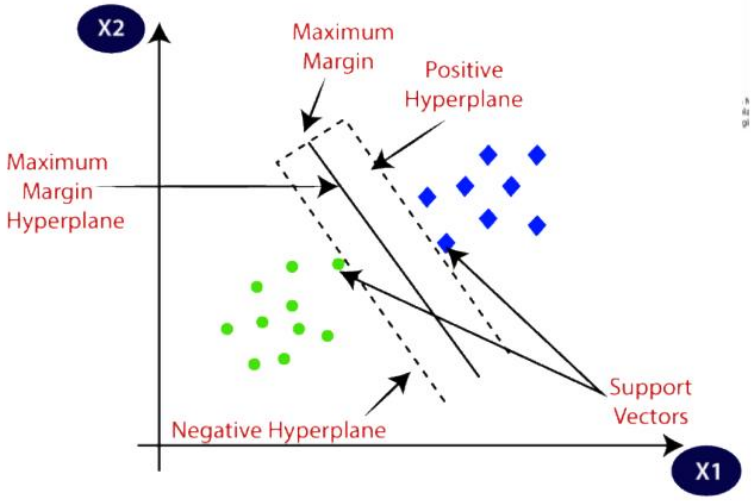
Image & Video Analytics Question Bank

Set 1

Q1A. Explain Support Vector Machine.

- A Support Vector Machine is one of the most popular supervised machine learning algorithm used for both classification and regression.
- An SVM is primarily used for classification problems in Machine Learning.
- SVM algorithm creates a best line or a decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
- SVM chooses the extreme points/vectors that help in creating hyperplane, these extreme cases are known as support vectors, hence the algorithm Support Vector Machine.



- For Example: Suppose a strange cat having certain features of a dog, in order to identify we create a model such that it can accurately identify whether it is a cat or a dog, such model can be created using the SVM Algorithm. We first train our model with images of cats and dogs, so that it can learn about different respective features of cat or dog, and then it is tested. So, an SVM creates a decision boundary between the two data and chooses extreme cases, on the basis of that support vector will classify it as a cat or a dog.

- Types of SVM:

- Linear SVM: Used for linearly separated data, which means if dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data.

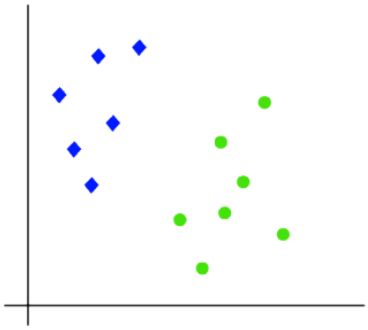
- Non-Linear SVM: Used for Non-Linearly separated data, which means if a data cannot be classified by a straight line.

- Hyperplane

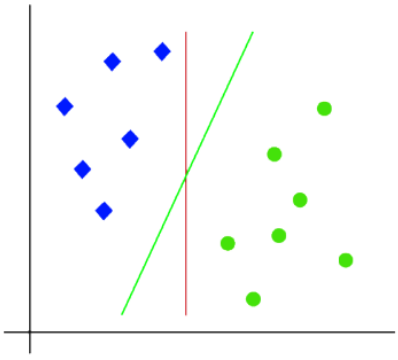
There can be multiple decision boundaries to segregate the class in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. The dimensions of a hyperplane depend on the features present in the dataset which means if there are 2 features hyperplane is a straight line and if there are 3 features it will be in a 2D plane.

- Working of SVM:

Suppose we have a dataset that has two tags (green and blue) and the dataset has two features x1 and x2. We want classifier that can classify the pair (x1, x2) of coordinates in either green or blue. Consider the below image:



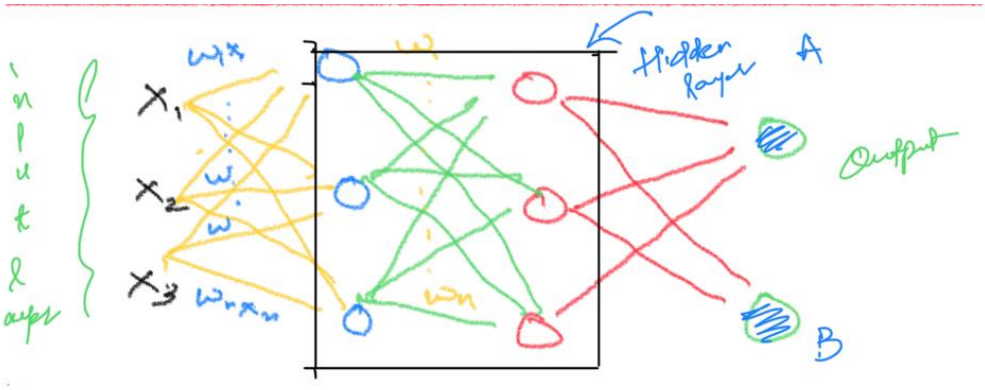
The hyperplane with maximum margin is called the optimal hyperplane. Since it is a 2-D space by just using a straight line, we can easily separate two classes.



Hence SVM helps find out the best line or decision boundary, this best boundary or region is called hyperplane. SVM algorithm finds the closest point of lines from both the classes. These points are called support vectors. The distance between vectors and hyperplane is called as margin.

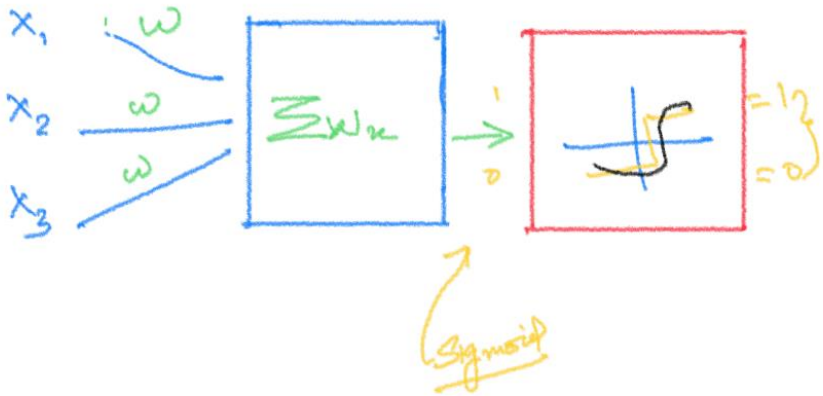
Q1B. Explain Neural Networks.

Neural Networks is comprised of TensorFlow, Keras & Deep Learning. It consists of input layer, output layer and hidden layer. Neural networks are programmed in the same fashion as the human brain. Neural networks are used for classification purposes.



Neural Networks can be of 3 types as follows:

- ANN
- CNN
- RNN



Some of the important terminologies associated to neural networks are as follows:

- **Sequential**
  ML model that sequences input/output of data. Example: Audio Streams, sound clips & time series data
- **Flatten**
  Connects 2D arrays from pooled feature maps into a continuous long array. The flattened matrix is fed as input to fully connected layer to classify the image.
- **Normalize**
  Normalize data to obtain mean close to 0. Speeds up learning and leads to faster convergence.
- **Layers**
  There are 4 types of layers as follows: Fully Connected Layer, Convolution Layer, Deconvolution Layer & Recurrent Layer.

- **Activation Functions**
  Decides whether a neuron can be activated or not.
- **Feed Forward**
  Flow of info in forward direction.
- **Back Propagation**
  Weights of network connections are repeated by adjusting to minimize the difference between the actual and desired output.
- **Loss Function**
  Prediction of error of neural networks. The types of loss functions are as follows: Mean Square Error, Binary CrossEntropy, Categorical CrossEntropy & Sparse Categorical CrossEntropy.
- **Metrics**
  Function to judge model performance. They are as follows: Accuracy Metrics, Probabilistic Metrics, Regression Metrics & Image Segmentation Metrics.

Q1C. Explain Keras.

• Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation. Keras is relatively easy to learn and work with because it provides a python frontend with a high level of abstraction while having the option of multiple back-ends for computation purposes. This makes Keras slower than other deep learning frameworks, but extremely beginner-friendly.

• Keras allows you to switch between different back ends.

• Out of these five frameworks, TensorFlow has adopted Eras as its official high-level API.

• Keras is embedded in TensorFlow and can be used to perform deep learning fast as it provides inbuilt modules for all neural network computations.

• At the same time, computation involving tensors, computation graphs, sessions, etc can be custom made using the TensorFlow Core API, which gives you total flexibility and control over your application and lets you implement your ideas in a relatively short time.

Q1D. Explain Python Image Library.

PIL stands for Python Image Library. It is another library for Image processing and filtering and it is a free and open-source additional library for the python programming language. PIL adds support to opening, manipulating and saving files into many different formats. PIL offers several standard procedures for image manipulations like:

1. Pre-pixel manipulations
2. Masking & Transparency Handling
3. Image Filtering
4. Image Blurring
5. Image Contouring
6. Image Smoothing
7. Image Edge Detection
8. Image Enhancing
9. Image Sharpening
10. Adjusting Brightness
11. Adjusting Contrast
12. Adjusting Colour

Syntax to install PIL:

!pip install pillow

Syntax to import PIL:

from PIL import Image

- In order to read an image via PIL, Image.open('img.ext')
- In order to convert an image into grayscale: Image.open('img.ext').convert('L')
- In order to resize an image, Image.resize((128,128))
- In order to crop an image: Image.crop(top, bottom, right, left)
- In order to rotate an image, Image.rotate(angle)

Q2A. Explain any 2 Image Processing Algorithms.

The Image Processing Algorithms are as follows:

- GAUSSIAN IMAGE PROCESSING
  • Gaussian blur which is also known as gaussian smoothing, is the result of blurring an image by a Gaussian function.
  • It is used to reduce image noise and reduce details. The visual effect of this blurring technique is similar to looking at an image through the translucent screen. It is sometimes used in computer vision for image enhancement at different scales or as a data augmentation technique in deep learning.
  • Formula:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

  • In practice, it is best to take advantage of the Gaussian blur's separable property by dividing the process into two passes. In the first pass, a one-dimensional kernel is used to blur the image in only the horizontal or vertical direction. In the second pass, the same one-dimensional kernel is used to blur in the remaining direction. The resulting effect is the same as convolving with a two-dimensional kernel in a single pass. Let's see an example to understand what gaussian filters do to an image.

- EDGE DETECTION

  • Finds boundaries of objects within images

  • Works by its discontinuities in brightness

  • Most of shape information is enclosed in edges, hence beneficial for extracting useful information

  • Can rapidly react if some noise is detected in image while detecting variations of grey levels. Edges are defined as local maxima of grey levels

  • Common edge algorithm is Sobel Edge Detection Algorithm

- Morphological Image Processing
- Fourier Transform Image Processing
- Wavelet Image Processing

Q2B. Explain Gaussian Image Processing Algorithm. Mention the syntax.

GAUSSIAN IMAGE PROCESSING

• Gaussian blur which is also known as gaussian smoothing, is the result of blurring an image by a Gaussian function.

• It is used to reduce image noise and reduce details. The visual effect of this blurring technique is similar to looking at an image through the translucent screen. It is sometimes used in computer vision for image enhancement at different scales or as a data augmentation technique in deep learning.

• Formula:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

• In practice, it is best to take advantage of the Gaussian blur's separable property by dividing the process into two passes. In the first pass, a one-dimensional kernel is used to blur the image in only the horizontal or vertical direction. In the second pass, the same one-dimensional kernel is used to blur in the remaining direction. The resulting effect is the same as convolving with a two-dimensional kernel in a single pass. Let's see an example to understand what gaussian filters do to an image.

• Syntax: img.filter(ImageFilter.GaussianBlur)

Q2C. Explain PIL. Mention Any two operations of PIL with code.

Ans. Same as Set 1 Q1D.

Q2D. Explain Open CV.

- OPENCV stands for Open-Source Computer Vision.
- OPENCV is a library of programming functions aimed at real time computer vision.
- OPENCV was developed by Intel.
- OPENCV is a cross-platform library.
- OPENCV has one main feature i.e., GPU acceleration for real-time operations.
- First version of OPENCV was released in 2006 and currently opencv2 is in used

In order to install opencv library syntax is:

```
!pip install opencv-python
```

In order to access the library syntax is:

```
Import cv2
```

- OPENCV functions:

1. Image Enhancement
2. Image Filtering
3. Object Detection
4. Image Resizing
5. Image Reshaping
6. Real Time Video Processing

- OPENCV Applications

1. Egomotion Estimation
2. Object Detection
3. Augmented Reality
4. Gesture Recognition
5. Structure from Motion
6. Human Computer Interaction
7. Motion Tracking

Q3A. Explain KNN. Mention the syntax and the library used for KNN. How is the value of "K" decided? What value of K is ideal?

• K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
• K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
• K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
• K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
• K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
• It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
• KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
• Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So, for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cat and dog images and based on the most similar features it will put it in either cat or dog category.

The syntax for KNN is as follows:

```
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
```

The value of K is decided as follows:

K is usually decided by finding the square root of N, where N is the total number of samples.

The square root of N is usually the ideal value for K.

Q3B. Explain Object Detection.

- Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.
- Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.
- It is also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, or tracking a person in a video.
- Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features.
- For example, when looking for circles, objects that are at a particular distance from a point (i.e., the centre) are sought.
- Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed.
- A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin colour and distance between eyes can be found.
- Methods for object detection generally fall into either neural network-based or non-neural approaches.
- For non-neural approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as support vector machine (SVM) to do the classification. On the other hand, neural techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN).

• Non-neural approaches:
◦ Viola–Jones object detection framework based on Haar features
◦ Scale-invariant feature transform (SIFT)
◦ Histogram of oriented gradients (HOG) features

• Neural network approaches:
◦ Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN, cascade R-CNN.)
◦ Single Shot MultiBox Detector (SSD)
◦ You Only Look Once (YOLO)
◦ Single-Shot Refinement Neural Network for Object Detection (RefineDet)
◦ Retina-Net
◦ Deformable convolutional networks

Q3C. Explain Fine Tuning.

• Fine-tuning is a technique of model reusability in addition to feature extraction.
• Fine-tuning consists of unfreezing few of the top layers of the frozen model base in neural network used for feature extraction and jointly training both the newly added part of the model (for example, a fully connected classifier) and the top layers.
• This technique is called fine-tuning as it slightly adjusts the more abstract representations of fine-tuning model being reused so that it can be made more relevant for the problem at hand.
• Only the top layers of the convolutional base are possible to be fine-tune once the classifier on top has already been trained because to be able to train a randomly initialized classifier, freezing of pretrained convnets like VGG16 should have to done.

-Steps to fine-tune a network are as follows:

1. Add custom network on top of an already-trained base network.
2. Freezing the base network.
3. Training the part added.
4. Unfreezing some layers in base network.
5. Jointly train both of these layers and the part added.

• It is useful to fine-tune more specialized features as these are the ones that need to be repurposed on the new problem. Fast-decreasing returns in fine-tuning lower layers will occur.

Q3D. Explain Feature Extraction.

-Feature extraction is a part of the dimensionality reduction process, in which, an initial set of the raw data is divided and reduced to more manageable groups.
-The most important characteristic of these large data sets is that they have a large number of variables. These variables require a lot of computing resources to process.
-So, feature extraction helps to get the best feature from those big data sets by selecting and combining variables into features, thus, effectively reducing the amount of data.
-These features are easy to process, but still able to describe the actual data set with accuracy and originality.
-The technique of extracting the features is useful when you have a large data set and need to reduce the number of resources without losing any important or relevant information.
-Feature extraction helps to reduce the amount of redundant data from the data set.
-The reduction of the data helps to build the model with less machine effort and also increases the speed of learning and generalisation steps in the machine learning process.

Applications

-Bag of Words

Bag of Words is the most used technique for natural language processing. In this process they extract the words or the features from a sentence, document, website, etc. and then they classify them into the frequency of use. So, in this whole process feature extraction is one of the most important parts.

-Image Processing

Image processing is one of the best and most interesting domain. In this domain basically you will start playing with your images in order to understand them. So, here we use many techniques which includes feature extraction as well and algorithms to detect features such as shaped, edges, or motion in a digital image or video to process them.

-Auto-encoders

The main purpose of the auto-encoders is efficient data coding which is unsupervised in nature. this process comes under unsupervised learning. So, feature extraction procedure is applicable here to identify the key features from the data to code by learning from the coding of the original data set to derive new ones.

Q4A. Suppose you were to run MNIST Data set, answer the below questions
i) Mention the libraries which will be used.
ii) Is there a need for using Neural Networks?
iii) Provide a step wise flow for running the MNIST Dataset.
iv) Which library is important for MNIST dataset?
v) Mention the syntax to normalise the data.

The libraries which will be used are TensorFlow for the actual analysis and pandas, numpy and matplotlib for pre-processing.

Yes, neural networks are necessary for using the MNIST dataset.

The step wise flow for running the MNIST dataset is given as follows:

- Importing the dataset from the TensorFlow library.
- Dividing into train and test.
- Normalizing the values.
- Modelling the data, converting to sequential format, flattening the data array, making the data dense using the relu and SoftMax functions.
- Compiling the model with the optimizer, loss and metric arguments.
- Fitting the model.
- Validating and saving the function.

The TensorFlow library is important for the MNIST dataset.

The syntax to normalise the data is as follows: tf.keras.utils.normalize(data)

Q5A. Explain the below code. Explain the flow of the code, functions and activation functions

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, tf.nn.relu))
model.add(tf.keras.layers.Dense(128, tf.nn.relu))
model.add(tf.keras.layers.Dense(10, tf.nn.softmax))
```

A sequential model, as the name suggests, allows you to create models layer-by-layer in a step-by-step fashion. The first line of the code converts the model into a sequential format. The second line flattens the data array i.e., it converts 2D array into 1D continuous array by unrolling the values beginning at the last dimension. It is used to reshape the tensor to have a shape that is equal to the number of elements contained in the tensor. The third, fourth and fifth lines are used to add dense layers to the model using the dense activation function. Relu is used in hidden layer to avoid vanishing gradient problem and better computation performance, and SoftMax function is used in the last output layer.

Q5B. Fill in the details for the below mentioned code:

model.compile(optimiser = '**adam**', loss_function = '**sparse_categorical_crossentropy**', metrics = ['**accuracy**']

Explain each parameter.

Adam optimizer involves a combination of two gradient descent methodologies:

Momentum

This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients. Using averages makes the algorithm converge towards the minima in a faster pace.

Root Mean Square Propagation (RMSP)

Root mean square prop or RMSprop is an adaptive learning algorithm that tries to improve AdaGrad. Instead of taking the cumulative sum of squared gradients like in AdaGrad, it takes the 'exponential moving average'.

Sparse Categorical CrossEntropy is a loss function which computes categorical cross-entropy. When using Sparse Categorical CrossEntropy the targets are represented by the index of the category (starting from 0).

The accuracy metric is used to calculate the accuracy. It calculates how often predictions matches labels based on two local variables it creates: total and count , that are used to compute the frequency with which logits matches labels.

Set 2

Q1A. Explain Edge Detection with syntax.

• Finds boundaries of objects within images.

• Works by its discontinuities in brightness.

• Most of shape information is enclosed in edges, hence beneficial for extracting useful information.

• Can rapidly react if some noise is detected in image while detecting variations of grey levels. Edges are defined as local maxima of grey levels.

• Common edge algorithm is Sobel Edge Detection Algorithm.

Syntax: img.filter(ImageFilter.FIND_EDGES)

Q1B. Explain Gaussian Image Processing.

Same as Set 1 Q2B.

***Q1C. Explain Random Forest.**

Random forest, like its name implies, consists of many individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction i.e., random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance. Random forests are frequently used as "Blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

Q1D. Fill in the details for the below mentioned code: 5

model.compile(optimiser = '_____', loss_function = '_____',

metrics = ['_____']

Same as Set 1 Q5B.

Q2A. Explain KNN with syntax.

Same as Set 1 Q3A.

***Q2B. Explain TensorFlow.**

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. It is a Python-friendly open-source library for numerical computation that makes machine learning and developing neural networks faster and easier. Created by the Google Brain team and initially released to the public in 2015, TensorFlow is an open-source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning models and algorithms (aka neural networks) and makes them useful by way of common programmatic metaphors. It uses Python or JavaScript to provide a convenient front-end API for building applications, while executing those applications in high-performance C++. TensorFlow allows developers to create dataflow graphs—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor.

Q2C. Explain Python Image Library and provide an example to crop an image.

Same as Set 1 Q1D.

Q2D. Explain CV2 mention the syntax to install open cv.

Same as Set 1 Q2D.

*Q3A. Explain Tesseract and provide the flow for OCR.

Tesseract is an open-source text recognition (OCR) Engine, available under the Apache 2.0 license. It can be used directly, or (for programmers) using an API to extract printed text from images. It supports a wide variety of languages. Tesseract does not have a built-in GUI, but there are several available from the 3rdParty page. Tesseract is compatible with many programming languages and frameworks through wrappers that can be found here. It can be used with the existing layout analysis to recognize text within a large document, or it can be used in conjunction with an external text detector to recognize text from an image of a single text line. Tesseract 4.00 includes a new neural network subsystem configured as a text line recognizer. It has its origins in OCRopus' Python-based LSTM implementation but has been redesigned for Tesseract in C++. The neural network system in Tesseract pre-dates TensorFlow but is compatible with it, as there is a network description language called Variable Graph Specification Language (VGSL), that is also available for TensorFlow. To recognize an image containing a single character, we typically use a Convolutional Neural Network (CNN). Text of arbitrary length is a sequence of characters, and such problems are solved using RNNs and LSTM is a popular form of RNN. Read this post to learn more about LSTM.

The flow for OCR is as follows: Capture Image, Save Image, Adaptive Thresholding, Identifying Foreground Pixels, Segment in Blobs, Filter Blobs, Estimating Base Lines, Merge Blobs into Lines, Find Fixed Pitch, Chop Words into Characters, Measure Gaps into Characters, Identify Outlines, Apply Classifier, Match with Prototype and Recognizing Characters.

Q3B. Explain Support Vector Machine.

Same as Set 1 Q1A.

Q3C. Explain Keras.

Same as Set 1 Q1C.

*Q3D. Write a short note on activation functions for neural networks.

Activation function decides whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron. We know, neural network has neurons that work in correspondence of weight, bias, and their respective activation function. In a neural network, we would update the weights and biases of the neurons based on the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases. A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

Some of the activation functions are given below:

- ReLU (Rectified Linear Unit) Activation Function
- Leaky ReLU
- SoftMax Activation Function
- Linear Activation Function
- Non-Linear Activation Function

Q4. Consider the Cat Dog Classifier Dataset

Provide the important libraries, mention the syntax for random function to call an image from the dataset, mention another method to normalise the data/image except for the normalise function.

The important libraries are as follows: TensorFlow & Keras. Some other pre-processing libraries used for the cat dog classifier dataset are numpy, pandas, matplotlib & random.

The syntax for random function to call an image from the dataset is as follows:
random.randint(0, len(data))
plt.imshow(data[])

Another method to normalise the data or image except for the normalise function is to divide by 255.

Q5. Explain the below code

```
model = Sequential([

Conv2D(32,(3,3),activation='relu', input_shape= (100,100,3)),

MaxPooling2D((2,2)),

Conv2D(32,(3,3),activation = 'relu'),

MaxPooling2D((2,2)),

Flatten(),

Dense(64, activation = 'relu'),

Dense(1, activation = 'sigmoid')])
```

A sequential model, as the name suggests, allows you to create models layer-by-layer in a step-by-step fashion. The first line of the code converts the model into a sequential format. The second line. The Conv2D layer creates a convolution kernel that is convolved with the layer input 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. The Relu activation function is used in hidden layer to avoid vanishing gradient problem and better computation performance. The specified input shape function reshapes the data or provides the shape of the input data. The maxpooling2D function down samples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool_size) for each channel of the input. The window is shifted by strides along each dimension. The sixth line flattens the data array i.e., it converts 2D array into 1D continuous array by unrolling the values beginning at the last dimension. It is used to reshape the tensor to have a shape that is equal to the number of elements contained in the tensor. The seventh and eight lines are used to add dense layers to the model using the dense activation function. The activation functions used are relu and sigmoid.

Set 3

Q1A. Explain Edge Detection with syntax.

Same as Set 2 Q1A.

Q1B. Explain Gaussian Image Processing.

Same as Set 1 Q2B.

Q1C. Explain Random Forest.

Same as Set 2 Q1C.

Q1D. Fill in the details for the below mentioned code: 5

model.compile(optimiser = '_____', loss_function = '_____',

metrics = ['_____']

Same as Set 1 Q5B.

Q2A. Explain any 2 Image Processing Algorithms.

Same as Set 1 Q2A.

Q2B. Explain Gaussian Image Processing Algorithm. Mention the syntax.

Same as Set 1 Q2B.

Q2C. Explain PIL. Mention any 2 operations of PIL with code.

Same as Set 1 Q1D.

Q2D. Explain Open CV.

Same as Set 1 Q2D.

Q3A. Explain KNN.

Same as Set 1 Q3A.

Q3B. Explain Support Vector Machine.

Same as Set 1 Q1A.

Q3C. Explain Random Forest.

Same as Set 2 Q1C.

Q3D. Explain various PIL functions with syntaxes.

Same as Set 1 Q1D.

*Q4. Explain the below mentioned functions of neural networks.

i) Activation Function (Enlist and explain any 2)

ii) Loss Function (Enlist and explain any 2)

iii) Explain Normalisation of images. Why do we need it?

Activation function decides whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron. We know, neural network has neurons that work in correspondence of weight, bias, and their respective activation function. In a neural network, we would update the weights and biases of the neurons based on the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases. A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

Some of the activation functions are given below:

- ReLU (Rectified Linear Unit) Activation Function
  The Relu activation function is used in hidden layer to avoid vanishing gradient problem and better computation performance.
- SoftMax Activation Function
  The softmax, or "soft max," mathematical function can be thought to be a probabilistic or "softer" version of the argmax function. The softmax function is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution. That is, softmax is used as the activation function for multi-class classification problems where class membership is required on more than two class labels.

A loss function is a function that compares the target and predicted output values; measures how well the neural network models the training data. When training, we aim to minimize this loss between the predicted and target outputs. The hyperparameters are adjusted to minimize the average loss. Some of the loss functions are listed as below:

- Mean Squared Error
  MSE loss is used for regression tasks. As the name suggests, this loss is calculated by taking the mean of squared differences between actual(target) and predicted values.
- Binary Cross Entropy
  BCE loss is used for the binary classification tasks. If you are using BCE loss function, you just need one output node to classify the data into two classes. The output value should be passed through a sigmoid activation function and the range of output is (0 – 1).

Image normalization is a process, often used in the preparation of data sets for artificial intelligence (AI), in which multiple images are put into a common statistical distribution in terms of size and pixel values; however, a single image can also be normalized within itself. The process usually includes both spatial and intensity normalization. The point from normalization comes behind calibrating the different pixels intensities into a normal distribution which makes the image looks better for the visualizer. Main purpose of normalization is to make computation efficient by reducing values between 0 to 1. This also makes the images easier to analyse.

Q5. Fill in the details for the below mentioned code: 15
Consider the data set for cat dog classifier, mention the steps for classification of cats/dogs images.
i) Mention necessary libraries.
The necessary libraries are numpy, pandas, random, matplotlib, tensorflow and keras (Sequential from keras.models and Conv2D, MaxPooling2D, Flatten, Dense from keras.layers)

ii) from google.colab import drive
drive.mount('/content/drive')

iii) Read the data using numpy arrays
X_train =(a)**np.loadtxt**('/content/drive/MyDrive/Cat Dog Classification/input.csv',
(b)**delimiter** = ',')
Y_train =**np.loadtxt**('/content/drive/MyDrive/Cat Dog Classification/labels.csv',
delimiter= ',')
X_test =**np.loadtxt**('/content/drive/MyDrive/Cat Dog Classification/input_test.csv',
delimiter= ',')
Y_test =**np.loadtxt**('/content/drive/MyDrive/Cat Dog Classification/labels_test.csv',
delimiter = ',')
Mention what is (a) and (b)
a = np.loadtxt & b = delimiter

iv) Can the image data be normalised used normalise function only? If No, mention another method for it.
No, the image data can be normalised using other functions too. For one, it can be normalised by dividing the data by 255.

v) model = Sequential([
Conv2D(32,(3,3),activation=(a)**relu**, input_shape= (100,100,3)),
MaxPooling2D((2,2)),
Conv2D(32,(3,3),activation =(b)**relu**),
MaxPooling2D((2,2)),
Flatten(),
Dense(64, activation = (c)**relu**),
Dense(1, activation = (d)**sigmoid**)
])
Mention (a),(b),(c) and (d)
a, b, c = relu & d = sigmoid

vi) model.compile(optimizer = '(a)**adam**', loss = '(b)**binary_crossentropy**, metrics= ['accuracy'])
Mention (a) and (b)
a = adam, b = binary_crossentropy

vii) Will the model.compile work if optimizer is not mentioned?
No, the model.compile does not work if optimizer is not mentioned.