PPT 1

- **COMPUTER VISION PIPELINE**

We start by collecting the data over which we want to perform the algorithm or any other operation. This data can be collected with the help of cameras using mobile phones or other devices etc. or they can directly be downloaded from the internet if the type of images that you want are available over the internet.

Then we perform preprocessing over the images by performing resampling by adjusting the size of all the images to a similar size, performing noise reduction to remove the noise from the images, contrast enhancement by increasing or decreasing the brightness in some areas of the images and by performing augmentation which basically creates new images by rotating the images at some angle , by scaling the images up and down and flipping the images.

Then we extract features that are important to us such as lines, ridges, blobs etc. using transfer learning and CNN's.

And then we can analyze the result on the basis we want to and make decision according to that.

PPT 2

- **EDGE DETECTION AND CANNY EDGE DETECTION ALGORITHM**

Edge detection helps us in identifying the edges, curves in an images where there are discontinuities or where the brightness changes sharply using a variety of mathematical methods. It is a fundamental tool of computer vision useful for feature extraction and feature detection.

Now if we talk about about Canny Edge Detection then:

CONVERT IMAGE TO GRAYSCALE

We start by converting the image into grayscale whose edges we want to find out.

APPLY GAUSSIAN FILTER

Then we apply gaussian blur over the image to reduce the noise in the image by smoothening the image.

APPLY HORIZONTAL AND VERTICAL GRADIENT

After applying gaussian blur we apply horizontal and vertical gradients over the image and calculate the magnitude to detect the intensity changes in the image.

APPLY NON MAXIMUM SUPPRESSION

Now after applying the gradient the edges that we got are thick edges but we should have thin images in the final output so we apply non maximum suppression based on comparison of neighboring pixel values.

APPLY DOUBLE THRESHOLDING

After performing non maximum suppression we get some edges that are not really edges so to remove them we perform double thresholding. Now for example if we select a low threshold as 50 and high threshold as 100 then the pixel values of edges having value less than 50 will be removed by setting the values of those pixels to 0, the pixel values of the edges having values greater than 100 are considered as strong edges and the pixel values that lie between 50 and 100 are considered as weak edge and they will be dealt with in the next step.

EDGE TRACKING USING HYSTERISIS

Now here if the weak edges are connected with string edges then they we will be considered as edges and if they are not connected with strong edges then they will also be removed.

LINE DETECTION AND HOUGH TRANSFORM

Line detection refers to collecting all the n edge points and detecting the lines over which these edge points lie.

Now If we talk about hough transform algorithm:

So here we first detect edges in the image using canny edge detection

Then we initialize a parameter space p(rho) and theta using a 2Darray which is basically an accumulator array.

Now for each edge point we calculate the values of rho and theta that correspond to all possible lines passing through that point and we increment the value in the accumulator array.

Then we find the maximum value of theta and these will be the parameters rho and theta for your line.

PPT 3

BLOBS AND PERFROMING SIFT

Blob are a group of pixel values that somewhat form a colony that is easily distinguishable from the background of the image. To make a blob useful we first locate it, determine its size and orientation and the assign a descriptor or signature to it.

For this we use sift algorithm that stands for scale invariant feature transform algorithm.

It helps in locating the features tin the image which are rotation and scale invariant that is they are not affected even after rotating the image or after scaling the image commonly know as key points.

STEPS TO PERFORM SIFT:

CONSTRUCTING A SCALE SPACE

Convert the image into grayscale and apply gaussian filter over it.

Now to ensure that the features are scale independent we keep on scaling down the images to its half and create 4 octaves through this where the images of one octave is half of its previous octave and in each octave we will have images in increasing order of gaussian blur. Now we will try to enhance the features by subtracting the images from the previous image in the same octave that is will find difference of gaussians.

KEYPOINT LOCALIZATION

Now here we find the important keypoints by looking for local maxima and minima in the result of difference of gaussians step.

To locate a local maxima or local minima we select and pixel and compare its value with the its neighboring pixel values along with pixel values present in the same position of the image present in the next octave and the image present In the previous octave.

KEYPOINT SELECTION

Now out of all the keypoints we remove those keypoints that are low in contrast or that are too close to the edges. Now after this we have the most accurate scale invariant keypoints to which we will now assign orientation.

ORIENTATION ASSIGNMENT

Now we will assign orientation to all those selected key points to make them rotation invariant by using gradient operators and create a histogram with angles as bins. The bin with the peak value in the histogram will be considered as principal orientation of the blob.

ASSIGNING DESCRIPTOR

Now descriptor is basically a fingerprint for the key points which is used for comparing the keypoints in different images. So here we take the neighboring values around the blob and divide the whole area into sub-blocks. For eg. If we take the neighboring value around the blob and it has 16*16 blocks so we will divide this into different sub blocks of equal sizes like 16 sub-blocks of size 4*4. And then we create a histogram for all these values which will be represented as a vector to form a keypoint descriptor.

PPT 4

BATCH GRADIENT DESCENT

In batch gradient descent all the training examples are taken into consideration and then the gradient of the loss function is calculated using the entire training dataset and then the weights are updated based on the gradient.

STOCHASTIC GRADIENT DESCENT

Here it randomly selects one training example at a time and calculate the gradient of the loss function using that selected example and then the weights are updated based on the gradient.

MINI BATCH GRADIENT DESCENT

Here the training dataset is divided into small random subsets, each containing fixed number of examples and then the gradient for the loss function of each mini batch is calculated and then the weights are updated based on the average gradient over the mini batch.

CHALLENGES THAT CNNs SOLVE:

LOCAL RECEPTIVE FIELDS

Local receptive fields play a key role in convolutional neural networks (CNNs) by allowing the network to effectively learn features from input images. Here's a simple explanation of how they work:

In a CNN, each layer consists of a set of filters that slide over the input image and compute a set of activations, known as a feature map. Each filter is associated with a small region of the input image, known as its local receptive field.

WEIGHT SHARING

Weight sharing in Convolutional Neural Networks (CNNs) refers to the idea that the same set of learnable weights is used for different receptive fields in a layer.

In a convolutional layer, the filters slide over the input image to produce feature maps. Each filter is responsible for detecting a specific feature in the image. Weight sharing means that each filter in a layer is applied to the entire input image, using the same set of weights. This allows the network to learn a set of shared features that are useful for recognizing patterns in the input.

POOLING

Reduces dimensionality by reducing the size of the output from previous layer in order to reduce the number of computations.

PPT 5

DIAGNOSIS OF CNN

Input image: The input image is fed into the network, typically as a matrix of pixel values.

Convolutional layers: The input image is processed by a series of convolutional layers, where each layer applies a set of filters to the input image to extract features at different spatial scales. The filters are learned by the network during the training process.

Pooling layers: After each convolutional layer, a pooling layer is applied to reduce the spatial dimensionality of the output. This helps to make the network more robust to small variations in the input image, such as changes in lighting or perspective.

Fully connected layers: The output of the final pooling layer is flattened into a vector and fed into one or more fully connected layers. These layers use the features extracted by the convolutional layers to make a prediction about the content of the input image, such as the presence of a particular object or the classification of the image into a particular category.

Output: The output of the fully connected layer is passed through an activation function, such as softmax, to produce a set of probabilities that represent the network's prediction about the content of the input image.

During the training process, the network learns to adjust the weights of the convolutional and fully connected layers to minimize the difference between its predicted output and the actual output for a given set of training examples. This allows the network to become more accurate at making predictions over time.

DROPOUT

Dropout as the same suggests refers to dropping some random nodes from the fully connected layer in the neural network. It is usually done to prevent overfitting due to over learning of the dataset. We provide a probability value(p) in the dropout layer that tells what percentage of the total nodes present in the fully connected layer are to be dropped out randomly. Eg.p=0.5 means that if there are a total of 8 nodes in the fully connected layer then 4 random nides out of those 8 nodes will not be used during the training process. And as we do not apply a dropout layer in the testing process so to compensate for this we multiply the weights of the nodes by p and then summed together to get the final weights.

FEATURE SCALING

Feature scaling is an important step in pre processing the input data for CNNs. It helps to normalize the input features so that they have similar ranges and magnitudes. This can lead to better performance and faster convergence during training. Here's a more detailed explanation of why feature scaling is important:

Range of input features: Input features can have different ranges and magnitudes, which can affect the performance of the CNN. For example, one feature may range from 0 to 1000, while another feature ranges from 0 to 1. This can cause some features to dominate over others, leading to suboptimal performance.

Gradient descent: CNNs use gradient descent to update the weights of the network during training. Gradient descent involves calculating the gradient of the loss function with respect to the weights, and using this gradient to update the weights. If the input features have different ranges, the gradients can become unbalanced, making it difficult for the network to converge to an optimal solution.

Normalization: Feature scaling helps to normalize the input features so that they have similar ranges and magnitudes.

INTERNAL COVARIANT SHIFT

Internal covariate shift refers to the phenomenon of the distribution of input data to each layer of a neural network changing during training. The input to the CNN is a set of images, which are fed into the first convolutional layer. The convolutional layer applies a set of filters to the input images, extracting features that are useful for the given task. The resulting feature maps are then passed through a pooling layer, which reduces their size while preserving important features. Finally, the pooled feature maps are flattened and fed into a fully connected layer, which produces the output.

During training, the distribution of input data to each layer can change as the weights of the network are updated. This can result in slower convergence and poorer performance. To address this issue, batch normalization was introduced, which normalizes the inputs to each layer to have zero mean and unit variance. This helps to stabilize the distribution of inputs and improve the convergence of the network.

In summary, internal covariate shift refers to the change in distribution of input data to each layer of a neural network during training. Batch normalization can be used to address this issue and improve the stability and convergence of the network.

BATCH NORMALIZATION

Batch normalization is a technique used in deep neural networks, including CNNs, to improve the stability and convergence of the network during training.

In simple words, batch normalization is a technique that normalizes the inputs to each layer of the network to have zero mean and unit variance. It works by calculating the mean and standard deviation of the inputs in each mini-batch of training data, and then they are scaled and shifted using two new parameters.

During testing we might not have a way to calculate the mean and variance of the input for mini batches instead we might calculate it for the entire dataset and then the parameters that were learned during the training process responsible for shifting and scaling will be used in testing too.

GLOBAL AVERAGE POOLING

Global average pooling, on the other hand, applies a pooling operation to the output of the convolutional layers that computes the average value of each feature map. This is typically done by taking the mean of each feature map across all of its spatial locations, resulting in a single value for each feature map. Reduces the increase in number of params caused by flattening layer

Takes an average of all the feature maps of the layer

E.g. if the last conv layer in the network has 512 filters thus producing 512 feature maps, the global average pooling will produce an output vector of 512

Here's a summary of the differences between global average pooling and flattening:

Flattening converts the output of the convolutional layers into a one-dimensional vector, while global average pooling computes the average value of each feature map.

Flattening results in a high-dimensional vector with many parameters, while global average pooling results in a low-dimensional vector with fewer parameters.

Global average pooling is more computationally efficient and less prone to overfitting than flattening.

Global average pooling is often used in CNNs for image classification, while flattening is used in more general-purpose CNNs.

TRANSFER LEARNING

Transfer learning is a technique in which a pre-trained neural network model is used as a starting point to solve a similar task to the one it was originally trained on. Transfer learning can be useful in deep learning tasks when the size of the training data is limited, or when the computational resources required for training a deep neural network are not available.

There are three main approaches to using transfer learning in CNNs:

Using a Pretrained Network as a Classifier: In this approach, a pre-trained CNN model is used as a feature extractor. The output of the last layer of the pre-trained model is flattened and passed through a new fully connected layer for the new classification task. The weights of the pre-trained layers are kept fixed during training, while only the weights of the new fully connected layer are learned from scratch.

Using a Pretrained Network as a Feature Extractor: In this approach, the pre-trained CNN model is used as a feature extractor, similar to the previous approach. However, instead of adding a new fully connected layer, the extracted features are used as input to a new model architecture. This new architecture can be trained from scratch, or with a small number of training samples.

To use a pre-trained model as a feature extractor, we remove the final output layer of the pre-trained model and replace it with a new output layer that corresponds to the number of classes in our new classification problem. We then freeze the weights of the pre-trained model so that they are not updated during training.

During training, we pass the input images through the pre-trained model and obtain the output of the desired layer. We then use these outputs as inputs to the new classifier model, which is trained to predict the correct class labels for the new problem.

Fine Tuning: In this approach, the weights of the pre-trained CNN model are fine-tuned during training for the new task. We just extract the correct feature maps from the source domain pretrained network and fine tune them to the target domain. This is known as fine tuning.

Fine tuning can be formally defined as freezing a few of the network layers that are used for feature extraction and transferring all of its high level feature maps (outputs of later convolutional layers) to your domain.

WHY SHOULD WE KEEP LEARNING RATE SMALL DURING TRANSFER LEARNING

When performing transfer learning, it is common to keep the learning rate small when training the new layers of the network. This is because the pre-trained layers of the network already contain useful and well-tuned features that have been learned on a large dataset, and the goal is to adapt these features to the new dataset rather than completely relearn them.

If the learning rate is set too high, it can cause the new layers to make large updates that will disrupt the existing features learned by the pre-trained layers. This can cause the network to converge to a suboptimal solution or even completely forget the pre-trained features, resulting in poor performance on the new task.

By keeping the learning rate small, the new layers can gradually adapt to the new dataset while preserving the useful features learned by the pre-trained layers. This can result in faster convergence and better performance on the new task.

It is also common to use a smaller batch size when fine-tuning a pre-trained network, as this can help prevent overfitting and allow the network to better adapt to the new dataset. Additionally, it is often necessary to freeze some or all of the pre-trained layers during fine-tuning to prevent them from being disrupted by the updates to the new layers.

PPT 6

INCEPTION

VANISHING GRADIENT

The vanishing gradient problem is a common issue that occurs during the training of artificial neural networks, particularly recurrent neural networks (RNNs) and deep neural networks with many layers. The problem arises when the gradients of the loss function with respect to the weights of the network become very small as they are propagated backwards through the layers of the network during training.

In simpler terms, the vanishing gradient problem occurs when the signals that are used to update the weights of the network become too small to effectively change the network's behavior. This can make it difficult or impossible for the network to learn complex patterns or relationships in the input data, especially if those patterns or relationships span multiple layers of the network.

To solve the vanishing gradient problem, He et al. created a shortcut that allows the gradient to be directly back propagated to earlier layers. These shortcuts are called skip connections.

They are used to flow information from earlier layers in the network to later layers, creating an alternate shortcut path for the gradient to flow through.

Skip connections also allow the model to learn an identity function which ensures that the layer will perform at least as well as the previous layer

Skip connections provide an additional path for the gradients to flow through the network. Instead of directly feeding the output of one layer into the next layer, the output is added to the input of a subsequent layer further along in the network. This creates a shortcut that allows the gradient to bypass the problematic layers and flow more easily through the network.

RESNET

RESIDUAL BLOCKS

In a residual block, the input to the block is passed through a series of convolutional layers, followed by a non-linear activation function. The output of these layers is then added to the original input, and the result is passed through another activation function. This approach allows the network to learn a residual mapping, which can help to avoid the vanishing gradient problem by allowing gradients to flow directly through the block.

He et al. decided to do dimension downsampling using bottleneck 1 × 1 convolutional layers, similar to the Inception network

Residual blocks starts with a 1 × 1 convolutional layer to downsample the input dimension volume, and a 3 × 3 convolutional layer and another 1 × 1 convolutional layer to downsample the output.

This is a good technique to keep control of the volume dimensions across many layers. This configuration is called a bottleneck residual block.

MOBILENET

DEPTHWISE CONVOLUTION

A depthwise convolution is basically a convolution along only one spatial channel of the image.

The key difference between a normal convolutional layer and a depthwise convolution is that the depthwise convolution applies the convolution along only one spatial dimension (i.e. channel) while a normal convolution is applied across all spatial dimensions/channels at each step.

Pointwise Convolution

A depthwise convolution is basically a convolution along only one spatial channel of the image.

The key difference between a normal convolutional layer and a depthwise convolution is that the depthwise convolution applies the convolution along only one spatial dimension (i.e. channel) while a normal convolution is applied across all spatial dimensions/channels at each step.

The output of a pointwise convolutional layer has the same spatial dimensions as the input, but with a different number of channels.

PPT 7

VISUAL EMBEDDINGS

An image embedding is a vector representation of an image that captures its high-level features and semantic information in a lower-dimensional space. These embeddings can be obtained by passing an image through a pre-trained neural network, which extracts the features of the image in a compressed form. The resulting vector is a numerical representation of the image that can be used for tasks such as image classification, object detection, and image retrieval.

These are basically the output that we get during the training but just before the final layer.

VISUAL SIMILARITY SEARCH

Extract visual embeddings using a pretrained network

Store the vector embeddings in a database

Provide input image -> calculate visual vector embedding

Calculate distance between input image embeddings and all the image embeddings in the database

Display the k-Nearest Neighbours based on the distance calculated

ONE SHOT LEARNING

SIAMESE NETWORK

A siamese network is a type of neural network architecture used for solving tasks related to similarity or matching.

The siamese network takes two input images and processes them through the same set of convolutional and pooling layers to extract features. The features from both input images are then concatenated and fed to a fully connected layer, which outputs a single value that represents the similarity between the two input images.

The training of a siamese network involves presenting pairs of input images along with their corresponding labels (e.g. "same" or "different") to the network. The network then learns to minimize the distance between the embeddings of two images if they are labeled as "same", and maximize the distance if they are labeled as "different".

The siamese network architecture is commonly used for tasks such as image recognition, object tracking, face recognition, and similarity-based retrieval systems.

PPT 8

ROC

A Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classifier, which is a model that predicts one of two possible outcomes. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for different classification thresholds.

The TPR is the proportion of positive examples that are correctly identified by the classifier, while the FPR is the proportion of negative examples that are incorrectly identified as positive by the classifier. By varying the classification threshold, we can obtain different TPR and FPR values, and plot them on the ROC curve.

An ideal classifier would have a TPR of 1 (all positive examples correctly identified) and an FPR of 0 (no negative examples incorrectly identified as positive).

AUC

An AUC (Area Under the Curve) curve is a graphical representation of the performance of a binary classifier. It is the area under the ROC (Receiver Operating Characteristic) curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for different classification thresholds.

The AUC measures the overall performance of the classifier, summarizing the trade-off between TPR and FPR for all possible classification thresholds. An AUC of 1.0 indicates a perfect classifier, while an AUC of 0.5 indicates a random classifier. A higher AUC means that the classifier has better performance in distinguishing between the positive and negative classes.

The AUC curve is commonly used in machine learning to evaluate the performance of binary classifiers. It provides a single number summary of the classifier's performance, which is useful for comparing different models or tuning hyperparameters. The AUC curve is also useful for visualizing the performance of the classifier across different thresholds and identifying the optimal threshold for a given problem.

https://towardsdatascience.com/demystifying-roc-curves-df809474529a

PPT 9

FASTER RCNN

Faster R-CNN (Region-based Convolutional Neural Network) is an improved version of the original R-CNN model for object detection in images. It was introduced by Shaoqing Ren, et al. in 2015.

Faster R-CNN uses a two-stage approach for object detection. The first stage generates region proposals and the second stage classifies the objects in those regions. Here are the main components of Faster R-CNN:

Backbone network: This is usually a pre-trained convolutional neural network (CNN) like VGG16 or ResNet. The backbone network processes the input image and extracts features from it.

Region Proposal Network (RPN): The RPN takes the feature maps generated by the backbone network and generates object proposals, which are regions of interest in the image where an object might be present. The RPN is trained to predict the probability of an object being present at a given location and to propose a set of bounding boxes that tightly fit the object.

Region of Interest (RoI) Pooling: Once the RPN has generated object proposals, they are passed to the RoI pooling layer, which extracts a fixed-length feature vector for each proposal. The RoI pooling layer extracts features from the feature maps generated by the backbone network using the proposals generated by the RPN.

Object Detection Network: The object detection network takes the feature vectors generated by the RoI pooling layer and classifies the objects in the regions of interest. This is usually a fully connected neural network that takes the fixed-length feature vectors as input and outputs the class probabilities and bounding box coordinates.

The main advantage of Faster R-CNN over the original R-CNN is that it eliminates the need for expensive region proposal computation for each CNN feature map. This results in a faster and more efficient object detection algorithm.

DIFFERENCE BETWEEN FAST RCNN AND FASTER RCNN

Fast R-CNN and Faster R-CNN are both object detection models, but Faster R-CNN is an improved version of Fast R-CNN with better performance and faster speed. The main differences between Fast R-CNN and Faster R-CNN are:

Architecture: In Fast R-CNN, a region proposal algorithm like Selective Search is used to generate candidate regions, and then a CNN is applied to each region to classify and refine the bounding boxes. In Faster R-CNN, a Region Proposal Network (RPN) is used to generate region proposals, which share convolutional features with the detection network. This makes Faster R-CNN faster and more accurate than Fast R-CNN.

Speed: Faster R-CNN is faster than Fast R-CNN because it eliminates the need for a separate region proposal step. The RPN generates region proposals directly from the convolutional features, which saves time and reduces computation.

Accuracy: Faster R-CNN is more accurate than Fast R-CNN because the RPN generates more accurate region proposals than selective search.

In summary, Faster R-CNN improves on Fast R-CNN by integrating the region proposal step into the model architecture, which makes it faster and more accurate.

REIGON PROPOSAL NETWORK

RPN identifies RoIs based on the last feature map of the pretrained CNN. Faster RCNN uses RPN to perform the region proposals instead of using selective search like its earlier versions

The architecture of RPN is composed of two layers

A 3x3 conv layer with 512 filters

Two parallel 1x1 conv layers

Classification layer: Predicts whether the is an object present in the proposed region (or just random background

Bounding box regression layer: bounding box of proposed RoIs .

MEAN AVERAGE PRECISION(Map)

mAP stands for mean Average Precision, which is a metric used to evaluate the performance of object detection algorithms. It measures how well the algorithm is able to detect and locate objects of interest in an image.

To understand mAP, we first need to understand precision and recall. Precision is the ratio of correctly detected objects to the total number of objects detected, while recall is the ratio of correctly detected objects to the total number of objects in the image.

To calculate mAP, we first calculate the precision and recall values for different confidence thresholds. A confidence threshold is a threshold value that is set to determine whether an object is detected or not. Then, we plot a precision-recall curve, which is a graph of precision values on the y-axis and recall values on the x-axis.

mAP is then calculated as the area under this precision-recall curve. A higher mAP value indicates that the object detection algorithm is better at detecting objects in an image.

In simple terms, mAP is a measure of how good an object detection algorithm is at finding objects in an image, and a higher mAP value means a better algorithm.

PPT 10

YOLOV1(YOU ONLY LOOK ONCE)

YOLOv1 (You Only Look Once version 1) is an object detection algorithm.It is a popular real-time object detection algorithm that is known for its simplicity and speed.

The main idea behind YOLOv1 is to take an entire image as input and pass it through a single neural network, which divides the image into regions and predicts bounding boxes and class probabilities for each region.

The architecture of YOLOv1 consists of 24 convolutional layers followed by 2 fully connected layers. The input image is divided into a fixed grid of cells, and each cell predicts a fixed number of bounding boxes (usually two in the original YOLOv1 paper). Each bounding box is defined by five values: the x and y coordinates of the box's center, the width and height of the box, and a confidence score that represents the likelihood that the box contains an object.

In addition to predicting bounding boxes, YOLOv1 also predicts class probabilities for each bounding box. The network is trained on a large dataset of labeled images using a loss function that penalizes both classification and localization errors.

Once the network has made predictions for each cell, a post-processing step is applied to remove overlapping bounding boxes using Non-Maximum Suppression (NMS). NMS is a technique that removes redundant bounding boxes that have a high overlap with another box and keeps only the box with the highest confidence score.

Overall, YOLOv1 is known for its fast inference speed and the ability to detect objects in real-time. However, it may not be as accurate as other more complex object detection algorithms, especially when it comes to detecting small objects.