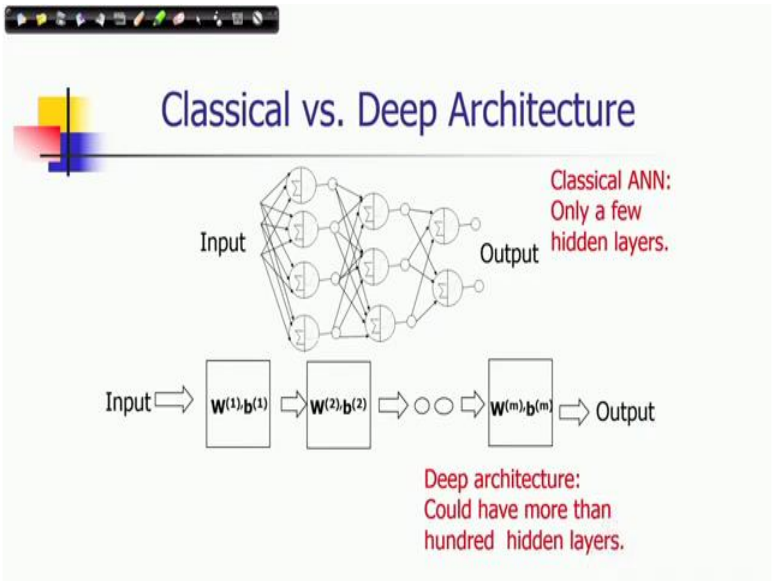
Computer Vision

Prof. Jayanta Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 55
Deep Neural Architecture and Applications Part – I

We will discuss a new topic today and that is on Deep Neural Architecture and Applications.

(Refer Slide Time: 00:21)



Classical vs. Deep Architecture

Classical ANN: Only a few hidden layers.

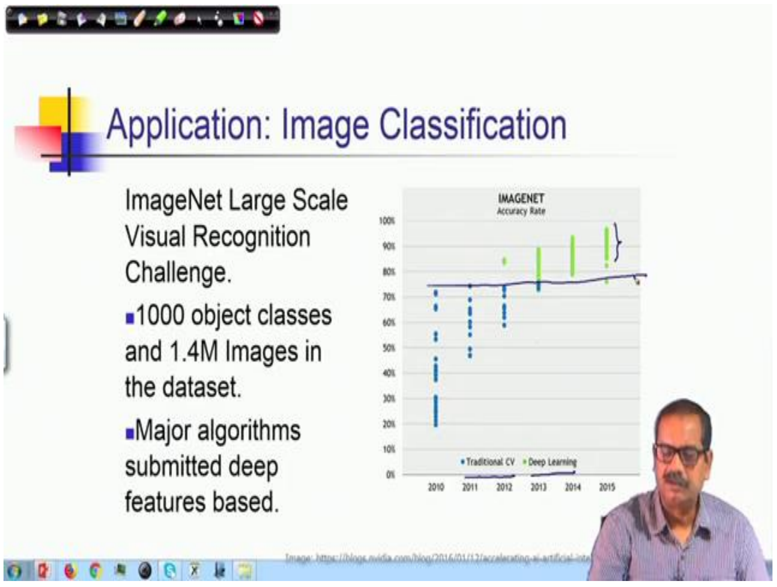
Deep architecture: Could have more than hundred hidden layers.

So, let us understand what is the difference between a classical neural architecture and a deep neural architecture. We have already discussed about the artificial neural network where you can see that it is a network of different neural nodes or perceptual nodes where you have the input at one end.

And it is the feed forward network, each node takes the weighted input vector and then the net sum of those weighted vectors are passed through a non-linear functions and the output is the response of that non-linear functions. And network of all these nodes that gives you the artificial neural network. And finally, there is an input and output layers of this particular architecture.

Now, the fact is that so far for classical artificial neural networks we had only a few hidden layers that is what we are used to do maximum three layers that we could see in many of the applications, but 1 or 2 layers are more common. Whereas, for deep neural architecture you have more number of layers, you can have more than 100 hidden layers and that is one of the major difference.

(Refer Slide Time: 01:41)



Application: Image Classification

ImageNet Large Scale Visual Recognition Challenge.

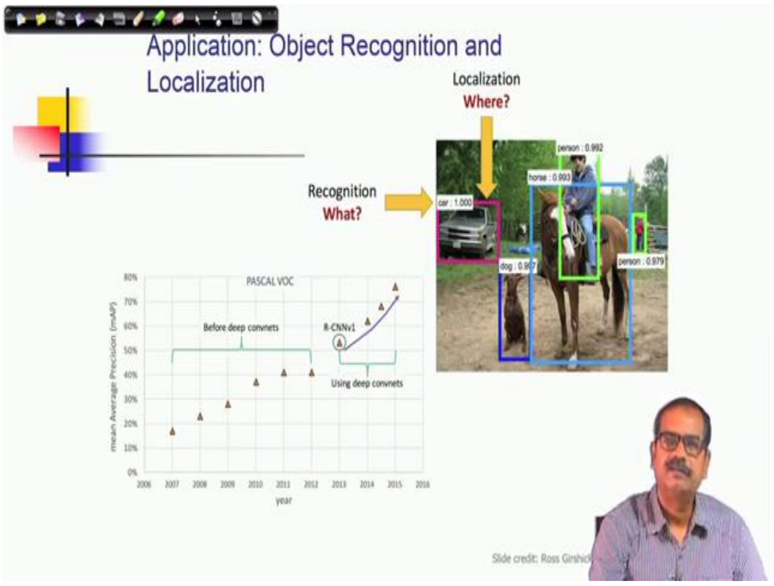
- 1000 object classes and 1.4M Images in the dataset.
- Major algorithms submitted deep features based.

So, there are different applications of this deep neural architecture. So, before describing the features of this architecture let me give a brief overview what are the applications which are reported by this kind of architecture. So, one of the applications is an image classification and there is a data set image net large scale visual recognition challenge data set which was there for improving the classification algorithms for testing. And for a long time there are researchers they are working on these data sets and they are reporting their performance on this data set.

So, in this data set you have 1000 object classes and there are about 1.4 million images the design in the data set. And major algorithms which have been found to provide good performances they are deep features based which means they have used deep neural architecture while classifying this data.

So, here is a brief summary that what are the performances of different algorithms. As you can see that you have these traditional competition algorithms, so this is a traditional algorithm, and this is a deep learning algorithms. So, all the performance indexes of these deep learning algorithms they are very high where the traditional algorithms and as we are moving ahead with time. So, it is still 2050 it was shown after that also the performance has been improved. So, the strain is increasing and you are getting better and better deep architecture for giving this kind of classification performance.

(Refer Slide Time: 03:35)



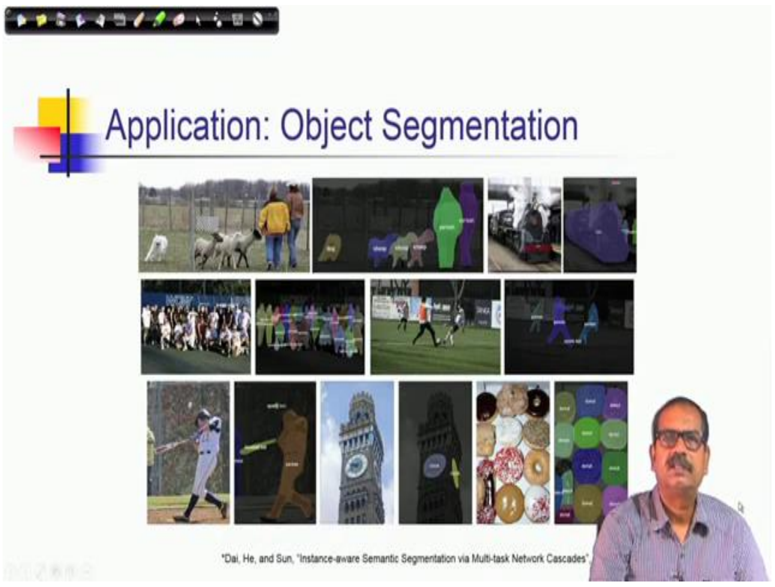
Application: Object Recognition and Localization

Slide credit: Ross Girshick

Another kind of application that we can discuss that is object recognition and localization. So, this is one example how this problem has been solved what is the kind of response. You can see that in this scene the system tries to localize an object for example, it has been shown car and there is a pink box rectangular box where the car has been shown so, this is the localization. And also it recognizes that what objects it is the car with certain confidence level or some probability value.

So, it is definitely car because it is 1.00 whereas, you can see that horse is 0.993, so there is some confusion with horse like this. And this kind of there is also another data set where these recognition and localization tasks are performed. And you can see that using deep conclusion neural networks or we will discuss what is a conclusion neural network in this under this topic, so they have provided they have given better performance in this case.

(Refer Slide Time: 04:51)

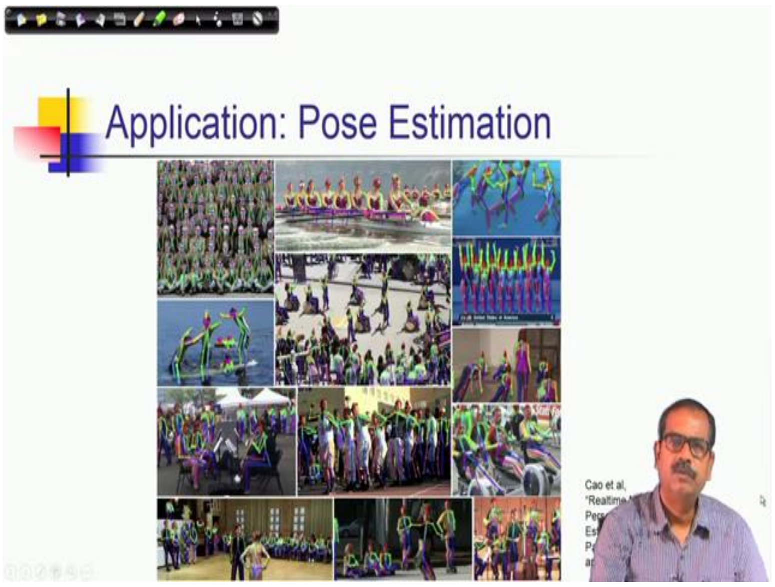


Application: Object Segmentation

"Dai, He, and Sun, "Instance-aware Semantic Segmentation via Multi-task Network Cascades"

Object segmentation is another task in fact, the segmentation with respect to deep neural architecture it is called semantic segmentation. Because you are not only segmenting the objects you are also telling that what is that object, it is like localization almost similar thing. But, now you are doing pixel level declarations that is a segment of which type of object. So, this is one kind of application,

(Refer Slide Time: 05:19)



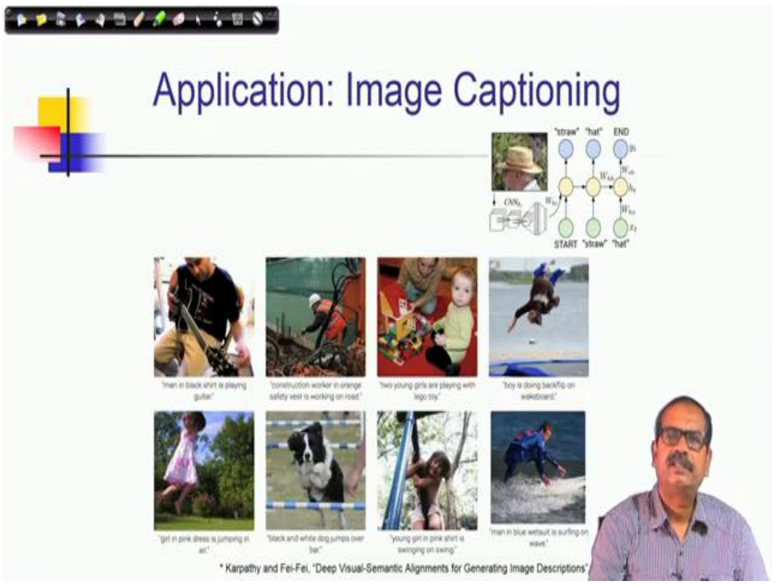
Application: Pose Estimation

Cao et al, "Realtime

Pose estimation is another, you can see in this diagram that poses of different persons are shown by a skeleton figure. So, skeletons are 3-dimensional information, 3 dimensional it is a description of a human body, human limbs with the graphical descriptions with 3D coordinate vertices. And connected by their edges and with our joints at different limbs those are representing our human skeleton.

And depending upon the image of a different person, we try to guess at what kind of skeleton configurations could provide that kind of stance or that kind of pose of a person. So, this is estimation and that gives you idea that in what way the person is standing or doing some activity.

(Refer Slide Time: 06:17)



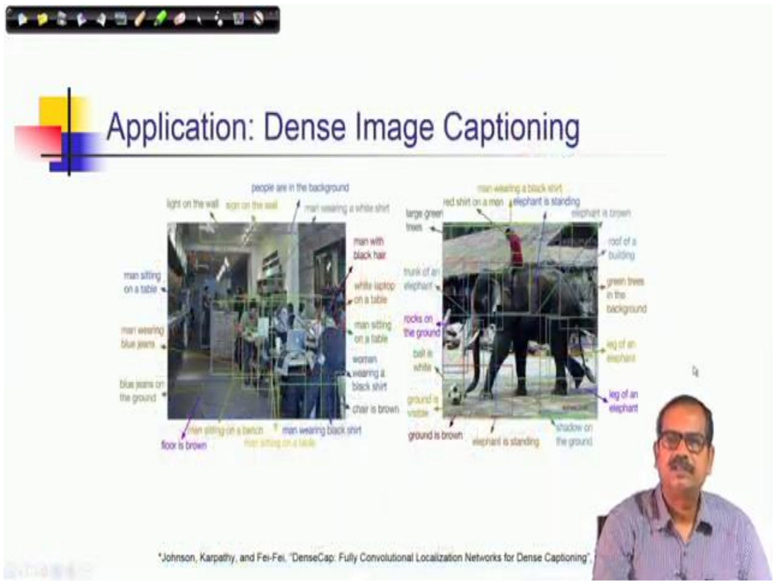
Application: Image Captioning

"straw" "hat" END
START "straw" "hat"
"man in black shirt is playing guitar."
"construction worker in orange safety vest is working on road."
"two young girls are playing with lego toy."
"boy is doing backflip on wakeboard."
"girl in pink dress is jumping in air."
"black and white dog jumps over bar."
"young girl in pink shirt is swinging on swing."
"man in blue wetsuit is surfing on wave."

* Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions"

Another interesting application is image captioning it is a cross disciplinary problem as you can see that not only image, understanding of images and videos we are trying to describe using natural languages. So, you can see that with respect to different scenes there is some language, some description like you consider this particular image where a child where a girl is jumping. So, it is written here girl in pink dress is jumping in air, so this kind of descriptions those are generated by this kind of system.

(Refer Slide Time: 06:55)

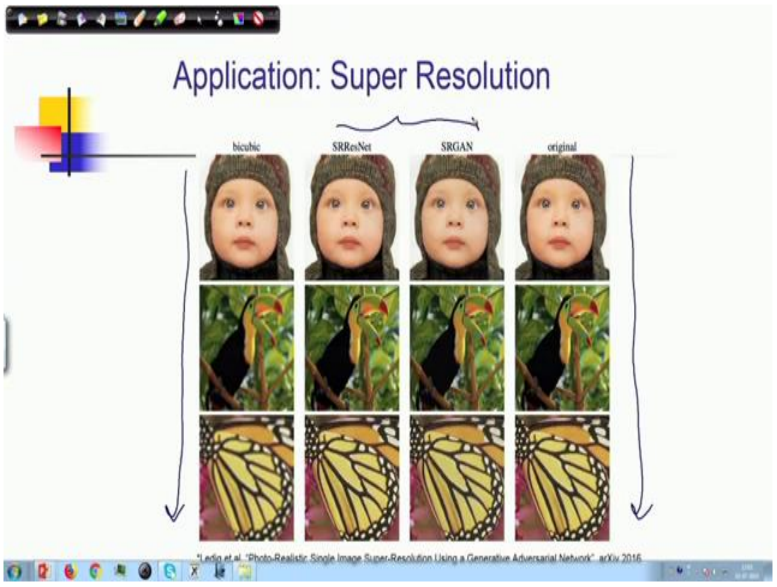


Application: Dense Image Captioning

*Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning",

Similarly, dense image captioning, so it is further complex. So, you would like to have not only you are describing the main motto of the same. But also you are describing every part of the image that is a elephant task of an elephant it is a football, there is a man sitting on top of an elephant like that the man is wearing a red shirt. So, this kind of description and this kind of captioning could be there using this kind of architecture.

(Refer Slide Time: 07:31)

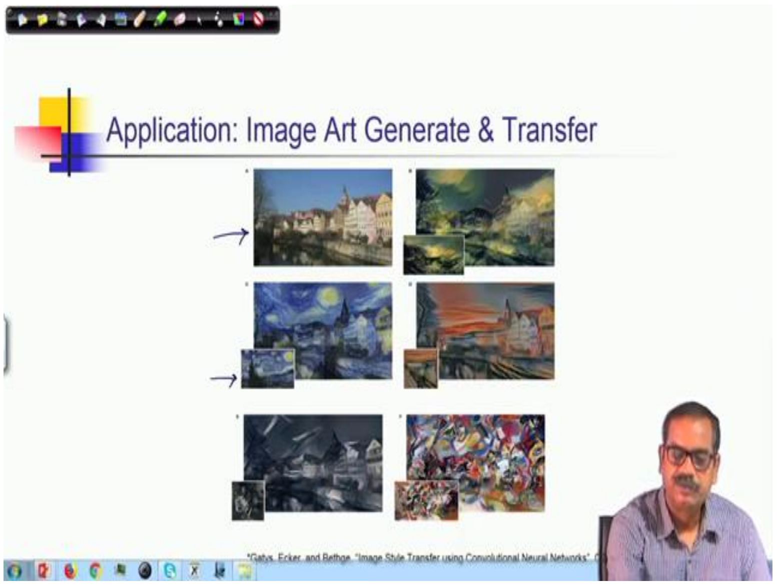


Application: Super Resolution

*Ledig et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", arXiv 2016

Another application is super resolution, so here you can see that examples are given the original image is shown in the right most column. So, this is the original image and this is the classical algorithm by cubic interpolation algorithm which produces the super resolutions or interpolated images. Whereas, using deep neural architecture you have this middle two columns those are the results and you can choose you can actually see the visual quality of these two interpolated images or super reserved images they are much better than what we have obtained by cubic interpolation.

(Refer Slide Time: 08:15)



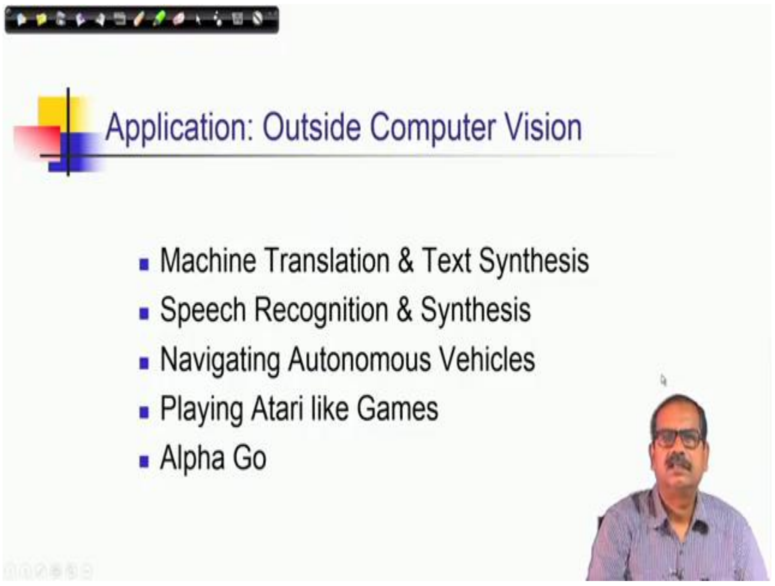
Application: Image Art Generate & Transfer

*Gatys, Ecker, and Bethge, "Image Style Transfer using Convolutional Neural Networks"

Image art generation and transfer this is also another very interesting applications. In fact, using deep neural architecture you can synthesize images, you can render images and here in particular you can see that there is a photograph. So, this is the original photograph which is taken by a camera and you consider the style of painter say this is a Hancock painting and I think this is starry night or that is a name.

And using Hancock painting, so this photograph is with that style itself painting style this photograph has been generated. So, all these buildings etcetera you can observe here, but they have very high similarity with this style with which it has been painted. And similarly you can get the examples that same scene is rendered by different styles of painters.

(Refer Slide Time: 09:11)

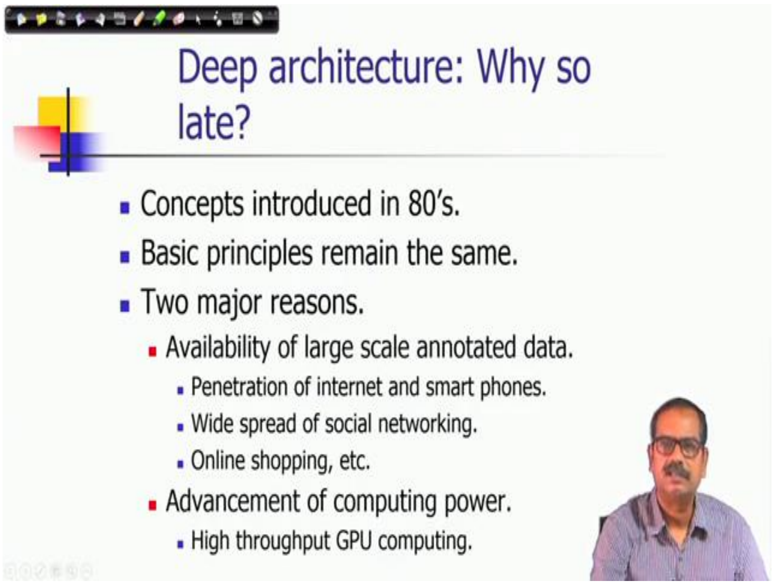


Application: Outside Computer Vision

- Machine Translation & Text Synthesis
- Speech Recognition & Synthesis
- Navigating Autonomous Vehicles
- Playing Atari like Games
- Alpha Go

Many other applications are there not only in computer vision deep learning deep neural architecture is used for in other areas. So, machine translation and text synthesis, speech recognition and synthesis, navigating autonomous vehicles, playing different kinds of games, even playing chess alpha go these are things.

(Refer Slide Time: 09:31)



Deep architecture: Why so late?

- Concepts introduced in 80's.
- Basic principles remain the same.
- Two major reasons.
  - Availability of large scale annotated data.
    - Penetration of internet and smart phones.
    - Wide spread of social networking.
    - Online shopping, etc.
  - Advancement of computing power.
    - High throughput GPU computing.

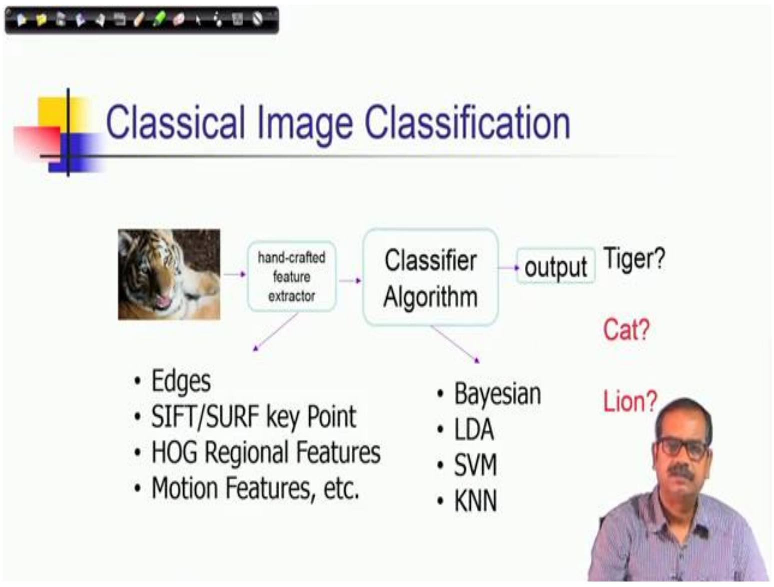
So, one of the very one of the questions will come will naturally this question will arise to us that why this architecture why when it is giving. So, higher performance compared to classical techniques, but why did it take so much of time that to come to the stage. Because the concepts what you are using in deep neural competitions they are very old they are the same neural network competition, they are the same neural network model. So, they are introduced in 80's and they are basic principles they remain the same.

So, there are two major reasons that is why it has arrived today it is an historical reason; that means, the progress of technology and science that is what led to every kind of new technological advancement. So, it is a new technological advancement because, we have; now we are have the capacity of generating large scale annotated data and because of that availability of the data.

So, you can train large networks to develop any kind of model which can work in a very general, to solve any general and difficult problems. And then the advancement of computing power because this training also requires high computations and it has to be done within a feasible time. So, because, so there are say there are gpu computing which has come with a you know which has been which has given this advancement and which has been very much used for these kinds of competitions.

So, availability of large scale annotated data is possible because of penetration of internet and smartphones, widespread of social networking, online shopping etcetera and advancement of computing power as I mentioned high throughput GPU computing like this is one such example.

(Refer Slide Time: 11:35)


Classical Image Classification

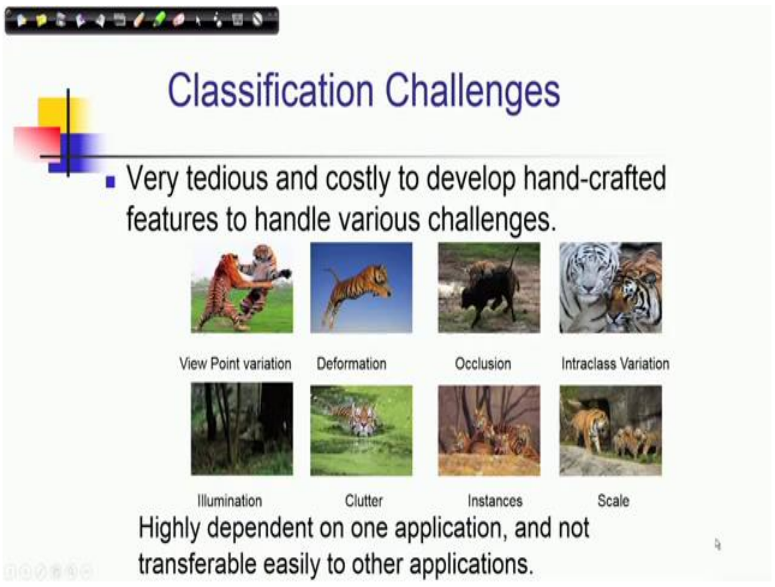
- Edges
- SIFT/SURF key Point
- HOG Regional Features
- Motion Features, etc.

- Bayesian
- LDA
- SVM
- KNN

Now, let us understand that what is the difference between classical image classification techniques and the deep features based classification techniques or deep neural competition based classification techniques. In classical image classification you consider you have an image and then you would like to represent that this image by some handcrafted features. Which means, we have studied what could be the distinct features are characterized in this image and we developed computational algorithms to extract those features that is what is called handcrafted features. So, it is designed by as the algorithm is also designed accordingly and then we extract them this features.

For example edges, SIFT, SURF key points, HOG hog regional features, motion features we have discussed some of these features in my previous lectures. And then this feature representation is input to our classification algorithm, so use the classifier. And there are different classifier like bayesian classifier linear discriminate analysis, support vector machine, KNN algorithms. So, different such you know classifier could be there and then we produce the output. Output is something like whether it is tiger, cat, lion it is a standard classification problem, so this is the classical approach.

(Refer Slide Time: 13:07)


Classification Challenges

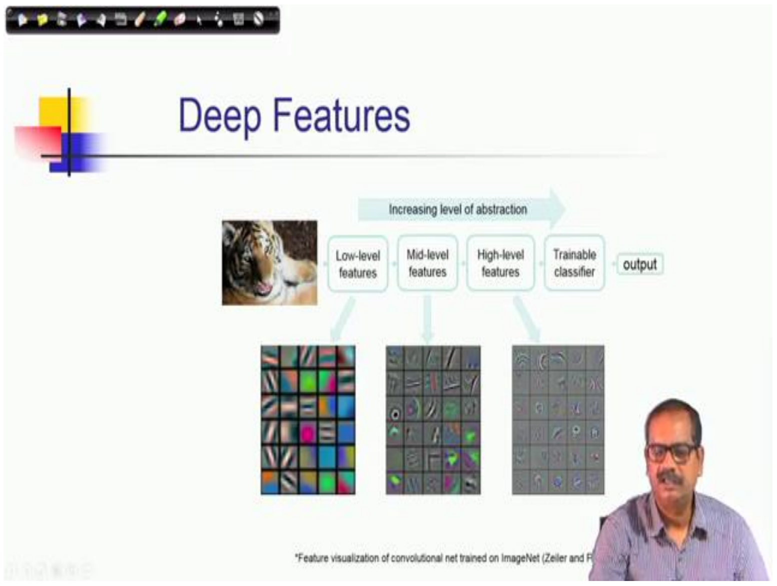
- Very tedious and costly to develop hand-crafted features to handle various challenges.

View Point variation
Deformation
Occlusion
Intraclass Variation
Illumination
Clutter
Instances
Scale

Highly dependent on one application, and not transferable easily to other applications.

Whereas, so the problem here is that it is very tedious and costly to develop handcrafted features to handle various challenges, because there is a large variance of input image of the same object. So, these variance causes due to different kind of variations, you have viewpoint variation deformation, occlusion, interclass variation, illumination, clutter, instances, scale; some nice examples are shown here with respect to that tiger. Say, you can see one by one that what kind of variations are there and, so that is why this is very difficult to develop a very general algorithm, it is dependent on one application and it is not easily transferable to other applications.

(Refer Slide Time: 13:59)



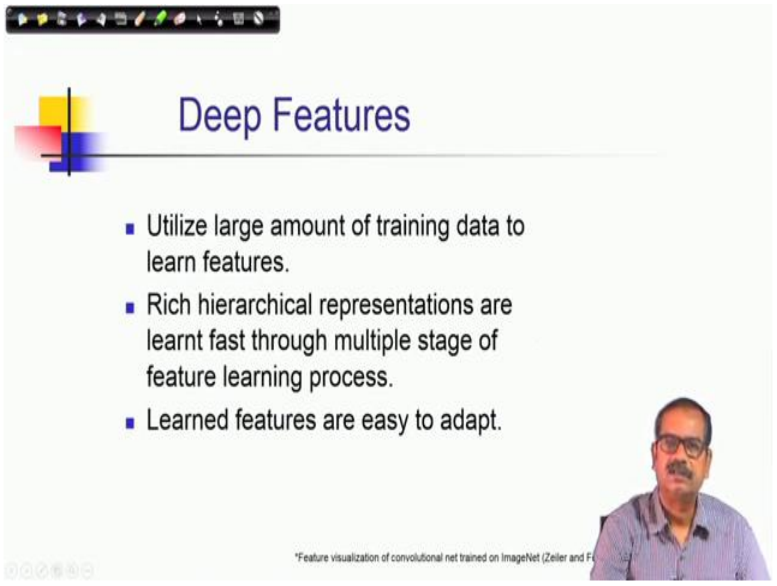
Deep Features
Increasing level of abstraction
Low-level features
Mid-level features
High-level features
Trainable classifier
output
*Feature visualization of convolutional net trained on ImageNet (Zeiler and F

In deep feature based algorithm we learn the feature representation first using the it is a automated learning it is unsupervised learning. So, your mechanism is such that they extract the features depending upon the you know problems statement depending upon the applications.

So, here usually it has been shown we will discuss that part that how you are learning even the feature extractions that is not handcrafted, your neural network itself is learning to represent the features. So, there are different levels of features representations when you are doing deep feature representation.

So, youwill have low level features and as you increase the level of abstraction you can have mid level features, you can have high level features and then you train the classifier using this features. The whole thing you are doing in one shot; that means, it is augmented it is coupled in the previous case classification scheme is independent of feature representation scheme. But now feature representation and classification they go hand in hand they go in one shot and then you get the output.

(Refer Slide Time: 15:23)



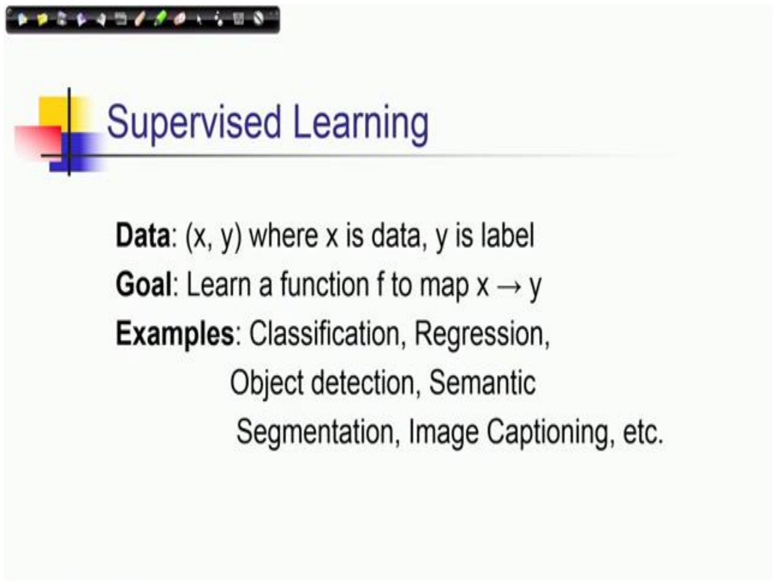
Deep Features

- Utilize large amount of training data to learn features.
- Rich hierarchical representations are learnt fast through multiple stage of feature learning process.
- Learned features are easy to adapt.

*Feature visualization of convolutional net trained on ImageNet (Zeiler and F

So, it utilizes large amount of training data to learn features and it has reached hierarchical representations as I have mentioned that from low level, mid level to high level feature representation. So, multiple stages of feature learning processes involved and it learns features adoptively, so in the context of your application this features are learnt.

(Refer Slide Time: 15:47)



Supervised Learning

Data: (x, y) where x is data, y is label

Goal: Learn a function f to map x → y

Examples: Classification, Regression, Object detection, Semantic Segmentation, Image Captioning, etc.

Mostly we will be considering supervised learning and in the deep neural architecture, particularly classification problem as we have discussed that you have the training samples, you are training the network and then you are classifying. So, this is then ideal supervised classification setup and we call it also supervised learning.

So, the problem here is that you have x data and y label, so I am just revisiting this we have discussed also in the context of classification previously. So, x is the data, y is the label and our goal is to learn a function f which will map the data to the label. And different kinds of problems like classification, and regression, object detection, semantic, segmentation, image captioning all could be mapped to this kind of learning framework.

(Refer Slide Time: 16:39)

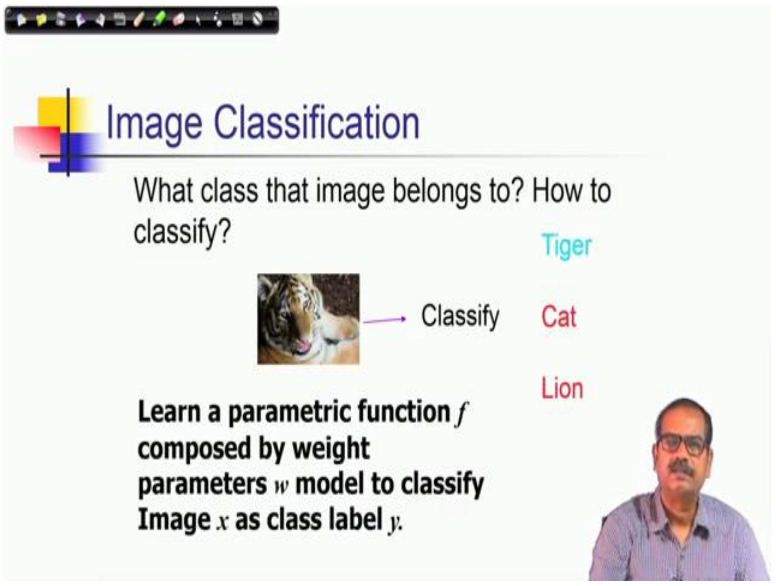


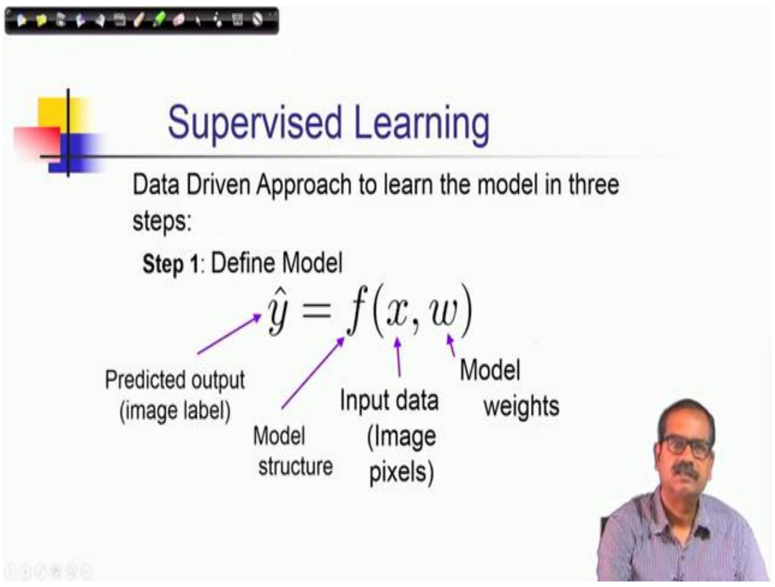
Image Classification

What class that image belongs to? How to classify?

Learn a parametric function f composed by weight parameters w model to classify Image x as class label y.

So, let us consider image classification and in the image classification the problem is at what class that image belongs to and how to classify, we have already understood this problem statement. So, in this case we would learn we would like to learn a parameter function f which composed by weight parameters of the model to classify image x as class label y.

(Refer Slide Time: 17:03)



Supervised Learning

Data Driven Approach to learn the model in three steps:

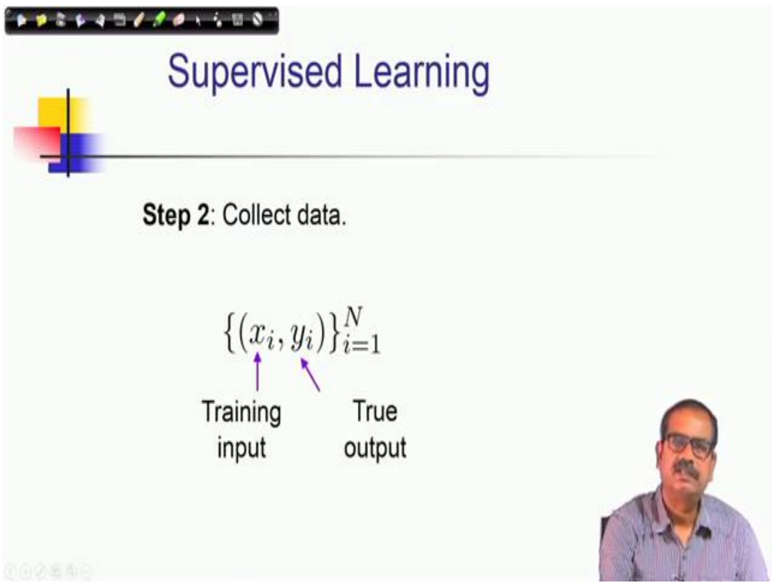
Step 1: Define Model

$$\hat{y} = f(x, w)$$

Predicted output (image label)
Model structure
Input data (Image pixels)
Model weights

So, it is a data driven approach to learn the model in three steps; first you have to define the model like we have defined the prediction of class y given the input and the model w. So, artificial neural network we discussed that is one example of model and as I mentioned that deep neural architecture is nothing but the same artificial neural network model only you have more number of hidden layers. So, you have input data as image pixels here and you have the model weights and this is the model structure, and you have to this is a predicted output or image label.

(Refer Slide Time: 17:43)



Supervised Learning

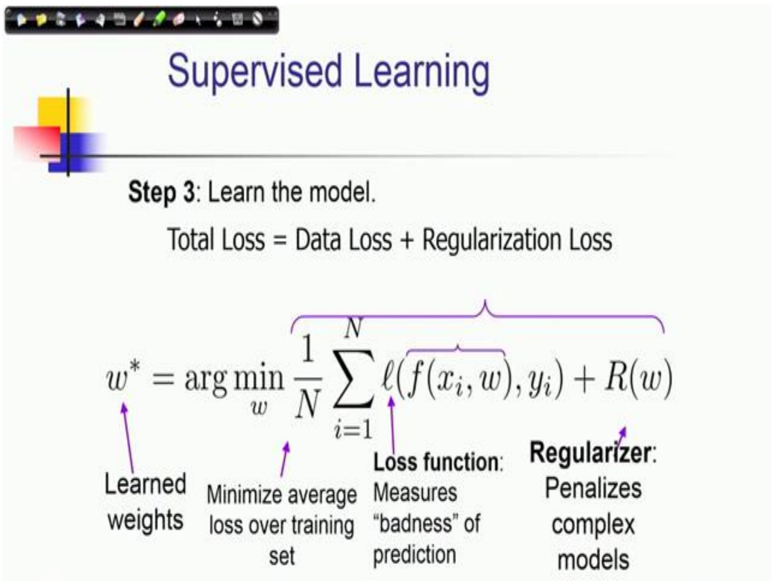
Step 2: Collect data.

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Training input
True output

So, first step is defined model next, step is you have to collect data. So, this is for this supervised learning it is important that you know you should have annotated data, xi is instance, and yi is the label of the data. So, this is the training input and this is the true output yi and then you learn the model.

(Refer Slide Time: 18:03)



Supervised Learning

Step 3: Learn the model.

Total Loss = Data Loss + Regularization Loss

$$\overbrace{w^* = \arg\min_w \frac{1}{N}\sum_{i=1}^{N} \ell(\overbrace{f(x_i, w)}, y_i) + R(w)}$$

Learned weights

Minimize average loss over training set

Loss function: Measures "badness" of prediction

Regularizer: Penalizes complex models

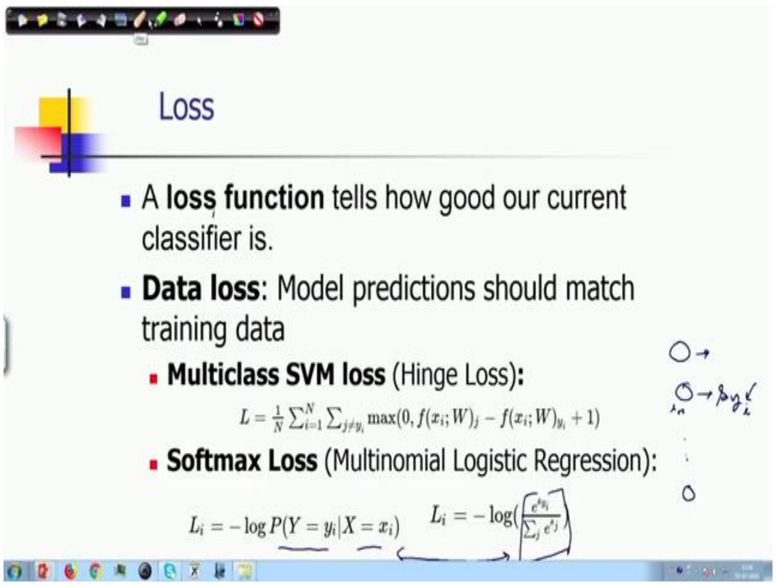
So, here we consider that that there is some mapping function that you have learn and how good is that mapping function and what is the gap then we try to update that gap, so in terms of weights. So, whatever weights are there what is the performance? If the loss is more then we need to update, so that loss should be minimized or loss should be reduced.

So, we use a loss function here to just the to assess the performance of a model, so which is represented here by these notation. So, this is the loss function what is the predicted output, and what is the ground truth depending upon this loss function is defined. And for a set of training samples we accumulate this loss functions and this is the regularization term, because your model should not be too complex we will discuss this part also.

So, let us see, so this is predicted output and this is a loss function and we would like to minimize the average loss over training set and this is called regularizer or panel, which penalizes the complex models and this is learned weights. So, total loss can be considered as the data loss and the regularization loss. So, data loss is given by the functions l here that is the data loss and regularization loss is given by R(w).

$$w^* = \arg\min \frac{1}{N} \sum_{i=1}^{N} l(f(x_i, w), y_i) + R(w)$$

(Refer Slide Time: 19:39)



Loss

- A **loss function** tells how good our current classifier is.
- **Data loss**: Model predictions should match training data
  - **Multiclass SVM loss** (Hinge Loss):

    $$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max(0, f(x_i;W)_j - f(x_i;W)_{y_i} + 1)$$
  - **Softmax Loss** (Multinomial Logistic Regression):

    $$L_i = -\log P(\underline{Y = y_i}|\underline{X} = x_i) \qquad L_i = -\log(\boxed{\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}})$$

So, a loss function tells how good is the current classifier and data loss is the model prediction should match training data, so what is the deviation. So, there are different kinds of examples of data loss functions like in multiclass support vector machine we can use hinge loss. Here, now we can see that each wrongly classified sample produces a loss. So it is the difference of the functional values of the predicted sample predicted function functional value predicted sample and the models sample.

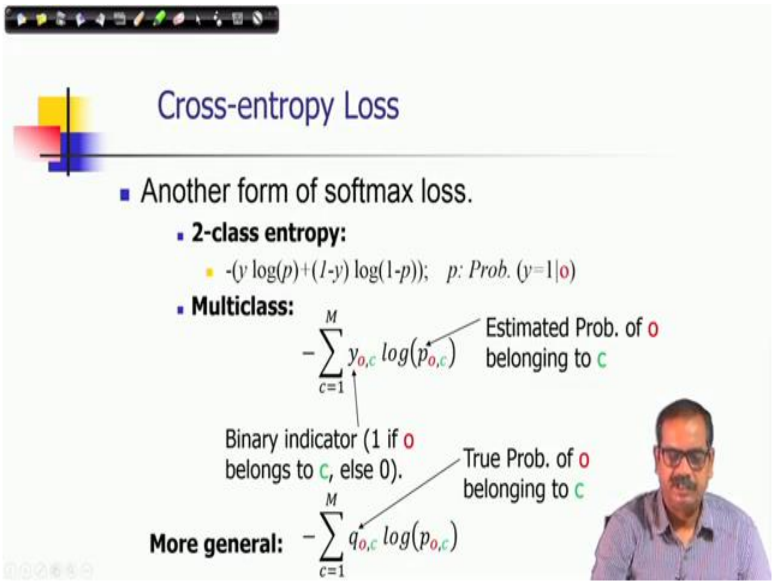
So, you can see that functional form here in this case and the softmax loss or multinomial logistic regression laws that is given in this form. So, in this function we are estimating the probability of a level true level given the input data. So, if this probability is very high then this loss will be less, so using this negation, negative sample we make it a positive function.

Sometimes this probability is computed with respect to the responses of all other classes. So, in the output layer see if you have in the output layer say N classes and the response is provided by say syi of the ith class. Then we consider that see yi is a true class, so we consider the syi of the true response of the true class.

Now, the probability of that class is given by this expression this is the softmax expression you raise it to the exponentiation and somehow normalize it by the sum of all these exponentiations. So, that is that would give you the probability value and it is actually modelling this particular term itself it is giving this probability of y given data this is (Refer Time: 21:55).

$$L_i = -\log P(Y = y_i | X = x_i) \rightarrow L_i = -\log(\frac{e^{y_i}}{\sum e^{y_i}})$$

(Refer Slide Time: 21:57)



Cross-entropy Loss

- Another form of softmax loss.
  - **2-class entropy:**
    - $-(y\log(p)+(1-y)\log(1-p))$; p: *Prob.* $(y=1|o)$
  - **Multiclass:**

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Estimated Prob. of o belonging to c
Binary indicator (1 if o belongs to c, else 0).
True Prob. of o belonging to c

More general: $-\sum_{c=1}^{M} q_{o,c} \log(p_{o,c})$

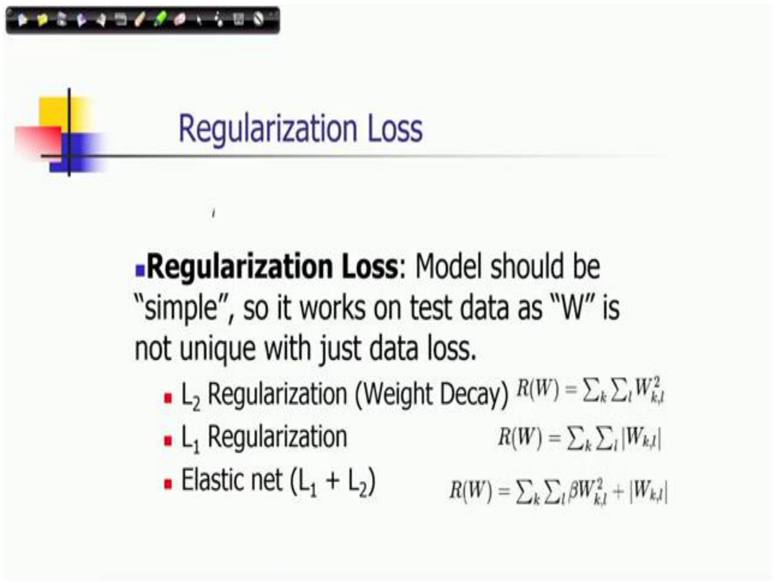
So, cross entropy loss is also another form of softmax loss when you have 2-class entropy then you have either say class 1 or class 2 that will object class and background class for example.

$$-(ylog(p) + (1-y)\log(1-p)); p: Prob. (y = 1|0)$$

So, when your object class y is 1, label is 1 then you are only considering if it is predicted as you know y then you are considering the, when the object is true class of the object is; true class of the sample is the object then you are using log(p) otherwise you are using log (1-p).

In the multi class you are extending this particular computations with this expression where you can see that y o, c is a binary indicator it is one if the object belongs to class c. And we are considering only probability of that class for the kind of sample. Here o is the input sample, so estimating probability of o belonging to class c. In a more general framework you may have a distribution of probability distribution of the true class also, so it is not binary always. So, it is true probability of o belonging to c use this, you can use it for the cross entropy loss.

(Refer Slide Time: 23:39)



Regularization Loss

- **Regularization Loss**: Model should be "simple", so it works on test data as "W" is not unique with just data loss.
  - L_2 Regularization (Weight Decay) $R(W)=\sum_k\sum_l W_{k,l}^2$
  - L_1 Regularization $R(W)=\sum_k\sum_l |W_{k,l}|$
  - Elastic net (L_1 + L_2) $R(W)=\sum_k\sum_l \beta W_{k,l}^2+|W_{k,l}|$

So, in the regulation laws we would like to have the model simple and there we would like to have the W, the size of the w as small as possible. And it is not an unique with the data loss, see if there are different kinds of regularization function which tries to put a constraint on the values of the W.

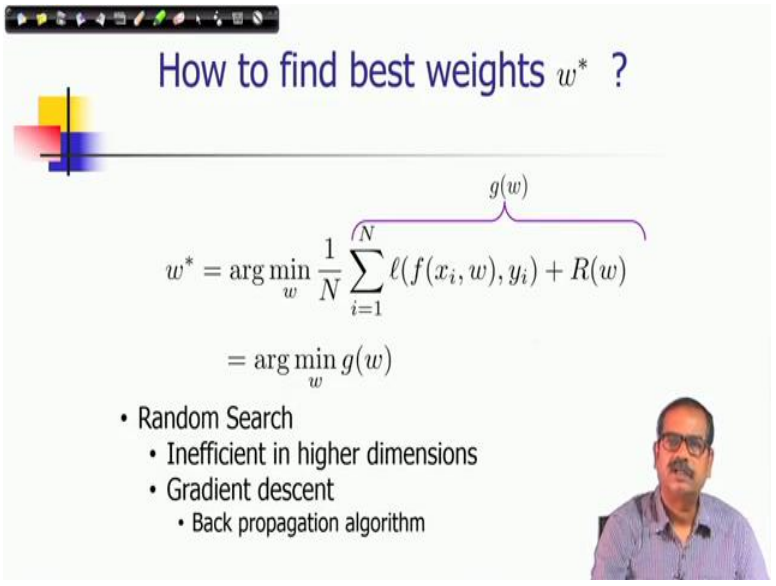
For example, you would like to minimize the

$$L2\ regularization: R(W) = \sum_k \sum_j {W_{k,j}}^2$$

$$L1\ regularization: R(W) = \sum_k \sum_j |W_{k,j}|$$

$$Elastic\ net\ (L_1 + L_2): R(W) = \sum_k \sum_j \beta {W_{k,j}}^2 + |W_{k,j}|$$

(Refer Slide Time: 24:27)



How to find best weights w^* ?

$$w^* = \arg\min_w \frac{1}{N} \overbrace{\sum_{i=1}^{N} \ell(f(x_i, w), y_i) + R(w)}^{g(w)}$$

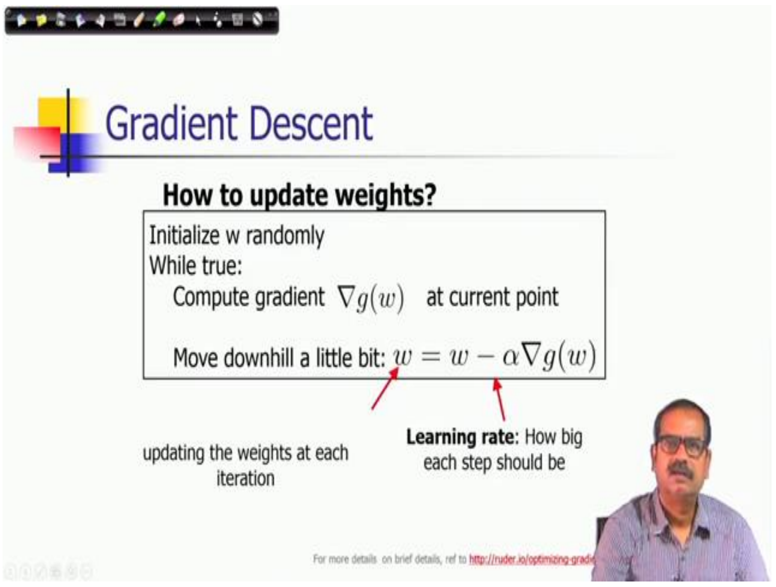
$$= \arg\min_w g(w)$$

- Random Search
  - Inefficient in higher dimensions
  - Gradient descent
    - Back propagation algorithm

So, finally, for the training we would like to solve this optimization problem. So, given this loss function which has the given this objective function which has data loss term and regularization loss term and then we would like to minimize with respect to a chosen weight. So, what is the best weight which can minimize it, so this is g(w) is your the sum of these losses and we would like to minimize g(w).

So, what we could have done that we can use random search; that means, once again we need to choose some initial guess of w and or pick here and there and try to see what kind of loss you are getting and try to get the minimum out of them, but it is very inefficient. So, standard algorithm would be gradient descent again the same back propagation algorithm what we can use it here.

(Refer Slide Time: 25:29)



Gradient Descent

How to update weights?

```
Initialize w randomly
While true:
    Compute gradient $\nabla g(w)$ at current point
    Move downhill a little bit: $w = w - \alpha \nabla g(w)$
```

updating the weights at each iteration

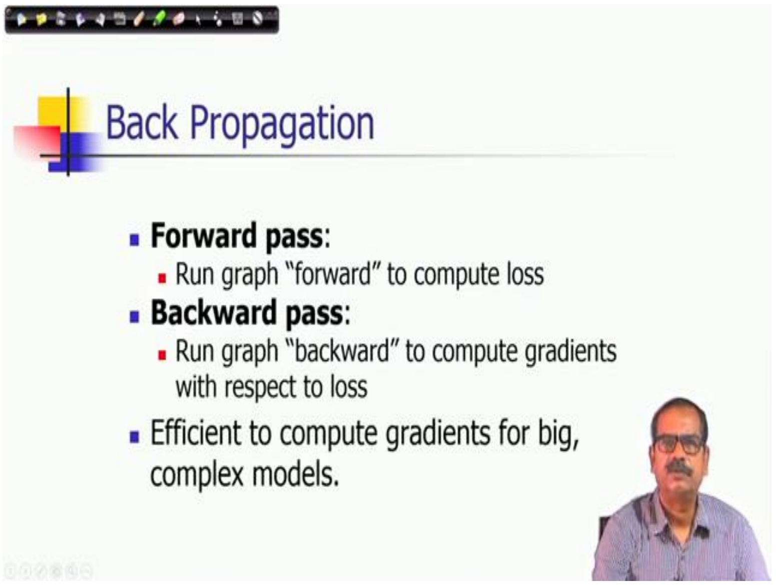
Learning rate: How big each step should be

For more details on brief details, ref to http://ruder.io/optimizing-gradi

So, gradient descent algorithm we have discussed earlier let us revisit once again in the context of this particular topic. So, we can initialize w randomly and then we can go ahead computing the gradient of that w at the current point. So, which is expressed as this grad of g(w) and then move down in a little bit because that is updation process.

So, move along in the opposite to the steepest increase of the gradient, so which means steepest descent of the gradient value, so $-\alpha\nabla g(w)$ is a learning factorial. So, this is the updating the weights at each iteration and this is the learning rate alpha is the learning rate we get step should be that needs to decided.

(Refer Slide Time: 26:17)

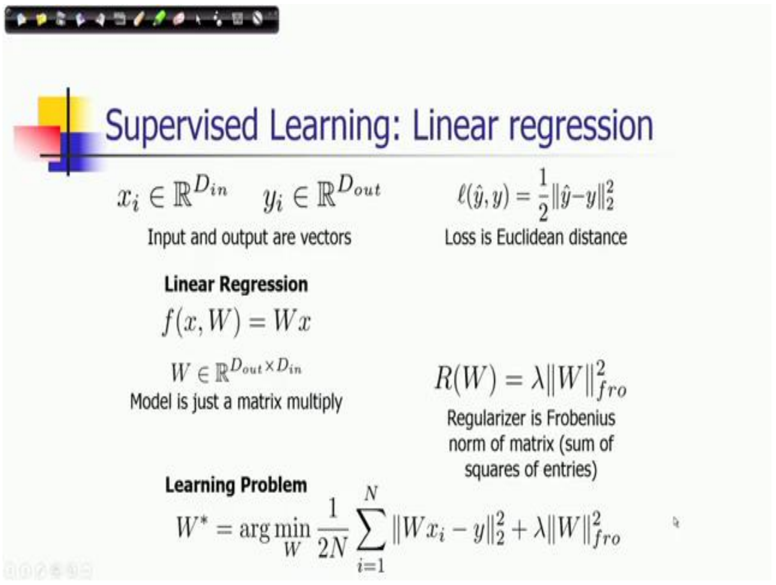


Back Propagation

- **Forward pass**:
  - Run graph "forward" to compute loss
- **Backward pass**:
  - Run graph "backward" to compute gradients with respect to loss
- Efficient to compute gradients for big, complex models.

And back propagation algorithm also we discussed that it has two passes, one is forward pass, another is backward pass. In the forward pass we run the graph forward to compute loss and the backward pass we run the graph backward to compute gradients with respect to loss. And we know that it is efficient to compute gradients for big complex models.

(Refer Slide Time: 26:41)



Supervised Learning: Linear regression

$x_i \in \mathbb{R}^{D_{in}}$ $y_i \in \mathbb{R}^{D_{out}}$

Input and output are vectors

$\ell(\hat{y}, y) = \frac{1}{2}\|\hat{y}-y\|_2^2$

Loss is Euclidean distance

Linear Regression

$f(x, W) = Wx$

$W \in \mathbb{R}^{D_{out} \times D_{in}}$

Model is just a matrix multiply

$R(W) = \lambda\|W\|_{fro}^2$

Regularizer is Frobenius norm of matrix (sum of squares of entries)

Learning Problem

$$W^* = \arg\min_{W} \frac{1}{2N}\sum_{i=1}^{N}\|Wx_i - y\|_2^2 + \lambda\|W\|_{fro}^2$$

So, supervised learning could be also considered as a linear regression problem where your input is a vector and output is also a vector. So, in a linear regression one of the linear regression problem is that now the model is linear.

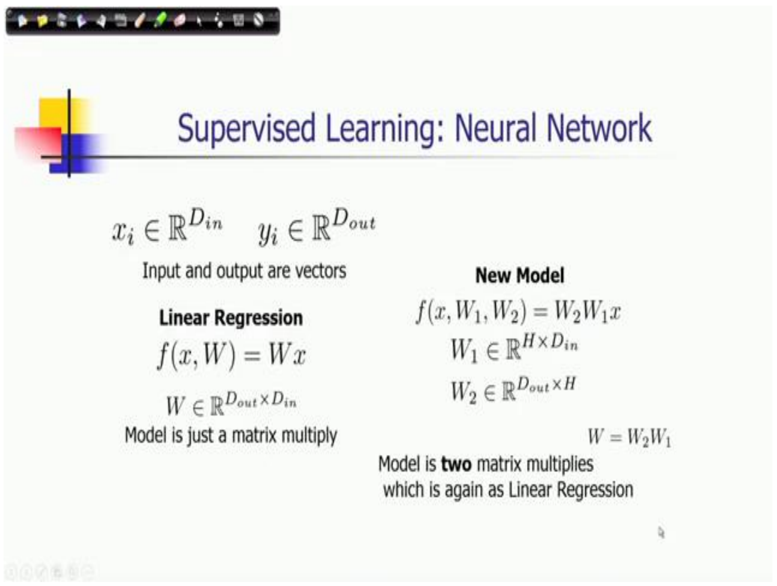
$$Linear\ regression: f(x, W) = Wx$$

$$Learning\ problem: W^* = \arg\min \frac{1}{2N} \sum_{i=1}^{N} ||Wx_i - y||_2^2 + \lambda ||W||_{fro}^2$$

So, the if your input is x, then use the weight linear combination of input that is Wx it is non-linear combination you multiply with any matrix w that would give you another vector and there is a linear model here.

So, dimension of W should be $D_{out}XD_{in}$, because your output vector is of size D_{out}. So, it is just a matrix multiplication that is the model here and you can use the euclidean distance as a loss norm of the deviations. So, you are learning problem here is a linear regression learning problem and coupled with regularization the corresponding objective function as been shown here. And regularization is using frobenius norm of matrix or sum of squares of this entries.

(Refer Slide Time: 27:51)



Supervised Learning: Neural Network

$x_i \in \mathbb{R}^{D_{in}}$ $y_i \in \mathbb{R}^{D_{out}}$

Input and output are vectors

Linear Regression

$f(x, W) = Wx$

$W \in \mathbb{R}^{D_{out} \times D_{in}}$

Model is just a matrix multiply

New Model

$f(x, W_1, W_2) = W_2 W_1 x$

$W_1 \in \mathbb{R}^{H \times D_{in}}$

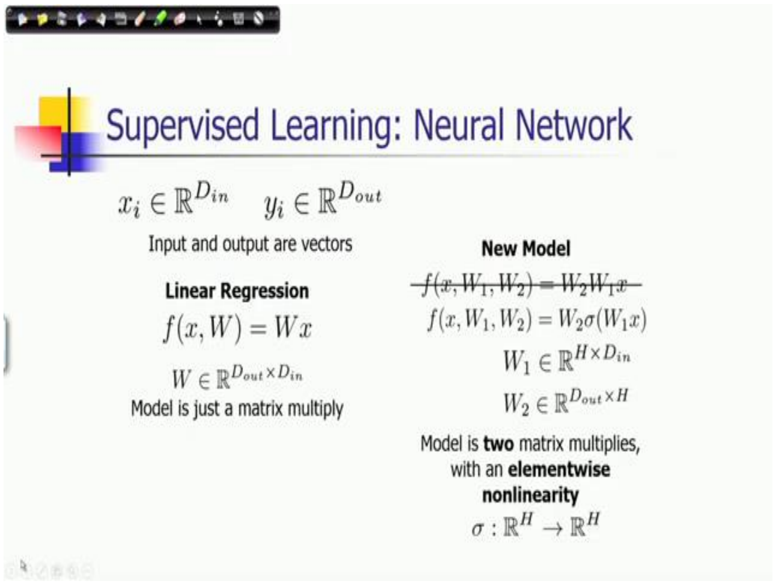
$W_2 \in \mathbb{R}^{D_{out} \times H}$

$W = W_2 W_1$

Model is **two** matrix multiplies which is again as Linear Regression

And supervised learning you can use the neural network where you can consider it is a sequence of linear layers. So, it is a new model once again as you can see that you are learning problem it is almost remaining same, because what you are doing if you are using multi layer neural network. So, first layer W1 since this is linear, next layer also you can multiply W2, but this $W=W_2W_1$, so it remains the same learning problem. So, even if you increase the layer finally, it is not increasing the capacity of problem solving it remains the same power of your problem solving remains the same.

(Refer Slide Time: 28:37)



Supervised Learning: Neural Network

$x_i \in \mathbb{R}^{D_{in}}$ $y_i \in \mathbb{R}^{D_{out}}$

Input and output are vectors

Linear Regression

$f(x, W) = Wx$

$W \in \mathbb{R}^{D_{out} \times D_{in}}$

Model is just a matrix multiply

New Model

~~$f(x, W_1, W_2) = W_2 W_1 x$~~

$f(x, W_1, W_2) = W_2 \sigma(W_1 x)$

$W_1 \in \mathbb{R}^{H \times D_{in}}$

$W_2 \in \mathbb{R}^{D_{out} \times H}$

Model is **two** matrix multiplies, with an **elementwise nonlinearity**

$\sigma : \mathbb{R}^H \to \mathbb{R}^H$

So, by introducing non-linearity here, you are giving another complexity to your model and increases the power of solving this problem. So, let me take a break here and we will continue this topic in the next lecture.

Thank you very much for listening this lecture.

Keywords: Deep architecture, loss function, regularization loss, supervised learning.