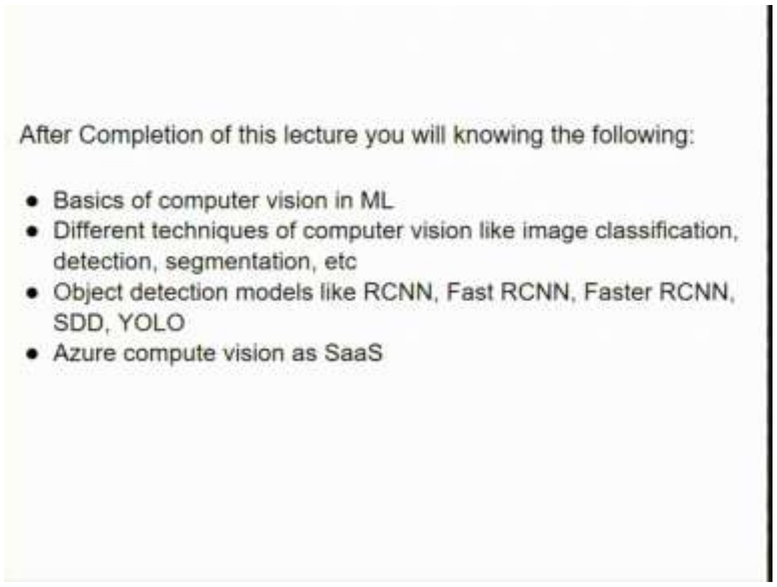
Foundation of Cloud IoT Edge ML

Professor Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture 06

ML-based Image Classifier at IoT-Edge

(Refer Slide Time: 0:26)

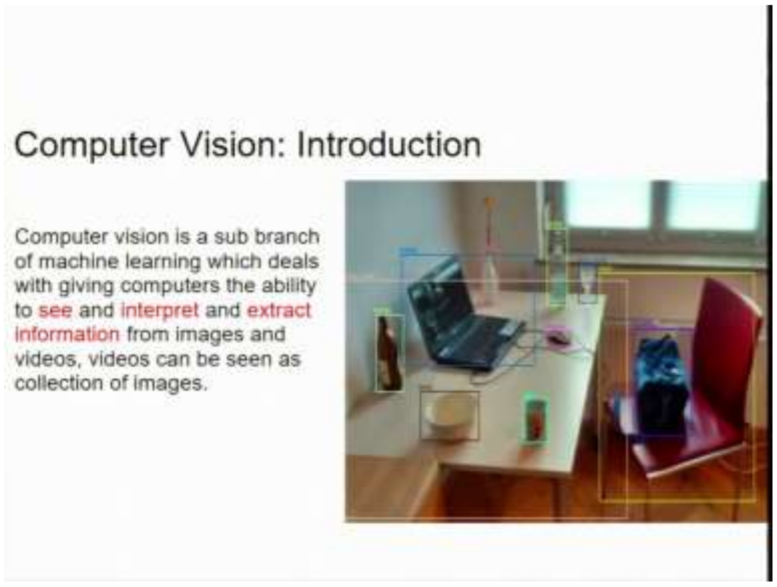


After Completion of this lecture you will knowing the following:

- Basics of computer vision in ML
- Different techniques of computer vision like image classification, detection, segmentation, etc
- Object detection models like RCNN, Fast RCNN, Faster RCNN, SDD, YOLO
- Azure compute vision as SaaS

I am Doctor Rajiv Mishra from IIT Patna, the topic of today's lecture is ML based image classifier at IoT Edge. In this lecture we will cover the following topics, basics of computer vision in machine learning, different techniques of computer vision like image classification, detection, segmentation, object detection models like RCNN, faster RCNN, so YOLO, SDD and so on, then we will look into the computer vision from Azure that is how that is being supported as software as a service SAAS.

(Refer Slide Time: 1:02)



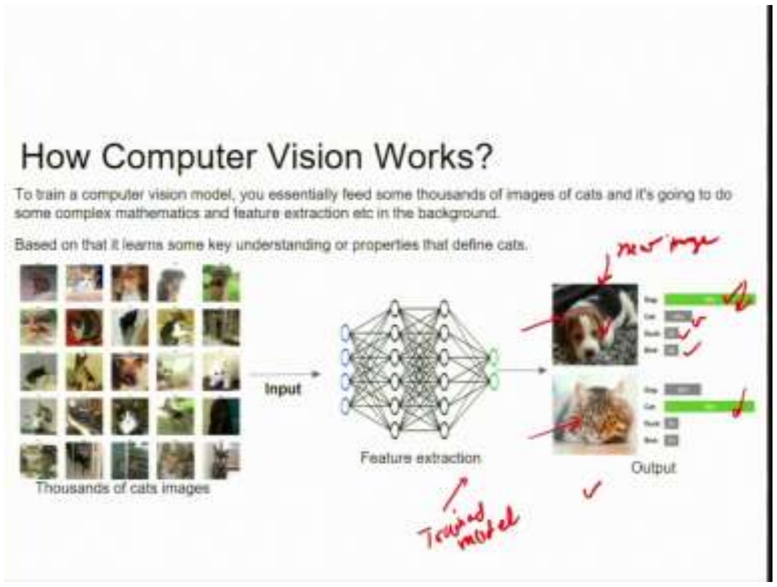
Computer Vision: Introduction

Computer vision is a sub branch of machine learning which deals with giving computers the ability to see and interpret and extract information from images and videos, videos can be seen as collection of images.

So let us give an introduction to the computer vision so as you know that computer vision is a sub branch of machine learning and this deals with giving the computer the ability to see and interpret and extract the information from the images and the videos so the videos can be seen as the collection of images, so here in this particular figure you can see there are different objects which you can see in an image.

So how to interpret whether it is a bottle or it is a laptop or some bowl or it is cup or it is a bag, chair and so on, so this particular ability is quite natural in human beings but how this capability is to be enabled into the machine using the concept which is called computer vision and that can be enabled with the help of machine learning and for that we are going to discuss some of the preliminaries and the basics and then we will see how these capabilities can be enabled into an IoT edge.

(Refer Slide Time: 2:20)



How Computer Vision Works?

To train a computer vision model, you essentially feed some thousands of images of cats and it's going to do some complex mathematics and feature extraction etc in the background.

Based on that it learns some key understanding or properties that define cats.

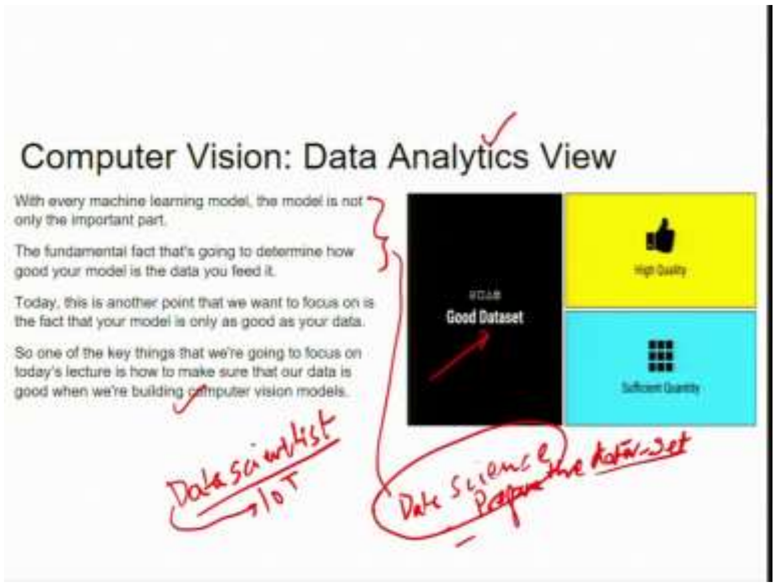
So let us understand a bit deeper about computer vision, its working, now here to train the computer vision in a machine learning model you essentially need to feed hundreds and thousands of images of interest so for example if you want to find out in an image whether it is a cat or it is some other animal so you have to now get such images as an example of a cat or some other animal so these particular thousands of images will now be used as the data to train the computer vision model for that this computer vision model will apply some complex mathematics and feature extraction to find out in contrast what is the background and what is the object into that image.

So based on that the model learns the key understanding or the properties of the objects that defines the cat or some other animal and distinguish that this is the basic understanding of the computer vision and it mimics the way humans can also understand it with the help of their vision and understanding through the neural networks the only difference here is that now instead of neural network it becomes an artificial neural network that is being simulated into the machines. But the capabilities are means inspired from human beings.

So take this example you have the rich data set of thousands of images of a particular object cat or animal and this particular so many number of images of a cat will be fed to a machine learning model that is running that computer vision that is the algorithm this particular algorithm or a machine learning model for computer vision is trained is called a trained once it goes through all these particular images and understands the features of this object that is called a cat.

So you can see the output that means when a new image is given to the trained model it will be able to detect with some percentage what is there into that image, whether it is a dog, it is cat or it is a duck or a bird, these are different classes they are called a classes, now you can see that if you apply a threshold that if it is more than 90 percent probability of a particular class then you can declare that this is an image of a dog and the other image is the image of a cat so this is a simple way to understand how the computer vision with the help of algorithm and the data set works.

(Refer Slide Time: 5:40)



Computer Vision: Data Analytics View

With every machine learning model, the model is not only the important part.

The fundamental fact that's going to determine how good your model is the data you feed it.

Today, this is another point that we want to focus on is the fact that your model is only as good as your data.

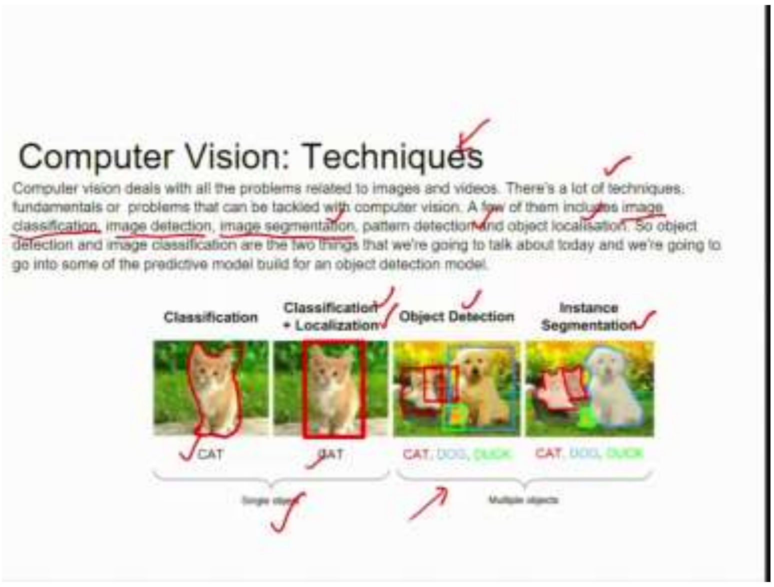
So one of the key things that we're going to focus on today's lecture is how to make sure that our data is good when we're building computer vision models.

So now let us see the other view that is called the data analytics part which is associated with this computer vision, so with every machine learning model the model is not only the important part the fundamental fact is that it is going to determine how good your model is the data which you feed to it, so therefore this particular aspect which is called the data science is to prepare the data or it is also called the data set and this particular data set if it is properly prepared by the domain experts and with the help of a data scientists then your model will be performing a very good.

So therefore there is another point that we have to focus on the fact that your model is only as good as your particular data it is not that machine learning model is doing good or bad it is the data set which becomes very important factor to train the machine learning model and this aspect is called the data science, so to perform this computer vision you have to have a good understanding of a data science and those experts are called data scientists.

Now if you want to do an IoT internet of things which requires, which is enabled with this computer vision capabilities then those people who are working for the IoT they have to be a good understanding or a data scientist working for an IoT these are different job roles for this computer vision role, now this particular lecture will make you sure that how your, how you can build your good data, how you can prepare the good data set so that you can build a very good computer vision model. So here you can see that what we mean by good data set is a data set which can train the model with a good set of examples is called a good data set.

(Refer Slide Time: 8:03)



Computer Vision: Techniques

Computer vision deals with all the problems related to images and videos. There's a lot of techniques: fundamentals or problems that can be tackled with computer vision. A few of them includes image classification, image detection, image segmentation, pattern detection and object localisation. So object detection and image classification are the two things that we're going to talk about today and we're going to go into some of the predictive model build for an object detection model.

Now let us see what are the techniques when we say that you want to train the model, machine learning models for computer vision what are the techniques which are needed for doing this kind of training, before that what are the techniques which are needed to get this computer vision working, so computer vision as you know that deals with all the problems related to the images and the videos and you know that images comprises of pixels that is in the digital forms and the videos comprises of set of several images.

So these are the fundamental principles has to be understood in the computer vision so with the help of so many pixels in the images you have to now decide whether the object in the image is a cat or it is some other animal that is cat or a dog, so how to understand from these particular collection of pixels to get this object understood, so there are a lot of techniques or the fundamentals that can be tackled with the computer vision and few of them include this technique which is called the image classification.

So that means image classification means that the object which is build out of several pixel together is able to classify as the cat, so this is called the image classification then the next issue is called the image detection, so image detection means that in a particular image where that object is located this is also sometimes called the localization, so classification that is the there is a cat into that image but where in that image that cat locates called localization.

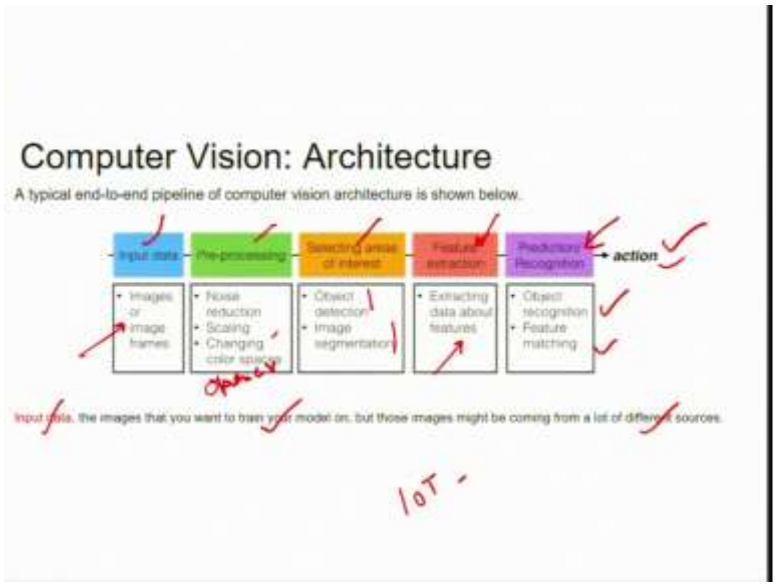
Now to get this particular classification and localization done for that you require to build a segment or segment that particular object into the image this technique is called segmentation, so segmentation is that given a image you have to now segment the image with a different color in contrast to the all other things in the image becomes the background so that is called the image segmentation similarly inside that image which is segmented now you have to do the image classification then comes the next issue called pattern detection and object localization that we have already defined.

So the object detection and image classification are the two things that we are going to discuss in this particular session in this lecture so we are going to, to go into some models they are called predictive models that means they will do a predictions based on this object detection methods.

So, let us complete that for a single object so far so good but what about if let us say that your image has more than one objects whether it is having a cat it is having a dog and it is having a duck also there are different classes of objects of your interest in your computer vision and all the objects how they are going to be means identified, segmented, localized and classified all these things together is called the object detection.

So if several objects are there so all the objects are to be now identified and this entire technique is called object detection, so object segmentation instance is also shown over here for object detection you have to segment the images as I told you that there are different objects so they are all segmented here in this case so whether it is the single object or there are multiple objects all this object detection techniques will be able to handle that, so these are the techniques which are very much needed to build a computer vision application.

(Refer Slide Time: 12:19)


Computer Vision: Architecture

A typical end-to-end pipeline of computer vision architecture is shown below.

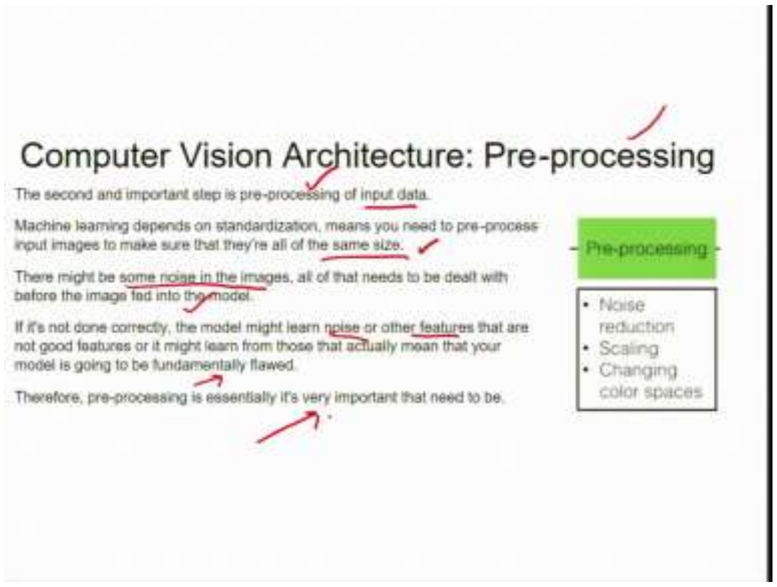
Input data, the images that you want to train your model on, but those images might be coming from a lot of different sources.

So let us see this what are the architecture for computer vision, so typical end to end pipeline for computer vision architecture is that once first you have to get the input data that is nothing in the form of images of those particular objects which need to be classified first, these objects require a pre processing that means the noise part in the images need to be reduced that object into the image or that image need to be scaled and also sometimes requires to change the color so openCV is the software now that is used for doing this preprocessing before this particular object becomes usable for the training of particular computer vision model.

Then the next process is called selecting the areas of interest that is to do the object segmentation and the object detection that is this particular technique will be applied once the pre-processing of that particular data set is complete then the next part is that how to identify the features of the objects of interest that is from extracting the data about the feature and then based on this learning the inferencing will be performed that is about the prediction or the recognition that is called object recognition and the feature matching.

And finally based on this some action has to be taken if it is an IoT device so the action could be that it will send some information to actuator for example if the IoT device is monitoring a particular traffic in the road and it founds that too many cars are there and almost a traffic jam then the signal at the junctions has to work that means this action has to be now fed to that particular junction so therefore these input data which is very much needed for this computer vision the images that you want to train your model but these images might be coming from lot of different sources so all these pipeline will be used to build that particular data set in the computer vision.

(Refer Slide Time: 14:52)



Computer Vision Architecture: Pre-processing

The second and important step is pre-processing of input data.

Machine learning depends on standardization, means you need to pre-process input images to make sure that they're all of the same size.

There might be some noise in the images, all of that needs to be dealt with before the image fed into the model.

If it's not done correctly, the model might learn noise or other features that are not good features or it might learn from those that actually mean that your model is going to be fundamentally flawed.

Therefore, pre-processing is essentially it's very important that need to be.

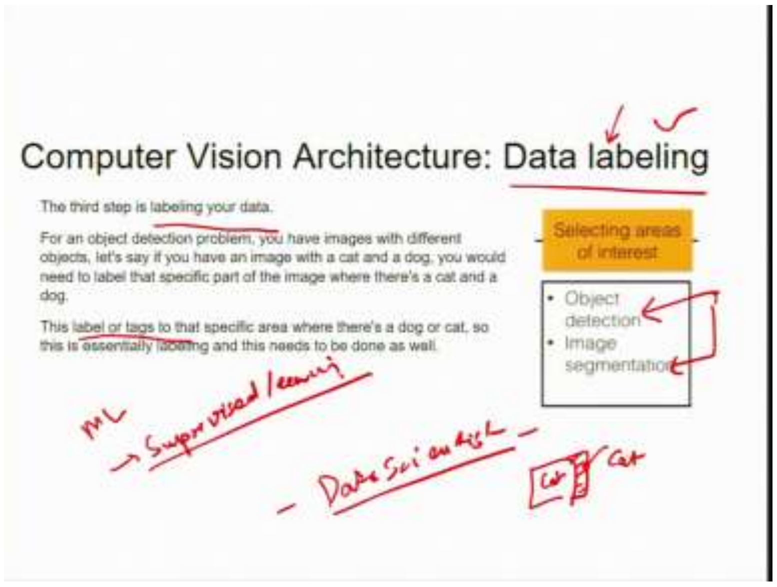
Pre-processing

- Noise reduction
- Scaling
- Changing color spaces

Now let us see how we are going to handle this pre processing which is very very important so the input data set to overcome from various issues, noise, scaling and color changing all this is being preprocessed in that raw data so after pre-processing you will get a data, a prepared data which will be fed for machine learning model training, so machine learning often depends upon the standardization that means you need to preprocess the input image to make sure that they are of same size so that the machine learning model for training will not find any issues.

Similarly there are some other fine kinds of noise while taking that image from the camera so all that need to be dealt before being used in a data set for model training, so if it is not correctly done the model might learn the noise part and also other features which are not good for image recognition or object detection and therefore the flawed the fundamentally it may go wrong and the model may not give the accurate predictions so therefore pre-processing is very very essential and we have discussed this.

(Refer Slide Time: 16:16)



Computer Vision Architecture: Data labeling

The third step is labeling your data.

For an object detection problem, you have images with different objects, let's say if you have an image with a cat and a dog, you would need to label that specific part of the image where there's a cat and a dog.

This label or tags to that specific area where there's a dog or cat, so this is essentially labeling and this needs to be done as well.

Selecting areas of interest

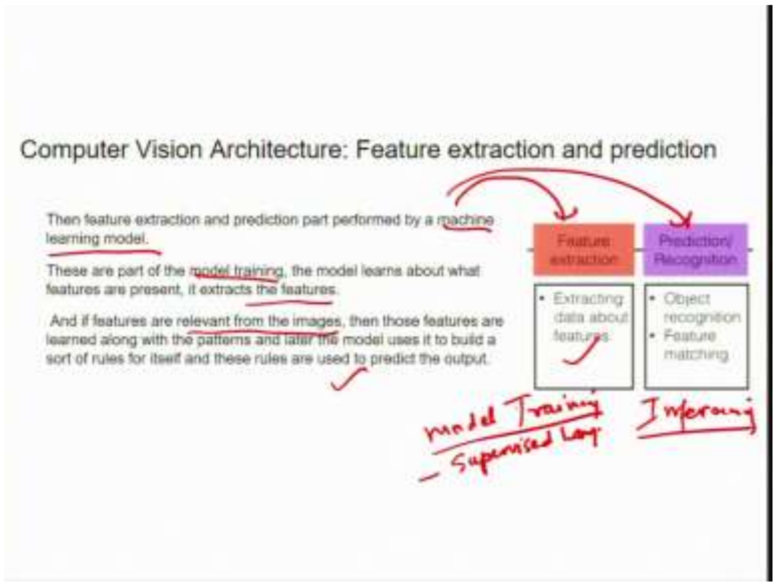
- Object detection
- Image segmentation

Once the data is pre-processed the next part is about the data scientists, they have to label the data so what do you mean by the label so that means if a picture of a cat is presented so that label is attached and this label will say that this is a image of a cat, so you require a domain expert to see the image and tell similarly the data scientist also is required so how many minimum number of images are needed to train the model so these are the different things which is called the labeling your data.

So for object detection you have the images that is taken in different way of the same object and if you have a image of a cat or a dog you need to label that, that means train you have to tell, label means you have to tell the model that this is the image of a cat. So, this label or the tag to the specific area where the dog or a cat is lying is essentially put as the label and this needs to be done very well by.

So, this particular labeling is very much essential in a supervised learning that is one part of machine learning which handles this computer vision that is called supervised learning so these supervised learning model require this data labeling and that is that is of our interest that is for object detection and the image segmentation will require this aspect to be completed that is called data labeling.

(Refer Slide Time: 18:11)



Computer Vision Architecture: Feature extraction and prediction

Then feature extraction and prediction part performed by a machine learning model.

These are part of the model training, the model learns about what features are present, it extracts the features.

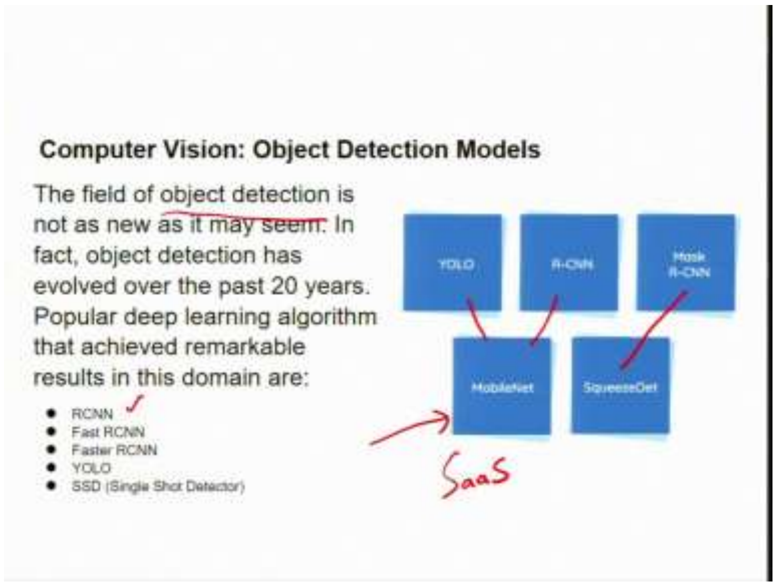
And if features are relevant from the images, then those features are learned along with the patterns and later the model uses it to build a sort of rules for itself and these rules are used to predict the output.

Now feature extraction and prediction is the next step in that process of computer vision, so the feature extraction and the prediction part will be performed by the machine learning model for that you require, you have to finalize which machine learning model you are going to do to apply on for with a data set for computer vision, so these are the parts of the model training so the model learns about the features which are present to distinguish between the cat and the dog and it will extract the features so if it is the deep learning then these features are automatically learned, if it is a machine learning algorithm then this particular data scientists they have to now do a feature engineering and identifies what features are important to recognize whether it is classified, whether it is a cat or a dog.

And if the features are relevant in the images then those features are learned through the pattern and later the model uses it to build a sort of rules for itself and these rules are then used to do the predictions when a new data set is given so this particular part that is called a feature extraction is a part of a machine learning algorithm to do this.

Similarly, for prediction and recognition that is also a inferencing part after the training the model will do this kind of recognition that is object recognition and a feature engineering so this part feature extraction is used at during the training process and in the supervised, this requires a supervised learning and then the next part which is called the prediction or the recognition this is called inferencing and this is used to do a predictions so we are going to see these kind of things.

(Refer Slide Time: 20:25)



Computer Vision: Object Detection Models

The field of object detection is not as new as it may seem. In fact, object detection has evolved over the past 20 years. Popular deep learning algorithm that achieved remarkable results in this domain are:

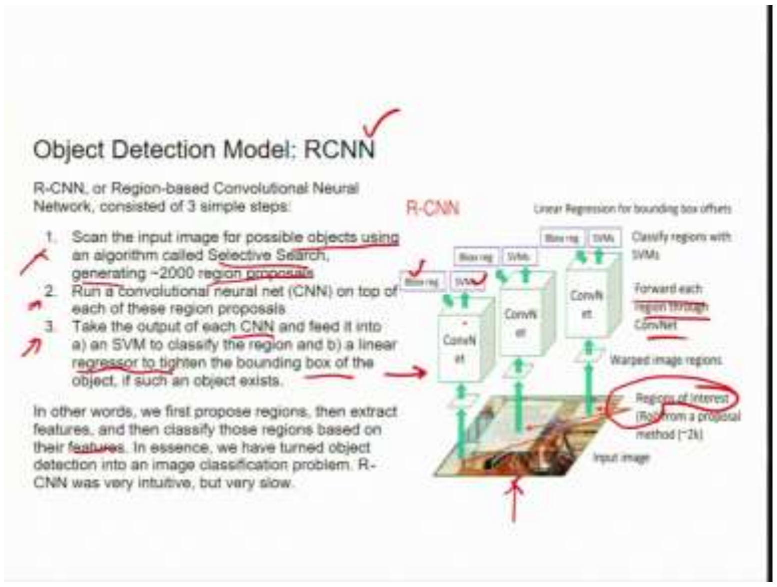
- RCNN ✓
- Fast RCNN
- Faster RCNN
- YOLO
- SSD (Single Shot Detector)

YOLO
R-CNN
Mask R-CNN
MobileNet
SqueezeDet
SaaS

Now let us see what are the different standard models which are now over the period evolved for doing this computer vision, so the field of object detection is has gone to a tremendous transformation over coming from the earlier days of very low resolution and very low accuracy models using the algorithms, why, because it requires artificial intelligence or the learning based system like which mimics the human beings not the fixed algorithm which can learn all that details of objects.

So therefore, over the last 20 years a lot of innovations happened and those models are now finalized and decided as the key building blocks for object detection in computer vision, so let us know them they are called regional RCNN, fast RCNN, faster RCNN, YOLO and SSD that is single shot detector, these models are now standardized and now put in the cloud and they are now given as the service that is the software as a service platform they are running these one of these algorithms into these kind of cloud-based services.

(Refer Slide Time: 21:56)



Object Detection Model: RCNN

R-CNN, or Region-based Convolutional Neural Network, consisted of 3 simple steps:

1. Scan the input image for possible objects using an algorithm called Selective Search, generating ~2000 region proposals
2. Run a convolutional neural net (CNN) on top of each of these region proposals
3. Take the output of each CNN and feed it into a) an SVM to classify the region and b) a linear regressor to tighten the bounding box of the object, if such an object exists.

In other words, we first propose regions, then extract features, and then classify those regions based on their features. In essence, we have turned object detection into an image classification problem. R-CNN was very intuitive, but very slow.

R-CNN
Linear Regression for bounding box offsets
Bbox reg
SVMs
Classify regions with SVMs
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

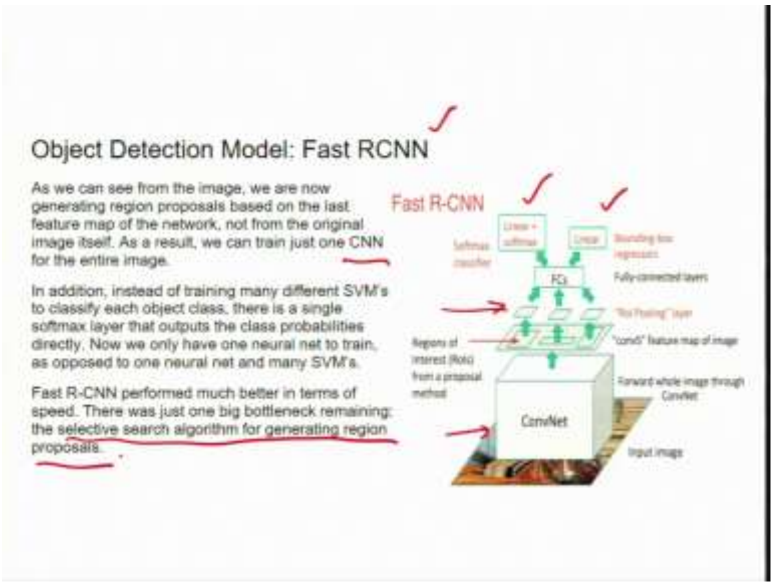
Let us go one by one very briefly about them, why, because this is a very little background which we intend to prepare so that we will understand this particular lecture further in a great details so RCNN is let us say that first model which you can use for object detection which is developed, so RCNN the full form is region based convolutional neural network, so region based convolutional neural network uses three different steps.

One is that it will scan the image for looking up a possible objects using the algorithm which is called selective search and those kind of regions where there is a possibility of the object so they will identify and they are called region proposals then it will run a convolutional neural network CNN on top of each of these regions of interest and then take this output of each RCNN and feed it into a support vector machine to classify the region and or a linear regressor to find out the bounding box around that particular object if that object exists.

In other words, so the first proposed the regions and then extract the features and then classify those features for those objects, so in essence these object detection uses this image classification and RCNN is very intuitive and very slow because it has to first identify those regions of interest that is called RoI from the proposal methods from the input image so if the input image has three, four dogs and cat so for every object region of interest will be earmarked and then so many number of, for example if five regions are there five different CNN will run on each region proposal.

Now this is a duplicacy that so many convolutional neural network if let us say image has many objects so so many number of convolutional neural network will run and then the output of that convolutional neural network require a classifier so that is it is whether it is a SVM classifier and SVM classifier will classify whether it is a cat or a dog and then where that particular object lies into the image so it also require a regressor, a regressor means it will give the coordinates height and length of that particular object so that particular object so therefore both the classifier and regressor will be attached after the convolution. So this becomes quite slow to classify this kind of images using RCNN.

(Refer Slide Time: 25:03)



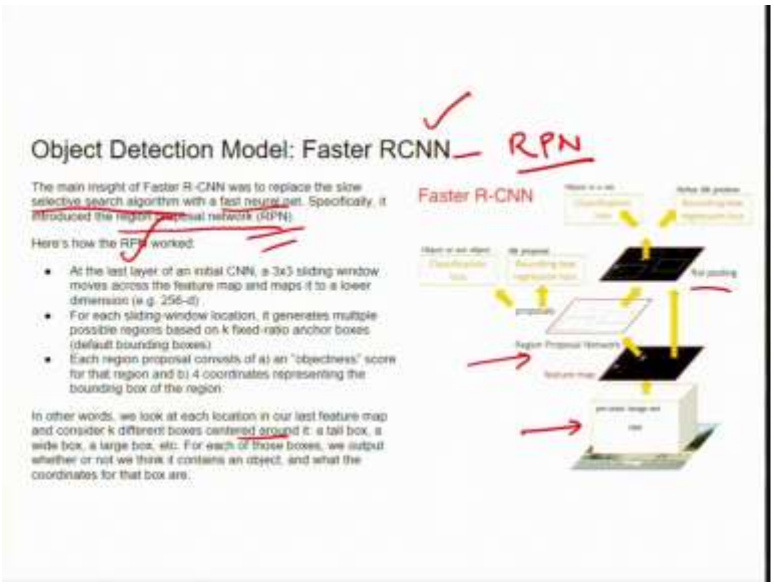
Object Detection Model: Fast RCNN

As we can see from the image, we are now generating region proposals based on the last feature map of the network, not from the original image itself. As a result, we can train just one CNN for the entire image.

In addition, instead of training many different SVM's to classify each object class, there is a single softmax layer that outputs the class probabilities directly. Now we only have one neural net to train, as opposed to one neural net and many SVM's.

Fast R-CNN performed much better in terms of speed. There was just one big bottleneck remaining: the selective search algorithm for generating region proposals.

Fast R-CNN
Linear + softmax
Linear
Softmax classifier
Bounding box regressors
FCs
Fully-connected layers
"RoI Pooling" layer
Regions of interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image



Object Detection Model: Faster RCNN

RPN

The main insight of Faster R-CNN was to replace the slow selective search algorithm with a fast neural net. Specifically, it introduced the region proposal network (RPN).

Here's how the RPN worked:

- At the last layer of an initial CNN, a 3x3 sliding window moves across the feature map and maps it to a lower dimension (e.g. 256-d)
- For each sliding-window location, it generates multiple possible regions based on k fixed-ratio anchor boxes (default bounding boxes)
- Each region proposal consists of a) an "objectness" score for that region and b) 4 coordinates representing the bounding box of the region.

In other words, we look at each location in our last feature map and consider k different boxes centered around it: a tall box, a wide box, a large box, etc. For each of those boxes, we output whether or not we think it contains an object, and what the coordinates for that box are.

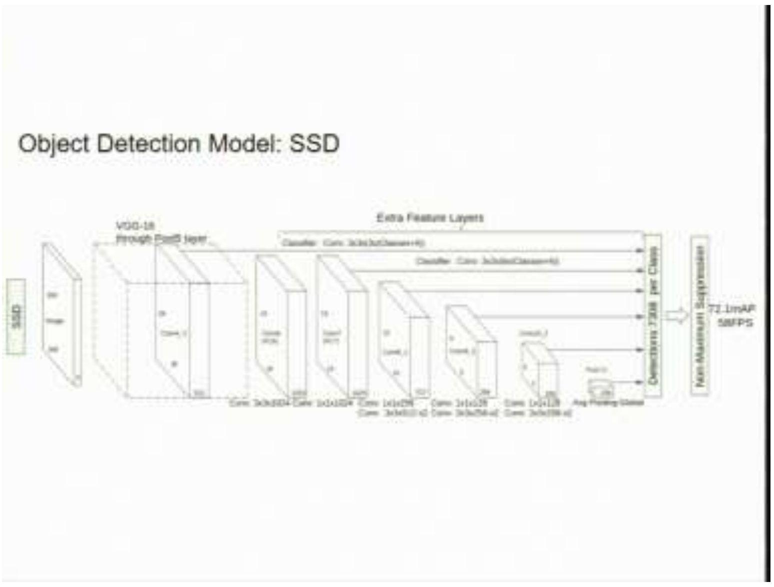
So faster RCNN is an improvement so you can see here there is only one convolution neural network and using this convolutional neural network it will scan and identify the regions of interest and for every region you will be having the fully connected layer and this fully connected layer will now do the both the classifier and the regressor, so this becomes faster because of the fact that so many neural convolutional neural networks CNN is not required only one CNN will do that.

So therefore, this improvement is called the fast RCNN so fast RCNN perform much better in terms of the speed but there was one bottleneck remaining is that the selective search algorithm for generating the region proposal was not efficient and therefore led to the faster RCNN, so let us see that faster RCNN used to replace the slow selective search algorithm with the fast neural network.

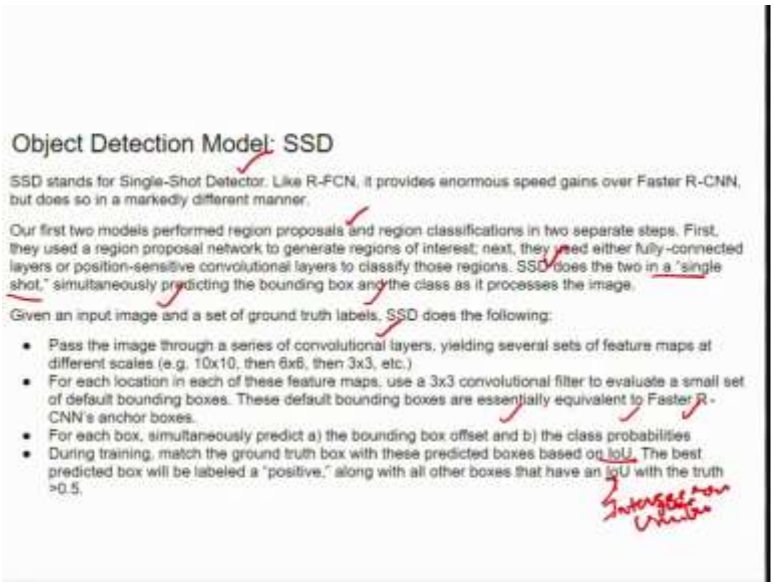
So it introduced the region proposal network so it has, so this selective search algorithm along with that region search together is called region proposal network RPN so here you can see that after the CNN doing this scan what it will do it will identify those selective search using region proposal network so these regional proposals network will contain a region of interest and then with a pooling layer then it will go ahead for a classifier and a regressor.

Classifier will classify the object and regressor will then bound a box on top of it, so let us see the unique thing about faster RCNN is region proposal network so this in the last layer of the initial CNN the sliding window moves around across the feature map and maps it to a lower dimension for each sliding window location it generates multiple possible regions and each region proposal now consists of an objects and the coordinates. In other words we have to look at each location and consider K different block center around that so this was the faster RCNN.

(Refer Slide Time: 27:31)



Object Detection Model: SSD



Object Detection Model: SSD

SSD stands for Single-Shot Detector. Like R-FCN, it provides enormous speed gains over Faster R-CNN, but does so in a markedly different manner.

Our first two models performed region proposals and region classifications in two separate steps. First, they used a region proposal network to generate regions of interest; next, they used either fully-connected layers or position-sensitive convolutional layers to classify those regions. SSD does the two in a "single shot," simultaneously predicting the bounding box and the class as it processes the image.

Given an input image and a set of ground truth labels, SSD does the following:

- Pass the image through a series of convolutional layers, yielding several sets of feature maps at different scales (e.g. 10x10, then 6x6, then 3x3, etc.)
- For each location in each of these feature maps, use a 3x3 convolutional filter to evaluate a small set of default bounding boxes. These default bounding boxes are essentially equivalent to Faster R-CNN's anchor boxes.
- For each box, simultaneously predict a) the bounding box offset and b) the class probabilities
- During training, match the ground truth box with these predicted boxes based on IoU. The best predicted box will be labeled a "positive," along with all other boxes that have an IoU with the truth >0.5.

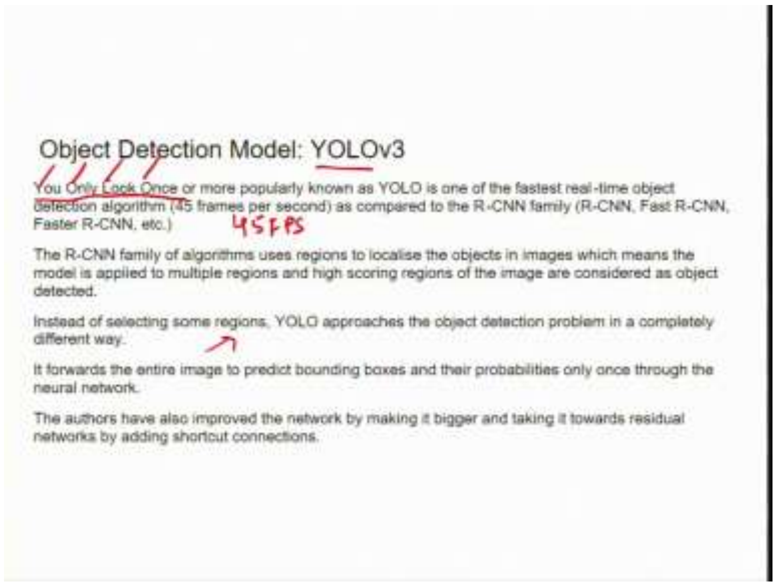
Intersection over Union

Now then there is another model which comes called object detection model called SSD, so SSD stands for single shot detector like faster RCNN it provides enormous speed gains over the faster RCNN but does so in a remarkably different manner, so we are not going in much details about that but only thing is that we have to tell that these two models will which perform the region proposal and region classification in separate steps and they use region proposal network to generate the regions of interest.

Next, they are fully connected layers are positive sensitive convolutional layer to classify these regions, so SSD that is single shot detector does the two in a single shot and that is why it is called a single shot detector simultaneously predicting the bounding box and the class it processes the image, so given an input image and the set of ground truth labels SSD does the following, pass the image through a series of convolutional layers yielding the several set of feature maps at different scales.

Now for each locations in each of these feature maps the convolutional filter to evaluate a small set of default bounding box, these default bounding box are essentially equivalent to a faster RCNN anchor box and for each box simultaneously predict the bounding box offset and the class probability and during the training, the match the ground truth box with the predicted box using IoU, so IoU full form is intersection over union.

(Refer Slide Time: 29:28)



Object Detection Model: YOLOv3

You Only Look Once or more popularly known as YOLO is one of the fastest real-time object detection algorithm (45 frames per second) as compared to the R-CNN family (R-CNN, Fast R-CNN, Faster R-CNN, etc.)

45FPS

The R-CNN family of algorithms uses regions to localise the objects in images which means the model is applied to multiple regions and high scoring regions of the image are considered as object detected.

Instead of selecting some regions, YOLO approaches the object detection problem in a completely different way.

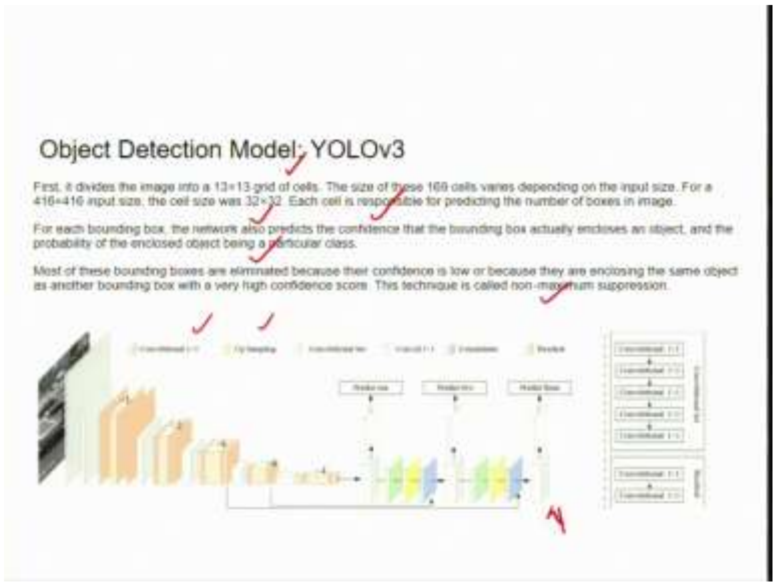
It forwards the entire image to predict bounding boxes and their probabilities only once through the neural network.

The authors have also improved the network by making it bigger and taking it towards residual networks by adding shortcut connections.

So let us see all these concepts in very standard model and very efficient also called YOLO version 3, YOLO full form is You Look Once, so You Only Look Once YOLO, so is popularly known as YOLO is one of the fastest real time object detection algorithm that is the speed is measured in frames per second FPS so this is 45 FPS speed in which this real time object detection takes place in YOLO as compared to RCNN, faster RCNN, etcetera, they are quite slow compared to the YOLO.

So RCNN family of algorithm uses regions to localize the objects in the images which means that the model is applied to multiple regions with a high scoring regions of the image are considered as object detected, instead of selecting some regions YOLO approaches the object detection problem in a different manner it forwards the entire image to a predicting bounding boxes and their probabilities only once through the neural network so there are many things are now optimized here in YOLO.

(Refer Slide Time: 30:44)



Object Detection Model: YOLOv3

First, it divides the image into a 13×13 grid of cells. The size of these 169 cells varies depending on the input size. For a 416×416 input size, the cell size was 32×32. Each cell is responsible for predicting the number of boxes in image.

For each bounding box, the network also predicts the confidence that the bounding box actually encloses an object, and the probability of the enclosed object being a particular class.

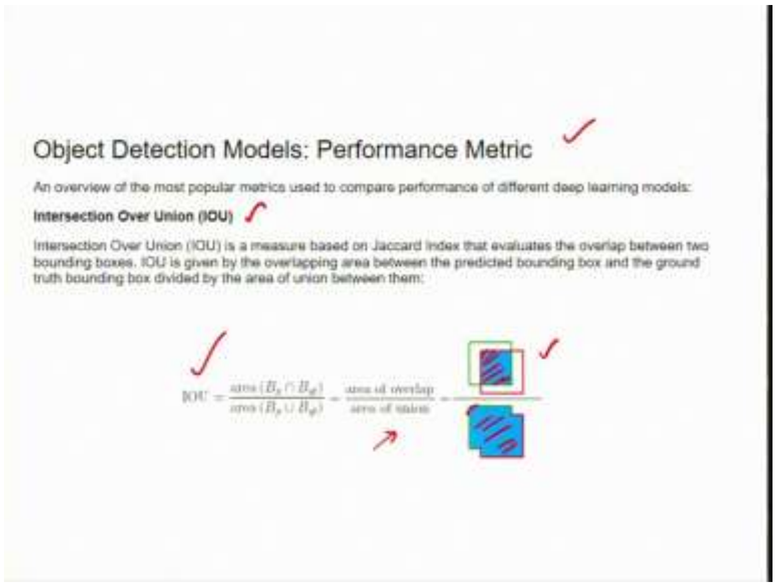
Most of these bounding boxes are eliminated because their confidence is low or because they are enclosing the same object as another bounding box with a very high confidence score. This technique is called non-maximum suppression.

So let us go ahead and see some of the details of about YOLO version 3 because this particular model which we will be discussing in the computer vision model of doing this computer vision at the edge of an IoT, so YOLO first divides the image into the grid of 13 by 13 and the size of these grid cells varies depending upon the input size and these cells each cell is responsible for predicting the number of boxes in that particular image.

So for each bounding box the network also predicts the confidence that the bounding box actually encloses the object and probability of enclosing the objects being a particular class so all these things are done simultaneously that is in one go, so most of these bounding box are eliminated because of their confidence is low or because they are enclosing the same objects as another bounding box with a very high confidence score and this is called non maximum suppression.

So therefore, what you can see over here is that this is an optimization of bounding boxes efficiently done in YOLO and also this particular protection and that is the classification and the regression both are done in one go so all things are being merged together and optimized in YOLO becoming a very fast and a very less space consuming model, so it comprises of so many layer, so many convolutional layer and then you can see that this will have this particular layer of this predictions.

(Refer Slide Time: 32:38)



Object Detection Models: Performance Metric

An overview of the most popular metrics used to compare performance of different deep learning models:

Intersection Over Union (IOU)

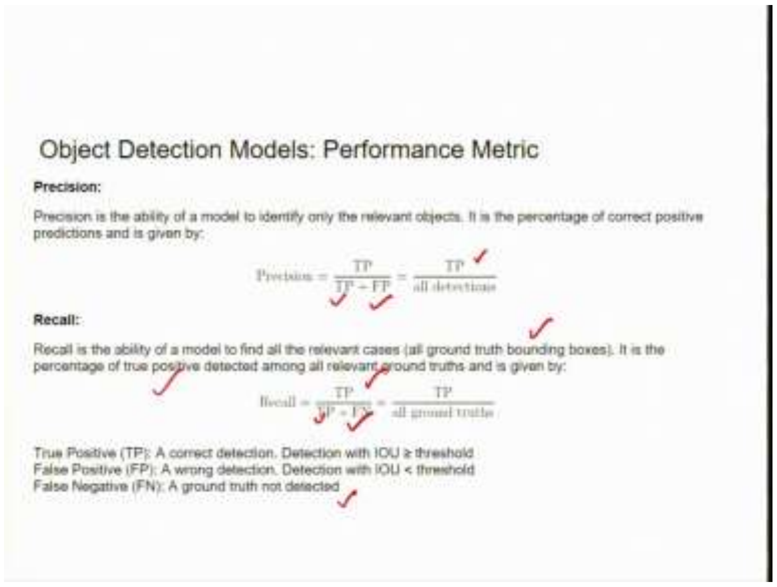
Intersection Over Union (IOU) is a measure based on Jaccard Index that evaluates the overlap between two bounding boxes. IOU is given by the overlapping area between the predicted bounding box and the ground truth bounding box divided by the area of union between them:

$$\text{IOU} = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} = \frac{\text{area of overlap}}{\text{area of union}} =$$

Now various models which we have discussed can be compared their performance with the metric there are several metrics which is now, we have to now understand before we go ahead so I have already mentioned about intersection over the union for bounding the box this is the measure which is used or which is based on Jaccard Index and evaluates the overlap between the bounding box.

So Intersection over union is given by overlapping area between the predicted bounding box and the ground truth bounding box divided by the area of union between them so here you can see through the example intersection or union is the area of the overlap so this is shown over here divided by the area of the union so with IO intersection of the union you have to now decide about the overlapping area to be fixed so that the bounding box fits well with the object.

(Refer Slide Time: 33:43)



Object Detection Models: Performance Metric

Precision:

Precision is the ability of a model to identify only the relevant objects. It is the percentage of correct positive predictions and is given by:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{all detections}}$$

Recall:

Recall is the ability of a model to find all the relevant cases (all ground truth bounding boxes). It is the percentage of true positive detected among all relevant ground truths and is given by:

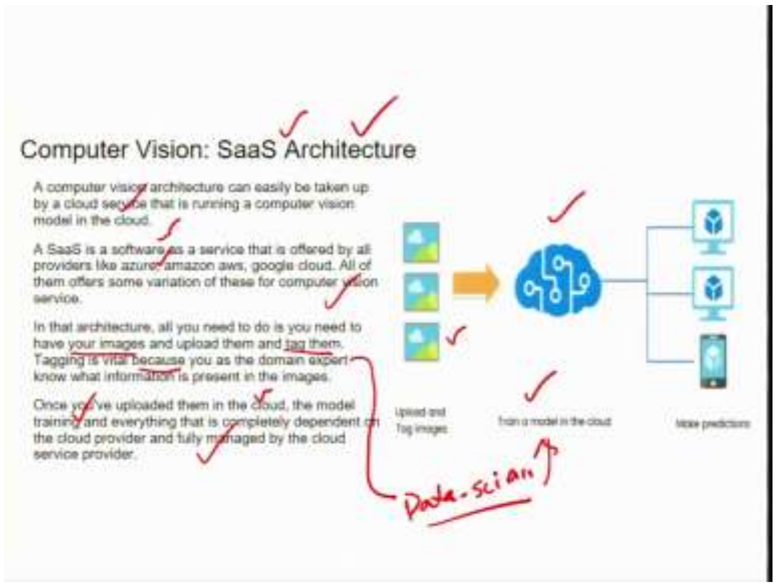
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{all ground truths}}$$

True Positive (TP): A correct detection. Detection with IOU ≥ threshold
False Positive (FP): A wrong detection. Detection with IOU < threshold
False Negative (FN): A ground truth not detected

The second method about performance is used that is called precision and recall, so precision is the ability of a trained model to identify only the relevant objects that is the percentage of the correct positive predictions and is given by the true positives divided by the true positives and the false positive that is called the precision, another measure is called the recall for understanding the performance of the model that is the trained model.

So recall is the ability of a model to find all the relevant cases that is all the ground truth bounding boxes so it is the percentage of true positives detected among all the relevant ground truths, so recall is given by this particular formula so recall is true positives divided by true positive plus false negatives so here this particular false negatives means the ground truth not detected correctly and false positives means a wrong detection that is detected and (false positive) true positive means a correct detection.

(Refer Slide Time: 34:59)



Computer Vision: SaaS Architecture

A computer vision architecture can easily be taken up by a cloud service that is running a computer vision model in the cloud.

A SaaS is a software as a service that is offered by all providers like azure, amazon aws, google cloud. All of them offers some variation of these for computer vision service.

In that architecture, all you need to do is you need to have your images and upload them and tag them. Tagging is vital because you as the domain expert know what information is present in the images.

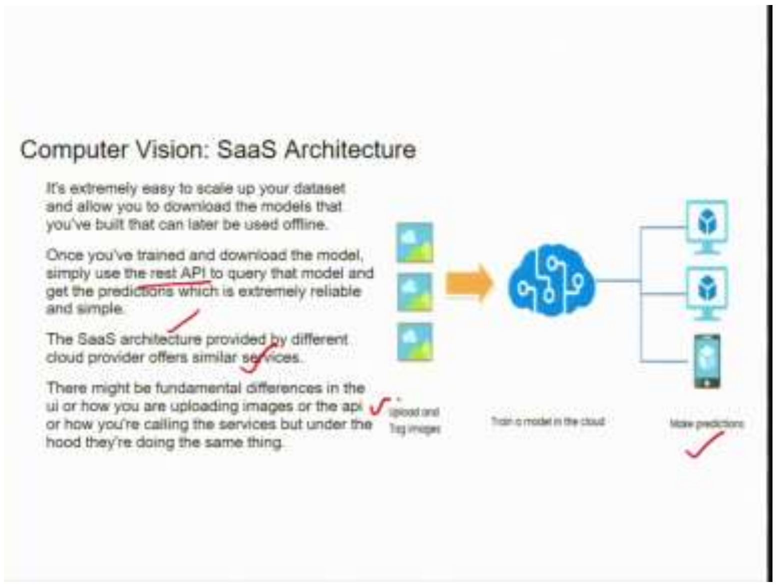
Once you've uploaded them in the cloud, the model training and everything that is completely dependent on the cloud provider and fully managed by the cloud service provider.

So this particular background is being built now we have to see about the computer vision services which are available by the cloud providers and they are called software as a service architecture, so computer vision architecture can easily be taken up by the some cloud service that is running a computer vision model in the cloud so that you can see that these particular model is provided by these different cloud providers using the technology, cloud technology called SaaS that is the software as a service.

So most of the prominent cloud provider they provide some sort of computer vision APIs or the services that is Azure, Amazon, Google and so on, they offer some variation of these kind of computer vision service but the basic fundamentals or the models you have to now decide which can fit into these kind of services so in that architecture all you need to do is to need to have your images available upload your images and then label them so labeling is vital why because it is to be done with the help of domain experts and the data science.

So data science is quite important again even you are choosing the model or the service through the platform that is software as a service, so once you upload your tagged image or a labeled image that is your data set for supervised learning then it will be applied with some machine learning model or a model which we have covered so far and everything that is completely dependent upon now the cloud provider fully managed that particular cloud so that is what shown over here first you have to prepare your data with the label and that data is to be uploaded to the cloud there that particular data set is presented with one of the algorithms which we have covered for doing this computer vision or training and it will train the model into the cloud and the trained model is now available to make the predictions.

(Refer Slide Time: 37:33)



Computer Vision: SaaS Architecture

It's extremely easy to scale up your dataset and allow you to download the models that you've built that can later be used offline.

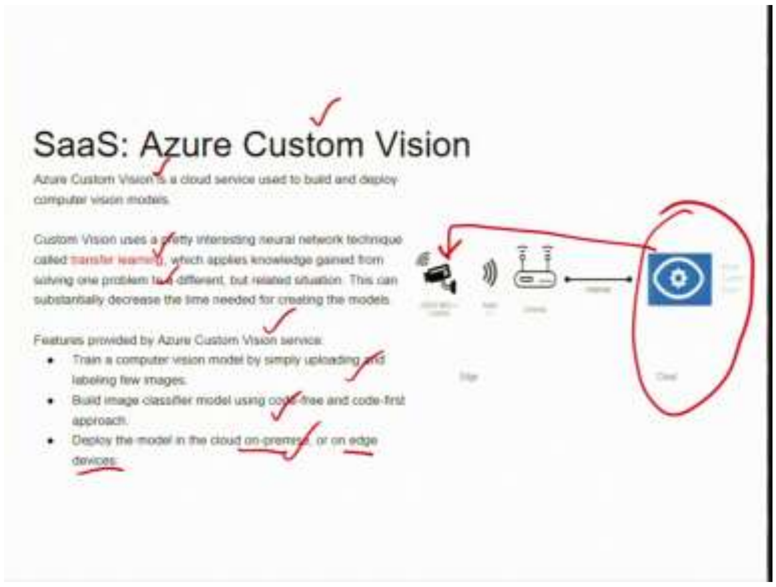
Once you've trained and download the model, simply use the rest API to query that model and get the predictions which is extremely reliable and simple.

The SaaS architecture provided by different cloud provider offers similar services.

There might be fundamental differences in the ui or how you are uploading images or the api or how you're calling the services but under the hood they're doing the same thing.

So this particular model which is trained can be accessible with the help of a service that is with the rest API and this particular model can do the predictions, so therefore software as a service which is being provided as a platform by different cloud providers for doing this computer vision service is very quite similar it depends upon the choice of the client or the developer. So there might be the fundamental difference in the UPIs and APIs which you are calling the service under the hood so we will be going in more detail discussing about that.

(Refer Slide Time: 38:18)



SaaS: Azure Custom Vision

Azure Custom Vision is a cloud service used to build and deploy computer vision models.

Custom Vision uses a pretty interesting neural network technique called transfer learning, which applies knowledge gained from solving one problem to a different, but related situation. This can substantially decrease the time needed for creating the models.

Features provided by Azure Custom Vision service:

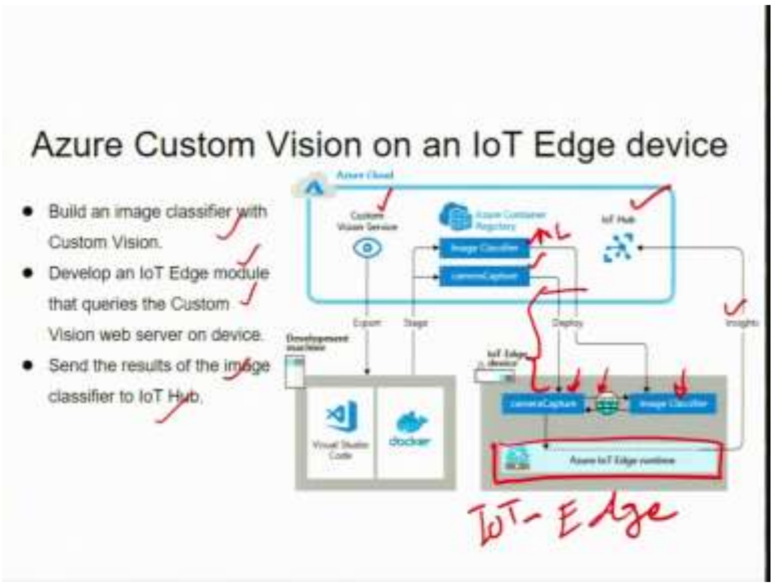
- Train a computer vision model by simply uploading and labeling few images.
- Build image classifier model using code-free and code-first approach.
- Deploy the model in the cloud on-premises, or on edge devices.

So let us see one such service which is called Azure custom vision service for doing the computer vision that is a software as a service platform, so Azure custom vision is a cloud service used to deploy, build the computer vision models this is one such example which we are now illustrating here, so computer vision uses the pretty interesting neural network technique and sometimes it is called the transfer learning because it uses a pre-trained model and then you can bring your own new set of images and you can retrain the model that is called transfer learning.

So which applies the knowledge gained out of solving one problem in a domain to a different problem which is also in a related situation this can substantially decrease the time of creating the models from the scratch, so the features which are provided by this Azure custom vision service is to train the computer vision model by simply uploading and labeling the few images that means you need not have to have a large number of image yet you can do this computer vision fairly well using this particular technique.

So you have to build a image classifier using code free and code first approach deploy the model in the cloud on premise or on the edge device so this is going to be a useful information for doing for enabling the edge with AI. So here in the picture all things are illustrated now you can see that this particular Azure custom vision application is running on the cloud as software as a service now over the internet the trained model can be either deployed on the edge and once the camera gets the new image this particular trained model will be able to do the predictions.

(Refer Slide Time: 40:27)



Azure Custom Vision on an IoT Edge device

- Build an image classifier with Custom Vision.
- Develop an IoT Edge module that queries the Custom Vision web server on device.
- Send the results of the image classifier to IoT Hub.

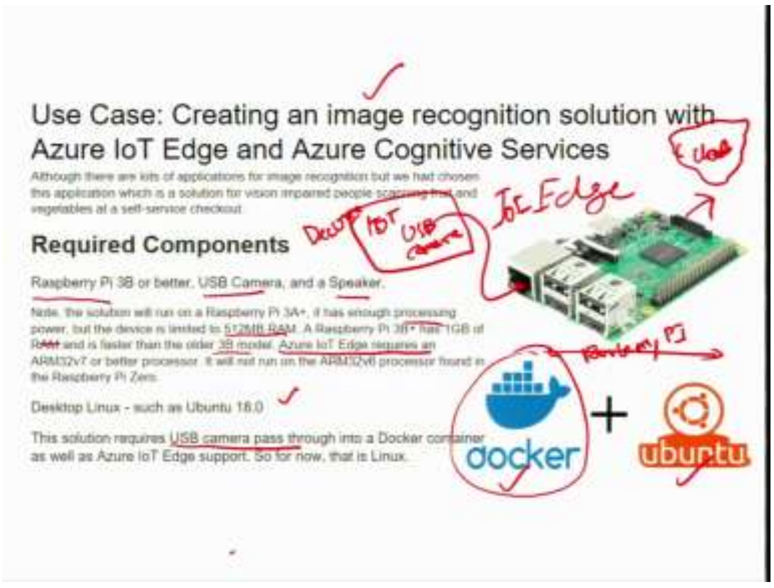
So Azure custom vision, let us see that how are you going to deploy on IoT edge device, so you can see this inside of this Azure cloud service so it is a custom vision service which runs inside the Azure cloud, now for that it will ask or it will build a image classifier based on one of the algorithms which we have covered earlier, so it will build a custom vision using custom vision and image classifier using some deep learning algorithm or a machine learning algorithm which we have discussed so it will only provide a framework and once the machine learning algorithm is being decided by the client or the developer then this particular model will automatically use that and perform this computer vision.

Now once this particular IoT edge module how that let us see how it will be prepared, so first this image classifier is to be built and then camera module which now will be taking the new images is being identified and these together is prepared as the container and using the Docker it will be now deployed into the edge so this is the edge, this is the IoT edge, so after containerization the camera module will come over here which is built into the cloud.

Similarly, the classifier is also now deployed on the edge and this is being enabled with the help of service which is running on the edge that is called Azure IoT edge runtime that we have already covered in the previous lecture, once these models that is for the image classifier and the camera capture is in place and this image camera capture and the image classifier they communicate with the help of a protocol called HTTP.

So HTTP is very close tight net protocol they have to work very closely, now this particular deployment will enable the edge to do the inside so this particular for doing more detail inside that particular pre-processed information or a data or inside what whatever is being inferenced by IoT Edge will be sent back to the IoT hub that is into the Azure Cloud so send the results of an image classifier to IoT hub for making the business decisions.

(Refer Slide Time: 43:09)



Use Case: Creating an image recognition solution with Azure IoT Edge and Azure Cognitive Services

Although there are lots of applications for image recognition but we had chosen this application which is a solution for vision impaired people scanning fruit and vegetables at a self-service checkout

Required Components

Raspberry Pi 3B or better, USB Camera, and a Speaker.

Note, the solution will run on a Raspberry Pi 3A+, it has enough processing power, but the device is limited to 512MB RAM. A Raspberry Pi 3B+ has 1GB of RAM and is faster than the older 3B model. Azure IoT Edge requires an ARM32v7 or better processor. It will not run on the ARM32v6 processor found in the Raspberry Pi Zero.

Desktop Linux - such as Ubuntu 18.0

This solution requires USB camera pass through into a Docker container as well as Azure IoT Edge support. So for now, that is Linux.

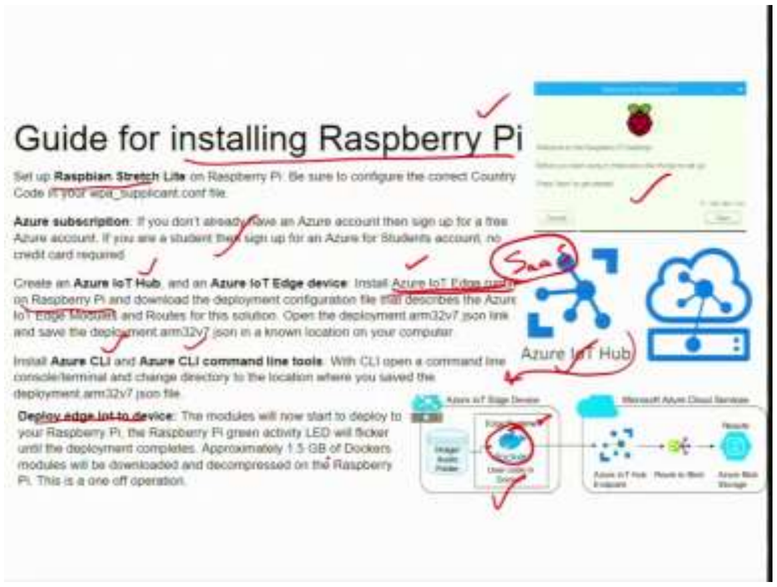
So the topic is to create an image recognition software or the solution with an IoT edge and Azure services, so you can see that lot of applications uses this image recognition and maybe that this particular application such application needs a solution that is vision based solution such as scanning the fruits for quality for segregating into the or classifying into the quality with a medium quality, low quality or high quality and vegetables and also there are various scanning at this at the airport for different baggages, self-service checkout, there are many many more applications, so let us see how that is all can be done with the help of this IoT, Edge IoT.

So in this particular lab session or lecture some of the components, some of the hardware which is needed is that Raspberry Pi 3B or better and an USB camera and a speaker so this is the picture of Raspberry Pi board, so Raspberry Pi board is a standalone computer and it has enough of processing power and it is the device has a limited memory within it, so Raspberry Pi 3 B plus board has 1 GB of RAM and now a newer version that is Raspberry Pi 4 has come now this particular board will be used to prepare to make this edge, IoT edge it will work as an IoT edge.

And other thing which you require is an is a desktop with a Ubuntu installed in it then you will be needing since it is a image recognition solution so you will be requiring a USB camera and that USB camera that is an IoT camera which is connected through a USB port here into this Raspberry Pi and this camera is a device or it is a thing, now it is this particular IoT, this will become an IoT edge, this is internet connected and this will be now in turn will be connected to the cloud, will have the connectivity to the, can be established.

Now there are two more icons which are shown you require a Ubuntu machine or a Linux based machine and then another thing is you have to need a docker, so all this is a requirement for this particular exercise.

(Refer Slide Time: 46:30)



Guide for installing Raspberry Pi

Set up **Raspbian Stretch Lite** on Raspberry Pi. Be sure to configure the correct Country Code in your wpa_supplicant.conf file.

Azure subscription: If you don't already have an Azure account then sign up for a free Azure account. If you are a student then sign up for an Azure for Students account, no credit card required.

Create an **Azure IoT Hub**, and an **Azure IoT Edge device**. Install Azure IoT Edge runtime on Raspberry Pi and download the deployment configuration file that describes the Azure IoT Edge Modules and Routes for this solution. Open the deployment.arm32v7.json link and save the deployment.arm32v7.json in a known location on your computer.

Install **Azure CLI** and **Azure CLI command line tools**. With CLI open a command line console/terminal and change directory to the location where you saved the deployment.arm32v7.json file.

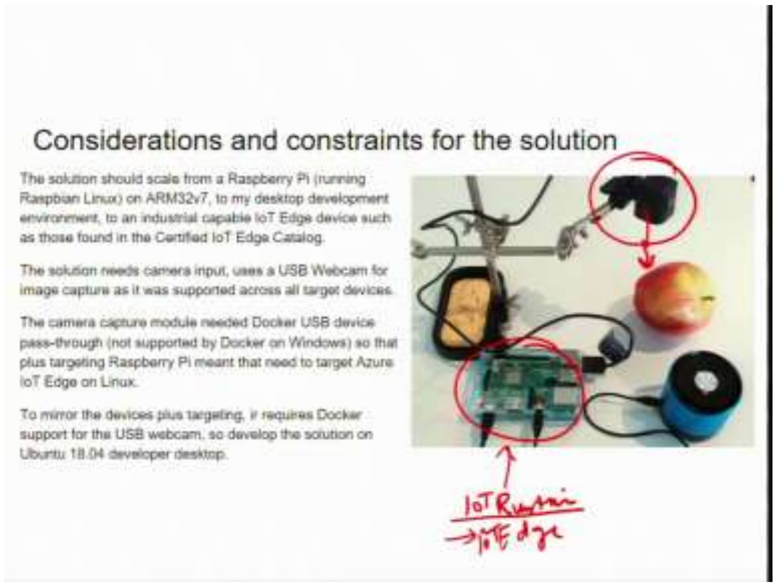
Deploy edge IoT to device: The modules will now start to deploy to your Raspberry Pi, the Raspberry Pi green activity LED will flicker until the deployment completes. Approximately 1.5 GB of Dockers modules will be downloaded and decompressed on the Raspberry Pi. This is a one off operation.

So let us see that once you have Raspberry Pi board then you have to install the Raspberry Pi operating system and for that you have to now do many things like installation so the installation of a Raspberry Pi operating system called Raspbian has to happen to run that particular board then on the cloud side you have to create an Azure services so that is called an Azure IoT Hub on the cloud platform that is offered as a software as a service so this Azure IoT Hub you have to create your details for that devices.

For example you have an IoT camera and Raspberry Pi so these devices need to be registered here in Azure IoT Hub runtime, so once it is done then you have to install on the Raspberry Pi Azure IoT Edge runtime on Raspberry Pi, so you have to download the configurations that is being described and then you have to deploy now then again you have to install the command line interface Azure CLI so that it may accept CLI commands from the console and then you have to deploy this IoT Edge to the device.

So you can see from once it is being prepared this IoT at the IoT Hub it will be containerized and that particular services are now deployed on IoT edge device, so this particular deployment of an IoT Edge runtime, IoT Edge runtime from the Azure IoT Hub is done through a technology which is called the docker so that is what we have already mentioned.

(Refer Slide Time: 48:35)



Considerations and constraints for the solution

The solution should scale from a Raspberry Pi (running Raspbian Linux) on ARM32v7, to my desktop development environment, to an industrial capable IoT Edge device such as those found in the Certified IoT Edge Catalog.

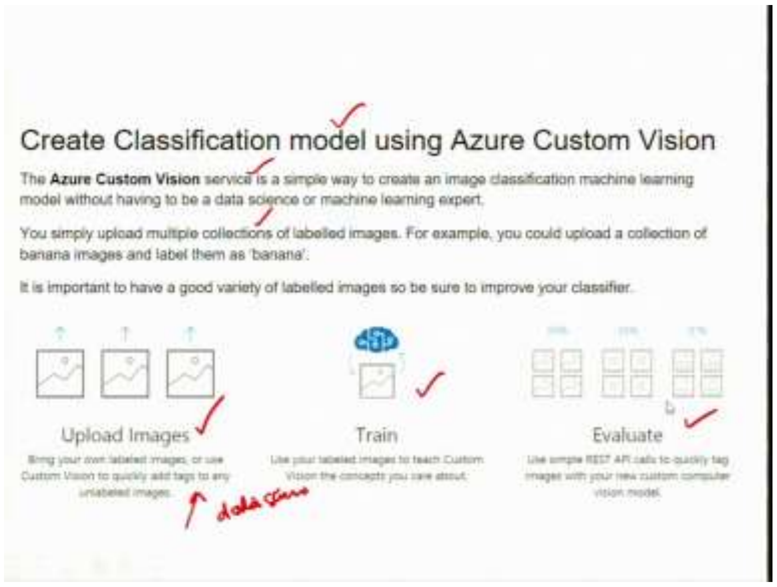
The solution needs camera input, uses a USB Webcam for image capture as it was supported across all target devices.

The camera capture module needed Docker USB device pass-through (not supported by Docker on Windows) so that plus targeting Raspberry Pi meant that need to target Azure IoT Edge on Linux.

To mirror the devices plus targeting, it requires Docker support for the USB webcam, so develop the solution on Ubuntu 18.04 developer desktop.

So let us see that once these requirements are done then let us see the solution how it look like so that solution will scale from Raspberry Pi to the desktop so let us see that this is an IoT camera and here you have the Raspberry Pi board which is running an Azure IoT Hub runtime on the board and so this is an environment which will become like in IoT Edge so this particular IoT Edge now will be installed with the using the docker, the trained machine learning model and this particular camera will take the picture of a fruit and it will be able to detect and display that this is the fruits.

(Refer Slide Time: 49:37)



Create Classification model using Azure Custom Vision

The **Azure Custom Vision** service is a simple way to create an image classification machine learning model without having to be a data science or machine learning expert.

You simply upload multiple collections of labelled images. For example, you could upload a collection of banana images and label them as 'banana'.

It is important to have a good variety of labelled images so be sure to improve your classifier.

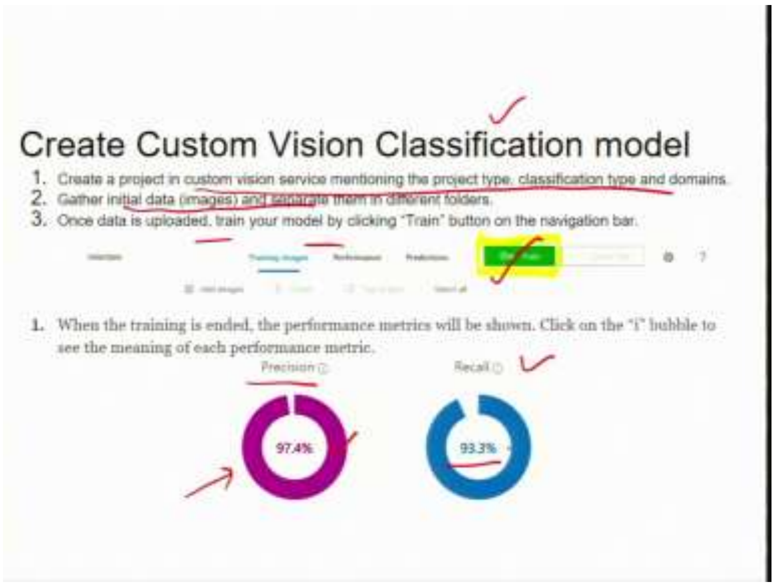
Upload Images ✓
Bring your own labeled images, or use Custom Vision to quickly add tags to any unlabeled images.

Train ✓
Use your labeled images to teach Custom Vision the concepts you care about.

Evaluate ✓
Use simple REST API calls to quickly tag images with your new custom computer vision model.

So let us see that now again come back to our point to understand more about these setup details so we have to create the classification model using the Azure custom vision and Azure custom vision service is simple to create an image classification machine learning model without having to do be, to be the data scientist or a machine learning expert, so what you have to do now you have to become like a data scientists, you have to upload, prepare and upload the labeled images here in the Azure custom vision service and then you can train your model and once the trained model, once the training is done then you have to evaluate the model whether the model is properly trained to do the correct predictions or not. So this particular classifier may be further be improved by adding more data set or a more samples.

(Refer Slide Time: 50:44)



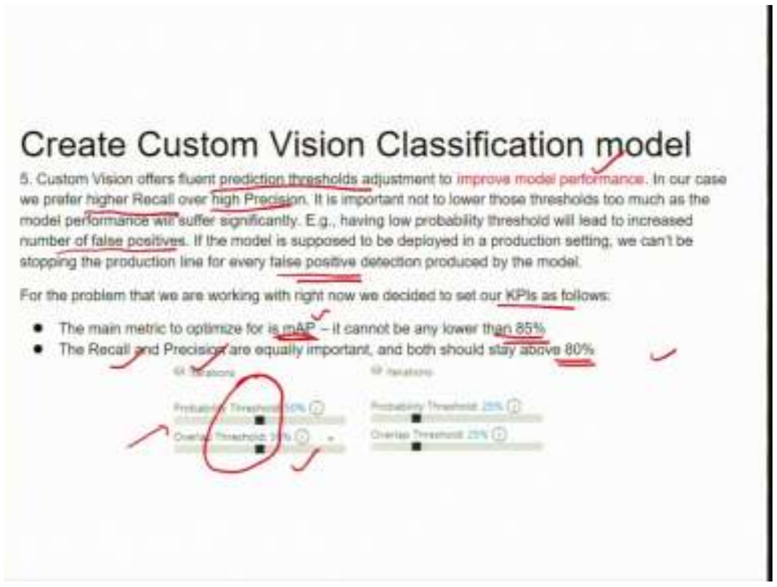
Create Custom Vision Classification model

1. Create a project in custom vision service mentioning the project type, classification type and domains.
2. Gather initial data (images) and separate them in different folders.
3. Once data is uploaded, train your model by clicking "Train" button on the navigation bar.

1. When the training is ended, the performance metrics will be shown. Click on the "i" bubble to see the meaning of each performance metric.

So let us see that how you are going to create this particular model, so you have to create the custom vision service and then you have to you have to gather the initial data or the images and then you have to upload and train your model and here there is a train button once you train the button then the model will be trained and once the training is done then then it will show the performance of performance metric of the trained model. So the precision is 97.4 and the recall is 93.3 both are fairly good.

(Refer Slide Time: 51:22)



Create Custom Vision Classification model

5. Custom Vision offers fluent prediction thresholds adjustment to improve model performance. In our case we prefer higher Recall over high Precision. It is important not to lower those thresholds too much as the model performance will suffer significantly. E.g., having low probability threshold will lead to increased number of false positives. If the model is supposed to be deployed in a production setting, we can't be stopping the production line for every false positive detection produced by the model.

For the problem that we are working with right now we decided to set our KPIs as follows:

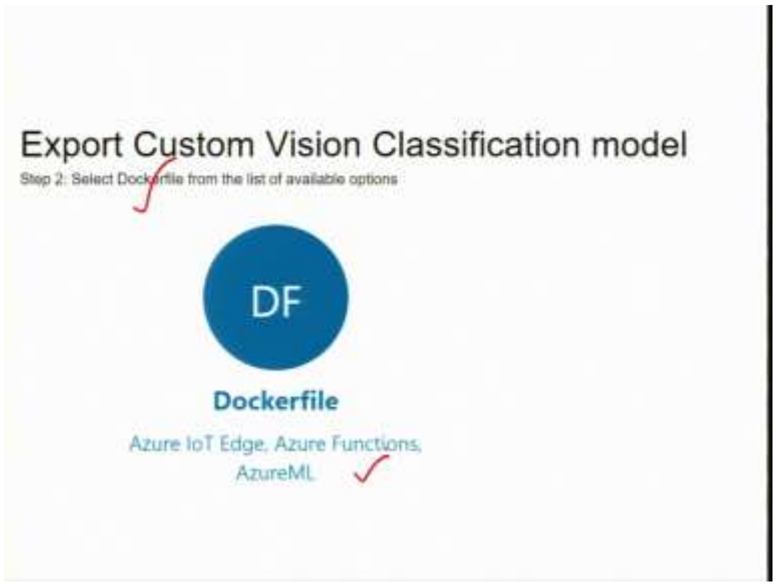
- The main metric to optimize for is mAP – it cannot be any lower than 85%
- The Recall and Precision are equally important, and both should stay above 80%

Now let us see that computer vision prediction thresholds you can set so here this is the example that if your model is not having the accuracy and the precision as per your need then you can adjust, you can train the model by modifying this threshold, let us understand about that so custom vision offer the fluent prediction threshold adjustment to improve the model performance.

So in this case we prefer the higher recall over the higher precision, this is important not to lower those threshold too much as the model performance will suffer significantly, having the low probability threshold will increase the number of false positives, so if the model is supposed to be deployed in a production settings we cannot stop the production line for every false positive detection by the model therefore the key performance indicators or index KPIs will be as follows and will be tunable to improve the model performance.

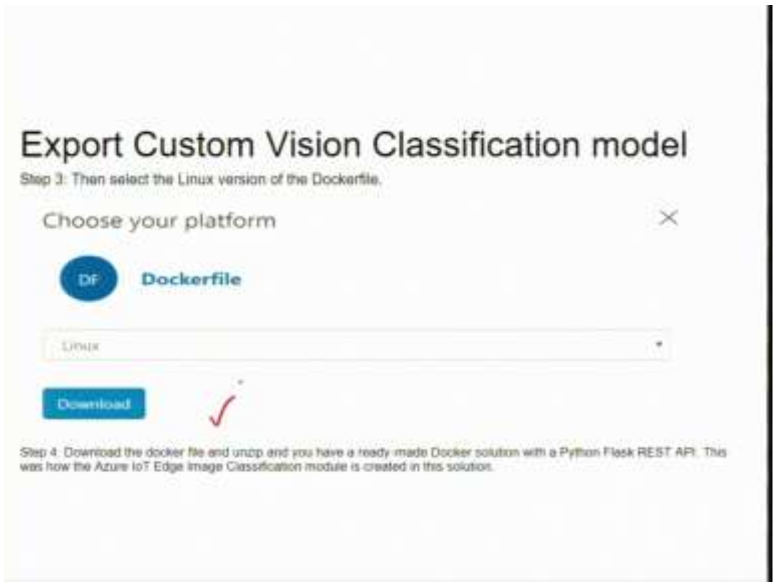
So the main metric to optimize is the mAP so this particular average of all that precision and recall is mAP metric so this particular metric cannot be lower than 85 percent let us say that if we optimize the mean metric so this means that recall and precision are also equally important and they should stay above 80 percent, if that is to be enforced in this particular threshold then your model will be giving a good accuracy and this is what is shown over here.

(Refer Slide Time: 53:16)



Export Custom Vision Classification model

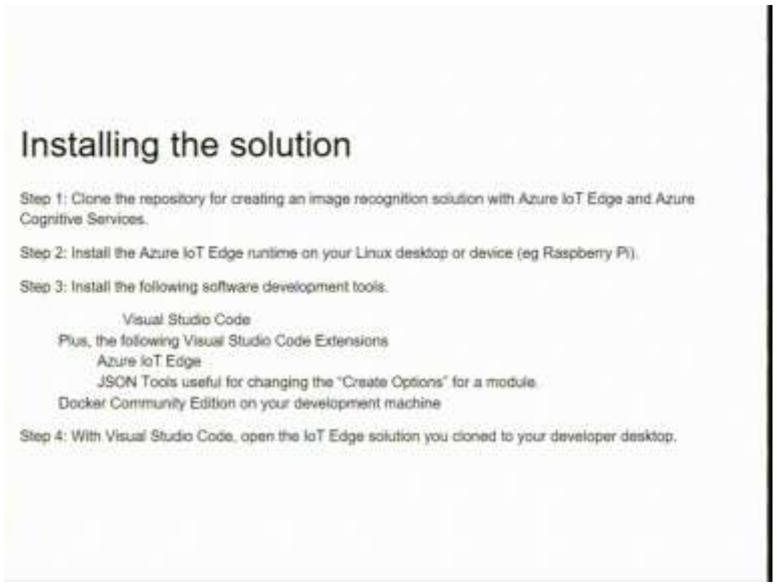
Step 2: Select Dockerfile from the list of available options



Export Custom Vision Classification model

Step 3: Then select the Linux version of the Dockerfile.

Step 4: Download the docker file and unzip and you have a ready made Docker solution with a Python Flask REST API. This was how the Azure IoT Edge Image Classification module is created in this solution.



Installing the solution

Step 1: Clone the repository for creating an image recognition solution with Azure IoT Edge and Azure Cognitive Services.

Step 2: Install the Azure IoT Edge runtime on your Linux desktop or device (eg Raspberry Pi).

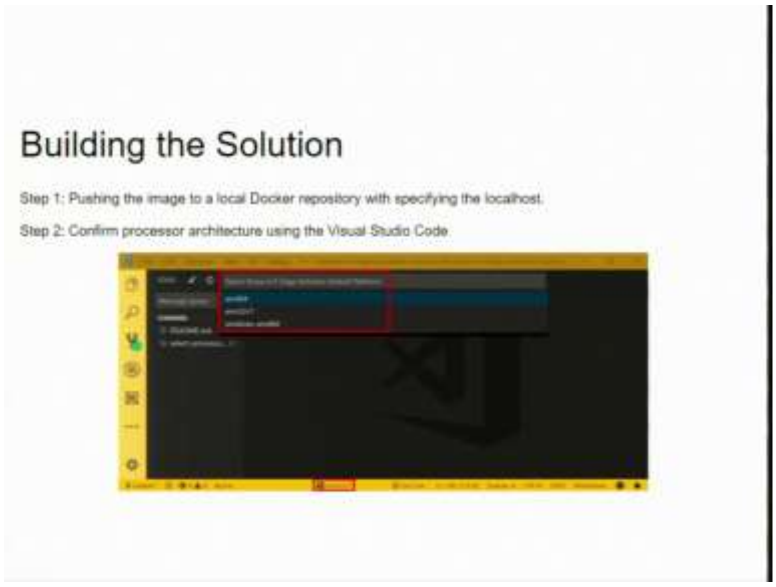
Step 3: Install the following software development tools.

Visual Studio Code
Plus, the following Visual Studio Code Extensions
Azure IoT Edge
JSON Tools useful for changing the "Create Options" for a module
Docker Community Edition on your development machine

Step 4: With Visual Studio Code, open the IoT Edge solution you cloned to your developer desktop.

So once these models are trained and trained up to the satisfying performance using this KPIs then this particular classification model classifier is now ready and you can now deploy with the help of Docker so you have to now containerize this trained model on the cloud and then deploy it on this particular edge, so that is what is the next step so once it is containerized then you have to download and it will be shipped or deployed to the IoT edge which is nothing but the Raspberry Pi board which is being built or registered with the Azure IoT hub.

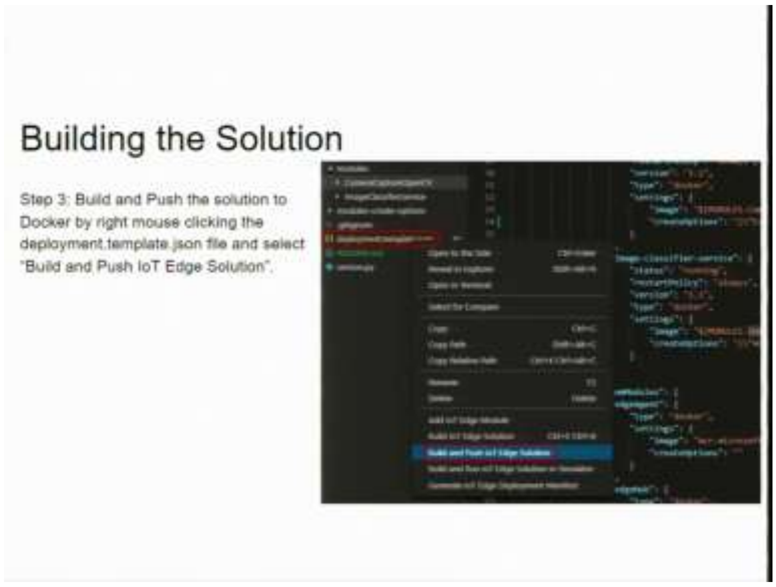
(Refer Slide Time: 54:09)



Building the Solution

Step 1: Pushing the image to a local Docker repository with specifying the localhost.

Step 2: Confirm processor architecture using the Visual Studio Code

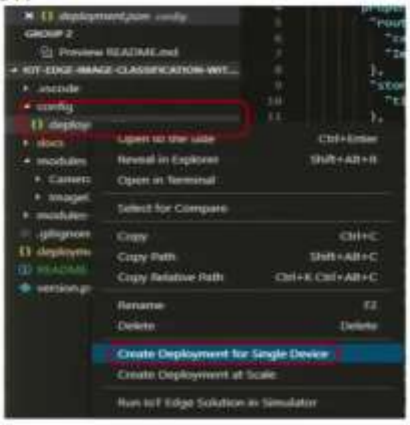


Building the Solution

Step 3: Build and Push the solution to Docker by right mouse clicking the deployment.template.json file and select "Build and Push IoT Edge Solution".

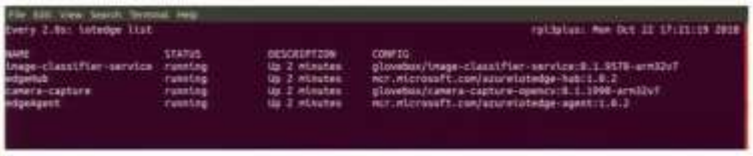
Deploying the Solution

When the Docker Build and Push process has completed select the Azure IoT Hub device you want to deploy the solution to. Right mouse click the deployment.json file found in the config folder and select the target device from the drop-down list.


deployment.json config
GROUP 2
Preview README.md
IOT-EDGE-IMAGE-CLASSIFICATION-WIT...
.vscode
config
deploy
docs
modules
Camera
ImageC
modules
.gitignore
deployment
README
version.py
Open to the side Ctrl+Enter
Reveal in Explorer Shift+Alt+R
Open in Terminal
Select for Compare
Copy Ctrl+C
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Alt+C
Rename F2
Delete Delete
Create Deployment for Single Device
Create Deployment at Scale
Run IoT Edge Solution in Simulator

Monitoring the Solution on the IoT Edge Device

Once the solution has been deployed you can monitor it on the IoT Edge device itself using the *iotedge list* command.



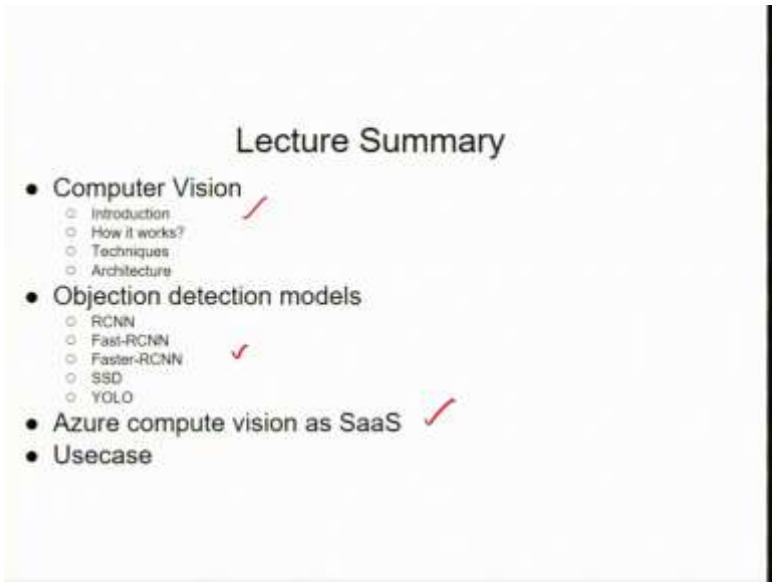
```
Every 2.0s: iotedge list                                          rpi3plus: Mon Oct 22 17:21:19 2018

NAME                     STATUS     DESCRIPTION    CONFIG
image-classifier-service running    Up 2 minutes   glovebox/image-classifier-service:0.1.9570-arm32v7
edgeHub                  running    Up 2 minutes   mcr.microsoft.com/azureiotedge-hub:1.0.2
camera-capture           running    Up 2 minutes   glovebox/camera-capture-opencv:0.1.1990-arm32v7
edgeAgent                running    Up 2 minutes   mcr.microsoft.com/azureiotedge-agent:1.0.2
```

Monitoring the Solution on the IoT Edge Device

Click on the device from the Azure IoT Edge blade to view more details about the modules running on the device.

So this is some of the more detail that you have to now build the solution and you have to deploy using Docker and container you have to push it on the IoT Hub or an edge IoT, IoT Edge and then you can also monitor it its functionality of that IoT edge, so that was a brief of the discussion on the practical side also.

(Refer Slide Time: 54:39)



Lecture Summary

- Computer Vision ✓
  - Introduction
  - How it works?
  - Techniques
  - Architecture
- Objection detection models
  - RCNN
  - Fast-RCNN
  - Faster-RCNN ✓
  - SSD
  - YOLO
- Azure compute vision as SaaS ✓
- Usecase

So let us summarize this lecture what we have understood in this lecture, we have understood about the computer vision fundamentals that computer vision needs some of the machine learning models they are called image classifier to build the image classifier, so different machine learning algorithm or the models which we have discussed is RCNN based models and YOLO based models.

Then we have seen about the Azure computer vision service that is the software as a service, we have also told that all other services provided by different cloud vendors are similar in nature so this particular use case taking from Azure is good enough to understand the functionality of every other solution provided by different vendors such as Amazon and Google, we have also covered the use case and the practical lab session here in this particular lecture. Thank you.