13 Computer Vision

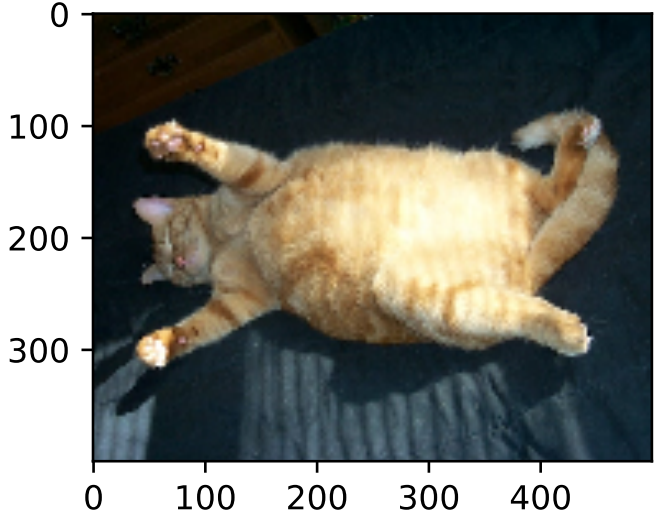
Lecture based on "Dive into Deep Learning" **http://D2L.AI** (Zhang et al., 2020)

Prof. Dr. Christoph Lippert

Digital Health & Machine Learning

- large-scale data sets are prerequisite for the successful application of deep neural networks.
- **Image augmentation** expands the scale of training data sets by random changes to the training images to produce similar, but different, training examples.
- reduce a model's dependence on certain properties, thereby improving its capability for generalization.
- crop the images, so that the objects of interest appear in different positions, reducing the model's dependence on the position.
- adjust the brightness, color, etc to reduce model's sensitivity to color.
- Most image augmentation methods have a certain degree of randomness.

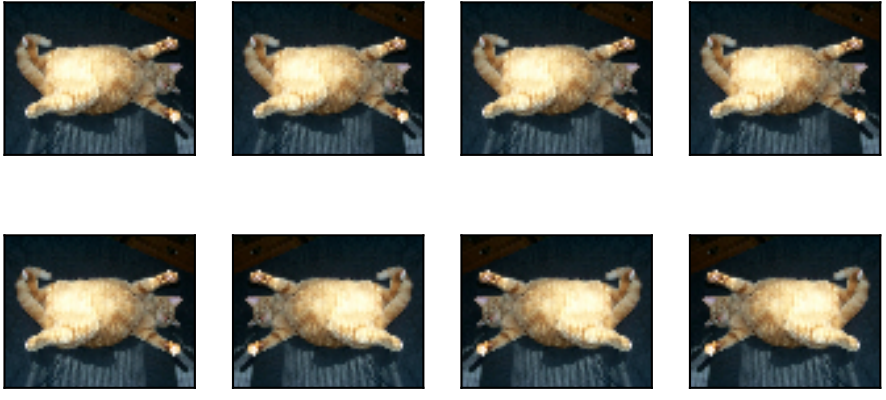
In this experiment, we will use an image with a shape of 400×500 as an example.



Flipping the image left and right usually does not change the category of the object.

Parameters:

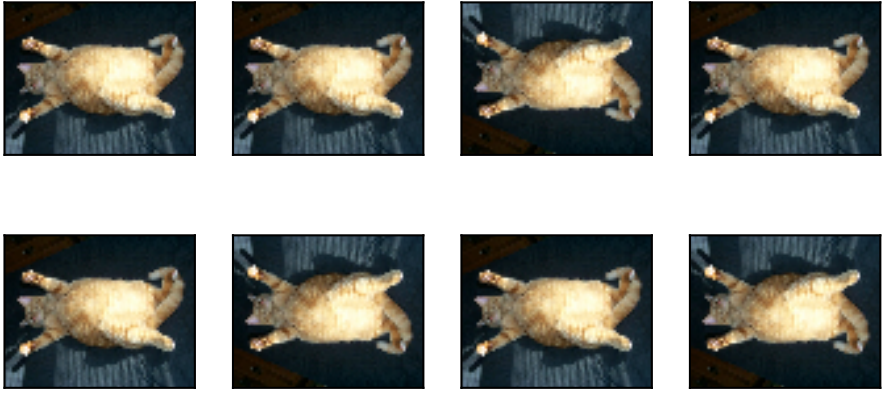
- chance that the image is flipped. (here 50%)



Flipping up and down might change category more frequently.

Parameters:

- chance that the image is flipped. (here 50%)



Random crop of a region and re-scale to original resolution

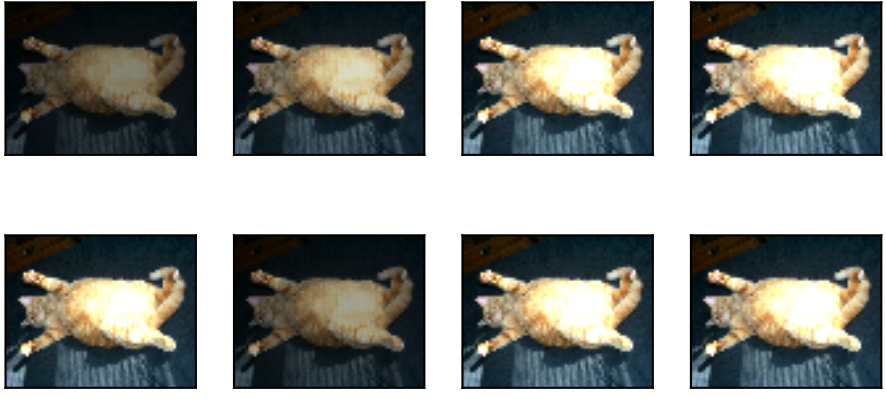
Parameters:

- interval size of the area of the crop $[a, b]$ (here 10% to 100%).
- ratio of width and height (here 0.5 to 2)

Changing colors: brightness

Parameters:

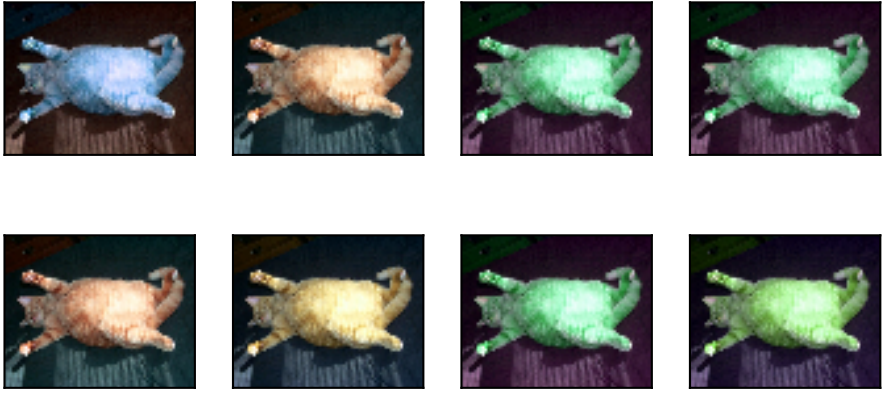
- brightness range relative to the original image (here between 50% $(1 - 0.5)$ and 150% $(1 + 0.5)$)



Changing colors: hue

Parameters:

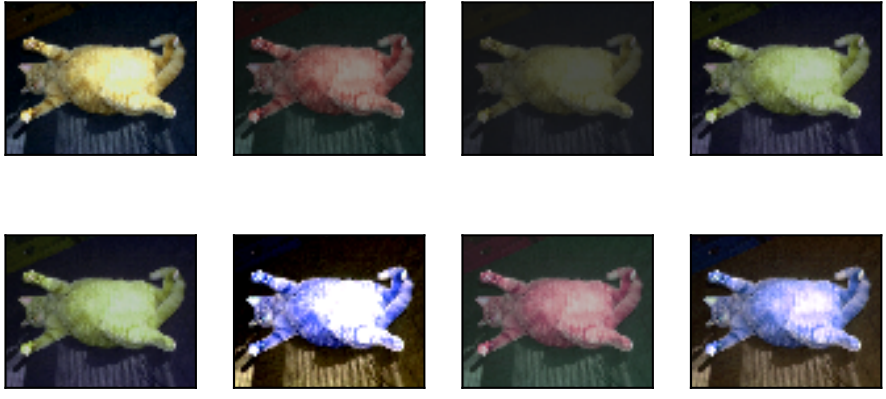
- hue range relative to the original image (here between 50% $(1 - 0.5)$ and 150% $(1 + 0.5)$)



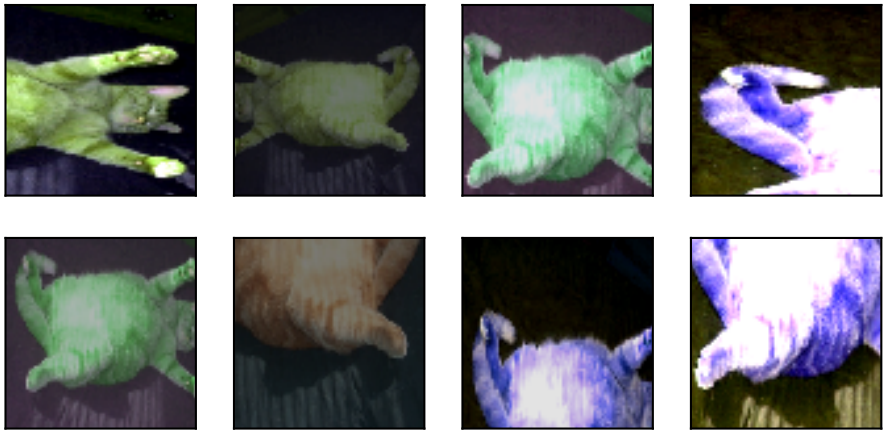
Changing colors: brightness, contrast, saturation, and hue.

Parameters:

- value range relative to the original image (here between 50% $(1 - 0.5)$ and 150% $(1 + 0.5)$)



In practice, we will overlay multiple image augmentation methods.



Summary

- Image augmentation generates random images based on existing training data to cope with overfitting.
- In order to obtain a definitive results during prediction, we usually only apply image augmentation to the training example, and do not use image augmentation with random operations during prediction.

- **ImageNet**, the most widely used large-scale image data set in the academic world, with more than 10 million images and objects of over 1000 categories.
- In order to collect the ImageNet data sets, researchers have spent millions of dollars of research funding.
- However, the size of data sets that we often deal with is usually smaller.
- A solution is to apply **transfer learning** to migrate the knowledge learned from the source data set to the target data set.

Example

Although the images in ImageNet are mostly unrelated to chest X-rays, models trained on this data set can extract more general image features that can help identify

- edges
- textures
- shapes
- object composition

These similar features may be equally effective for recognizing a lung tumor.

A common technique in transfer learning is **fine tuning**:

1. Pre-train a neural network model, i.e., the **source model**, on a **source data set** (e.g., the ImageNet data set).
2. Create a **target model** by replicating all model designs and their parameters on the source model, except the output layer.
   - We assume that these model parameters contain the knowledge learned from the source data set and this knowledge will be equally applicable to the target data set.
   - We assume that the output layer of the source model is closely related to the labels of the source data set and is therefore not used in the target model.
3. Add an **output layer** whose output size is the number of target data set categories to the target model, and randomly initialize the model parameters of this layer.
4. Train the target model on a target data set.
   - We will train the output layer from scratch.
   - The parameters of all remaining layers are fine tuned based on the parameters of the source model.

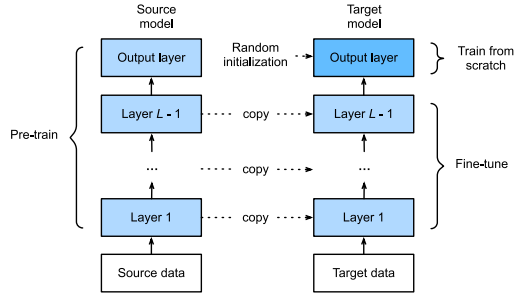


Figure: Fine tuning.

Summary

- Transfer learning migrates the knowledge learned from the source data set to the target data set.

  Fine tuning is a common technique for transfer learning.

- The target model replicates all model designs and their parameters on the source model, except the output layer, and fine tunes these parameters based on the target data set.

  In contrast, the output layer of the target model needs to be trained from scratch.

- Generally, fine tuning parameters use a smaller learning rate, while training the output layer from scratch can use a larger learning rate.

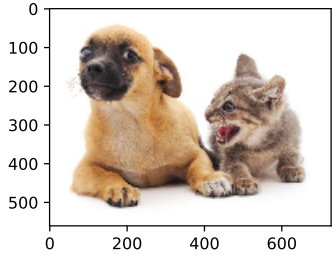
- In **image classification**, we assume that there is only one main target in the image and we only focus on how to identify the target category.
- Often there are multiple targets in the image that we are interested in.
- We want to classify and to obtain their specific positions in the image.
- We refer to such tasks as **object detection** (or **object recognition**).

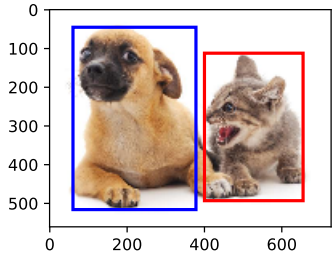
Example

In medical imaging, we are interested in detect any tumor in an image, and also predict its location.

Example

The left side of the image and a cat on the right are the two main targets in this image.





- The **bounding box** is a rectangular box that can be determined by the x and y axis coordinates in the upper-left corner and the x and y axis coordinates in the lower-right corner of the rectangle.
- The **origin** of the coordinates is the upper left corner of the image, and to the right and down are the positive directions of the x axis and the y axis, respectively.

Summary

- In object detection, we not only need to identify all the objects of interest in the image, but also their positions. The positions are generally represented by a rectangular bounding box.

- Object detection algorithms usually sample a large number of regions in the input image.
- Then determines whether these regions contain objects of interest and adjust the edges of the regions to predict the bounding box of the target.

Let the **input image** have

- a **height** of h
- a **width** of w.

We generate **anchor boxes** with different shapes centered on each pixel of the image.

- The relative **size** of the anchor box is $s \in (0, 1]$
- the **aspect ratio** is $r > 0$
- the **width** of the anchor box is $ws\sqrt{r}$
- the **height** is $hs/\sqrt{r}$.

Let $s_1, \ldots, s_n$ be a set of **sizes**.
Let $r_1, \ldots, r_m$ be a set of **aspect ratios**.

- A combination of all sizes and aspect ratios per pixel as the center, will yield $whnm$ anchor boxes.
- To reduce computational complexity, we have to reduce the number of boxes.

For example:

- only combine either s_1 or r_1 sizes and aspect ratios:

$$(s_1, r_1), (s_1, r_2), \ldots, (s_1, r_m), (s_2, r_1), (s_3, r_1), \ldots, (s_n, r_1).$$

- The number of anchor boxes per pixel is reduced from nm to $n + m - 1$.
- For the entire image, we get a total of $wh(n + m - 1)$ anchor boxes.

- Per sample the shape of the anchor box tensor y is
  (*number of anchor boxes*, 4)
- This can be reshaped to
  (image height, image width, number of anchor boxes centered on the same pixel, 4)

Each box has 4 elements in a range between 0 and 1:

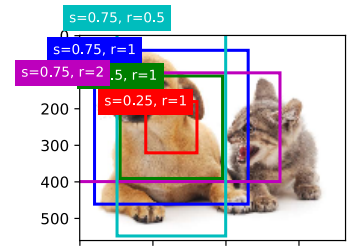
- the x, y axis coordinates in the upper-left corner
- the x, y axis coordinates in the lower-right corner The coordinate values of the x and y axis are divided by the width and height of the image, respectively.

Example

All anchor boxes at (250, 250).

Blue covers the dog well

- size 0.75
- aspect ratio 1



Given sets $\mathcal{A}$ and $\mathcal{B}$, their **Jaccard index** is the size of their intersection divided by the size of their union.

$$J(\mathcal{A}, \mathcal{B}) = \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}.$$

- consider the pixel area of a bounding box as a collection of pixels.
- In this way, we can measure the similarity of the two bounding boxes by the Jaccard index of their pixel sets.
- For bounding boxes, the Jaccard index measures **intersection over union (IoU)**.

  the ratio of the intersecting area to the union area
- The range of IoU is (0,1)
  - 0 means that there are no overlapping pixels between the two bounding boxes
  - 1 indicates that the two bounding boxes are equal.

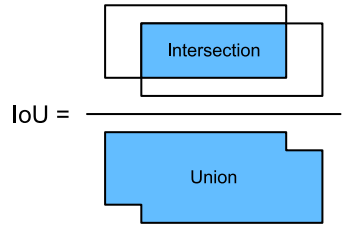


Figure: IoU is the ratio of the intersecting area to the union area of two bounding boxes.

During **training** each anchor box is a training example with 2 labels.

- the category of the target contained in the anchor box (category)
- the offset of the ground-truth bounding box relative to the anchor box (offset)

For training, we greedily assign n_b ground-truth bounding boxes to $n_a \geq n_b$ anchor boxes.

- anchor boxes in the image are $A_1, A_2, \ldots, A_{n_a}$
- ground-truth bounding boxes are $B_1, B_2, \ldots, B_{n_b}$
- matrix $\boldsymbol{X} \in \mathbb{R}^{n_a \times n_b}$
  where x_{ij} is the IoU of the anchor box A_i to the ground-truth bounding box B_j.

Greedy assignment to ground truth

Given $\boldsymbol{X}$ and threshold τ.

- For $b \in [1, n_b]$
  1. find the largest element x_{i_b, j_b} and record the row index and column index of the element as i_b, j_b
  2. Assign the ground-truth bounding box B_{j_b} to the anchor box A_{i_b}.
  3. Discard all elements in the i_bth row and the j_bth column in $\boldsymbol{X}$.
- for the remaining $n_a - n_b$ anchors:
  - If largest IoU is larger than τ:
    - assign anchor to ground truth bounding box with largest IoU

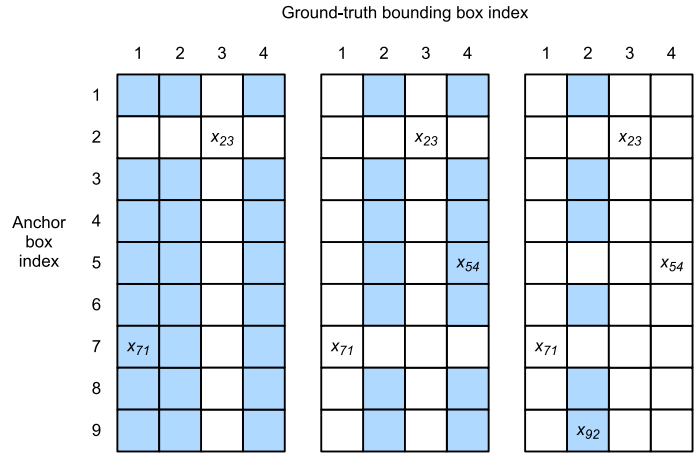
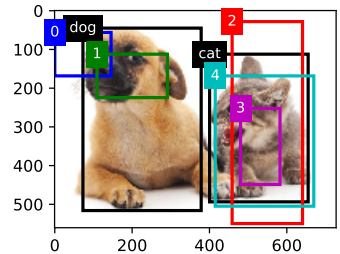


Figure: Assign ground-truth bounding boxes to anchor boxes.

Example

- The IoU of anchor box A_4 to the ground-truth bounding box of the cat is the largest, so anchor box A_4 is labeled as cat.
- Without A_4 or the ground-truth bounding box of the cat, the pair with the largest IoU is anchor box A_1 and the ground-truth bounding box of the dog, so anchor box A_1 is labeled as dog.
- For the remaining three unlabeled anchor boxes.
- The category of the ground-truth bounding box with the largest IoU with anchor box A_0 is dog, but the IoU is less than the threshold (0.5), so the category is labeled as background;
- the category of the ground-truth bounding box with the largest IoU with anchor box A_2 is cat and the IoU is greater than the threshold, so the category is labeled as cat;
- the category of the ground-truth bounding box with the largest IoU with anchor box A_3 is cat, but the IoU is smaller than the threshold, so the category is labeled as background.



5 anchor boxes:
$A_0, \ldots, A_4$

Labeling anchor boxes

1. If an anchor box A is assigned ground-truth bounding box is B
   - the **category** of the anchor box A is set to the category of B
   - the **offset** of the anchor box A is set according to the relative position of the central coordinates of B and A and the relative sizes of the two boxes.
     - **center coordinates** of anchor box A are (x_a, y_a)
     - center coordinates of assigned ground-truth bounding box B are (x_b, y_b)
     - the **widths** are w_a, w_b
     - **heights** are h_a, h_b, respectively.
     - In this case, a common technique is to label the **offset** of A as

$$\left(\frac{\frac{x_b - x_a}{w_a} - \mu_x}{\sigma_x}, \frac{\frac{y_b - y_a}{h_a} - \mu_y}{\sigma_y}, \frac{\log \frac{w_b}{w_a} - \mu_w}{\sigma_w}, \frac{\log \frac{h_b}{h_a} - \mu_h}{\sigma_h} \right),$$

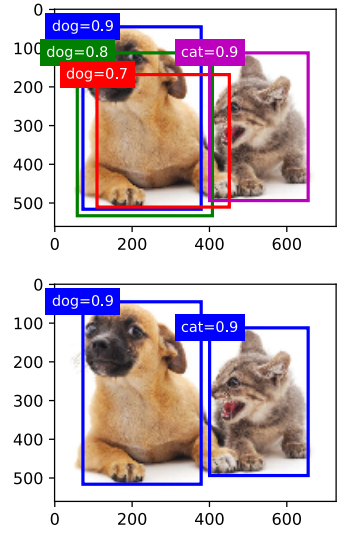
     The default values of the constants are
     $\mu_x = \mu_y = \mu_w = \mu_h = 0 \qquad \sigma_x = \sigma_y = 0.1 \qquad \sigma_w = \sigma_h = 0.2.$

2. If an anchor box is not assigned a ground-truth bounding box, the category is **background**.

During **prediction**:

1. predict categories and offsets for all anchor boxes
2. Generate **prediction bounding boxes** based on anchor boxes and their predicted offsets.
3. remove similar predictions using **non-maximum suppression** (NMS):
   - Calculate the predicted probabilities p for each category.
   - The largest probability is the **confidence level** for the box.
   - Sort the non-background predictions by confidence level to obtain the list L.
   - Repeat until all boxes in L have been used as a baseline:
     1. Select the prediction B_1 with highest confidence level from L as a baseline
     2. remove all non-benchmark prediction bounding boxes with an IoU with B_1 greater than a certain threshold from L.
4. Output remaining prediction bounding boxes in L.

   (The IoU of any pair of boxes in the output is less than the threshold.)



Summary

- We generate multiple anchor boxes with different sizes and aspect ratios, centered on each pixel.
- IoU, also called Jaccard index, measures the similarity of two bounding boxes. It is the ratio of the intersecting area to the union area of two bounding boxes.
- In the training set, we mark two types of labels for each anchor box: one is the category of the target contained in the anchor box and the other is the offset of the ground-truth bounding box relative to the anchor box.
- When predicting, we can use non-maximum suppression (NMS) to remove similar prediction bounding boxes, thereby simplifying the results.

Example (image with a height of 561 and a width of 728 pixels)

If five different shapes of anchor boxes are generated centered on each pixel, over two million anchor boxes need to be predicted and labeled.
$(561 \times 728 \times 5)$

We need to reduce the number of anchor boxes to reduce computational complexity:

- Apply uniform sampling on a small portion of pixels from the input image and generate anchor boxes centered on the sampled pixels.
- Generate anchor boxes of varied numbers and sizes on **multiple scales**.

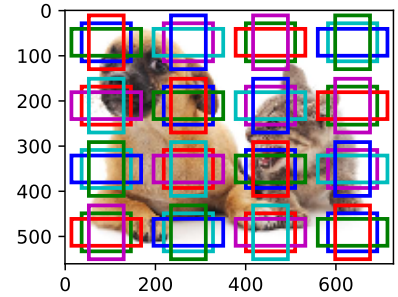
Example (image with the shape 2×2)

- 4 possible positions for objects with shape 1×1
- 2 possible positions for objects with shape 1×2
- 1 possible positions for an object with shape 2×2.

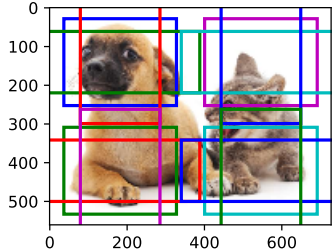
  - When using smaller anchor boxes to detect smaller objects, we can sample more regions
  - when using larger anchor boxes to detect larger objects, we can sample fewer regions.

- The 2D array output of the convolutional neural network (CNN) is called a **feature map**.
- We can determine the **midpoints** of anchor boxes uniformly sampled on any image by defining the **shape** of the **feature map**.
- We are going to generate anchor boxes `anchors` centered on each unit (pixel).

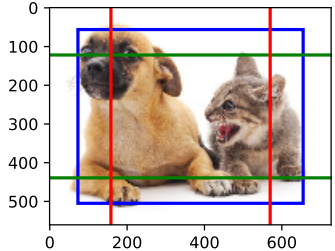
- We will first focus on the detection of small objects.
- In order to make it easier to distinguish upon display, the anchor boxes with different midpoints here do not overlap.
- We assume that the size of the anchor boxes is 0.15 and the height and width of the feature map are 4.
- We can see that the midpoints of anchor boxes from the 4 rows and 4 columns on the image are uniformly distributed.



We are going to reduce the height and width of the feature map by half and use a larger anchor box to detect larger objects. When the size is set to 0.4, overlaps will occur between regions of some anchor boxes.



Finally, we are going to reduce the height and width of the feature map by half and increase the anchor box size to 0.8. Now the midpoint of the anchor box is the center of the image.



Summary

- We can generate anchor boxes with different numbers and sizes on multiple scales to detect objects of different sizes on multiple scales.
- The shape of the feature map can be used to determine the midpoint of the anchor boxes that uniformly sample any image.
- We use the information for the input image from a certain receptive field to predict the category and offset of the anchor boxes close to that field on the image.

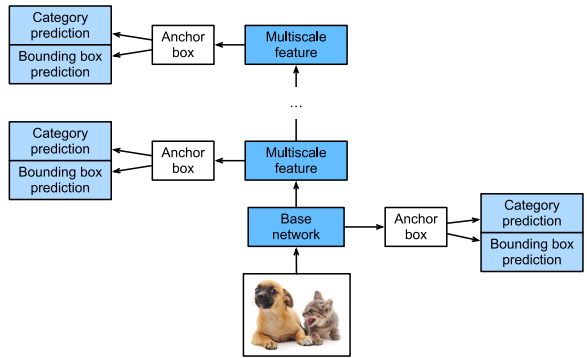
single shot multibox detection (SSD)[1].

- series of **multiscale feature blocks**
- each multiscale feature block reduces the height and width of the feature map from the previous layer (for example, reduce sizes by half).

  The blocks use each element in the feature map to expand the receptive field on the input image.
- the closer a multiscale feature block is to the top
  - the smaller its **output feature map**, the fewer the anchor boxes that are generated
  - the larger the **receptive field** of each element in the feature map and the better suited it is to detect larger objects

[1] `Liu.Anguelov.Erhan.ea.2016`

- At a scale feature map has height h and width w
- If we generate a anchor boxes per element, we get hwa anchor boxes
- SSD uses a convolutional layer that maintains the input h and w channels to predict categories.
- output and input have identical spatial coordinates (x, y)
  - There are $a(q+1)$ category prediction channels (q categories plus background)
    - indexed as $i(q+1)+j$ with $(0 \leq j \leq q)$
    - category j
    - anchor box i.
  - $a4$ offset prediction channels for bounding box prediction



Training

Object detection uses the sum of two losses.

- anchor box category loss
  cross-entropy loss
- anchor box offset loss
  $\|\cdot\|2^2$ or L_1 norm loss on the difference between the predicted value and the ground-truth value.

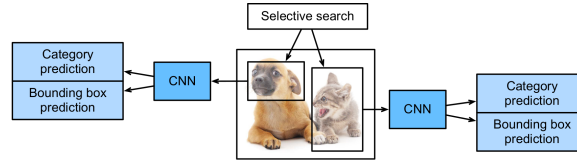
Summary

- SSD is a multiscale object detection model.
  - It generates different numbers of anchor boxes
  - different sizes based on the base network block and each multiscale feature block
  - It predicts the categories and offsets of the anchor boxes to detect objects of different sizes.
- During SSD model training, the loss function is calculated using the predicted and labeled category and offset values.

- Region-based convolutional neural networks or regions with CNN features (R-CNNs)Girshick.Donahue.Darrell.ea.2014.
  1. first select several proposed regions from an image
  2. use a CNN to extract features from each proposed area
  3. perform a classification in each area
  4. As these regions have a high degree of overlap, independent feature extraction results in a high volume of repetitive computations.
- Fast R-CNN Girshick.2015*1
  - improves on the R-CNN by only performing CNN forward computation on the image as a whole
- Faster R-CNN Girshick.2015
- Mask R-CNN He.Gkioxari.Dollar.ea.2017.

R-CNNs are composed of four main parts:

1. Selective search [1] is performed on the input image to select multiple high-quality regions
   - on multiple scales and different shapes
   - The category and ground-truth bounding box of each proposed region is labeled.
2. A pre-trained CNN without the output layer transforms each proposed region into the input dimensions required to output the features extracted from the proposed regions
3. The features and labeled category of each region are combined to train multiple support vector machines for object classification.
4. The features and labeled bounding box of each region are combined to train a linear regression model for ground-truth bounding box prediction.

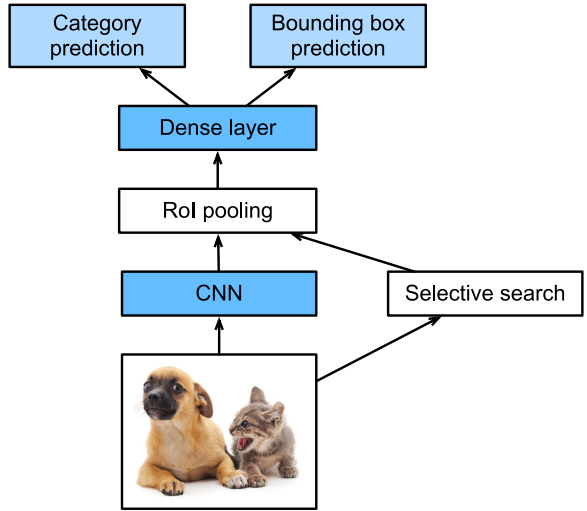


[1] `Uijlings.Van-De-Sande.Gevers.ea.2013.`

- The main downside is the slow speed.

Fast R-CNN extracts feature from the entire image.

1. As the input is an entire image, the CNN output shape is $1 \times c \times h_1 \times w_1$.
2. selective search generates n proposed **regions of interests (RoIs)** of **different shapes** on the CNN output.
   - Features of the **same shapes** must be extracted from the RoIs (height h_2, width w_2).
   - **RoI pooling**, transforms the CNN output for RoIs to output a concatenation of the features extracted from each proposed region with the shape $n \times c \times h_2 \times w_2$.
   - A fully connected layer transforms the output shape to $n \times d$
3. **category prediction**
   - the shape is transformed to $n \times q$.
   - Softmax is used to predict (q) categories
4. **bounding box prediction**
   - the shape is again transformed to $n \times 4$.



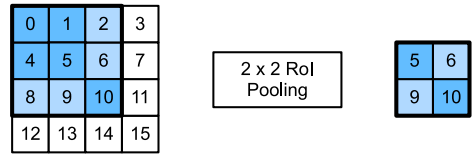
In the **RoI pooling layer** we can directly specify the output height h_2 and width w_2 of each output for an input RoI of height h and width w.

- window is divided into a grid of sub-windows with the shape $h_2 \times w_2$.
- The size of each sub-window is about $(h/h_2) \times (w/w_2)$ (integers).

Example

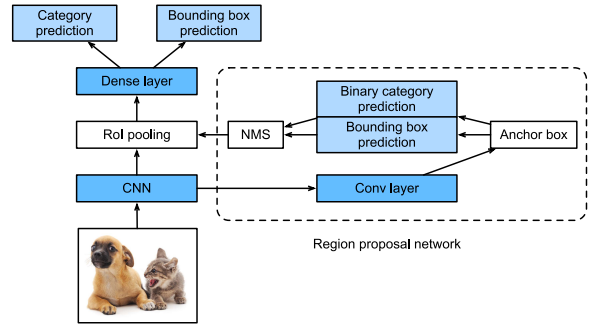
- we select an 3×3 region as an RoI of the 4×4 input.
- For this RoI, we use a 2×2 RoI pooling layer to obtain a single 2×2 output.
- divide the region into four sub-windows

  they respectively contain the elements
  - 0, 1, 4, and 5 (5 is the largest);
  - 2 and 6 (6 is the largest)
  - 8 and 9 (9 is the largest)
  - 10.

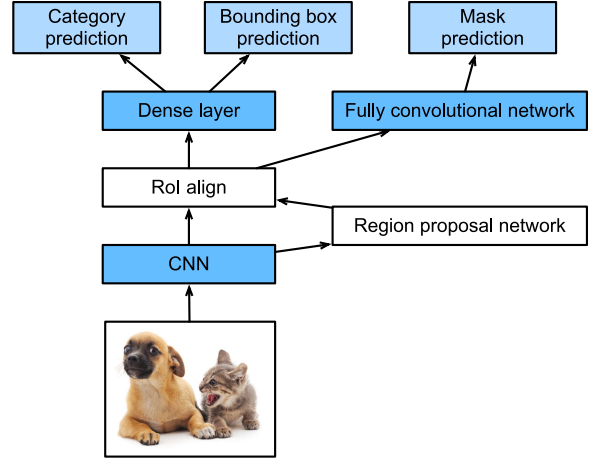


Faster R-CNN replaces selective search with a **region proposal network**.

1. 3×3 convolutional layer with a padding of 1 to transform the CNN output and set the number of output feature map channels to c.
2. generate multiple anchor boxes of different sizes centered at each element
3. use the c features at the center on the anchor boxes to predict the binary category (object or background) and bounding box.
4. use non-maximum suppression to remove similar bounding boxes for "object".



A **Mask R-CNN** can effectively use **pixel-level positions** of each object in an image to further improve the precision of object detection.



Summary

- An R-CNN model selects several proposed regions and uses a CNN to perform forward computation and extract the features from each proposed region. It then uses these features to predict the categories and bounding boxes of proposed regions.
- Fast R-CNN improves on the R-CNN by only performing CNN forward computation on the image as a whole. It introduces an RoI pooling layer to extract features of the same shape from RoIs of different shapes.
- Faster R-CNN replaces the selective search used in Fast R-CNN with a region proposal network. This reduces the number of proposed regions generated, while ensuring precise object detection.
- Mask R-CNN uses the same basic structure as Faster R-CNN, but adds a fully convolution layer to help locate objects at the pixel level and further improve the precision of object detection.

Semantic segmentation attempts to segment images into regions with different semantic categories.

- label and predict objects at the pixel level.

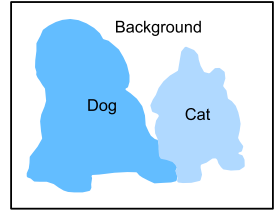


Image segmentation divides an image into several constituent regions.

- uses the correlations between pixels in an image.
- During training, labels are not needed for image pixels.
- This method cannot ensure that the segmented regions have the *semantics* we want.

Example

Image segmentation might divide the dog into two regions.

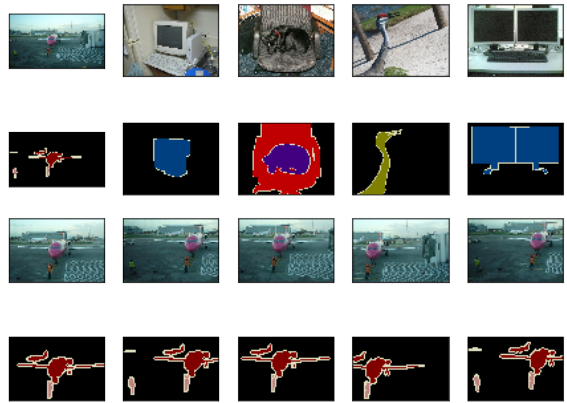
- the dog's mouth and eyes where black is the prominent color
- the rest of the dog where yellow is the prominent color.

Instance segmentation is also called simultaneous detection and segmentation.

- It attempts to identify the pixel-level regions of each *object instance* in an image.
- Instance segmentation not only distinguishes semantics, but also different object instances.
- If an image contains two dogs, instance segmentation will distinguish which pixels belong to which dog.

In semantic segmentation, one important data set is Pascal VOC2012.

- For example, in the first example image, the category index for the front part of the airplane is 1 and the index for the background is 0.



Summary

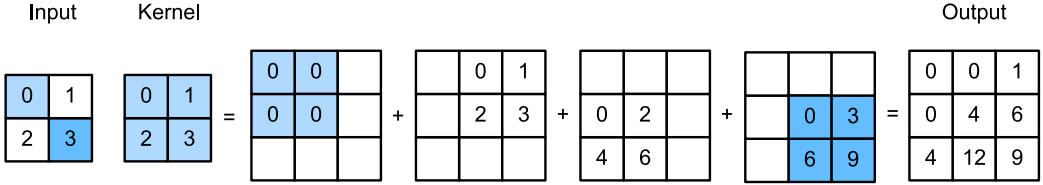
- Semantic segmentation looks at how images can be segmented into regions with different semantic categories.
- In the semantic segmentation field, one important data set is Pascal VOC2012.

- The CNN layers we introduced so far reduce the input width and height, or keep them unchanged.
  - convolutional layer
  - pooling layer
- **semantic segmentation** and **generative adversarial networks** require **pixel-level predictions** and therefore need to increase input width and height.

Transposed convolution, also named **fractionally-strided convolution** or **deconvolution**, allows to increase width and height.

Example

Let's consider a basic case that both input and output channels are 1, with 0 padding and 1 stride. transposed convolution with a 2×2 kernel is computed on the 2×2 input matrix.



- We apply padding elements to the *input* in convolution, while they are applied to the *output* in transposed convolution.

  A 1×1 padding means we first compute the output, then remove the first/last rows and columns.

- Strides are applied to outputs as well.

- The multi-channel extension of the transposed convolution is the same as the convolution.

  - c_i number of input channels
  - $k_h \times k_w$ kernel matrix
  - the transposed convolution assigns a kernel matrix to each input channel.
  - c_o number of output channels
  - $c_i \times k_h \times k_w$ kernel tensor for each output channel.

- If we feed X into a convolutional layer f to compute $Y = f(X)$

  transposed convolution layer g with the same hyper-parameters as f except for the output channel set to be the channel size of X, then $g(Y)$ should has the same shape as X.

Convolution can be represented by matrix multiplication.

Example

take a flattened 4-by-4 input matrix as a 16-dim vector and produce a 4-dim vector that is re-shaped as a 2-by-2 tensor.

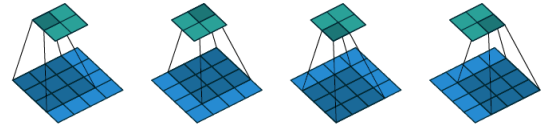


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

Transposed convolution can be represented by multiplication with the transposed matrix.

Example

take a flattened 2-by-2 input tensor as a 4-dim vector and produce a 16-dim vector that is re-shaped as a 4-by-4 tensor.

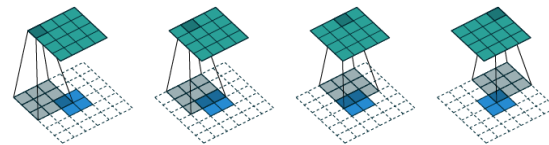


Figure 4.1: The transpose of convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$). It is equivalent to convolving a 3×3 kernel over a 2×2 input padded with a 2×2 border of zeros using unit strides (i.e., $i' = 2$, $k' = k$, $s' = 1$ and $p' = 2$).

Summary

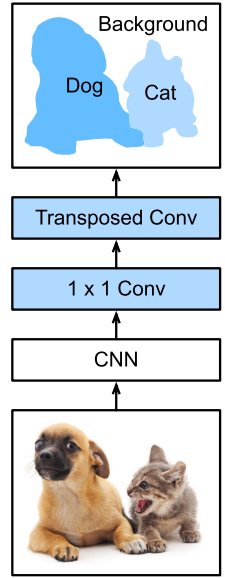
- Compared to convolutions that reduce inputs through kernels, transposed convolutions broadcast inputs.
- If a convolution layer reduces the input width and height by n_w and h_h time, respectively.

  Then a transposed convolution layer with the same kernel sizes, padding and strides will increase the input width and height by n_w and n_h, respectively.
- We can implement convolution operations by the matrix multiplication, the corresponding transposed convolutions can be done by transposed matrix multiplication.

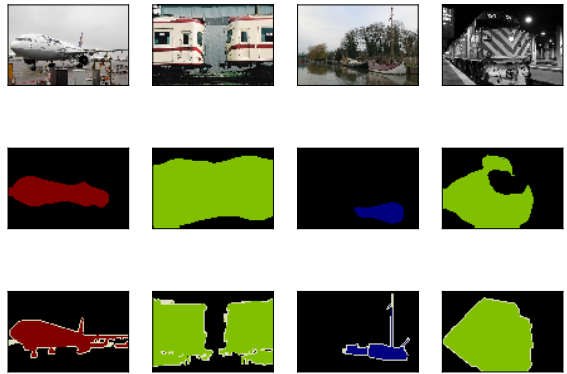
A **fully convolutional neural network (FCN)**

1. uses the convolutional neural network to extract image features
2. transforms the number of channels into the number of categories through the 1×1 convolution layer
3. transforms the height and width of the feature map to the size of the input image by using the transposed convolution layer.

- The model output has the same height and width as the input image and has a one-to-one correspondence in spatial positions.
- The final output channel contains the category prediction of the pixel of the corresponding spatial position.



- ResNet-18 model pre-trained on ImageNet to extract image features
- Given an input of a height and width of 320 and 480 respectively, the forward computation reduces the height and width of the input to $1/32$ of the original, i.e. 10 and 15.
- Transform the number of output channels to the number of categories of Pascal VOC2012 (21) through the 1×1 convolution layer.
- Magnify the height and width of the feature map by a factor of 32 to change them back to the height and width of the input image using transposed convolution.



Summary

The fully convolutional network

1. uses the convolutional neural network to extract image features
2. transforms the number of channels into the number of categories through the 1×1 convolution layer
3. transforms the height and width of the feature map to the size of the input image by using the transposed convolution layer to output the category of each pixel.