**Quick Notes On Object Detection in Neural Networks**

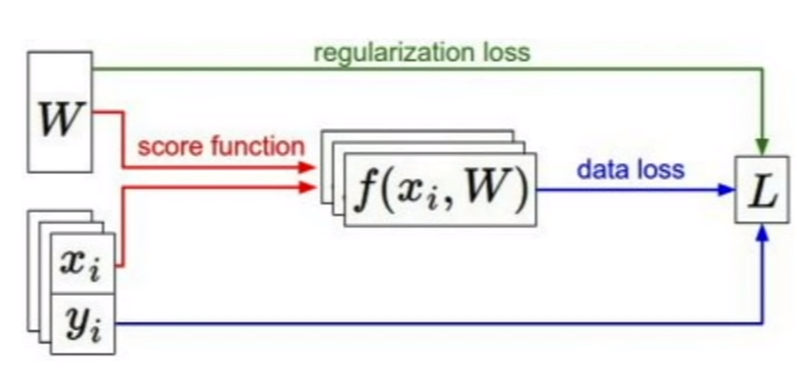
**Neural Networks and Loss Functions Quick Overview:**

**Multilabel SVM Loss:** Hingle loss, either 0, or the margin between the prediction and the correct output.

Check if all scores for the wrong categories has lower score than the actual ground truth label by some margin ( mostly 1 )

**Softmax:** Creates a probability distribution measure, normalize it to 1, and tries to minimize the negative log likelihood of the score of the correct class.

**Regularization:** Method for avoiding overfitting, a function of weights, by defining a tuning parameter lambda, penalize the complexity of the modal, that way, you can avoid overfitting models with high degrees.



**Figure 1- General overview of scores, errors and regularization**

**Gradient:** For multiple dimensions(multi variable functions), gradient is a vector of partial derivatives along each direction

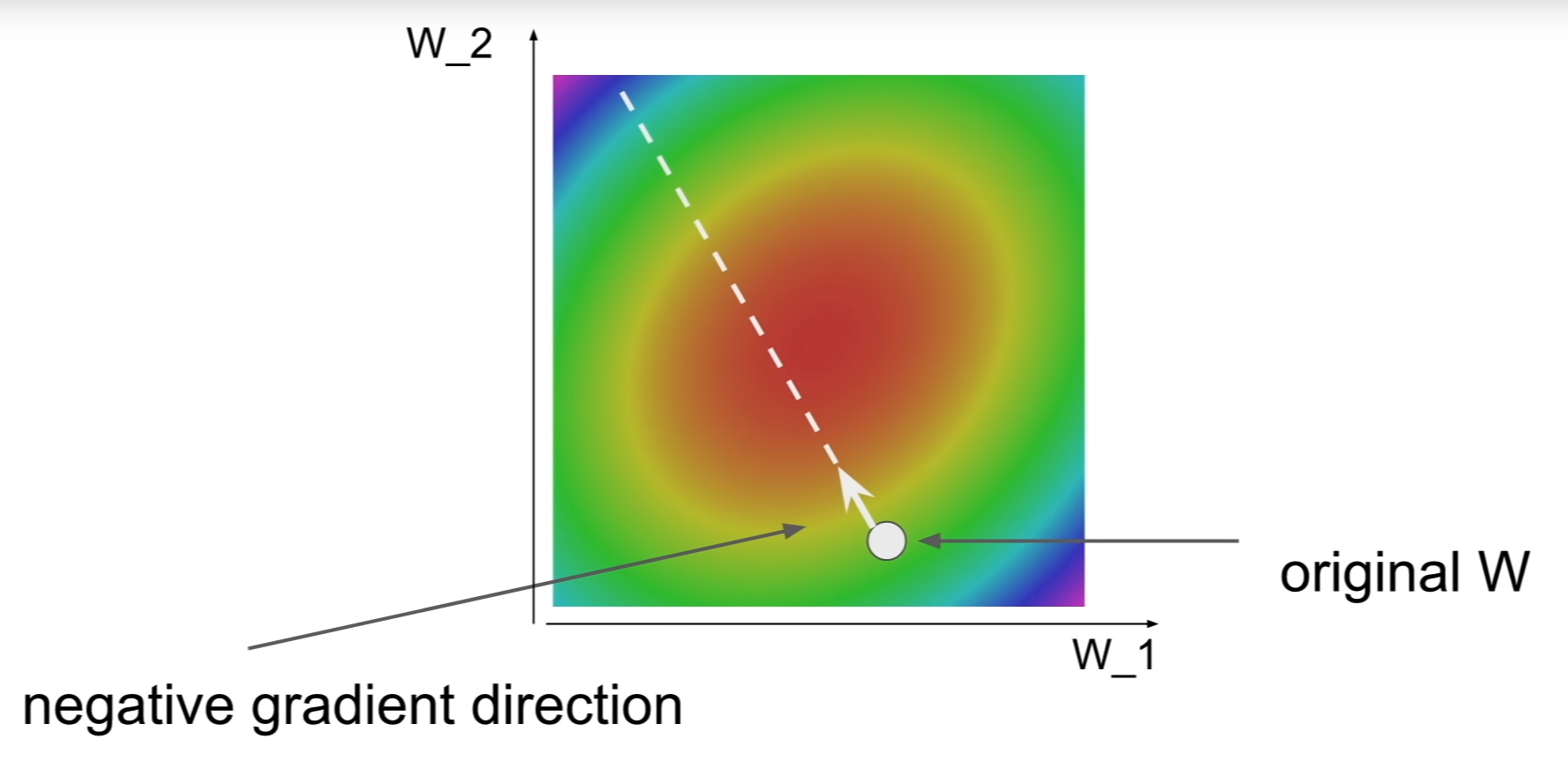
Each direction, meaning for each weight for corresponding feature, tells the slope of function f(), while moving into the direction of the corresponding element of the gradient

You can calculate the **numerical gradients** with the finite differences, just like in the limit approach, however this is too dumb and too slow. Instead, use calculus to calculte the **analytic gradient**, by making the loss as a function expression too. Also you can check the correctness of the analytic gradients by comparing to the numerical gradients when you are first building a model.

**Vanilla Gradient Descent:** An optimizing algorithm ( Finding the better weights prior to the gradients )

Starting from random weights, iteratively evaluate the gradients, and update each element(weights), in the reverse direction of the gradient( Since gradient gives the direction in where max. İncrease in a function occurs, so take the negative of that direction )

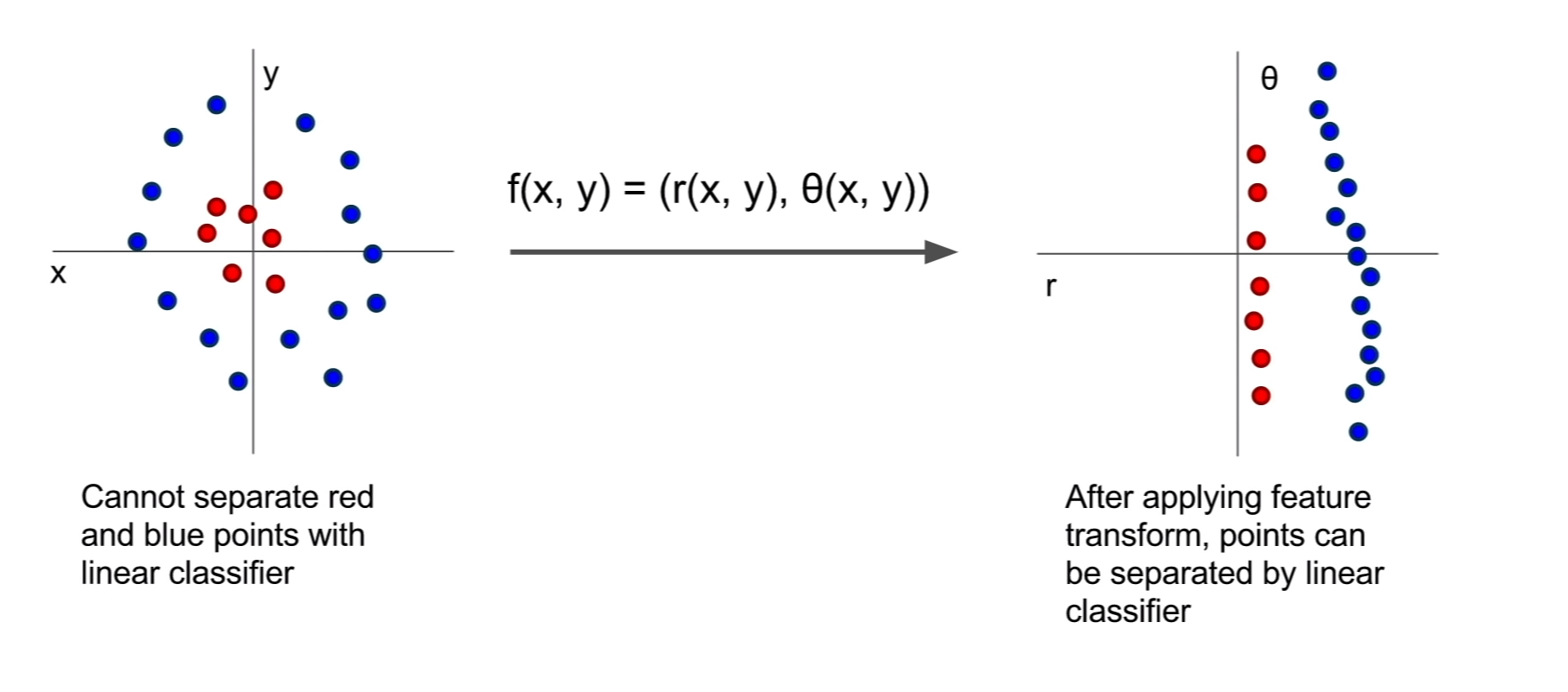
**IMPORTANT:** Gradients give the direction around each and every element(weight) you need to figure out the amount of change by some hyperparameter step size, mostly called **learning rate.** Thus, gradients just give directions with differing amounts prior to each parameter. Pick the learning rate early as possible.



**Figure 2.** Gradient visualization in 2D feature space

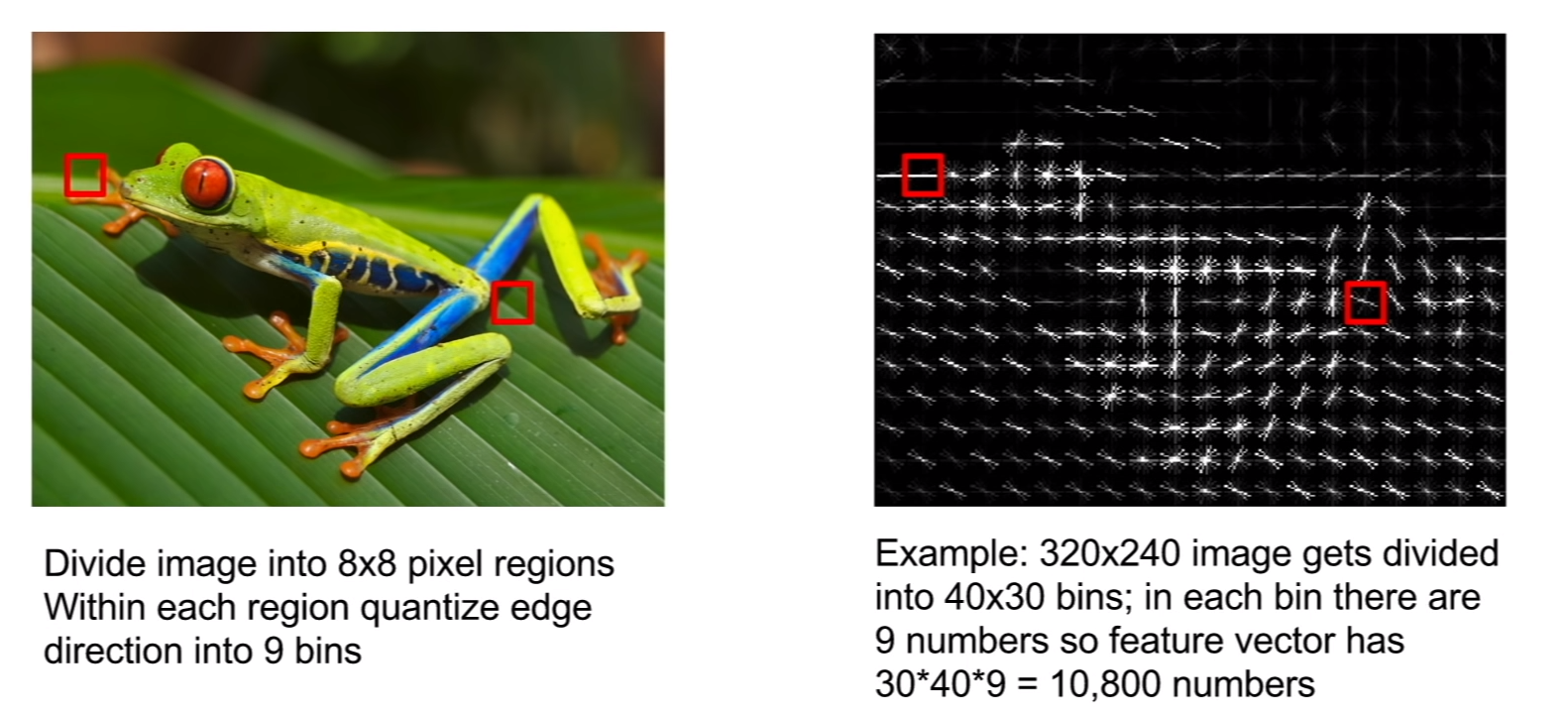
Gradient descent with momentum, and adam optimizer are some other more meaningful optimizer techniques.

However, waiting to calculate the gradients for the whole dataset is cumbersome,thus it would be more efficient to divide this set into batches **(minibatch),** so that, you can update the gradients for N number of data instances at one iteration. This optimizing technique is called **Stochastic Gradient Descent,** since the losses and the gradients change stochastically prior to the selected minibatches



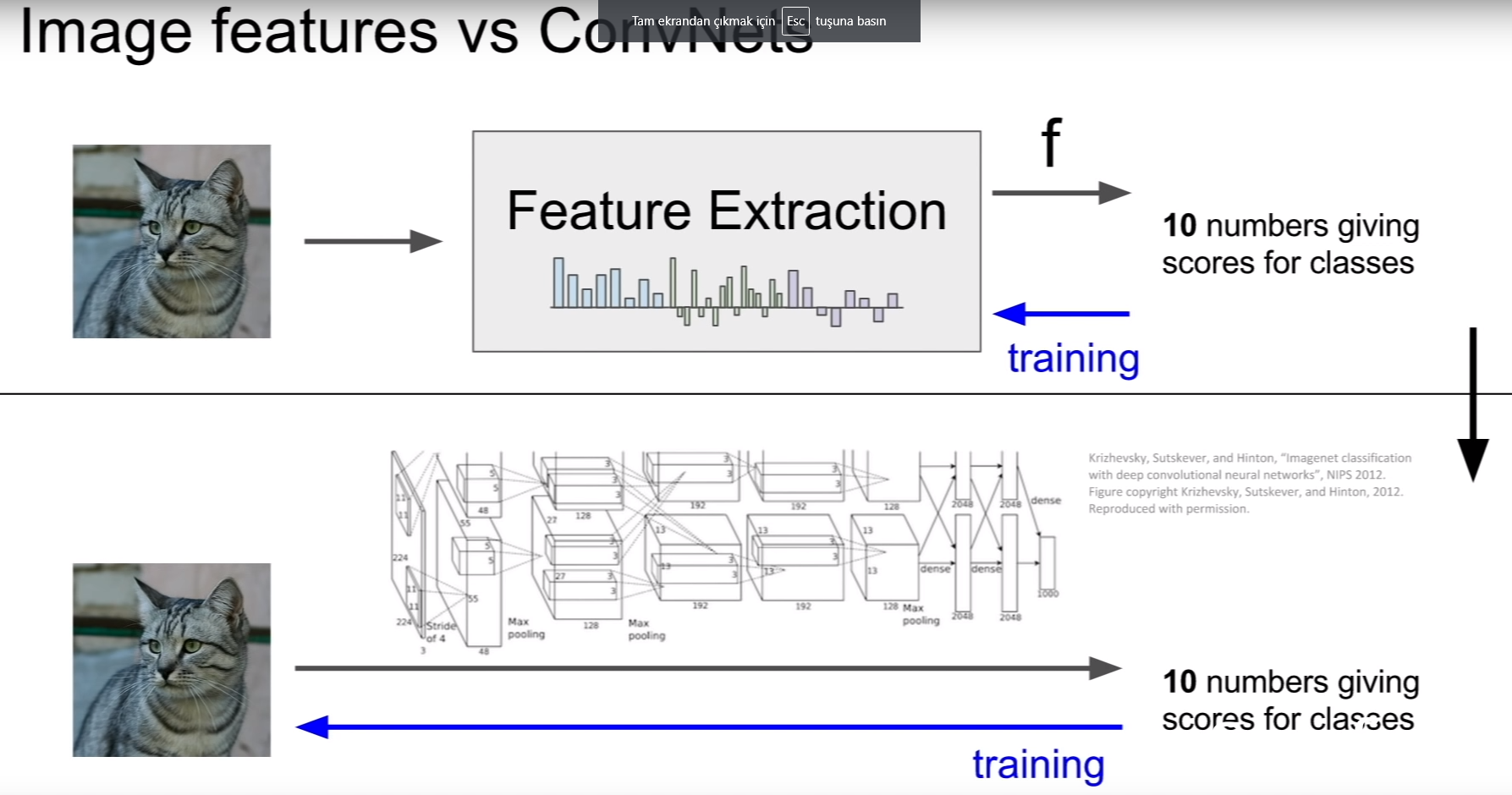
You can use a smart idea to tranform the features, since on the left is a non-linearly classifiable set of points. Thus, you can use the polar coordinates to differentiate those two distinct classes data. This is a one example of feature transformation and it can be used in some image features. ( Not has to be polar coordinates, but you can try to come up with, or extract some other different feature representations of your data that makes sense) (For example, some feature representation, like color histogram, instead of a raw pixel values for an image, visual data)

Histogram of Gradients (Finding the edge direction for each set of pixels of a fixed size) is also different technique to create new feautre vectors, before the deep neural networks’ rise.



Also,Bag of Words can also be used, influenced by mostly the NLP methods, bag of words is quantifying an occuring pattern by clusters. For images, you can take random crops, and assign to some clusters with algorithms such as K-Means.

So, You can find the feature map for a corresponding data, and come up with a classifier from those “intelligent” features instead of more dummy ones, like only raw pixels.



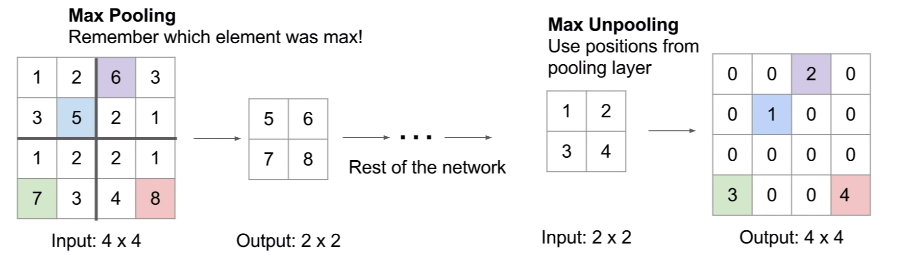
Before ConvNets, this procedure was being done like that, extracting more meaningful features by some fixed feature map extractor( which cannot be updated during the training procedure, which is dumb) and feed this extracted features to a linear classifier to come up with a succesful model. So you would only update the linear classifier ( weights of the model ) by some optimizing method, but you can not update the feature extraction mechanism

The difference and the succession of the convnets from the previous methods, is that, convnet is fully-data-driven, using raw pixels to create feature representations in the hidden layers and classify them at the end. However, convnets also updates the different feature representation creation process. Therefore you can say that it updates the whole weights, instead of just on the top( the linear classifiers)

Semantic Segmentation: Classify each pixel as an object, you can try to run a convolutional modal which has something like 3x3 kernel size and 0 padding such that the image resolution do not change, resulting output of the final conv layer is CxHxW, where C is the number of classes. Giving a probabalistic measure for C classes for each pixel. Loss for this would be the sum/average classification error for each and every pixel with cross entropy.

**A problem with this approach!** Too many operations since dimensions of the data never reduce. Instead, you can use upsampling and downsampling ( like autoencoder architecture ). You can downsample by pooling or stride convolutions

Upsampling can be done with unpooling( nearest neighbour, or bed of nails)Also max unpooling would be more intelligent, by remembering the max cell before the max pooling, then after couple of conv layers, while upsampling, only fill the elements that has the max value for the unpooling operation (Fig 1)



**Figure 3- Max unpooling**

**How does max pooling be differentiable, comes to my mind: (Explained at here:** <https://www.youtube.com/watch?v=d14TUNcbn1k&feature=youtu.be&list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk&t=1964>), it basically says, gradients only flow through the branch which has the maximum element, while others would have the gradients of 0.

Classification + Localization: Classify + Finding a bounding box

Object Detection: Find as many objects in one frame

Object Detection Algorithms:

RCNN - Runs along with a region proposal algorithm.

Region proposal algorithms finds N number of regions that can be an object candidate. Then, a ConvNet runs for each proposal in one pass, creating Objet Classes from SVM and Bounding Boxes from the regressor for each region. If It gives over %50 match, it is classified as an object

RCNN is very slow since ConvNet runs numerous times for a one frame.

Instead, one ConvNet pass can be used to extract convolution maps and then regions can be used for selecting the regions from it