
* It is a type of Recurrent Neural Network (RNN) but RNN have the problem of vanishing gradients which can make it harder to train.
* To counter the problem of vanishing gradients, we implement a model using Long Short Term Memory (LSTM).
* Long Short Term Memory (LSTM) cannot just process individual images, but can also process sequence of images or videos.
* It is a type of RNN model.
* An RNN is a type of neural network which is repeated again and again and the newer repetitions are only dependent on the previous one computation enabling the processing of a sequence of data.
* The Features from CNN and Sequences from LSTM are merged to generate the captions of any image.
* We merge both CNN and LSTM model to produce the captions.
* Softmax takes in a vector of numbers and converts them to probabilities which are then used for image generating results.
* Softmax converts logits into probabilities by taking the exponents from every output and then normalize each of these numbers by the sum of such exponents, such that the entire output vector adds up to one.

**Prediction:**

An image in natural scene is given as an input and a caption which describes the image in a sentence is generated.

**ALGORITHM:**

**CNN:**

Convolution Neural Networks is a type of Deep Neural Network. It is typically used for image visualization purposes. They can be defined as a regularized version of Multi Layer Perceptron where Multi Layer perceptron has all of its neurons fully connected which makes increases the likelihood of an overfitted model. It uses the mathematical operator convolution instead of the usual matrix multiplication.

Let us consider the basic part of any Neural Network:

**1. Input Layer:** It is the layer where we provide the input for the model. The number of features our input has is equal to the number of neuron in the input layer.

2. **Hidden Layer:** The input features are transferred to the hidden layer(s) where different processes/activities takes place. There can be multiple hidden layers. The layers undergoes mathematical operations like matrix multiplication, convolutions, pooling etc. along with an activation function.

3**. Output Layer:** They layer which is used to generate probability scores using sigmoid or softmax functions which is then converted to the output of our model.

A Convolutional Neural Network (CNN) is a Neural Network with at least one convolutional layers. They are used for various purposes including image processing, feature extraction, classification etc. Instead of taking an entire image at once to find features, it is more effective to look at small parts of the image. CNN is commonly used for feature extraction from images. CNNs also perform well at image segmentation, signal processing, speech recognition etc.

A CNN can also be implemented as a U-Net architecture, which are essentially two almost mirrored CNNs resulting in a CNN whose architecture can be presented in a U shape. U-nets are used where the output needs to be of similar size to the input such as segmentation and image improvement. Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map. These are linear transformations; each convolution is a type of affine function. In computer vision the input is often a 3 channel RGB image. For simplicity, if we take a greyscale image that has one channel (a two-dimensional matrix) and a 3x3 convolutional kernel (a two-dimensional matrix). The kernel strides over the input matrix of numbers moving horizontally column by column, sliding/scanning over the first rows in the matrix containing the images pixel values. Then the kernel strides down vertically to subsequent rows.

**Padding:**

To handle the edge pixels there are several approaches:

* Losing the edge pixels
* Padding with zero value pixels
* Reflection padding

Reflection padding is by far the best approach, where the number of pixels needed for the convolutional kernel to process the edge pixels are added onto the outside copying the pixels from the edge of the image. For a 3x3 kernel, one pixel needs to be added around the outside, for a 7x7 kernel then three pixels would be reflected around the outside. The pixels added around each side is the dimension, halved and rounded down.

Traditionally in many research papers, the edge pixels are just ignored, which loses a small proportion of the data and this gets increasing worse if there are many deep convolutional layers. For this reason, I could not find existing diagrams to easily convey some of the points here without being misleading and confusing stride 1 convolutions with stride 2 convolutions.

With padding, the output from an input of width w and height h would be width w and height h (the same as the input with a single input channel), assuming the kernel takes a stride of one pixel at a time.

**Strides:**

It is common to use a stride two convolution rather than a stride one convolution, where the convolutional kernel strides over 2 pixels at a time, for example our 3x3 kernel would start at position (1, 1), then stride to (1, 3), then to (1, 5) and so on, halving the size of the output channel/feature map, compared to the convolutional kernel taking strides of one. With padding, the output from an input of width w, height h and depth 3 would be the ceiling of width w/2, height h/2 and depth 1, as the kernel outputs a single summed output from each stride.

For example, with an input of 3x64x64 (say a 64x64 RGB three channel image), one kernel taking strides of two with padding the edge pixels, would produce a channel/feature map of 32x32.

The first step of creating and training a new convolutional neural network (Convnet) is to define the network architecture. This topic explains the details of Convent layers, and the order they appear in a ConvNet. For a complete list of deep learning layers and how to create them, see List of Deep Learning Layers. To learn about LSTM networks for sequence classification and regression, see Long Short-Term Memory Networks. To learn how to create your own custom layers, see Define Custom Deep Learning Layers. The network architecture can vary depending on the types and numbers of layers included.

**Image Input Layer:**

Create an image input layer using image input layer. An image input layer inputs images to a network and applies data normalization. Specify the image size using the input Size argument. The size of an image corresponds to the height, width, and the number of color channels of that image. For example, for a grayscale image, the number of channels is 1, and for a color image it is 3.

**Convolution Layer:**

Convolutional layers are the major building blocks used in convolutional neural networks.

A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image.

The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modeling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.

**Pooling Layer:**

It is common to periodically insert a Pooling layer in-between successive Convolution layers in a CNN architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged.

**Input Sequence Layer:**

A sequence input layer inputs sequence data to a network. In the general case, input sequences and output sequences have different lengths (e.g. machine translation) and the entire input sequence is required in order to start predicting the target. This requires a more advanced setup, which is what people commonly refer to when mentioning "sequence to sequence models" with no further context. Here's how it works:

In the general case, input sequences and output sequences have different lengths (e.g. machine translation) and the entire input sequence is required in order to start predicting the target. This requires a more advanced setup, which is what people commonly refer to when mentioning "sequence to sequence models". Here's how it works:

* A RNN layer (or stack thereof) acts as "encoder": it processes the input sequence and returns its own internal state. Note that we discard the outputs of the encoder RNN, only recovering the state.
* Another RNN layer (or stack thereof) acts as "decoder": it is trained to predict the next characters of the target sequence, given previous characters of the target sequence.

**LSTM Layer:**

Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they’ll have a hard time carrying information from earlier time steps to later ones. So if you are trying to process a paragraph of text to do predictions, RNN’s may leave out important information from the beginning.

During back propagation, recurrent neural networks suffer from the vanishing gradient problem. Gradients are values used to update a neural networks weights. The vanishing gradient problem is when the gradient shrinks as it back propagates through time. If a gradient value becomes extremely small, it doesn’t contribute too much learning.

With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long short Term Memory networks, a.k.a LSTMs have been observed as the most effective solution.

LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time.

**Fully Connected Layer:**

Fully connected layers connect every neuron in one layer to every neuron in another layer. The flattened matrix goes through a fully connected layer to classify the images.

Convolutional layers act to detect features that help classification, by picking up edges and curves and then from there detecting shapes and from there picking out ears for example, but as they can only filter images a dense layer is required to look at the output of the final convolutional neurons/filters and output a number (or numbers if one hot encoding is being used) as the classification. The outputs of all the neurons/filters in the last convolutional layer are joined together and then are “flattened” into 1D data ie it all becomes one row of data instead of the rows and columns of the data. The 1D data then acts as the input to the neuron(s) of the fully connected layer which performs a dot product of this input data and the neuron’s weights to produce a single number as output (a single number per neuron).

**Output Layers:**

**Softmax and Classification Layers:**

Softmax converts logits into probabilities by taking the exponents from every output and then norms each of these numbers by the sum of such exponents, such that the entire output vector adds up to one – every probability should be one. Generally, cross-entropy loss is the loss of such a problem in several classes. In the last layer of an image classification network such as CNN (e.g. VGG16) used in ImageNet competitions, softmax is also applied.

A softmax layer applies a softmax function to the input. A classification layer computes the cross-entropy loss for multi-class classification problems with mutually exclusive classes. Create a classification layer using classification Layer. For classification problems, a softmax layer and then a classification layer must follow the final fully connected layer. The softmax function is also known as the normalized exponential and can be considered the multi-class generalization of the logistic sigmoid function. For typical classification networks, the classification layer must follow the softmax layer. In the classification layer, train Network takes the values from the SoftMax function.

**STEPS FOR EXECUTING THE PROJECTS**

1. Import the Libraries/packages.
2. Load the labels of the training images.
3. Load the training images.
4. Load the captions of the training images. (We have 5 captions for each image in Flicker8k dataset)
5. Clean all the captions by lower casing them, removing punctuations etc.
6. Save all the captions in one image.
7. Compute the set of all unique words in the captions.
8. We resize the images.
9. Images are normalized.
10. We extract features of the images using pre trained CNN algorithm known as Xception.
11. We generate the vocabulary sequences using the LSTM model.
12. CNN and LSTM are merged together.
13. The model is then trained with the training dataset of images.
14. The model is evaluated by using the testing images.
15. The user interface is developed using the Flask architecture.
16. HTML5 and CSS3 templates are used to do so.
17. The webapp includes a login and registration system.
18. The data of the users is stored in MySQL database.
19. The generated caption can be viewed in the flask app.

**SYSTEM DESIGN**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**Collaboration Diagram:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization where as the collaboration diagram shows the object organization.



**DEPLOYMENT DIAGRAM**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application.



**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**COMPONENT DIAGRAM**

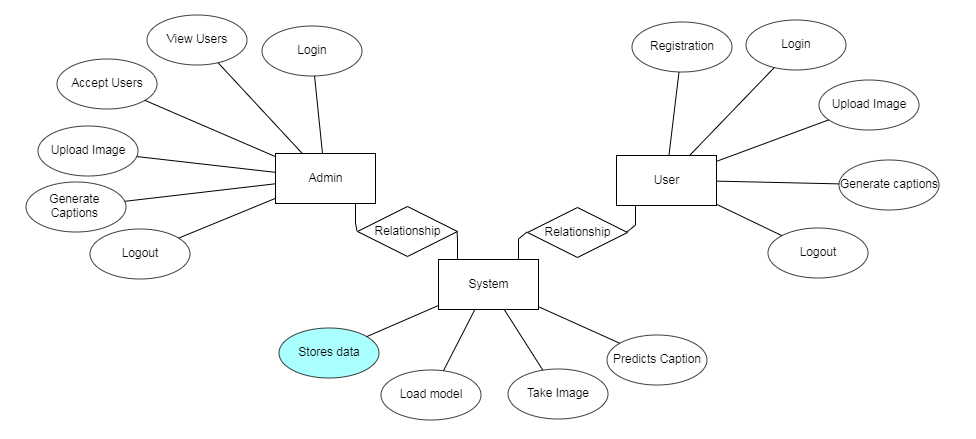
A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.



**ER DIAGRAM:**

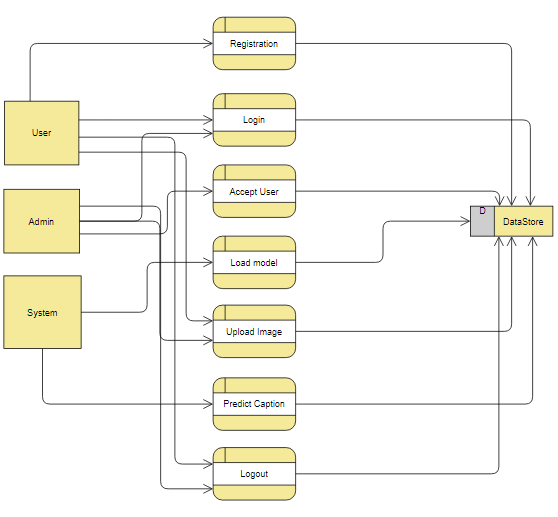
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

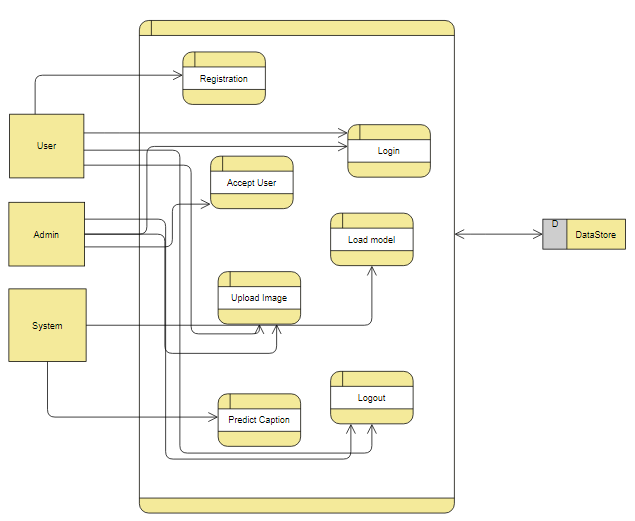
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.



**DFD DIAGRAM:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.





# INTRODUCTION TO PYTHON

* **Python**

### What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python.  This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

**Scripts are reusable**

Basically, a script is a text file containing the statements that comprise a Python program.  Once you have created the script, you can execute it over and over without having to retype it each time.

**Scripts are editable**

Perhaps, more importantly, you can make  different versions of the script by modifying the statements from one file to the next using a text editor.  Then you can execute each of the individual versions.  In this way, it is easy to create different programs with a minimum amount of typing.

**You will need a text editor**

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad, Microsoft WordPad, Microsoft Word,*or just about any word processor if you want to.

**Difference between a script and a program**

**Script:**

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, where as the applications they control are traditionally compiled to native machine code.

**Program:**

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived(e.g., compiled)

**Python**

what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

**Python concepts**

If your not interested in the thehows and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

• Open source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-ish to interface with C++ (via SWIG)

• Great interactive environment

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 it 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, 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.

**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.

**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

## Python 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.

## 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.

## 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.

## 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

**Some Python Libraries:**

1. Pandas
2. Numpy
3. Matplotlib
4. Seaborn
5. OpenCV
6. Keras
7. TensorFlow
8. NLTK
9. Scikit-Learn
10. SciPY
11. BeautifulSoup
12. TextBlob
13. Pillow
14. Request
15. SQLAlchamy
16. PyTorch
17. Selenium

**Pandas:**

* Pandas provide us with many Series and DataFrames. It allows you to easily organize, explore, represent, and manipulate data.
* Smart alignment and indexing featured in Pandas offer you a perfect organization and data labeling.
* Pandas has some special features that allow you to handle missing data or value with a proper measure.
* This package offers you such a clean code that even people with no or basic knowledge of programming can easily work with it.
* It provides a collection of built-in tools that allows you to both read and write data in different web services, data-structure, and databases as well.
* Pandas can support JSON, Excel, CSV, HDF5, and many other formats. In fact, you can merge different databases at a time with Pandas.

**Numpy:**

* Arrays of Numpy offer modern mathematical implementations on huge amount of data. Numpy makes the execution of these projects much easier and hassle-free.
* Numpy provides masked arrays along with general array objects. It also comes with functionalities such as manipulation of logical shapes, discrete Fourier transform, general linear algebra, and many more.
* While you change the shape of any N-dimensional arrays, Numpy will create new arrays for that and delete the old ones.
* This python package provides useful tools for integration. You can easily integrate Numpy with programming languages such as C, C++, and Fortran code.
* Numpy provides such functionalities that are comparable to MATLAB. They both allow users to get faster with operations.

**Matplotlib:**

* Matplotlib can create such quality figures that are really good for publication. Figures you create with Matplotlib are available in hardcopy formats across different interactive platforms.
* You can use MatPlotlib with different toolkits such as Python Scripts, IPython Shells, Jupyter Notebook, and many other four graphical user interfaces.
* A number of third-party libraries can be integrated with Matplotlib applications. Such as seaborn, ggplot, and other projection and mapping toolkits such as basemap.
* An active community of developers is dedicated to helping you with any of your inquiries with Matplotlib. Their contribution to Matplotlib is highly praisable.
* Good thing is that you can track any bugs, new patches, and feature requests on the issue tracker page from Github. It is an official page for featuring different issues related to Matplotlib.

**Seaborn:**

Seaborn is built on top of Python’s core visualization library Matplotlib. It is meant to serve as a complement, and not a replacement. However, Seaborn comes with some very important features. Let us see a few of them here. The features help in −

* Built in themes for styling matplotlib graphics
* Visualizing univariate and bivariate data
* Fitting in and visualizing linear regression models
* Plotting statistical time series data
* Seaborn works well with NumPy and Pandas data structures
* It comes with built in themes for styling Matplotlib graphics

In most cases, you will still use Matplotlib for simple plotting. The knowledge of Matplotlib is recommended to tweak Seaborn’s default plots.

**OpenCV:**

* OpenCV is an ideal image processing package that allows you to both read and write images at the same time.
* Computer Vision allows you to rebuild, interrupt, and comprehend a 3D environment from its respective 2D environment.
* This package allows you to diagnose special objects in any videos or images. Objects such as faces, eyes, trees, etc.
* You can also save and capture any moment of a video and also analyze its different properties such as motion, background, etc.
* OpenCV is compatible with many operating systems such as Windows, OS-X, Open BSD, and many others.

**Keras:**

* Keras is a powerful python library. It is capable of running on Microsoft Cognitive Toolkit, PaidML, TensorFlow, and other platforms as well.
* This python library features a variety of implementations from neural network forming blocks – functions, layers, optimizers, objectives, and others.
* Keras also features many useful tools that allow you to work with different images and texts easily.
* It doesn’t only support neural networks only but also provides a fully supportive environment for convolutional and re-current neural networks.
* Using Keras, you can build deep models for smartphones – both Android and iOS or for Java Virtual Machine also.

**TensorFlow:**

* TensorFlow uses automatic high-performance APIs such as – Keras. It offers an immediate iteration of machine learning models.
* This library features eager execution, which allows you to create, manipulate machine learning models, and make the debugging way easier.
* With TensorFlow, you can easily move your ML models in clouds, on any device and on-premises in any browser.
* TensorFlow comes with an easy to learn architecture. You can easily develop your concept into code and make your publications even easier.
* It has a solution to all of your common machine learning issues. You can easily implement it and go for giving your best.

**NLTK (Natural Language Tool Kit):**

* The text processing libraries of NLTK allow classification, tagging, tokenization, stemming, parsing, and semantic reasoning as well.
* NLTK contains a graphical illustration of data science. It also comes with a handbook for guiding through the principles of language processing for NLTK.
* It is open source and contains over fifty corpora and lexical resources such as open multilingual wordnet, question classification, SentiWordNet, SEMCOR, Stopwords Corpus, and many more.
* NLTK also features structure types, structure strings parsing, features different pathways, and re-entrance as well.
* This toolkit comes with a dynamic discussion forum where you can discuss and bring up any issues related to language NLTK.

**Scikit-Learn:**

* Scikit Learn comes with a clean and neat API. It also provides very useful documentation for beginners.
* It comes with different algorithms – classification, clustering, and regression. It also supports random forests, k-means, gradient boosting, DBSCAN and others
* This package offers easy adaptability. Once you get well with the general functionalities of Scikit Learn, switching to other platforms will be no problem at all.
* Scikit Learn offers easy methods for data representation. Whether you want to present data as a table or matrix, it is all possible with Scikit Learn.
* It allows you to explore through digits that are written in hands. You can not only load but also visualize digits-data as well.

**SciPY:**

* Scipy contains different modules. These modules are suitable for optimization, integration, linear algebra, and statistics, as well.
* It makes the best use of Numpy arrays for general data structures. In fact, Numpy is an integrated part of Scipy.
* Scipy can handle 1-d polynomials in two ways. Whether you can use poly1d class from numpy or you can use co-efficient arrays to do the job.
* High-level scipy contains not only numpy but also numpy.lib.scimath as well. But it is better to use them from their direct source.
* A supporting community of Scipy is always there to answer your regular questions and solve any issues if aroused.

**BeautifulSoup:**

* BeautifulSoup can easily parse data out of HTML and XML. However, to do so, it needs a package and an exterior parser.
* It can be easily taught and learned. Parsing can be nicely done with simple html.parser command.
* BeautifulSoup4 comes with good support both for Python 2 and 3. However, BeautiSoup3 works with Python 2 only.
* Moreover, it offers users proper documentation of the package, which helps us to learn things quite fast.
* While working with BeautifulSoup, if you ever need any support, there is a large community to help you at an instance.

**TextBlob:**

* TextBlob offers quite straight-forward tokenization. Tokenization is the process of dividing a large paragraph into many words or sentences.
* With TextBlob, it is easier than ever convert the words to their original form as they were in the dictionary. The process is called Lemmatization.
* This library offers you easily have Parts of Speech (PoS) tagging. However, this feature is noticeable in other NLP libraries, as well.
* With TextBlob, by using simple pluralize or singularize procedures, you can transform your text into single or plural.
* Also, you can easily extract different noun phrases in TextBlob using a simple noun\_phrase attribute.
* TextBlob also offers you word/phrase counts, uppercase and lowercase conversion, spelling correction, translation, N-grams detection, and many more.

**Pillow:**

* Using Pillow, you can not only open and save images but also influence the environment of images as well.
* Pillow supports a lot of file types such as PDF, WebP, PCX, PNG, JPEG, GIF, PSD, WebP, PCX, GIF, IM, EPS, ICO, BMP, and many others as well.
* With Pillow, you can easily create thumbnails for images. Thumbnails bear most of the valuable aspects of your image.
* Pillow supports a collection of image filters – FIND\_EDGES, DETAIL, SMOOTH, BLUR, CONTOUR, SHARPEN, SMOOTH\_MORE, and others.
* Pillow offers great support from the community who are eager to answer, challenge, and work through any of your inquiries.

**Requests:**

* Using basic Python Dictionaries in Requests, you can add parameters, headers, multi-part files, and form data as well.
* It is an easy library with tons of features that allow you to address custom headers, SSL certificate verifications, and sweep parameters towards URLs.
* With Requests, you can easily upload multiple files at a time. It allows you to work in a faster and efficient environment.
* Requests features automatic decompression that allows you to restore and revive compressed data into its authentic form in no time.
* Enjoy the benefits of HTTP proxy support with Requests. And allow your users with a faster and simpler route to your files and pages.
* Requests also features with value cookies, Unicode response bodies, Basic/Digest authentication, thread safety, connection pooling, and many more.

**SQLAlchamy:**

* SQLAlchemy is featured with a fully-featured core. It comes with SQL based abstraction toolkits.
* Another component of SQLAlchemy – ORM manages the insert/ update/ delete functionalities into a row to deliver them in a batch.
* SQLAlchemy makes communication between Python language and databases easier. It fastens the communication as well.
* It supports almost all modern platforms, including – Python 2.5 and above, Jython and Pypy as well.
* With SQLAlchemy, you can map classes in different ways. You can also develop database schemes and object models from scratch.

**PyTorch:**

* PyTorch uses TorchScript, which offers a flexible and simple eager mode. You can evaluate different functions and operations instantly.
* While in the graph mode, PyTorch provides absolute transitioning, fast optimizations, and offers a C++ run-time environment.
* PyTorch has a good support for async. execution for cumulative operations. This way, you can boost up your project performance.
* This library also allows P2P (Peer to Peer) communication, which can be gained by both Python and C++.
* PyTorch can be used with other popular libraries, as well. You can easily integrate it with libraries/packages like Cython and Numba.
* With PyTorch, you can get direct access to platforms, visualizers, and runtimes that are compatible with ONNX**.**

**Selenium:**

Selenium is an open source tool for web application testing. Selenium supports multiple languages including Python. Using Selenium with Python, we can communicate with browser, send keys and get the values. Python supports multiple browsers and we can compose our code based on our need.

* Simpler Installation process
* More realistic browser interaction
* A separate component such as RC server is inessential
* Faster Execution time
* Open Source
* Capability to run tests across different browsers
* Supports multiple operation systems
* Supports mobile devices
* Capability to execute tests in parallel

**Python class and objects**

These are the building blocks of OOP. class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

**Inheritance**

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by subclassing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components.

As we’ve seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python’s object model and the object interfaces are more consistent to Python’s built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python’s built-in operation methods

**Exceptions**

I’ve talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a KeyError exception.

• Searching a list for a non−existent value will raise a ValueError exception

. • Calling a non−existent method will raise an AttributeError exception.

• Referencing a non−existent variable will raise a NameError exception.

• Mixing datatypes without coercion will raise a TypeError exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don’t need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don’t put any overhead on the code processing.

Because exceptions aren’t supposed to happen very often, they aren’t processed until they occur.

Exceptions can be thought of as a special form of the if/elif statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren’t processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won’t have to do this very often but it’s nice to have the option when necessary.

However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

**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.

**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().

## 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).

An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames.   
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.

**Python XML Parser**

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs The Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX : Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

**Python Web Frameworks**

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

## Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format templating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

## Comparing web frameworks

There is also a repository called [compare-python-web-frameworks](https://github.com/mattmakai/compare-python-web-frameworks) where the same web application is being coded with varying Python web frameworks, templating engines and object.

## Web framework resources

* When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
* Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
* what is a web framework? is an in-depth explanation of what web frameworks are and their relation to web servers.
* Djangovs Flash vs Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
* This fascinating blog post takes a look at the  code complexity of several Python web frameworks by providing visualizations based on their code bases.
* Python’s web frameworks benchmarks  is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
* What web frameworks do you use and why are they awesome? is a language agnostic Reddit discussion on web frameworks. It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
* This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?". The votes aren't as important as the list of the many frameworks that are available to Python developers.

## Web frameworks learning checklist

1. Choose a major Python web framework (Django or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the [deployment](https://www.fullstackpython.com/deployment.html) section to make it accessible on the web.

**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 anSQLdatabase.   
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 developed 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 datetimeand 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.

**2. SYSTEM STUDY**

**2.1 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

### 6. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**SYSTEMTEST**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**6.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**6.3 Acceptance Testing**

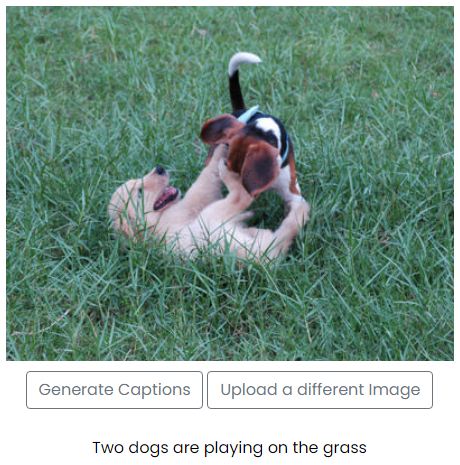
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

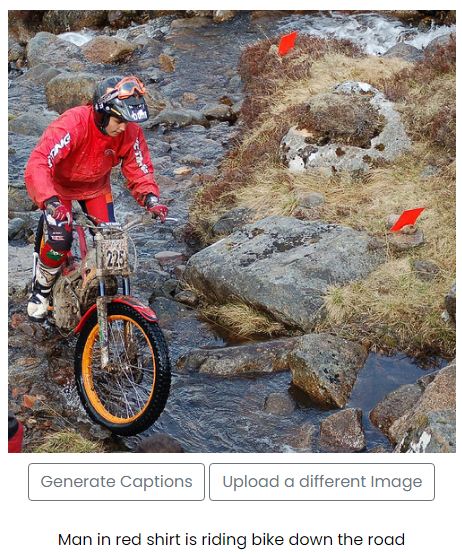
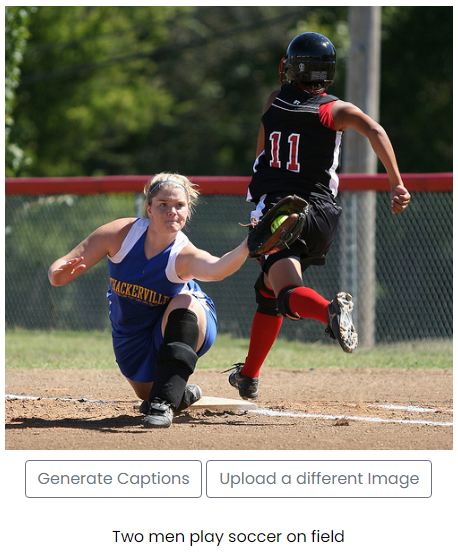
**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

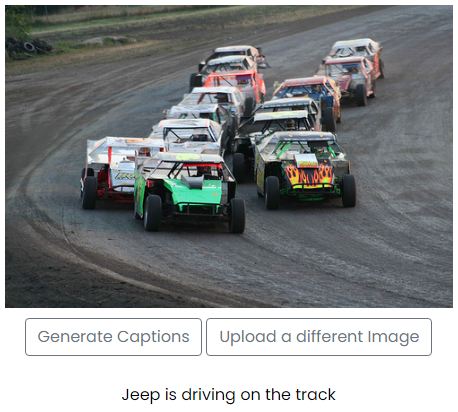
**CONCLUSION:**

We have successfully developed a deep learning model using the Xception architecture to generate automatic captions. So far, most of the image captioning models have used inception (v3). We have famed our project in a web based application using the Flask architecture. Some of the automatically generated captions based on our model are as shown below.

Please note that generating captions is subjective and captions for one same image can differ from person to person. This is also the reason why the algorithm which is trained on human typed captions can generate erratic results sometimes.

****

****

****

**FUTURE SCOPE:**

The accuracy of the current project can be further increased by adding a weightage system to the vocabulary by assigning low weight to high frequently occuring words. It can potentially create a better algorithm to generate captions.

**REFERENCES:**

[1] Sherstinsky, Alex. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network." *Physica D: Nonlinear Phenomena* 404 (2020): 132306. Print.

[2] Liu, Yu Han. "Feature Extraction and Image Recognition with Convolutional Neural Networks." *Journal of Physics: Conference Series* 1087 (2018): 062032. Print.

[3] Aung, San & Pa, Win & nwe, tin. (2020). "Automatic Myanmar Image Captioning using CNN and LSTM-Based Language Model." Proceedings of the 1st Joint SLTU and CCURL Workshop (SLTU-CCURL 2020), pages 139–143. Print.

[4] Chollet, Francois. "Xception: Deep Learning with Depthwise Separable Convolutions." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). Print.

[5] Hossain, M., Sohel, F., Shiratuddin, M. and Laga, H., 2019. A Comprehensive Survey of Deep Learning for Image Captioning. *ACM Computing Surveys*, 51(6), pp.1-36.

[6] O. Vinyals, A. Toshev, S. Bengio and D. Erhan, "Show and tell: A neural image caption generator," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 3156-3164, doi: 10.1109/CVPR.2015.7298935.