Object localization

- Loss:

$L = L_{reg} + L_{cls}$

L_{reg}: regression, e.g. cross entropy; L_{cls}: classification, e.g. L_2

no object – ignore all predictions

$$L_2(y^{(i)}, \hat{y}^{(i)}) = \begin{cases} (y_1^{(i)} - \hat{y}_1^{(i)})^2 & \text{if } y_1^{(i)} = 0 \\ \sum_j (y_j^{(i)} - \hat{y}_j^{(i)})^2 & \text{otherwise} \end{cases}$$

1st element in vector y -->Pc

j: cross entropy for class labels

- Suitable for one object in an image. What should we do if there are multiple objects?

Sliding window detection

- Slide a window through the image. At each location:
  - Crop sub window
  - Classify each sub window (object/background)
- To deal with different object scales, crop different sub windows with different sizes and aspect ratios at each location
- This may be slow because there are many sub windows to classify

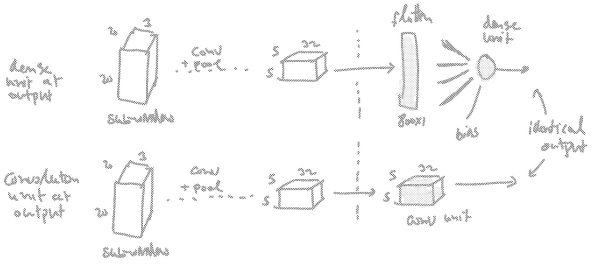
1000 × 1000 image with 10 sub-windows at each location → 10,000,000 sub-windows to classify for each image

Sliding window detection

- Problems:
  - Running dense layers for classifying each sub window is inefficient
  - In addition, dense layers need a fixed input size and so the image input size needs to be fixed
- Fully conversational implementation:
  - A fully convolutional implementation of the dense layers is more efficient: no need to process each sub window separately, and convolutions can be implemented efficiently (e.g. as in mobileNet)
  - A fully convolutional implementation can work on variable size images

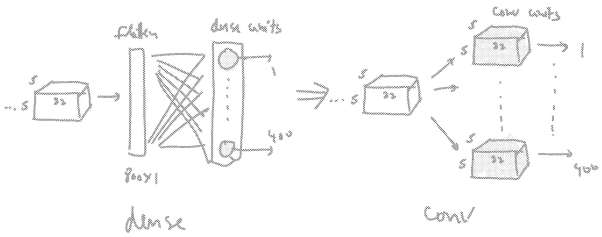
Sliding window detection

- A single fully connected (Dense) unit may be replaced by a single convolutional unit that has the exact dimensions of the input tensor (without flattening it)



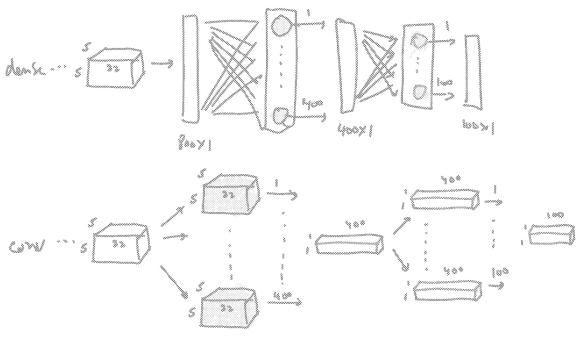
Sliding window detection

- Consider replacing 400 fully connected (FC) units
  - The output of 400 FC (dense) units is a 400×1 vector
  - We can replace 400 FC units with 5×5×32 convolution filters and so their output will be a 400×1 vector



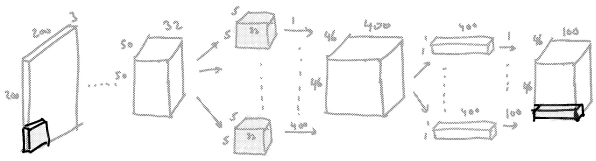
Sliding window detection

- Replacing a subsequent dense layer with 100 units:



Sliding window detection

- So far we performed a single sub window classification
- For a single sub window there is no reduction in computations
- The advantage of convolutions is when sharing computations between overlapping windows



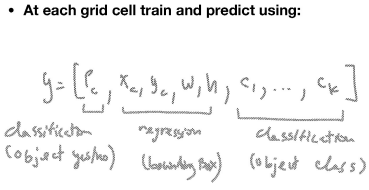
- Each column in the output provides probability for 100 classes in a corresponding image region (sub window)

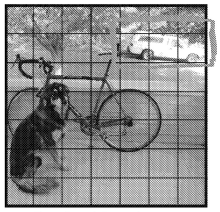
Sliding window detection

- Summary:
  - Use convolution + pooling to extract features and reduce spatial resolution
  - Each location in a reduced resolution layer corresponds to a larger patch in the original image (patches overlap)
  - Use a dense layer or 1×1 convolution to classify or regress at each location in reduced resolution layer (classify class and regress bounding box)
- Problem:
  - Detection using a sliding window does not produce accurate detection because:
    1. It is based on classification in a pooled (lower resolution) layer
    2. The region covered by filters is of fixed size and so we detect fixed size bounding boxes
  - To solve this we can, in addition to classifying each location, regress bounding boxes

Grid cell detection

- Instead of scanning the image, divide the image into grid cells and attempt to detect an object in grid cell
- Cells do not necessarily match objects and so use regression to find a bounding box for each detected object
- Similar to object localization (single object) except that now at each grid cell detect and localize objects
- At each grid cell train and predict using:



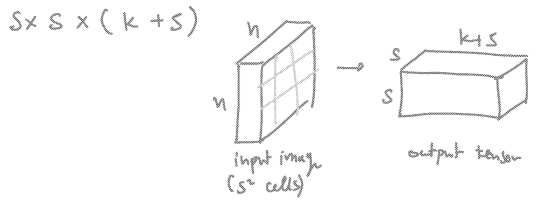


S × S grid on input

Refer image

Grid cell detection

- For an S x S grid and K classes the output is a tensor of size:



- At each grid cell with an object find a precise bounding box instead of using a fixed size window
- Only one object is detected at each grid cell

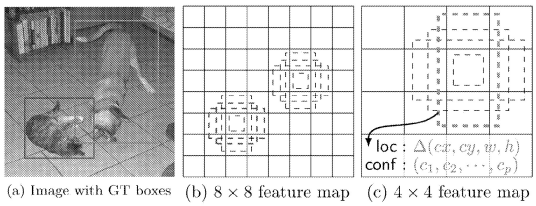
Refer image

Grid cell detection

- Problems:
  - What should be the grid cell size?
    - A too small cell will fail in classifying big objects
    - A too large cell will fail in classifying small objects
  - What should be the aspect ratio of candidate cells?
    - Different objects have different aspect ratios of boxes that should be used for their detection
  - How to detect more than one object in each grid cell?

Grid cell detection

- Anchor boxes:
  - To solve three problems we use multiple anchor boxes of each grid location
  - Anchor boxes have different sizes and different aspect ratios (selected according to what we see in the training data)



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Grid cell detection

- Multiple anchor boxes:
  - To allow detecting two different (overlapping) objects at each location (e.g. objects with different aspect ratios) double the target factor y
- Most grid cells will have zero or one anchor boxes

$y = [p_c^1, x_c^1, y_c^1, w^1, h^1, c_1^1, \ldots, c_K^1 ; p_c^2, x_c^2, y_c^2, w^2, h^2, c_1^2, \ldots, c_K^2]$

$p_c^1 = 0$ and $p_c^2 = 0$ if object does not exist

- Anchor box assignment during training:
  - During training, the anchor box assigned to an object is the one having the highest IOU with a ground truth box

Refer image

Single shot detectors

- Single shot versus double shot methods:
  - Single shot detectors perform object detection in a single network that looks at the image once
  - Two shot detectors look at the image twice: one to propose object regions (region proposals) and a second to refine and classify regions
  - Two shot methods have a higher computational cost (lower frame rate) but may be more accurate
- Single shot methods:
  - Consider predefined anchor boxes at grid cells
  - Refine and classify region proposals
- Two short methods:
  - Consider predefined anchor boxes at grid cells
  - Classify anchor boxes as objects or not to get region proposals
  - Refine and classify region proposals (higher quality classifier)
- Single shot approaches:
  - YOLO (You Only Look Once)
  - MultiBox detection
  - Single Shot MultiBox Detector (SSD)

YOLO (You Only Look Once)

- A fast single-shot detector
- Algorithm:
  - Divide image into (s x s) grid cells (e.g. s = 19). Each grid cell is responsible for detecting objects with centers in it.
  - Each grid cell predicts B boundary boxes and confidence scores (confidence that the box contains the object and is accurate).
- Confidence:
  - Confidence = Probability(object) + IOU(truth, prediction)
  - Confidence = 0 if no object in cell.
- Training:
  - Use a vector of B bounding boxes per cell.

$y = [p_c^1 b_x^1 b_y^1 h^1 w^1 c_1^1 \ldots c_K^1 ; \ldots ; p_c^B b_x^B b_y^B h^B w^B c_1^B \ldots c_K^B]$

$p_c^j = 0$ if bounding box does not contain object

- Inference:
  - Predict using dense layers or 1x1 convolutions.
  - Apply non-maximum suppression (NMS).

YOLO (You Only Look Once)

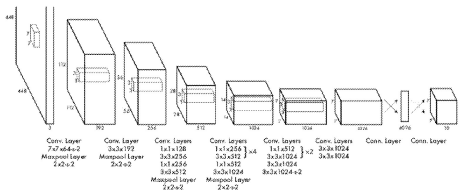


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

- **Use Leaky ReLU activations inside the network**

YOLO (You Only Look Once)

YOLO loss function:

ignore empty bounding boxes

center regression

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

H, W regression

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

object classification

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

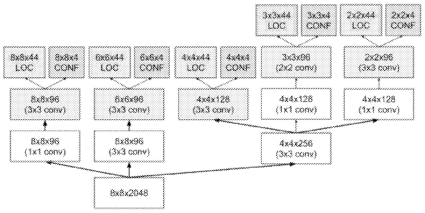
no-object classification

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

box classification

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

MultiBox detection

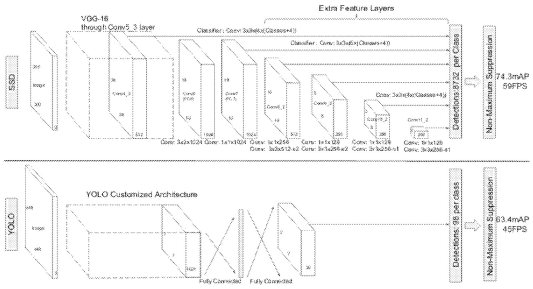


Architecture of multi-scale convolutional prediction of the location and confidences of multibox

- **Use prior for anchor boxes determined per class**
- **Inception-like detection at different scales**

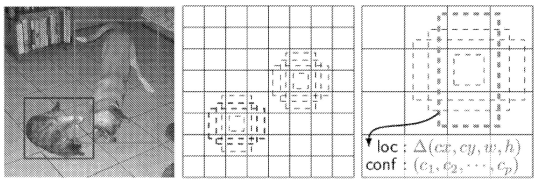
- **Loss**
  - **Confidence_loss = class categorical cross entropy**
  - **Location_loss = bounding box IoU**
  - **MultiBox_loss = Confidence_loss + alpha * Location_loss**

Single shot MultiBox detector (SSD)



- **SSD = YOLO + MultiBox**
- **Use feature maps from intermediate convolution layers**

Single shot MultiBox detector (SSD)



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

- **Use different priors for different convolution layers**
- **Apply non-maximum suppression (NMS)**
- **Use hard negatives (FP predicted by the model) to train instead of all negatives. This helps in balancing negative and positive examples.**
- **Loss:**

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

$$L_{conf} = \frac{1}{n_{\text{positives}}}\left(\sum_{positives} \text{cross-entropy loss} + \sum_{\text{hard negatives}} \text{cross-entropy loss}\right)$$