EE 628
Deep Learning
Fall 2019

Lecture 12

11/14/2019

Assistant Prof. Sergul Aydore

Department of Electrical and Computer Engineering

Overview

- Last lecture we covered
  - Finish implementing RNN
  - GRUs and LSTMs
  - Attention Mechanism

- Today, we will cover
  - Advanced topics in Computer Vision

Computer vision

- Here, we will introduce image augmentation and fine tuning methods

Computer vision

- Here, we will introduce image augmentation and fine tuning methods
- Then, we will explore various methods of object detection.

Computer vision

- Here, we will introduce image augmentation and fine tuning methods
- Then, we will explore various methods of object detection.
- After that, we will learn how to use fully convolutional networks to perform semantic segmentation on images.

Computer vision

- Here, we will introduce image augmentation and fine tuning methods
- Then, we will explore various methods of object detection.
- After that, we will learn how to use fully convolutional networks to perform semantic segmentation on images.
- Then, we explain how to use style transfer technology

Image Augmentation

- We mentioned that large-scale data sets are prerequisites for the successful application of deep neural networks

Image Augmentation

- We mentioned that large-scale data sets are prerequisites for the successful application of deep neural networks
- Image augmentation technology expands the scale of training data sets by making a series of random changes to the training images to produce similar, but different, training examples.

Image Augmentation

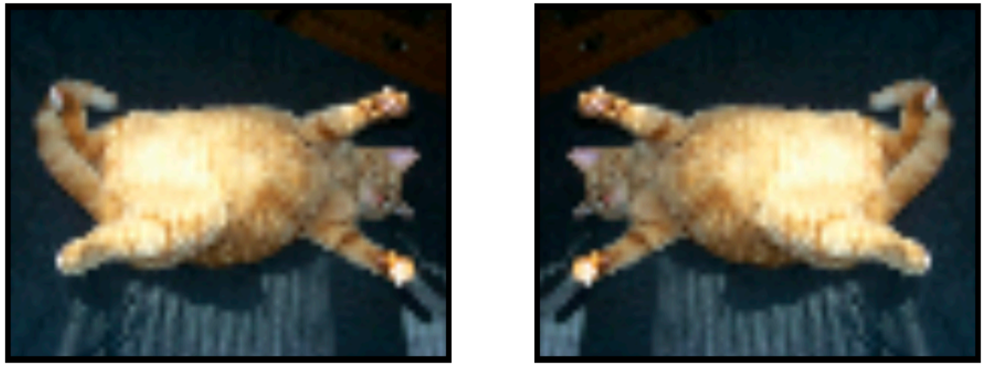
- We mentioned that large-scale data sets are prerequisites for the successful application of deep neural networks
- Image augmentation technology expands the scale of training data sets by making a series of random changes to the training images to produce similar, but different, training examples.
- Another way to explain image augmentation is that randomly changing training examples can reduce a model’s dependence on certain properties, thereby improving its capability for generalization.

Image Augmentation

- We mentioned that large-scale data sets are prerequisites for the successful application of deep neural networks
- Image augmentation technology expands the scale of training data sets by making a series of random changes to the training images to produce similar, but different, training examples.
- Another way to explain image augmentation is that randomly changing training examples can reduce a model’s dependence on certain properties, thereby improving its capability for generalization.
- It can be said that image augmentation technology contributed greatly to the success of AlexNet.

Common Image Augmentation Methods

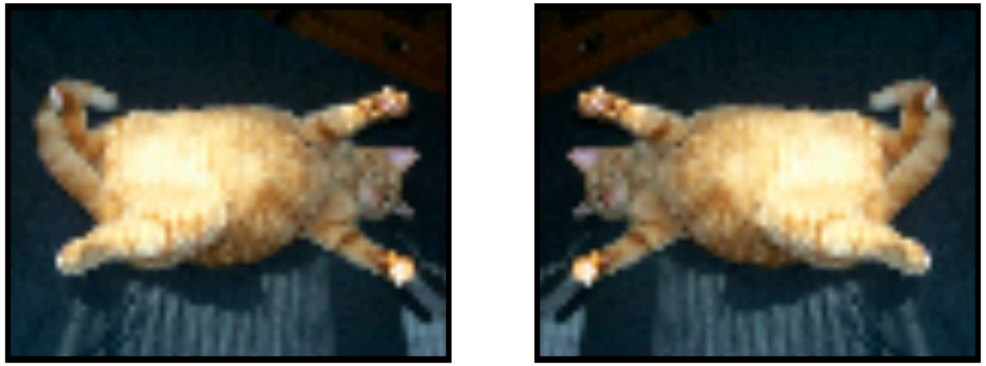
Flip



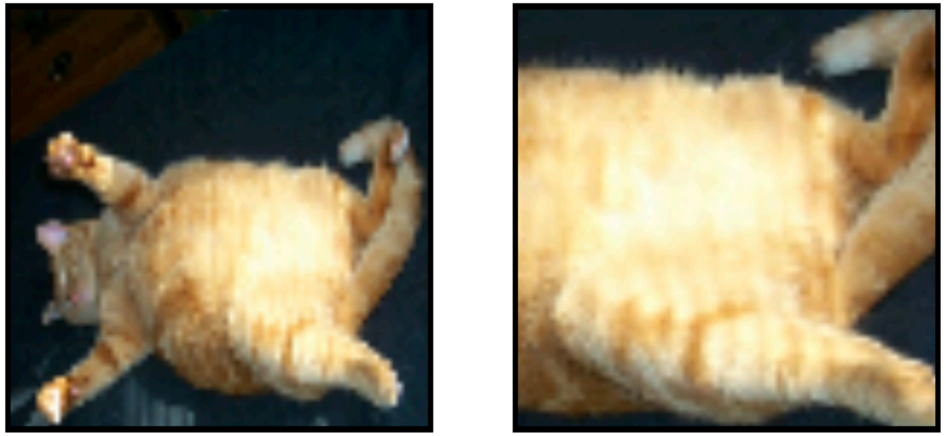
Crop

Common Image Augmentation Methods

Flip

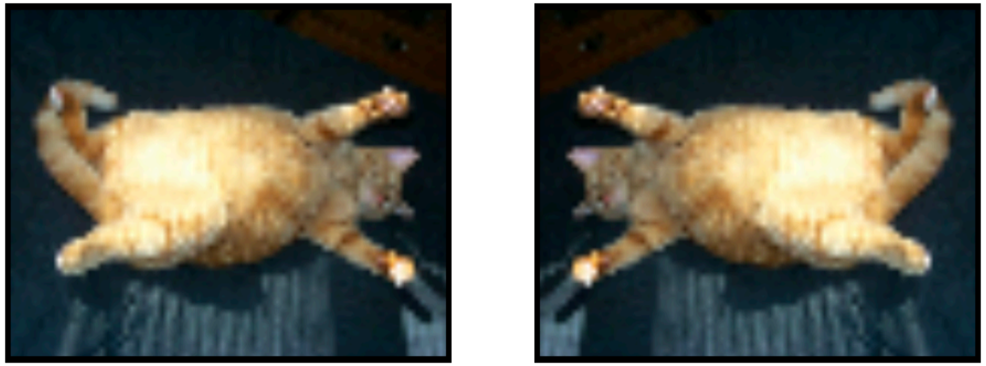


Crop

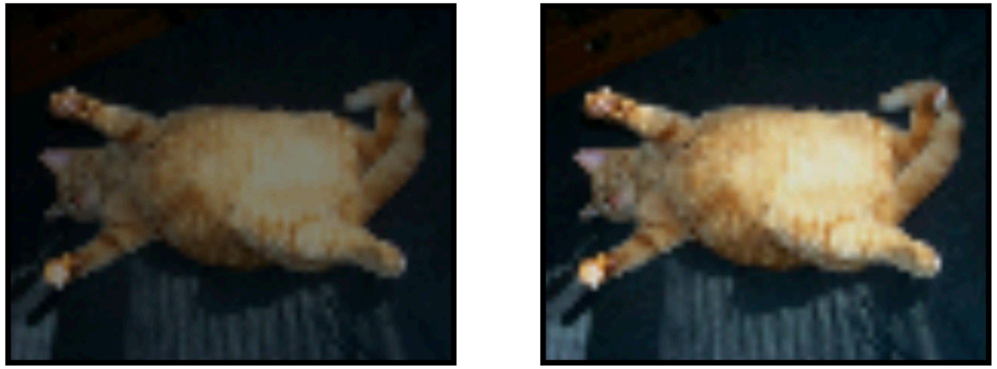


Common Image Augmentation Methods

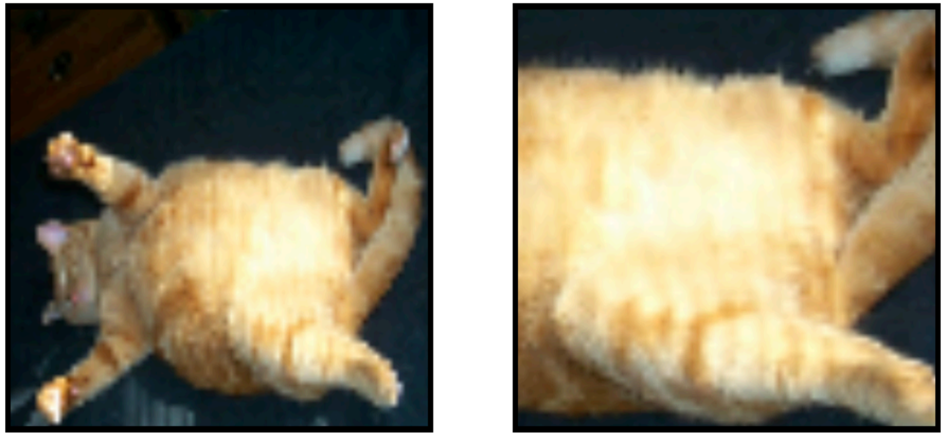
Flip



Change Color

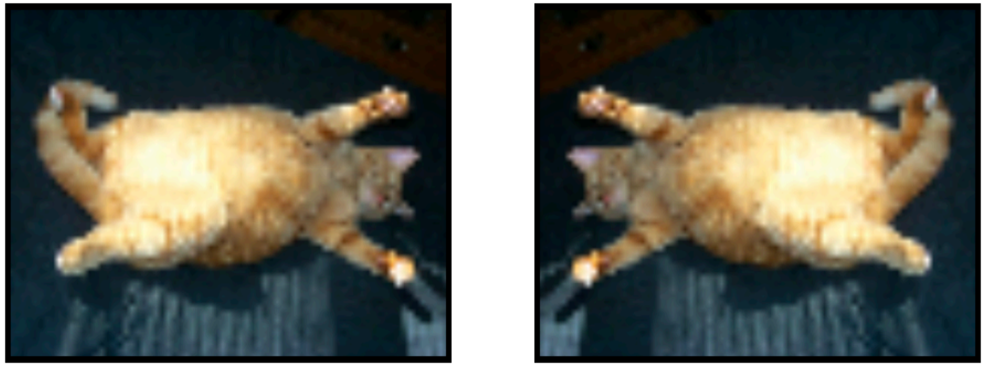


Crop

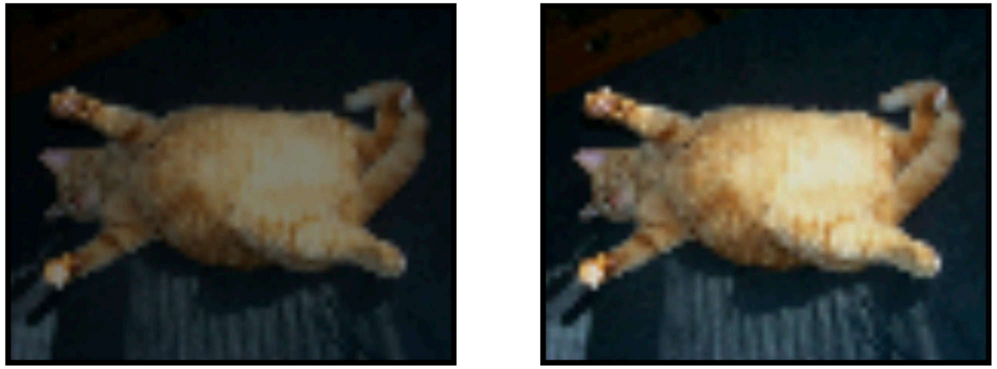


Common Image Augmentation Methods

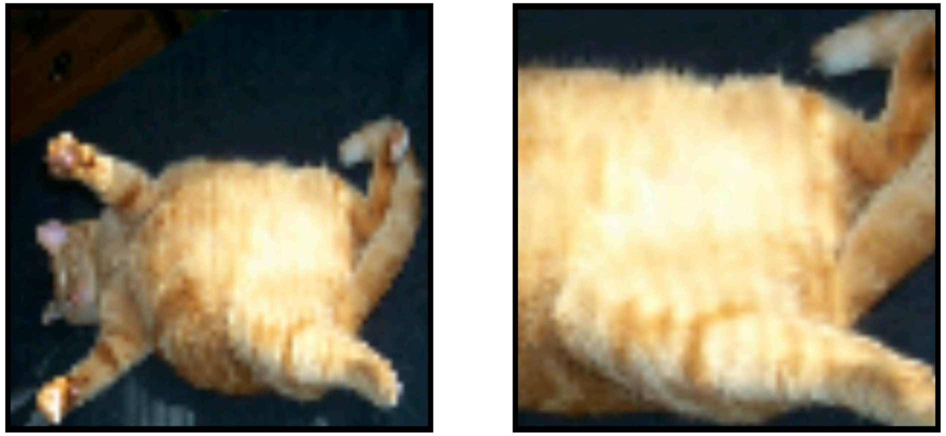
Flip



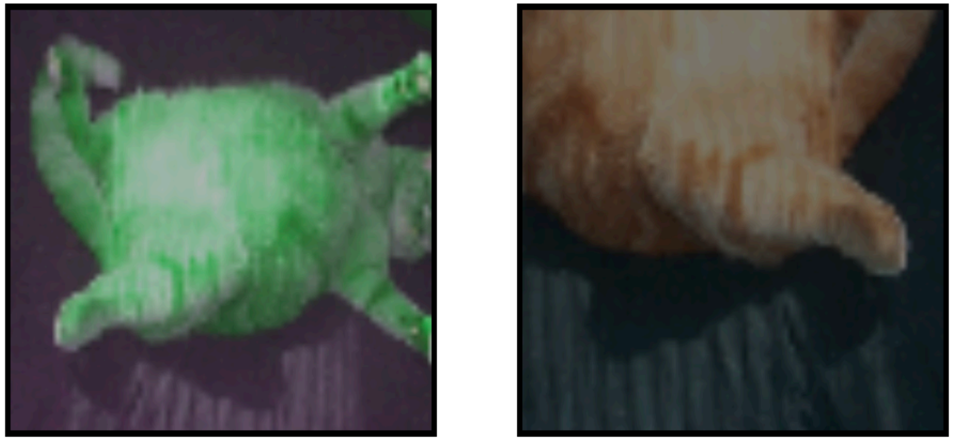
Change Color



Crop



Overlaying multiple methods



Fine Tuning

- Assume that we want to identify different kinds of chairs in images.

Fine Tuning

- Assume that we want to identify different kinds of chairs in images.
- One possible method is to first find a hundred common chairs, take one thousand different images with different angles for each chair, and then train a classification model on the collected image data set.

Fine Tuning

- Assume that we want to identify different kinds of chairs in images.
- One possible method is to first find a hundred common chairs, take one thousand different images with different angles for each chair, and then train a classification model on the collected image data set.
- Although this data set may be larger than Fashion-MNIST, the number of examples is still less than ImageNet.

Fine Tuning

- Assume that we want to identify different kinds of chairs in images.
- One possible method is to first find a hundred common chairs, take one thousand different images with different angles for each chair, and then train a classification model on the collected image data set.
- Although this data set may be larger than Fashion-MNIST, the number of examples is still less than ImageNet.
- This may result in the overfitting of the complicated model applicable to ImageNet on this data set.

Fine Tuning

- Assume that we want to identify different kinds of chairs in images.
- One possible method is to first find a hundred common chairs, take one thousand different images with different angles for each chair, and then train a classification model on the collected image data set.
- Although this data set may be larger than Fashion-MNIST, the number of examples is still less than ImageNet.
- This may result in the overfitting of the complicated model applicable to ImageNet on this data set.
- In order to deal with the above problems, an obvious solution is to collect more data.

Fine Tuning

- Assume that we want to identify different kinds of chairs in images.
- One possible method is to first find a hundred common chairs, take one thousand different images with different angles for each chair, and then train a classification model on the collected image data set.
- Although this data set may be larger than Fashion-MNIST, the number of examples is still less than ImageNet.
- This may result in the overfitting of the complicated model applicable to ImageNet on this data set.
- In order to deal with the above problems, an obvious solution is to collect more data.
- However, collecting and labeling data can consume a lot of time and money.

Fine Tuning

- Another solution is to apply transfer learning to migrate the knowledge learned from the source data set to the target data set.

Fine Tuning

- Another solution is to apply transfer learning to migrate the knowledge learned from the source data set to the target data set.
- For example, although the images in ImageNet are mostly unrelated to chairs, models trained on this data set can extract more general image features that can help identify edges, textures, shapes, and object composition.

Fine Tuning

- Another solution is to apply transfer learning to migrate the knowledge learned from the source data set to the target data set.
- For example, although the images in ImageNet are mostly unrelated to chairs, models trained on this data set can extract more general image features that can help identify edges, textures, shapes, and object composition.
- These similar features may be equally effective for recognizing a chair.

Fine Tuning

- Another solution is to apply transfer learning to migrate the knowledge learned from the source data set to the target data set.
- For example, although the images in ImageNet are mostly unrelated to chairs, models trained on this data set can extract more general image features that can help identify edges, textures, shapes, and object composition.
- These similar features may be equally effective for recognizing a chair.
- Now, we introduce a common technique in **transfer learning:** **fine tuning**.

Fine Tuning

- Fine tuning consists of the following four steps:

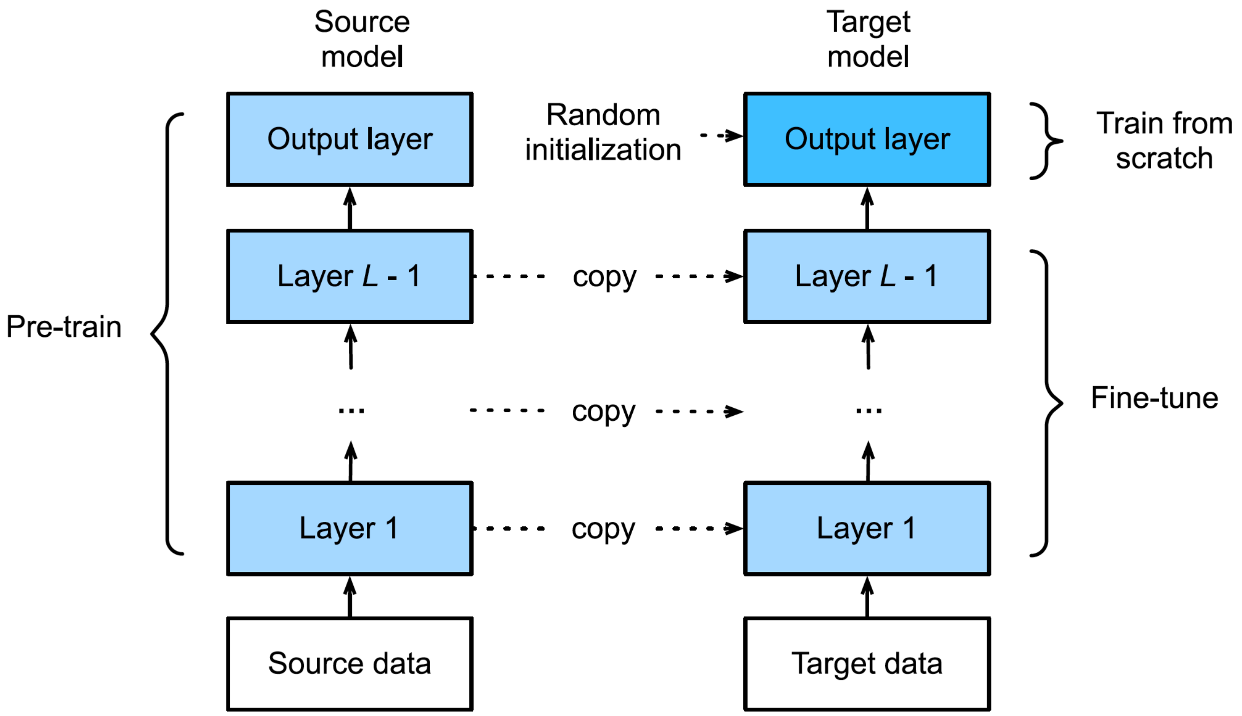


Fig. 14.2.1: Fine tuning.

Fine Tuning

- Fine tuning consists of the following four steps:

1. Pre-train a neural network model, i.e., the source model, on a source data set (e.g., the ImageNet dataset).

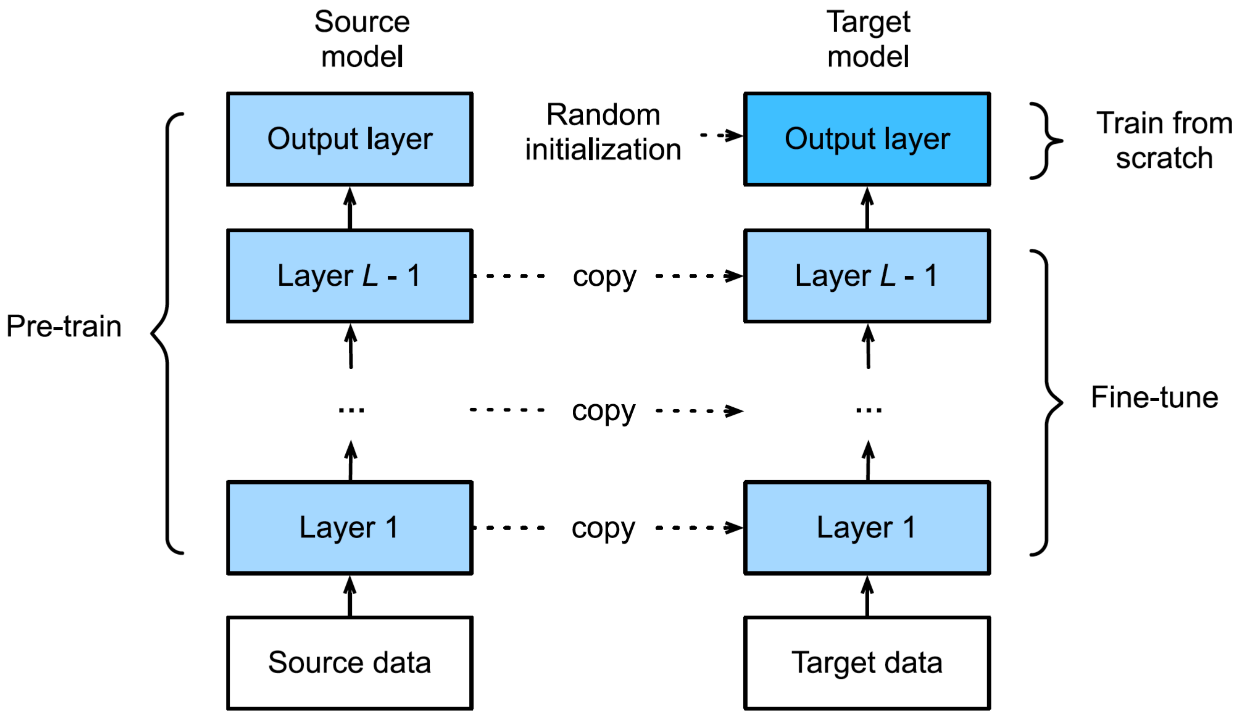


Fig. 14.2.1: Fine tuning.

Fine Tuning

- Fine tuning consists of the following four steps:

1. Pre-train a neural network model, i.e., the source model, on a source data set (e.g., the ImageNet dataset).
2. Create a new neural network model, i.e., the target model. This replicates all model designs and their parameters on the source model, except the output layer.

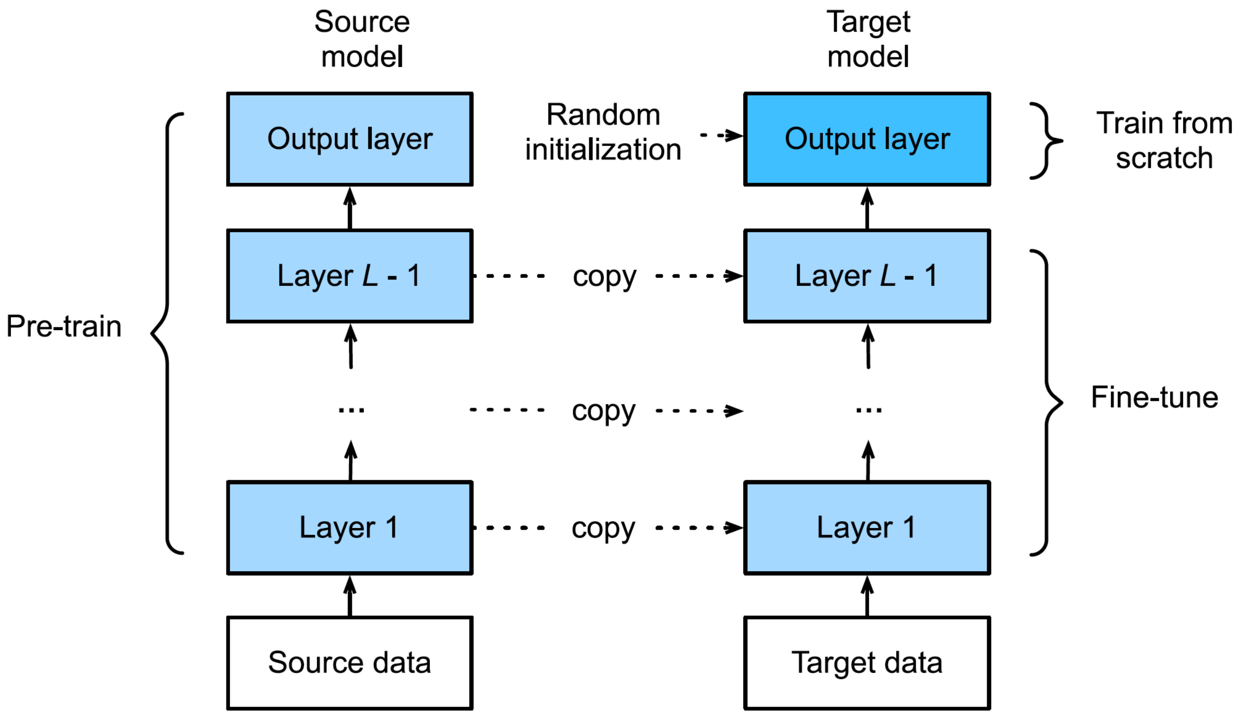


Fig. 14.2.1: Fine tuning.

Fine Tuning

- Fine tuning consists of the following four steps:

1. Pre-train a neural network model, i.e., the source model, on a source data set (e.g., the ImageNet dataset).
2. Create a new neural network model, i.e., the target model. This replicates all model designs and their parameters on the source model, except the output layer.
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.

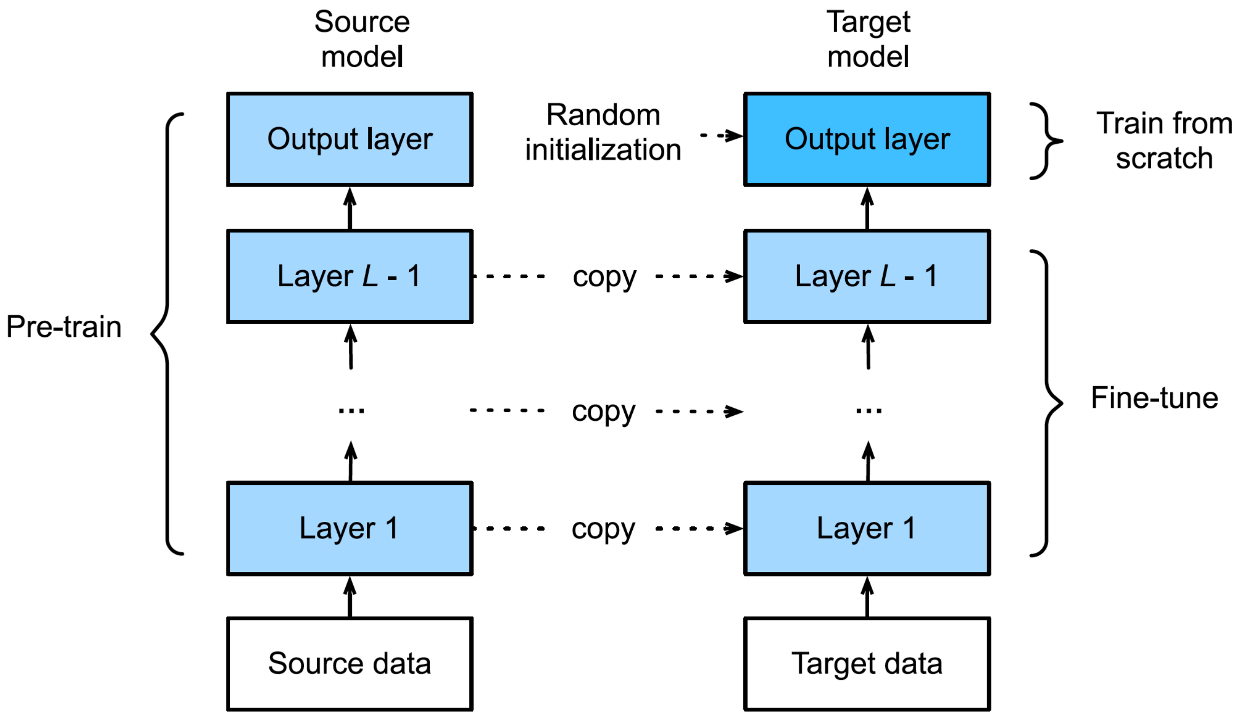


Fig. 14.2.1: Fine tuning.

Fine Tuning

- Fine tuning consists of the following four steps:

1. Pre-train a neural network model, i.e., the source model, on a source data set (e.g., the ImageNet dataset).
2. Create a new neural network model, i.e., the target model. This replicates all model designs and their parameters on the source model, except the output layer.
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, such as a chair data set.

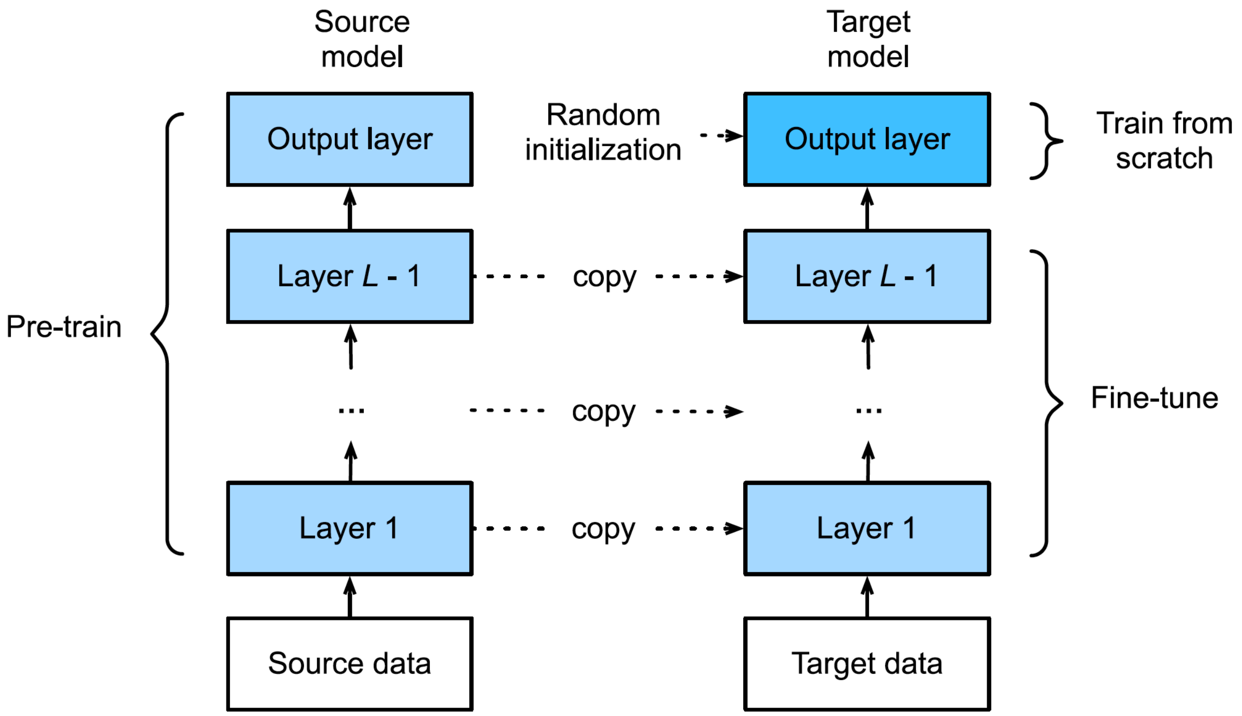


Fig. 14.2.1: Fine tuning.

Object Detection and Bounding Boxes

- In image classification tasks, we assume that there is only one main target in the image and we only focus on how to identify the target category.

Object Detection and Bounding Boxes

- In image classification tasks, we assume that there is only one main target in the image and we only focus on how to identify the target category.
- However, in many situations, there are multiple targets in the image that we are interested in.

Object Detection and Bounding Boxes

- In image classification tasks, we assume that there is only one main target in the image and we only focus on how to identify the target category.
- However, in many situations, there are multiple targets in the image that we are interested in.
- We not only want to classify them, but also want to obtain their specific positions in the image.

Object Detection and Bounding Boxes

- In image classification tasks, we assume that there is only one main target in the image and we only focus on how to identify the target category.
- However, in many situations, there are multiple targets in the image that we are interested in.
- We not only want to classify them, but also want to obtain their specific positions in the image.
- In computer vision, we refer to such tasks as object detection (or object recognition).

Object Detection and Bounding Boxes

- In image classification tasks, we assume that there is only one main target in the image and we only focus on how to identify the target category.
- However, in many situations, there are multiple targets in the image that we are interested in.
- We not only want to classify them, but also want to obtain their specific positions in the image.
- In computer vision, we refer to such tasks as object detection (or object recognition).
- Object detection is widely used in many fields

Object Detection and Bounding Boxes

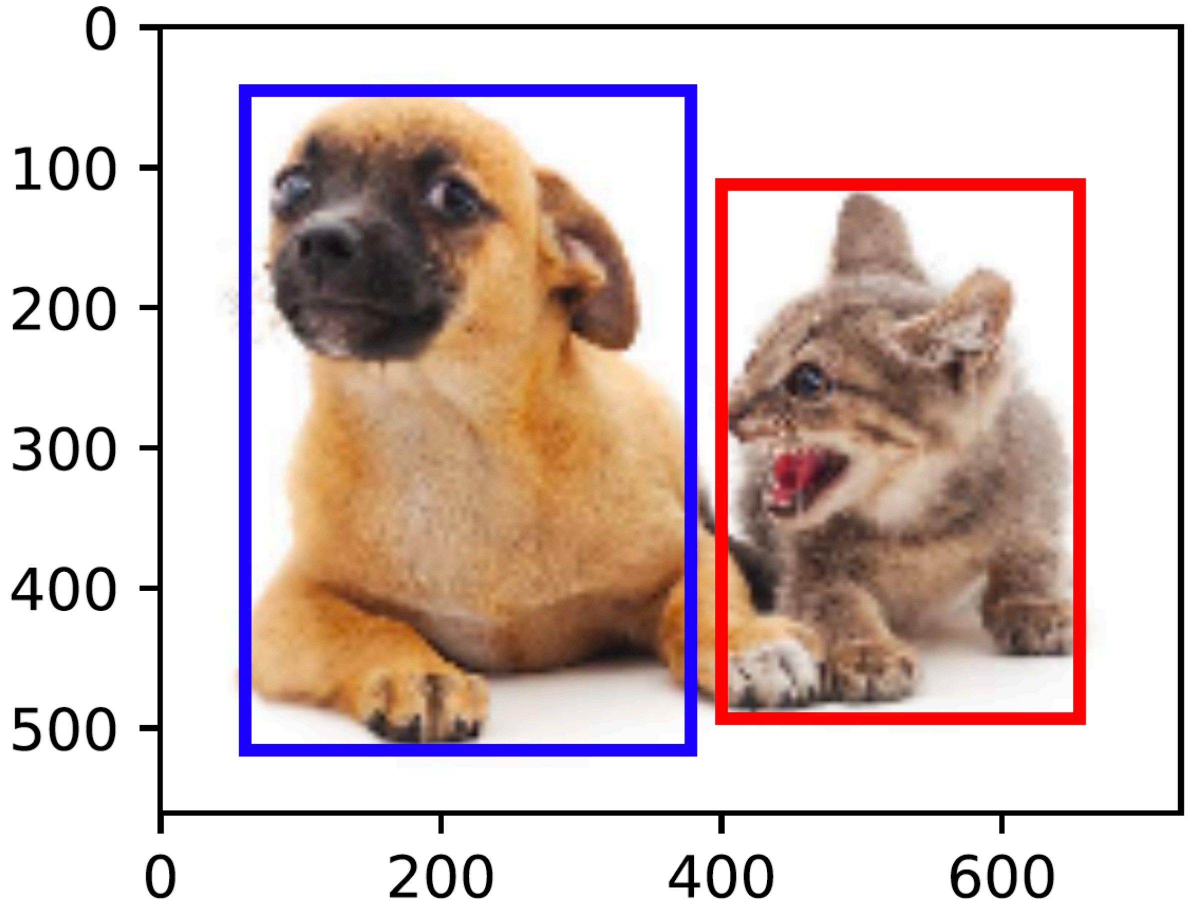
- In image classification tasks, we assume that there is only one main target in the image and we only focus on how to identify the target category.
- However, in many situations, there are multiple targets in the image that we are interested in.
- We not only want to classify them, but also want to obtain their specific positions in the image.
- In computer vision, we refer to such tasks as object detection (or object recognition).
- Object detection is widely used in many fields
  - in self-driving technology, we need to plan routes by identifying the locations of vehicles, pedestrians, roads, and obstacles in the captured video image.

Object Detection and Bounding Boxes

- In image classification tasks, we assume that there is only one main target in the image and we only focus on how to identify the target category.
- However, in many situations, there are multiple targets in the image that we are interested in.
- We not only want to classify them, but also want to obtain their specific positions in the image.
- In computer vision, we refer to such tasks as object detection (or object recognition).
- Object detection is widely used in many fields
    - in self-driving technology, we need to plan routes by identifying the locations of vehicles, pedestrians, roads, and obstacles in the captured video image.
    - Systems in the security field need to detect abnormal targets, such as intruders or bombs.

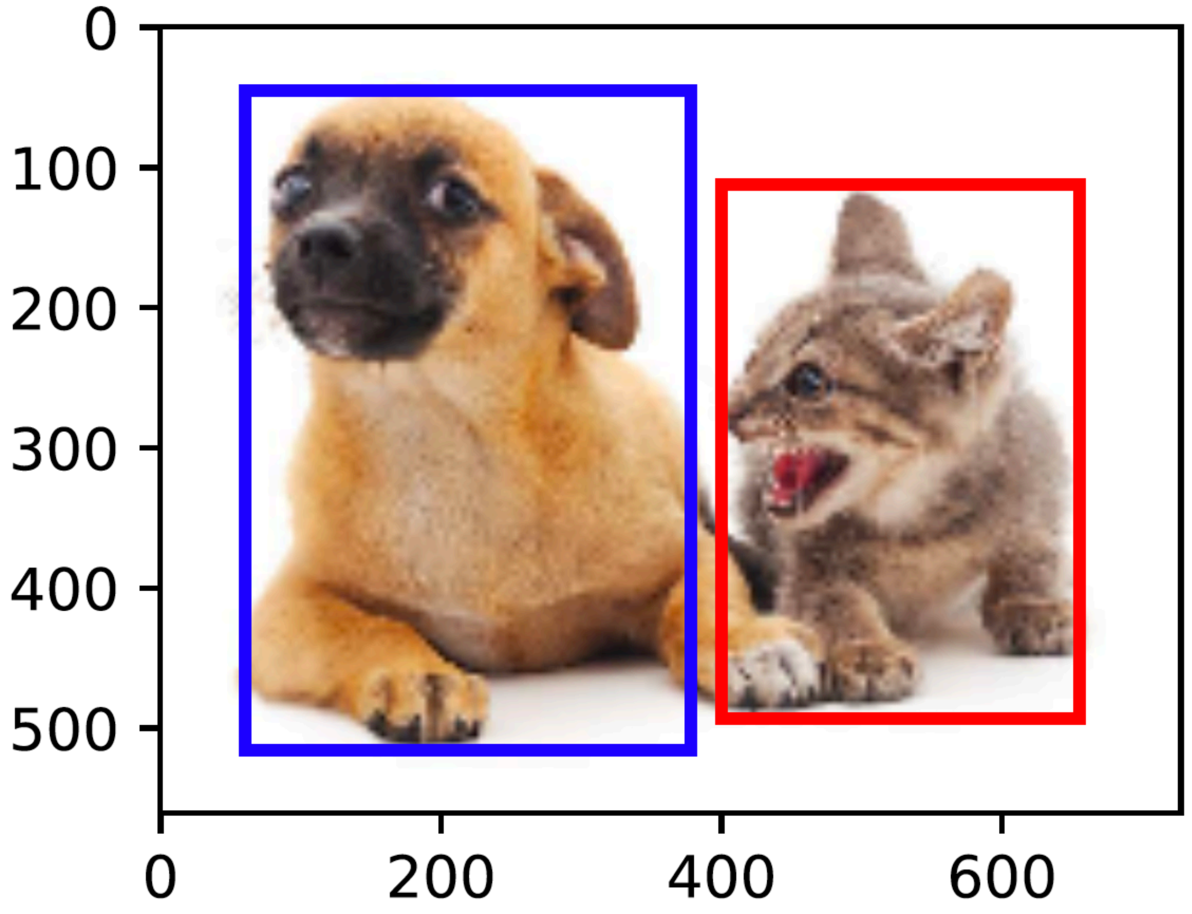
Bounding Boxes

- In object detection, we usually use a bounding box to describe the target location.



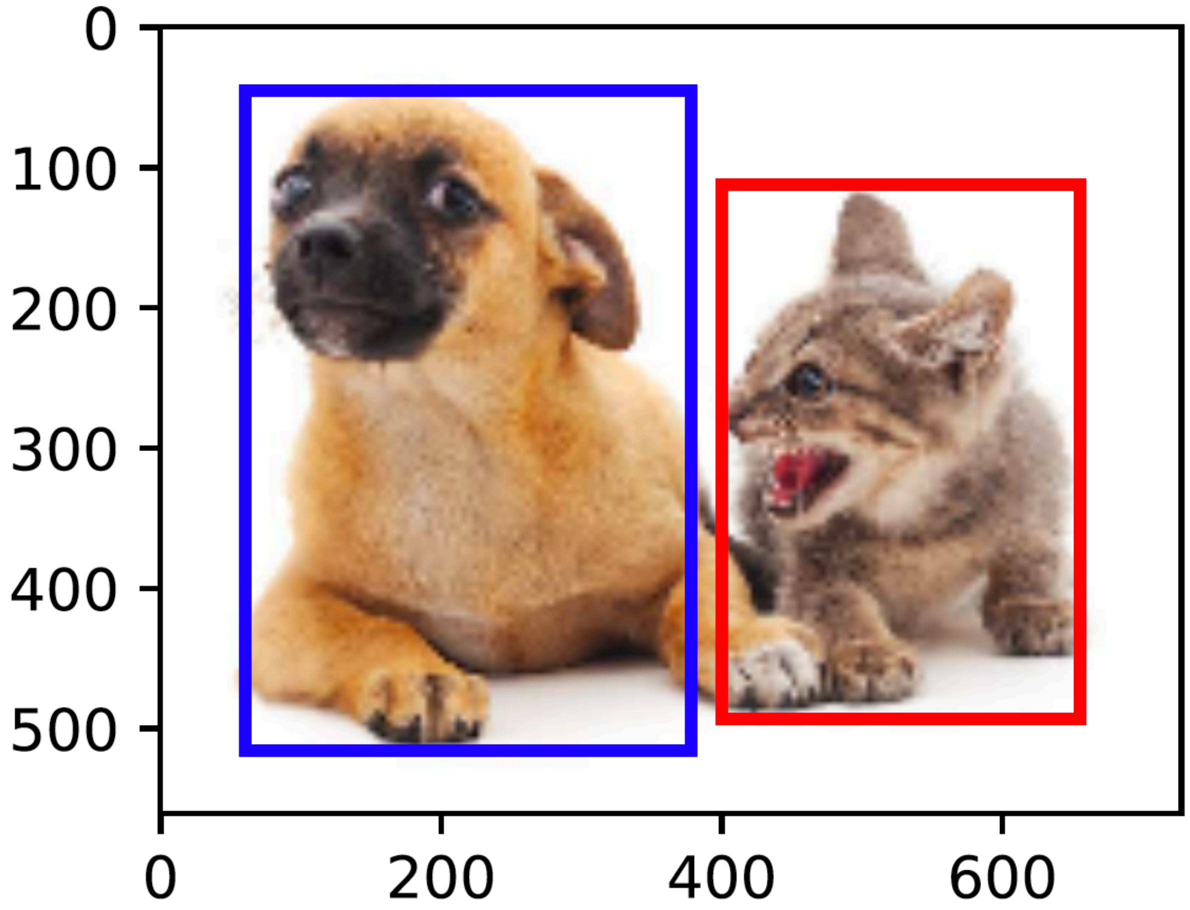
Bounding Boxes

- In object detection, we usually use a bounding box to describe the target location.
- 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.



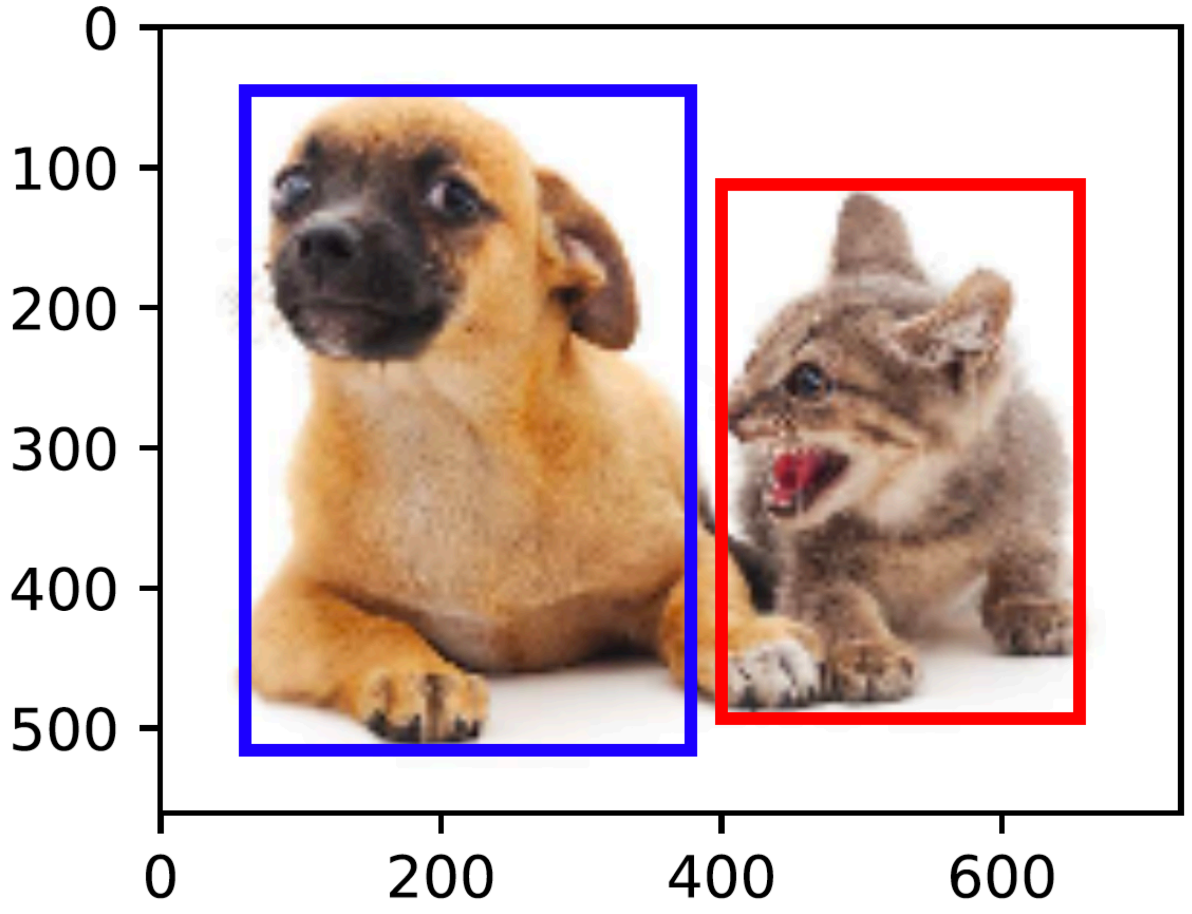
Bounding Boxes

- In object detection, we usually use a bounding box to describe the target location.
- 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 in the above image 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.



Bounding Boxes

- In object detection, we usually use a bounding box to describe the target location.
- 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 in the above image 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.




```python
# bbox is the abbreviation for bounding box
dog_bbox, cat_bbox = [60, 45, 378, 516], [400, 112, 655, 493]
```

Anchor Boxes

- Object detection algorithms usually sample a large number of regions in the input image, determine whether these regions contain objects of interest, and adjust the edges of the regions so as to predict the ground-truth bounding box of the target more accurately.

Anchor Boxes

- Object detection algorithms usually sample a large number of regions in the input image, determine whether these regions contain objects of interest, and adjust the edges of the regions so as to predict the ground-truth bounding box of the target more accurately.
- Different models may use different region sampling methods.

Anchor Boxes

- Object detection algorithms usually sample a large number of regions in the input image, determine whether these regions contain objects of interest, and adjust the edges of the regions so as to predict the ground-truth bounding box of the target more accurately.
- Different models may use different region sampling methods.
- Here, we introduce one such method: it generates multiple bounding boxes with different sizes and aspect ratios while centering on each pixel.

Anchor Boxes

- Object detection algorithms usually sample a large number of regions in the input image, determine whether these regions contain objects of interest, and adjust the edges of the regions so as to predict the ground-truth bounding box of the target more accurately.
- Different models may use different region sampling methods.
- Here, we introduce one such method: it generates multiple bounding boxes with different sizes and aspect ratios while centering on each pixel.
- These bounding boxes are called **anchor boxes**.

Generate Multiple Anchor Boxes

- Assume the input image has a height of h and width of w.

Generate Multiple Anchor Boxes

- Assume the input image has a height of h and width of w.
- We generate anchor boxes with different shapes centered on each pixel of the image.

Generate Multiple Anchor Boxes

- Assume the input image has a height of h and width of w.
- We generate anchor boxes with different shapes centered on each pixel of the image.
- Assume the size is $s \in (0, 1]$, the aspect ratio is $r > 0$, and the width and

height of the anchor box are $ws/\sqrt{r}$ and $\mathrm{h}s/\sqrt{r}$, respectively

Generate Multiple Anchor Boxes

- Assume the input image has a height of h and width of w.
- We generate anchor boxes with different shapes centered on each pixel of the image.
- Assume the size is $s \in (0, 1]$, the aspect ratio is r > 0, and the width and

height of the anchor box are $ws/\sqrt{r}$ and $\mathrm{h}s/\sqrt{r}$, respectively

- When the center position is given, an anchor box with known width and height is determined.

Generate Multiple Anchor Boxes

- Assume the input image has a height of h and width of w.
- We generate anchor boxes with different shapes centered on each pixel of the image.
- Assume the size is $s \in (0, 1]$, the aspect ratio is $r > 0$, and the width and

height of the anchor box are $ws/\sqrt{r}$ and $\mathrm{h}s/\sqrt{r}$, respectively

- When the center position is given, an anchor box with known width and height is determined.
- Let's set a set of sizes $s_1, \ldots, s_n$ and a set of aspect ratios $r_1, \ldots, r_m$. If we use a combination of all sizes and aspect ratios with each pixel as the center, the input image will have a total of $whmn$ anchor boxes.

Generate Multiple Anchor Boxes

- Assume the input image has a height of h and width of w.
- We generate anchor boxes with different shapes centered on each pixel of the image.
- Assume the size is $s \in (0, 1]$, the aspect ratio is $r > 0$, and the width and

height of the anchor box are $ws/\sqrt{r}$ and $\mathrm{h}s/\sqrt{r}$, respectively

- When the center position is given, an anchor box with known width and height is determined.
- Let's set a set of sizes $s_1, \ldots, s_n$ and a set of aspect ratios $r_1, \ldots, r_m$. If we use a combination of all sizes and aspect ratios with each pixel as the center, the input image will have a total of $whmn$ anchor boxes.
- Although these anchor boxes may cover all ground-truth bounding boxes, the computational complexity is often excessive.

Generate Multiple Anchor Boxes

- Therefore, we are usually only interested in a combination containing s_1 or r_1 sizes and aspect ratios, that is:

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

Generate Multiple Anchor Boxes

- Therefore, we are usually only interested in a combination containing s_1 or r_1 sizes and aspect ratios, that is:

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

- That is, the number of anchor boxes centered on the same pixel is $n + m - 1$. For the entire input image, we will generate a total of $wh(n + m - 1)$ anchor boxes.

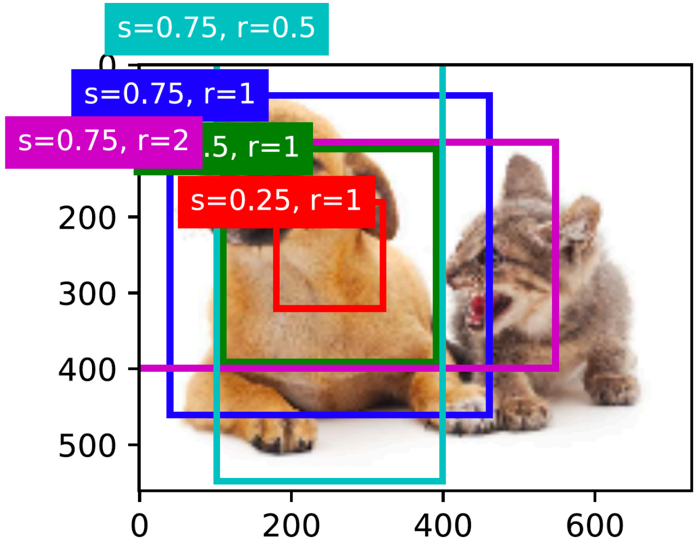
Generate Multiple Anchor Boxes

- Therefore, we are usually only interested in a combination containing s_1 or r_1 sizes and aspect ratios, that is:

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

- That is, the number of anchor boxes centered on the same pixel is $n + m - 1$. For the entire input image, we will generate a total of $wh(n + m - 1)$ anchor boxes.

```
Y = contrib.nd.MultiBoxPrior(X, sizes=[0.75, 0.5, 0.25], ratios=[1, 2, 0.5])
```



Intersection over Union

- How to measure how well the anchor box covers the target?

Intersection over Union

- How to measure how well the anchor box covers the target?
- We measure the similarity between anchor boxes and the ground-truth bounding box.

Intersection over Union

- How to measure how well the anchor box covers the target?
- We measure the similarity between anchor boxes and the ground-truth bounding box.
- Given set A and B, their Jaccard index is the size of their intersection divided by the size of their union

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

Intersection over Union

- How to measure how well the anchor box covers the target?
- We measure the similarity between anchor boxes and the ground-truth bounding box.
- Given set A and B, their Jaccard index is the size of their intersection divided by the size of their union

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

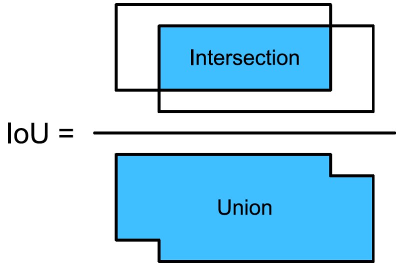
- In fact, we can consider the pixel area of a bounding box as a collection of pixels.

Intersection over Union

- How to measure how well the anchor box covers the target?
- We measure the similarity between anchor boxes and the ground-truth bounding box.
- Given set A and B, their Jaccard index is the size of their intersection divided by the size of their union

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

- In fact, we can consider the pixel area of a bounding box as a collection of pixels.
- When we measure the similarity of two bounding boxes, we usually refer the Jaccard index as intersection over union (IoU)



Labeling Training Set Anchor Boxes

- In the training set, we consider each anchor box as a training example.

Labeling Training Set Anchor Boxes

- In the training set, we consider each anchor box as a training example.
- In order to train the object detection model, we need to mark two types of labels for each anchor box:

Labeling Training Set Anchor Boxes

- In the training set, we consider each anchor box as a training example.
- In order to train the object detection model, we need to mark two types of labels for each anchor box:
  - first, the category of the target contained in the anchor box (category) and,

Labeling Training Set Anchor Boxes

- In the training set, we consider each anchor box as a training example.
- In order to train the object detection model, we need to mark two types of labels for each anchor box:
  - first, the category of the target contained in the anchor box (category) and,
  - second, the offset of the ground-truth bounding box relative to the anchor box (offset).

Labeling Training Set Anchor Boxes

- In the training set, we consider each anchor box as a training example.
- In order to train the object detection model, we need to mark two types of labels for each anchor box:
  - first, the category of the target contained in the anchor box (category) and,
  - second, the offset of the ground-truth bounding box relative to the anchor box (offset).
- In the training set, each image is labelled with the location of the ground truth bounding box and the category of the target contained.

Labeling Training Set Anchor Boxes

- In the training set, we consider each anchor box as a training example.
- In order to train the object detection model, we need to mark two types of labels for each anchor box:
  - first, the category of the target contained in the anchor box (category) and,
  - second, the offset of the ground-truth bounding box relative to the anchor box (offset).
- In the training set, each image is labelled with the location of the ground truth bounding box and the category of the target contained.
- After the anchor boxes are generated, we primarily label anchor boxes based on the location and category information of the ground-truth bounding boxes similar to the anchor boxes.

Labeling Training Set Anchor Boxes

- In the training set, we consider each anchor box as a training example.
- In order to train the object detection model, we need to mark two types of labels for each anchor box:
  - first, the category of the target contained in the anchor box (category) and,
  - second, the offset of the ground-truth bounding box relative to the anchor box (offset).
- In the training set, each image is labelled with the location of the ground truth bounding box and the category of the target contained.
- After the anchor boxes are generated, we primarily label anchor boxes based on the location and category information of the ground-truth bounding boxes similar to the anchor boxes.
- So how do we assign ground-truth bounding boxes to anchor boxes similar to them?

Labeling Training Set Anchor Boxes

- Assume the anchor boxes in the image are $A_1, \ldots, A_{n_a}$

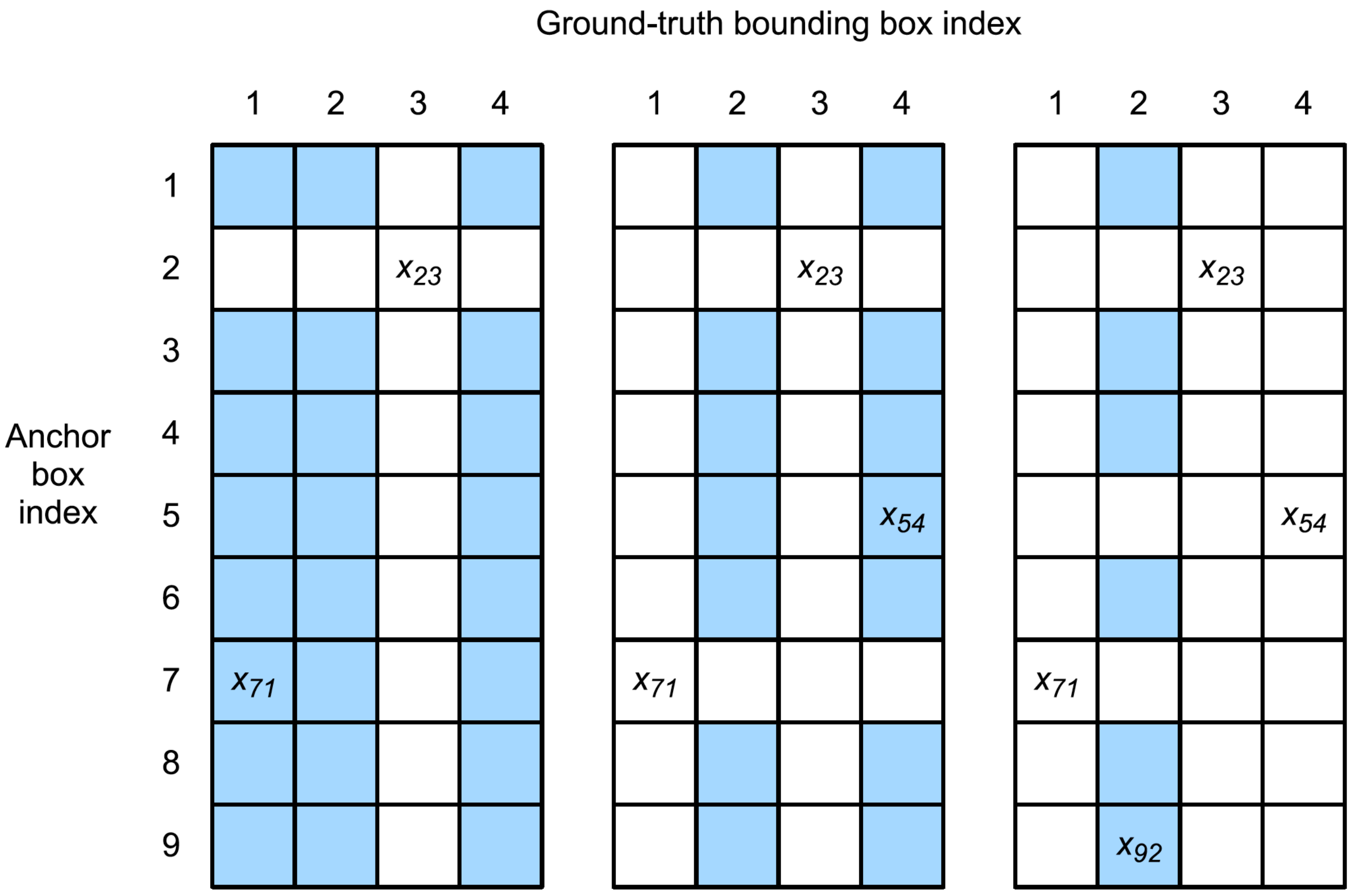


Fig. 14.4.2: Assign ground-truth bounding boxes to anchor boxes.

Labeling Training Set Anchor Boxes

- Assume the anchor boxes in the image are $A_1, \ldots, A_{n_a}$
- And the ground truth bounding boxes are $B_1, \ldots, B_{n_b}$

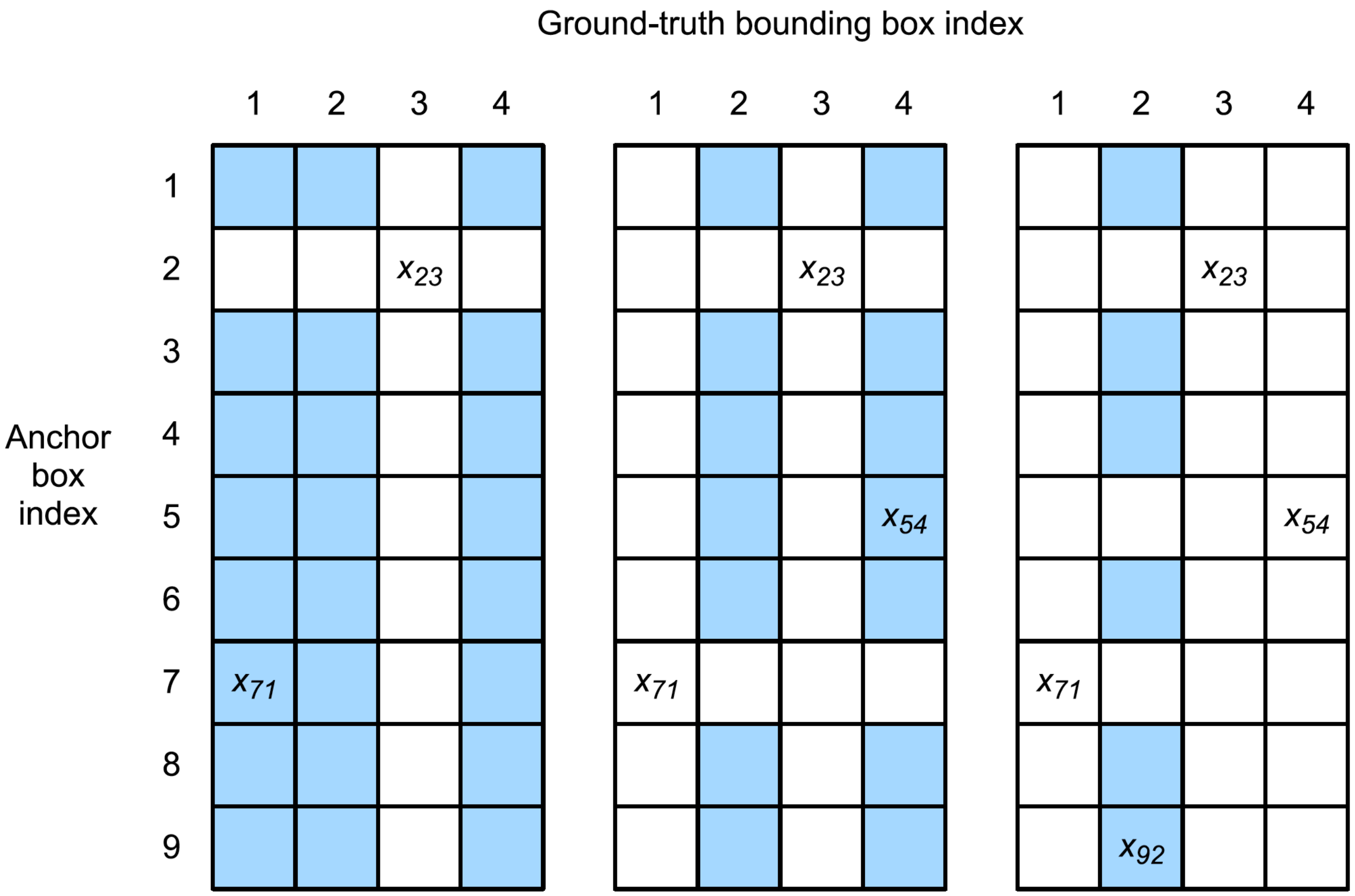


Fig. 14.4.2: Assign ground-truth bounding boxes to anchor boxes.

Labeling Training Set Anchor Boxes

- Assume the anchor boxes in the image are $A_1, \ldots, A_{n_a}$
- And the ground truth bounding boxes are $B_1, \ldots, B_{n_b}$
- And $n_a \geq n_b$

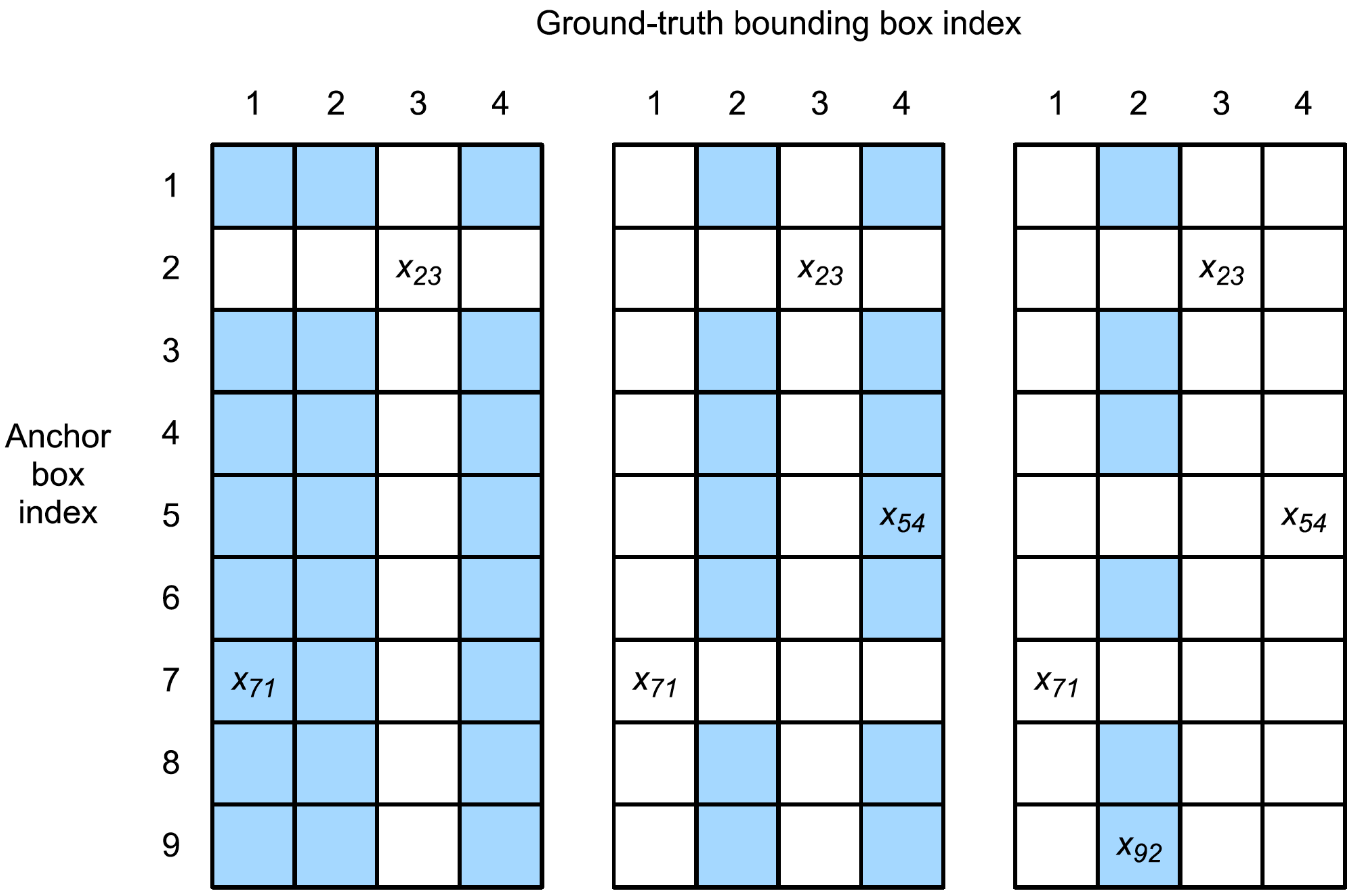


Fig. 14.4.2: Assign ground-truth bounding boxes to anchor boxes.

Labeling Training Set Anchor Boxes

- Assume the anchor boxes in the image are $A_1, \dots, A_{n_a}$
- And the ground truth bounding boxes are $B_1, \dots, B_{n_b}$
- And $n_a \geq n_b$
- Define a matrix $\mathbf{X} \in R^{n_a \times n_b}$, where element x_{ij} is the IoU of the anchor box A_i to the ground-truth bounding box B_j

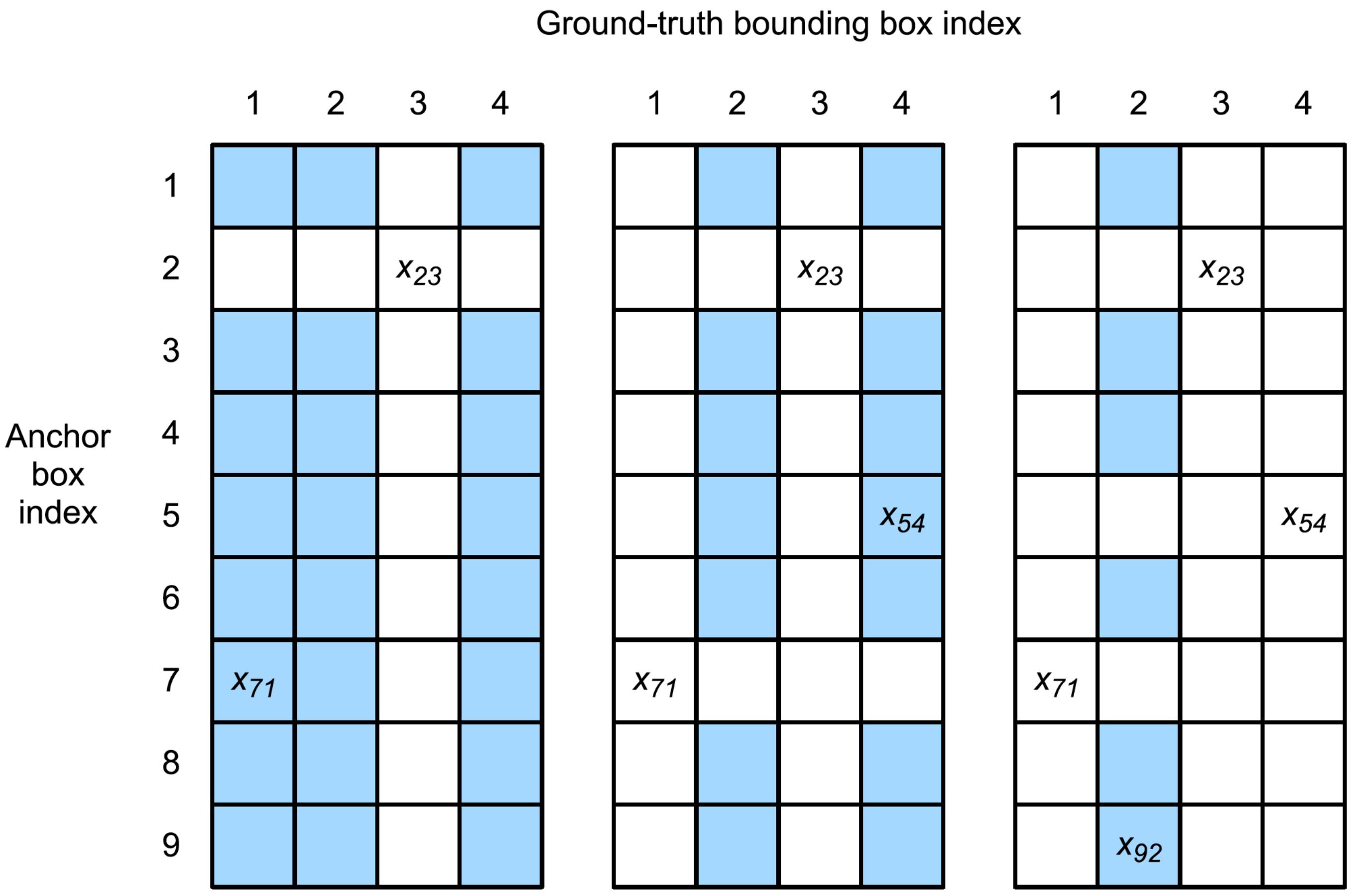
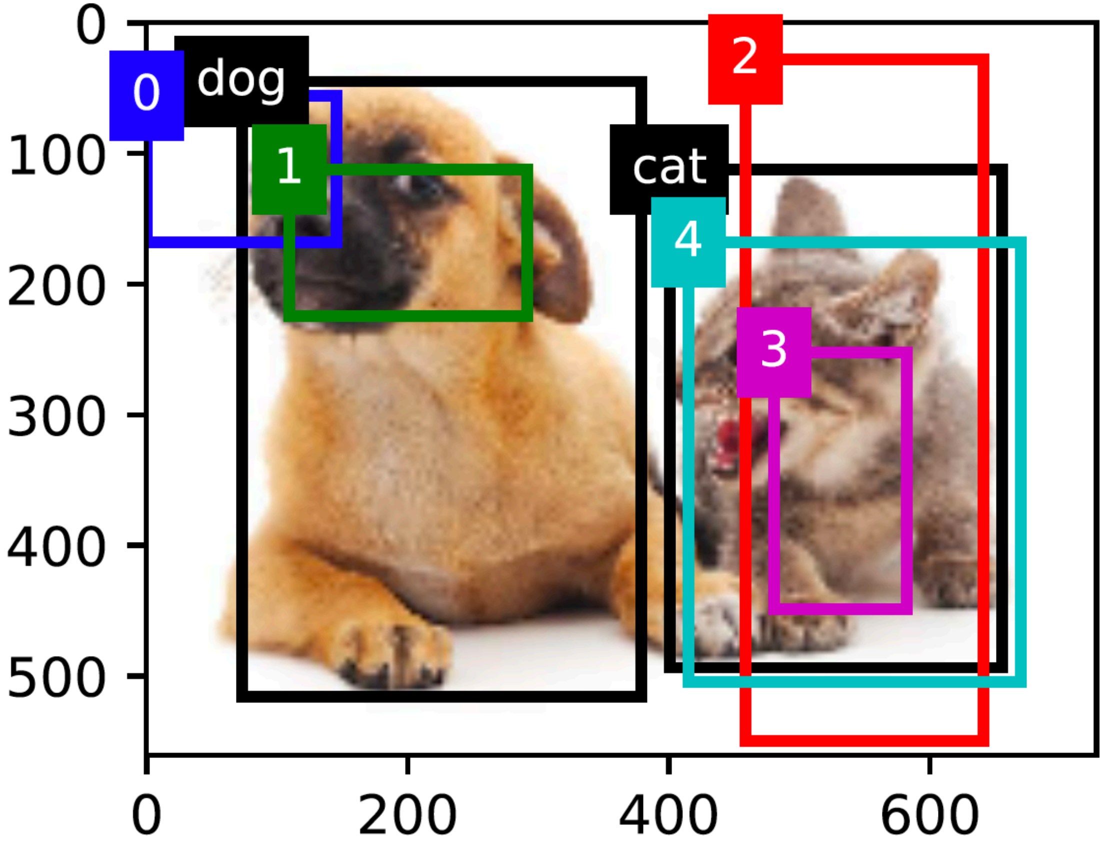


Fig. 14.4.2: Assign ground-truth bounding boxes to anchor boxes.

Labeling Training Set Anchor Boxes

1: dog
2: cat



[[0. 1. 2. 0. 2.]]

Labeling Training Set Anchor Boxes

- 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.

Labeling Training Set Anchor Boxes

- 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.
- Because the positions and sizes of various boxes in the data set may vary, these relative positions and relative sizes usually require some special transformations to make the offset distribution more uniform and easier to fit.

Labeling Training Set Anchor Boxes

- 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.
- Because the positions and sizes of various boxes in the data set may vary, these relative positions and relative sizes usually require some special transformations to make the offset distribution more uniform and easier to fit.
- Assume the center coordinates of anchor box A and its assigned ground-truth bounding box B are $(x_a, y_a), (x_b, y_b)$ the widths of A and B are w_a, w_b, and their heights are h_a, h_b, respectively.

Labeling Training Set Anchor Boxes

- 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.
- Because the positions and sizes of various boxes in the data set may vary, these relative positions and relative sizes usually require some special transformations to make the offset distribution more uniform and easier to fit.
- Assume the center coordinates of anchor box A and its assigned ground-truth bounding box B are $(x_a, y_a), (x_b, y_b)$ the widths of A and B are w_a, w_b, and their heights are h_a, h_b, respectively.
- A common technique is to label the offset of A as

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

Output Bounding Boxes for Prediction

- During model prediction, we first generate multiple anchor boxes for the image and then predict categories and offsets for these anchor boxes one by one.

Output Bounding Boxes for Prediction

- During model prediction, we first generate multiple anchor boxes for the image and then predict categories and offsets for these anchor boxes one by one.
- Then, we obtain prediction bounding boxes based on anchor boxes and their predicted offsets.

Output Bounding Boxes for Prediction

- During model prediction, we first generate multiple anchor boxes for the image and then predict categories and offsets for these anchor boxes one by one.
- Then, we obtain prediction bounding boxes based on anchor boxes and their predicted offsets.
- When there are many anchor boxes, many similar prediction bounding boxes may be output for the same target.

Output Bounding Boxes for Prediction

- During model prediction, we first generate multiple anchor boxes for the image and then predict categories and offsets for these anchor boxes one by one.
- Then, we obtain prediction bounding boxes based on anchor boxes and their predicted offsets.
- When there are many anchor boxes, many similar prediction bounding boxes may be output for the same target.
- To simplify the results, we can remove similar prediction bounding boxes.

Output Bounding Boxes for Prediction

- During model prediction, we first generate multiple anchor boxes for the image and then predict categories and offsets for these anchor boxes one by one.
- Then, we obtain prediction bounding boxes based on anchor boxes and their predicted offsets.
- When there are many anchor boxes, many similar prediction bounding boxes may be output for the same target.
- To simplify the results, we can remove similar prediction bounding boxes.
- A commonly used method is called non-maximum suppression (NMS).

Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.

Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.
- For a prediction bounding box B, the model calculates the predicted probability for each category.

Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.
- For a prediction bounding box B, the model calculates the predicted probability for each category.
- Assume the largest predicted category is p, the category corresponding to this probability is the predicted category of B.

Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.
- For a prediction bounding box B, the model calculates the predicted probability for each category.
- Assume the largest predicted category is p, the category corresponding to this probability is the predicted category of B.
- We also refer to p as the confidence level of prediction bounding box B.

Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.
- For a prediction bounding box B, the model calculates the predicted probability for each category.
- Assume the largest predicted category is p, the category corresponding to this probability is the predicted category of B.
- We also refer to p as the confidence level of prediction bounding box B.
- On the same image, we sort the prediction bounding boxes with predicted categories other than background by confidence level from high to low, and obtain the list L.

Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.
- For a prediction bounding box B, the model calculates the predicted probability for each category.
- Assume the largest predicted category is p, the category corresponding to this probability is the predicted category of B.
- We also refer to p as the confidence level of prediction bounding box B.
- On the same image, we sort the prediction bounding boxes with predicted categories other than background by confidence level from high to low, and obtain the list L.
- Select the prediction bounding box B_1 with highest confidence level from L as a baseline and remove all non-benchmark prediction bounding boxes with an IoU with B_1 greater than a certain threshold from L.

Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.
- For a prediction bounding box B, the model calculates the predicted probability for each category.
- Assume the largest predicted category is p, the category corresponding to this probability is the predicted category of B.
- We also refer to p as the confidence level of prediction bounding box B.
- On the same image, we sort the prediction bounding boxes with predicted categories other than background by confidence level from high to low, and obtain the list L.
- Select the prediction bounding box B_1 with highest confidence level from L as a baseline and remove all non-benchmark prediction bounding boxes with an IoU with B_1 greater than a certain threshold from L.
- At this point, L retains the prediction bounding box with the highest confidence level and removes other prediction boxes similar to it.

Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.
- For a prediction bounding box B, the model calculates the predicted probability for each category.
- Assume the largest predicted category is p, the category corresponding to this probability is the predicted category of B.
- We also refer to p as the confidence level of prediction bounding box B.
- On the same image, we sort the prediction bounding boxes with predicted categories other than background by confidence level from high to low, and obtain the list L.
- Select the prediction bounding box B_1 with highest confidence level from L as a baseline and remove all non-benchmark prediction bounding boxes with an IoU with B_1 greater than a certain threshold from L.
- At this point, L retains the prediction bounding box with the highest confidence level and removes other prediction boxes similar to it.
- Next, we select the prediction bounding box B_2 with the second highest confidence level from L as a baseline, and remove all non-benchmark prediction bounding boxes with an IoU with B_2 greater than a certain threshold from L.

Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.
- For a prediction bounding box B, the model calculates the predicted probability for each category.
- Assume the largest predicted category is p, the category corresponding to this probability is the predicted category of B.
- We also refer to p as the confidence level of prediction bounding box B.
- On the same image, we sort the prediction bounding boxes with predicted categories other than background by confidence level from high to low, and obtain the list L.
- Select the prediction bounding box B_1 with highest confidence level from L as a baseline and remove all non-benchmark prediction bounding boxes with an IoU with B_1 greater than a certain threshold from L.
- At this point, L retains the prediction bounding box with the highest confidence level and removes other prediction boxes similar to it.
- Next, we select the prediction bounding box B_2 with the second highest confidence level from L as a baselines, and remove all non-benchmark prediction bounding boxes with an IoU with B_2 greater than a certain threshold from L.
- Repeat this process until all prediction bounding boxes in L have been used as a baseline,

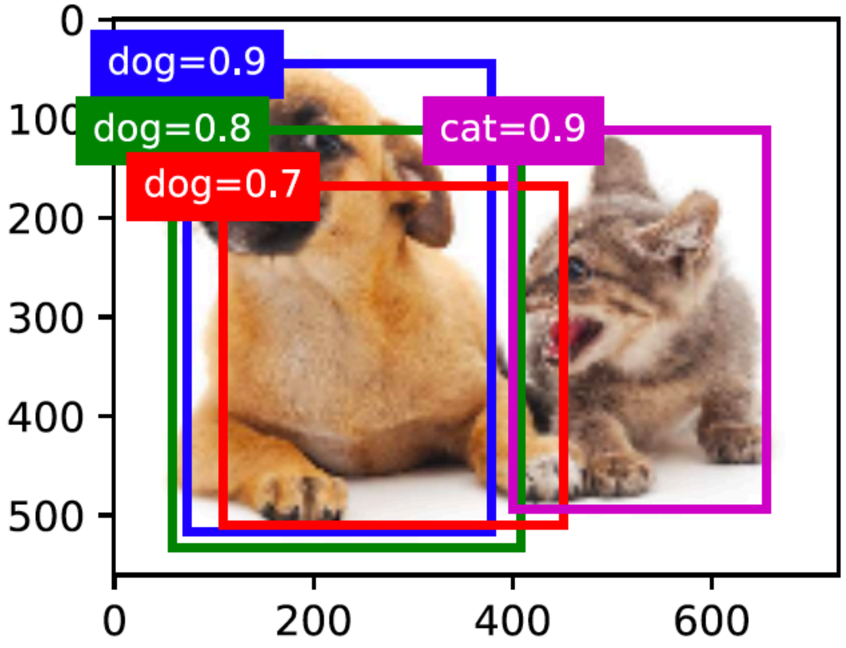
Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.
- For a prediction bounding box B, the model calculates the predicted probability for each category.
- Assume the largest predicted category is p, the category corresponding to this probability is the predicted category of B.
- We also refer to p as the confidence level of prediction bounding box B.
- On the same image, we sort the prediction bounding boxes with predicted categories other than background by confidence level from high to low, and obtain the list L.
- Select the prediction bounding box B_1 with highest confidence level from L as a baseline and remove all non-benchmark prediction bounding boxes with an IoU with B_1 greater than a certain threshold from L.
- At this point, L retains the prediction bounding box with the highest confidence level and removes other prediction boxes similar to it.
- Next, we select the prediction bounding box B_2 with the second highest confidence level from L as a baselines, and remove all non-benchmark prediction bounding boxes with an IoU with B_2 greater than a certain threshold from L.
- Repeat this process until all prediction bounding boxes in L have been used as a baseline,
- At this time, the IoU of any pair of prediction bounding boxes in L is less than the threshold.

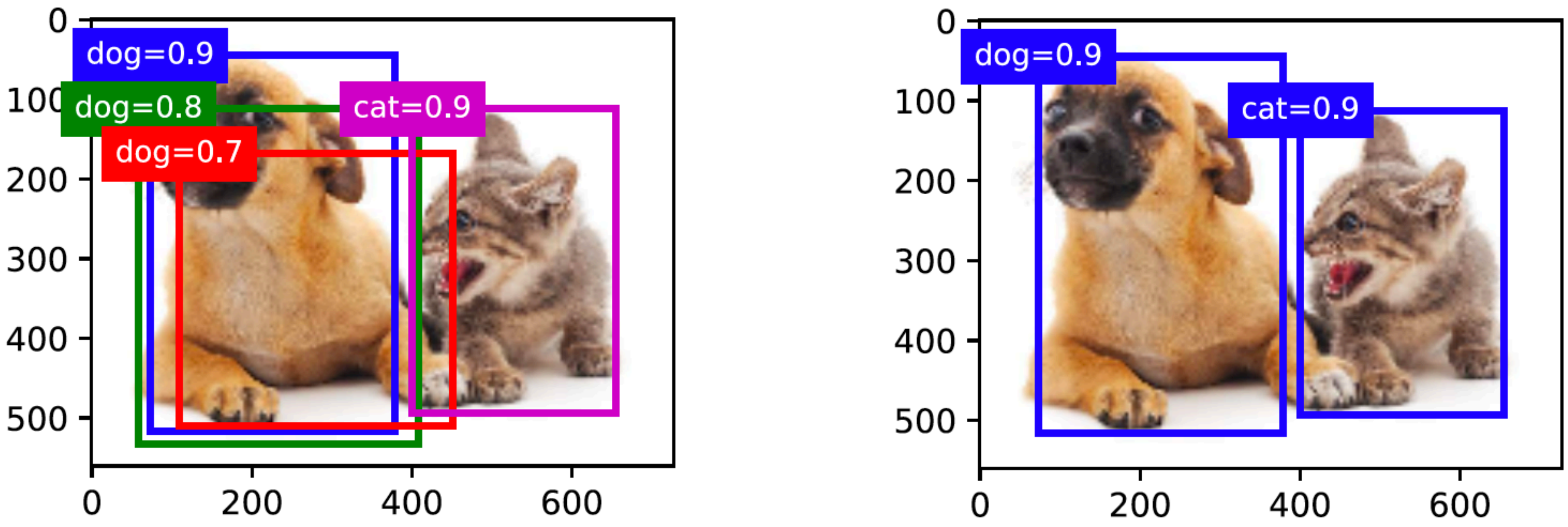
Output Bounding Boxes for Prediction

- Let us take a look at how NMS works.
- For a prediction bounding box B, the model calculates the predicted probability for each category.
- Assume the largest predicted category is p, the category corresponding to this probability is the predicted category of B.
- We also refer to p as the confidence level of prediction bounding box B.
- On the same image, we sort the prediction bounding boxes with predicted categories other than background by confidence level from high to low, and obtain the list L.
- Select the prediction bounding box B_1 with highest confidence level from L as a baseline and remove all non-benchmark prediction bounding boxes with an IoU with B_1 greater than a certain threshold from L.
- At this point, L retains the prediction bounding box with the highest confidence level and removes other prediction boxes similar to it.
- Next, we select the prediction bounding box B_2 with the second highest confidence level from L as a baselines, and remove all non-benchmark prediction bounding boxes with an IoU with B_2 greater than a certain threshold from L.
- Repeat this process until all prediction bounding boxes in L have been used as a baseline,
- At this time, the IoU of any pair of prediction bounding boxes in L is less than the threshold.
- Finally, output all prediction bounding boxes in the list L

Output Bounding Boxes for Prediction



Output Bounding Boxes for Prediction



Multiscale Object Detection

- So far, we generated multiple anchor boxes centered on each pixel of the input image.

Multiscale Object Detection

- So far, we generated multiple anchor boxes centered on each pixel of the input image.
- These anchor boxes are used to sample different regions of the input image.

Multiscale Object Detection

- So far, we generated multiple anchor boxes centered on each pixel of the input image.
- These anchor boxes are used to sample different regions of the input image.
- However, if anchor boxes are generated centered on each pixel of the image, soon there will be too many anchor boxes to compute.

Multiscale Object Detection

- So far, we generated multiple anchor boxes centered on each pixel of the input image.
- These anchor boxes are used to sample different regions of the input image.
- However, if anchor boxes are generated centered on each pixel of the image, soon there will be too many anchor boxes to compute.
- It is not difficult to reduce the number of anchor boxes.

Multiscale Object Detection

- So far, we generated multiple anchor boxes centered on each pixel of the input image.
- These anchor boxes are used to sample different regions of the input image.
- However, if anchor boxes are generated centered on each pixel of the image, soon there will be too many anchor boxes to compute.
- It is not difficult to reduce the number of anchor boxes.
- We can apply uniform sampling on a small portion of pixels from the input image and generate anchor boxes centered on the samples pixels.

Multiscale Object Detection

- In addition, we can generate anchor boxes of varied numbers and sizes on multiple scales.

Multiscale Object Detection

- In addition, we can generate anchor boxes of varied numbers and sizes on multiple scales.
- Notice that smaller objects are more likely to be positioned on the image rather than larger ones.

Multiscale Object Detection

- In addition, we can generate anchor boxes of varied numbers and sizes on multiple scales.
- Notice that smaller objects are more likely to be positioned on the image rather than larger ones.
- Consider an image of size 2x2:

Multiscale Object Detection

- In addition, we can generate anchor boxes of varied numbers and sizes on multiple scales.
- Notice that smaller objects are more likely to be positioned on the image rather than larger ones.
- Consider an image of size 2x2:
  - Say, there are objects with shapes of 1x1, 1x2, and 2x2

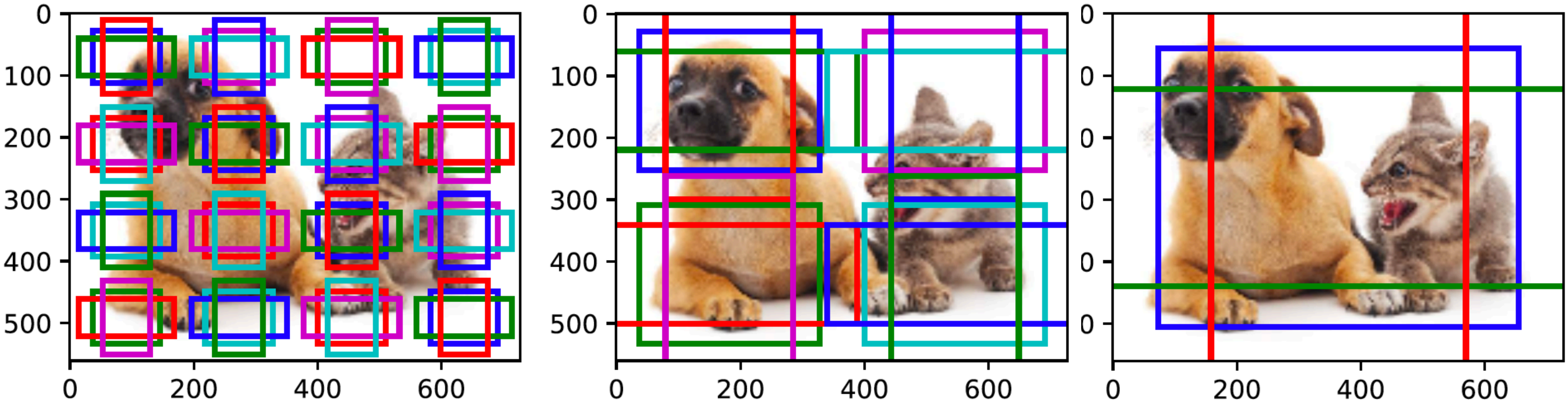
Multiscale Object Detection

- In addition, we can generate anchor boxes of varied numbers and sizes on multiple scales.
- Notice that smaller objects are more likely to be positioned on the image rather than larger ones.
- Consider an image of size 2x2:
  - Say, there are objects with shapes of 1x1, 1x2, and 2x2
  - Each of these objects 4, 2, and 1 possible positions on an image

Multiscale Object Detection

- In addition, we can generate anchor boxes of varied numbers and sizes on multiple scales.
- Notice that smaller objects are more likely to be positioned on the image rather than larger ones.
- Consider an image of size 2x2:
  - Say, there are objects with shapes of 1x1, 1x2, and 2x2
  - Each of these objects 4, 2, and 1 possible positions on an image
- Therefore, when using smaller anchor boxes to detect smaller objects, we can sample more regions; when using larger anchor boxes to detect larger objects, we cam sample fewer regions.

Multiscale object detection



Object Detection Data Set (Pikachu)

Single Shot Multibox Detection (SSD)

- Now, we will use our knowledge on bounding boxes, anchor boxes and multiscale object detection to construct an object detection model: single shot multibox detection (SSD).

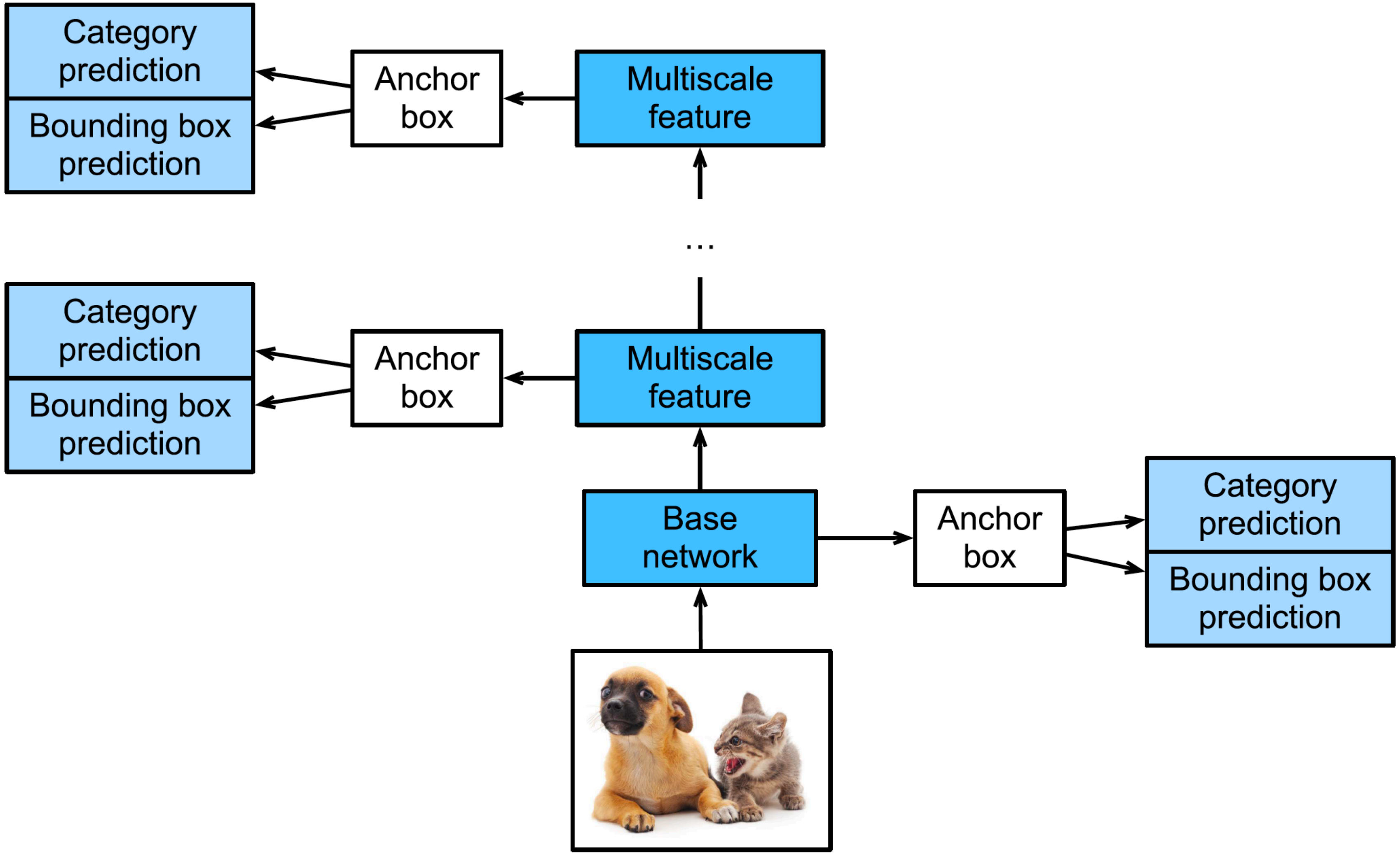
Single Shot Multibox Detection (SSD)

- Now, we will use our knowledge on bounding boxes, anchor boxes and multiscale object detection to construct an object detection model: single shot multibox detection (SSD).
- This quick and easy model is already widely used.

Single Shot Multibox Detection (SSD)

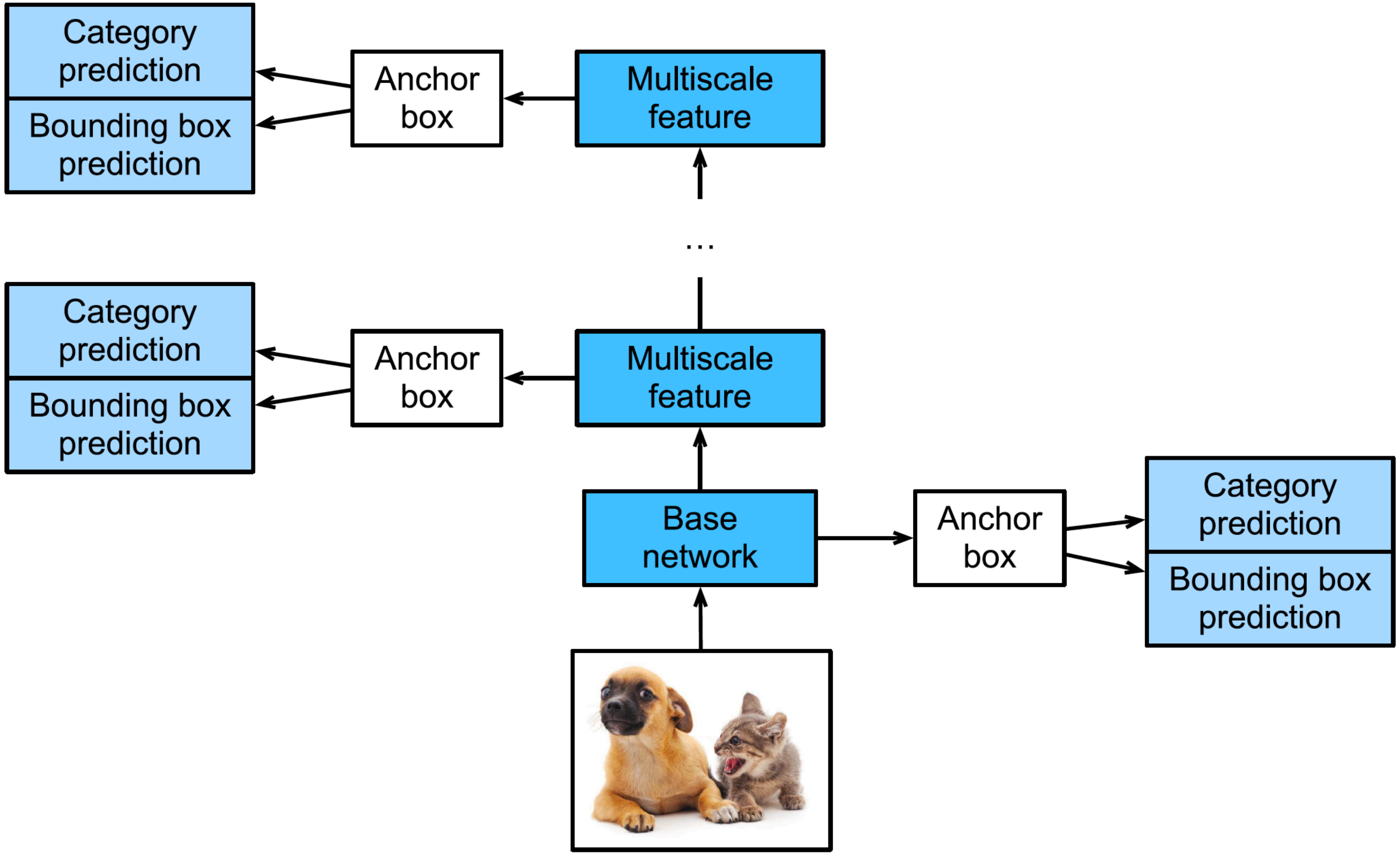
- Now, we will use our knowledge on bounding boxes, anchor boxes and multiscale object detection to construct an object detection model: single shot multibox detection (SSD).
- This quick and easy model is already widely used.
- Some of the design concepts and implementation details of this model are also applicable to other object detection models.

Single Shot Multibox Detection (SSD)



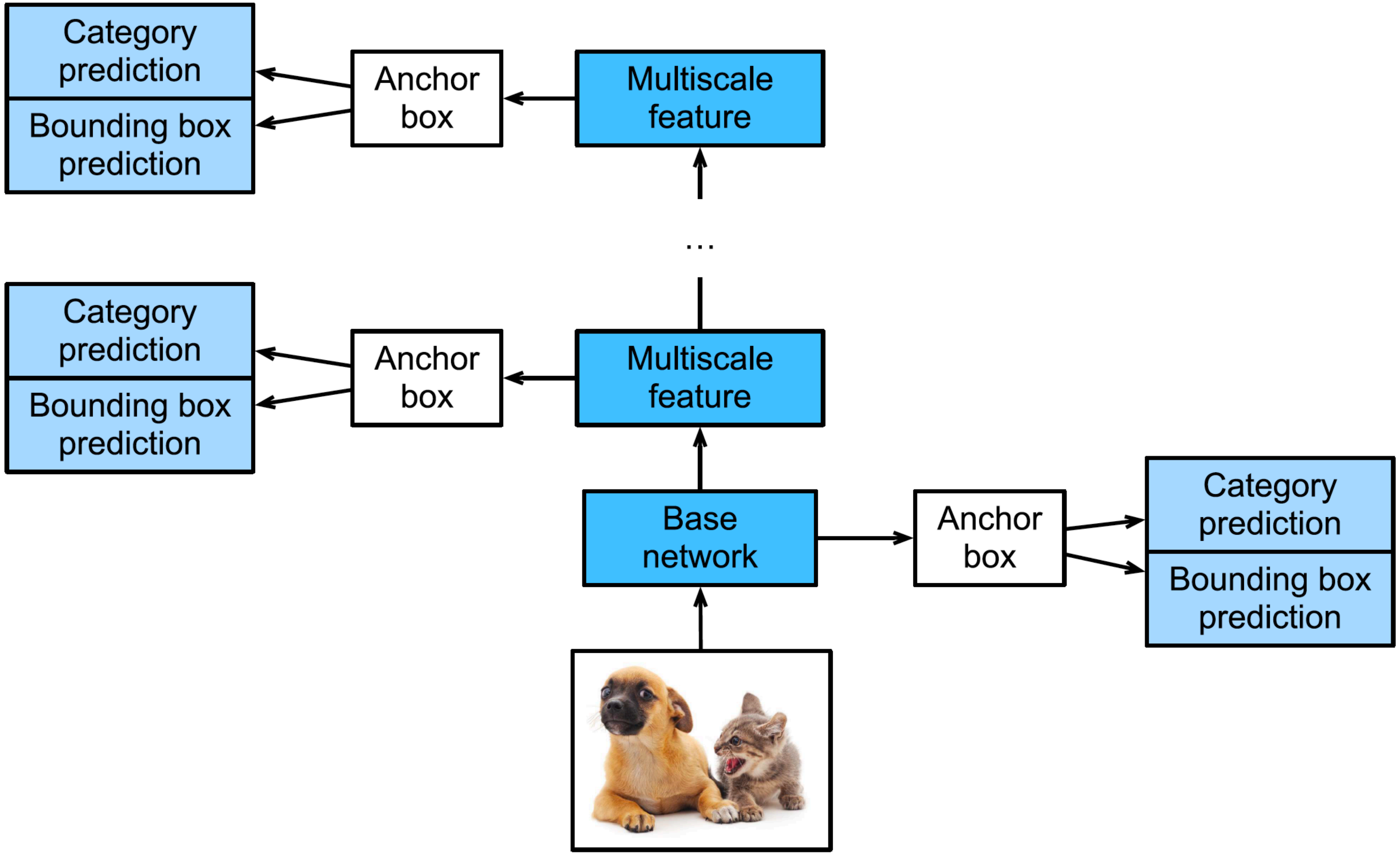
- The base network is used to extract features of original images.

Single Shot Multibox Detection (SSD)



- The base network is used to extract features of original images.
- The closer a feature block is to the top, the better suited it is to detect larger objects.

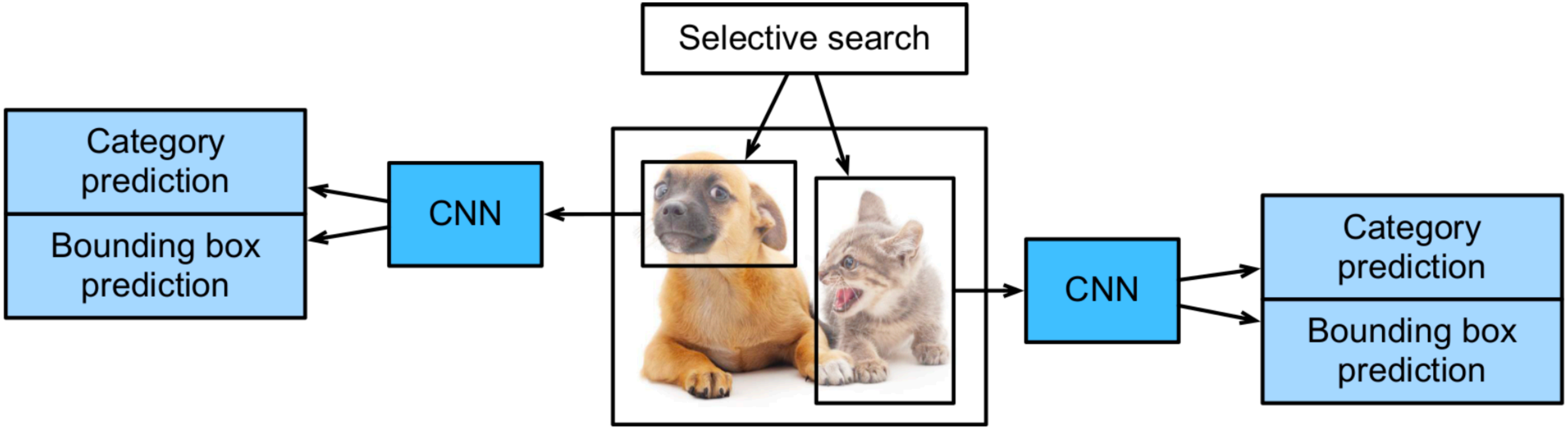
Single Shot Multibox Detection (SSD)



- The base network is used to extract features of original images.
- The closer a feature block is to the top, the better suited it is to detect larger objects.
- As the SSD generates different numbers of anchor boxes of different sizes based on the base network block and each multiscale feature block and then predicts the categories and offsets (i.e., predicted bounding boxes) of the anchor boxes in order to detect objects of different sizes, SSD is a multiscale object detection model.

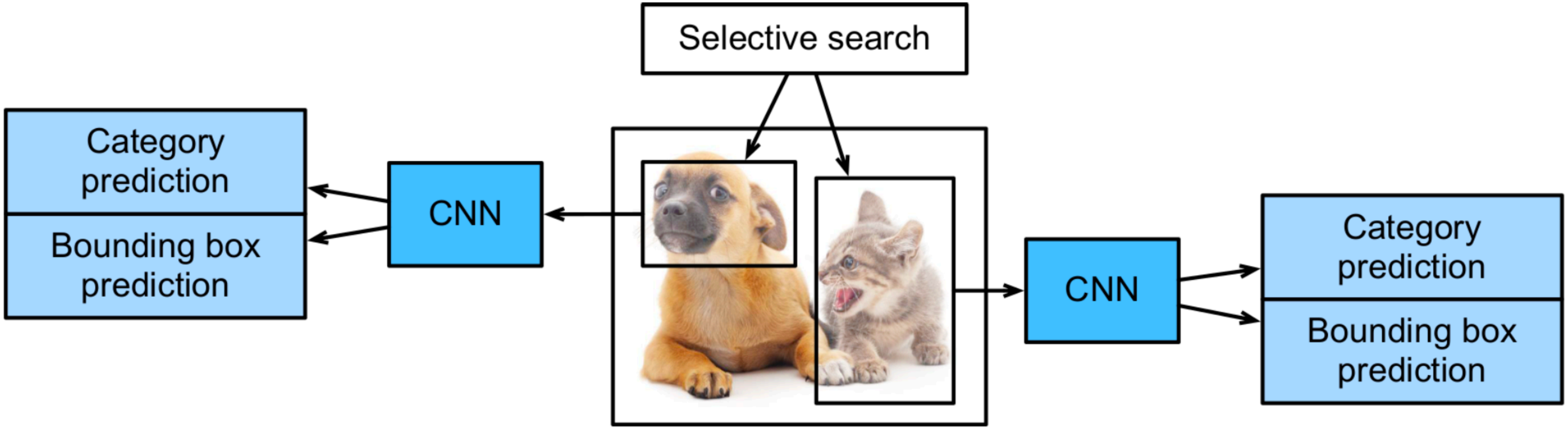
Region-based CNNs (R-CNNs)

- R-CNN models first select several proposed regions from an image (for example, anchor boxes are one type of selection method) and then label their categories and bounding boxes (e.g., offsets).



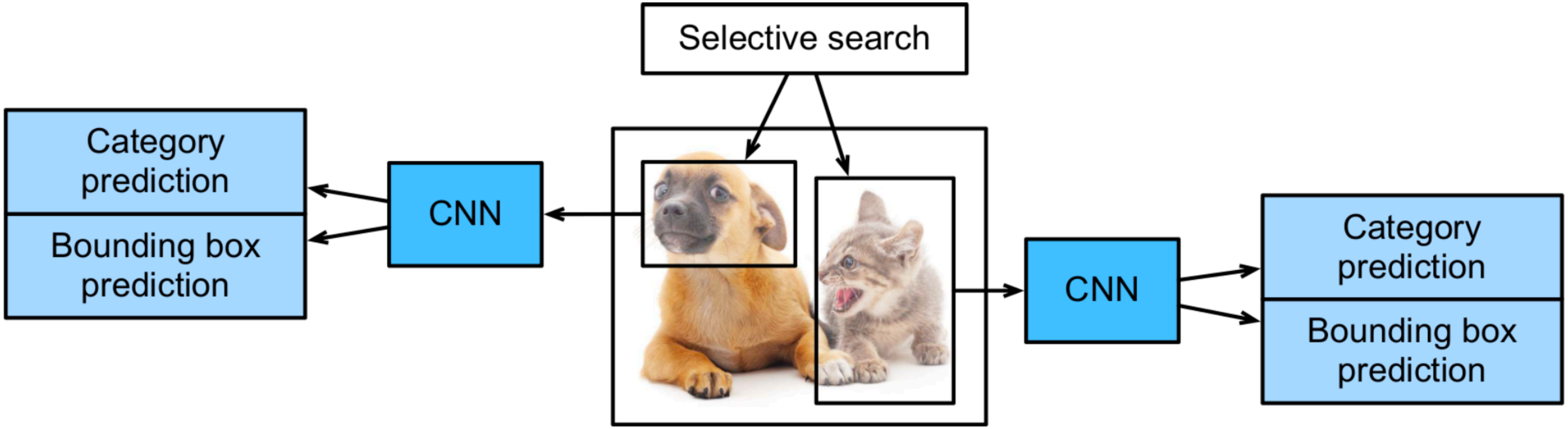
Region-based CNNs (R-CNNs)

- R-CNN models first select several proposed regions from an image (for example, anchor boxes are one type of selection method) and then label their categories and bounding boxes (e.g., offsets).
- Then, they use a CNN to perform forward computation to extract features from each proposed area.



Region-based CNNs (R-CNNs)

- R-CNN models first select several proposed regions from an image (for example, anchor boxes are one type of selection method) and then label their categories and bounding boxes (e.g., offsets).
- Then, they use a CNN to perform forward computation to extract features from each proposed area.
- Afterwards, we use the features of each proposed region to predict their categories and bounding boxes.



R-CNNs

- Specifically, R-CNNs are composed of:

R-CNNs

- Specifically, R-CNNs are composed of four main parts:
    1. Selective search is performed on the input image to select multiple high-quality proposed regions. The category and ground-truth bounding box of each proposed region is labeled.

R-CNNs

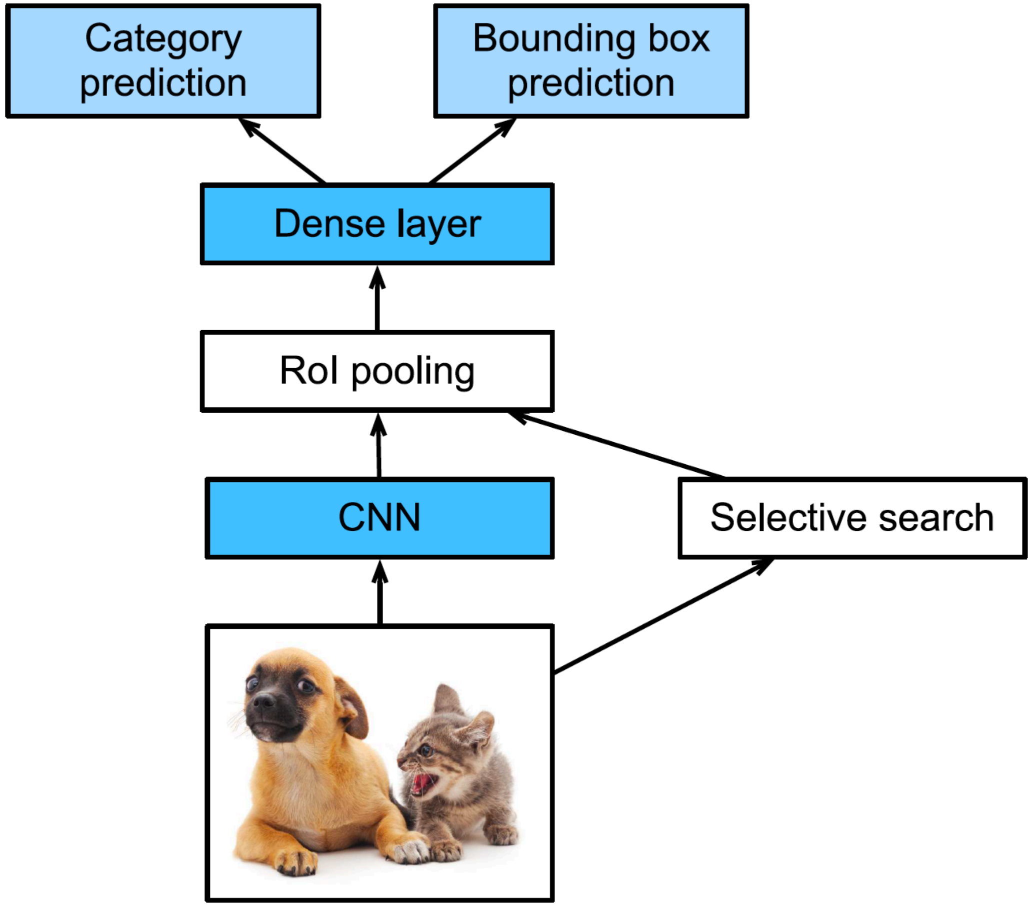
- Specifically, R-CNNs are composed of four main parts:
    1. Selective search is performed on the input image to select multiple high-quality proposed regions. The category and ground-truth bounding box of each proposed region is labeled.
    2. A pre-trained CNN is selected and placed, in truncated form, before the output layer. It transforms each proposed region into the input dimensions required by the network and uses forward computation to output the features extracted from the proposed regions.

R-CNNs

- Specifically, R-CNNs are composed of four main parts:
    1. Selective search is performed on the input image to select multiple high-quality proposed regions. The category and ground-truth bounding box of each proposed region is labeled.
    2. A pre-trained CNN is selected and placed, in truncated form, before the output layer. It transforms each proposed region into the input dimensions required by the network and uses forward computation to output the features extracted from the proposed regions.
    3. The features and labeled category of each proposed region are combined as an example to train multiple support vector machines for object classification. Here, each support vector machine is used to determine whether an example belongs to a certain category.

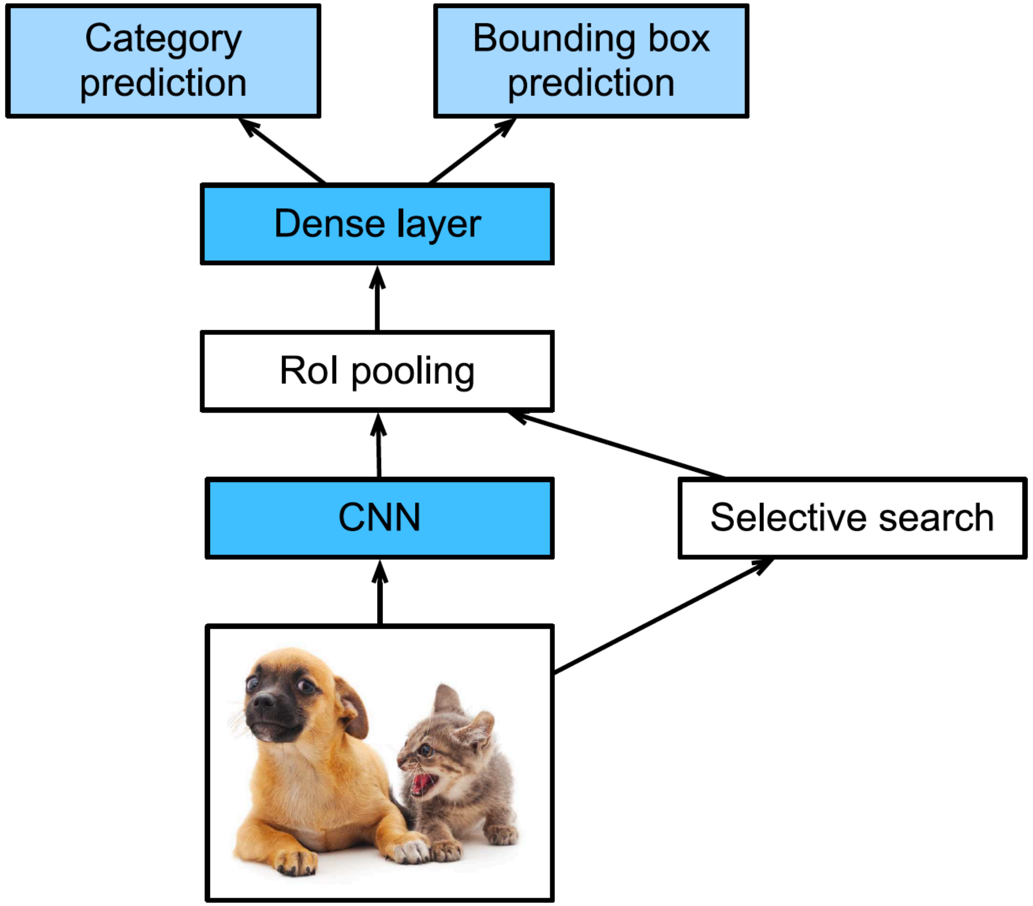
Fast R-CNN

- The main downside of R-CNN is the slow speed.



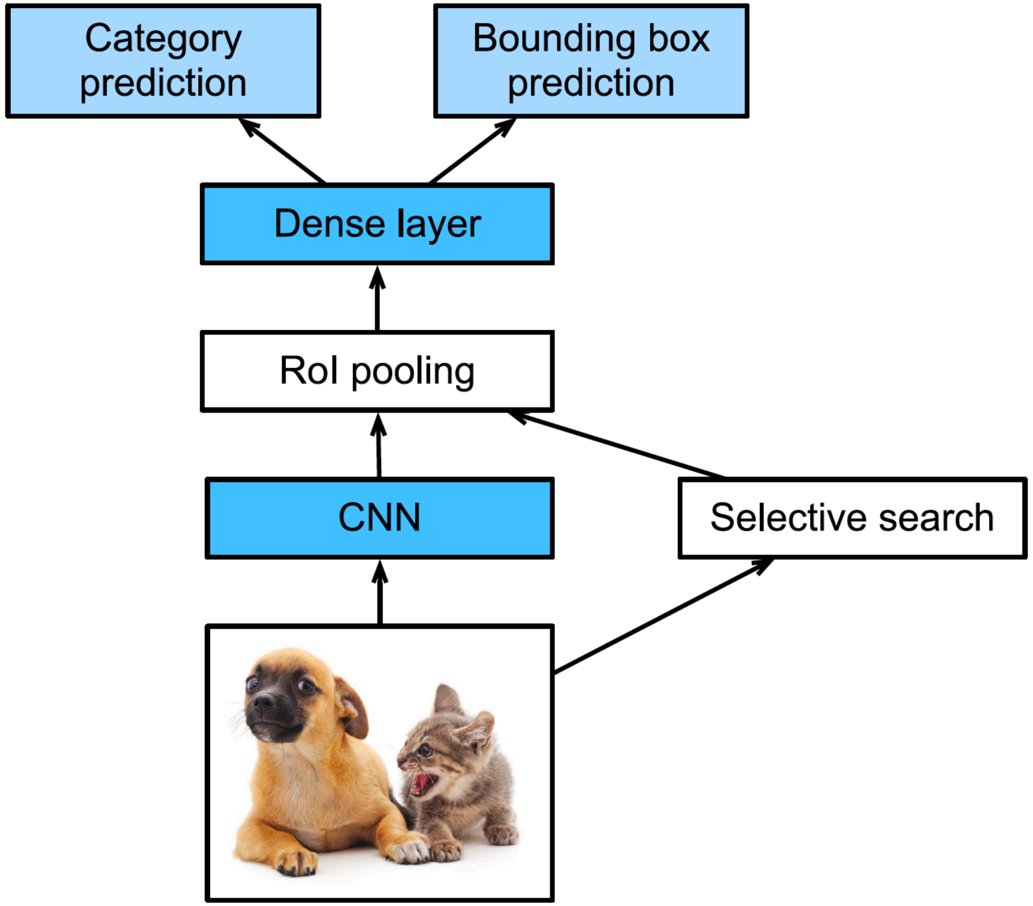
Fast R-CNN

- The main downside of R-CNN is the slow speed.
- The main performance bottleneck of an R-CNN model is the need to independently extract features for each proposed region.



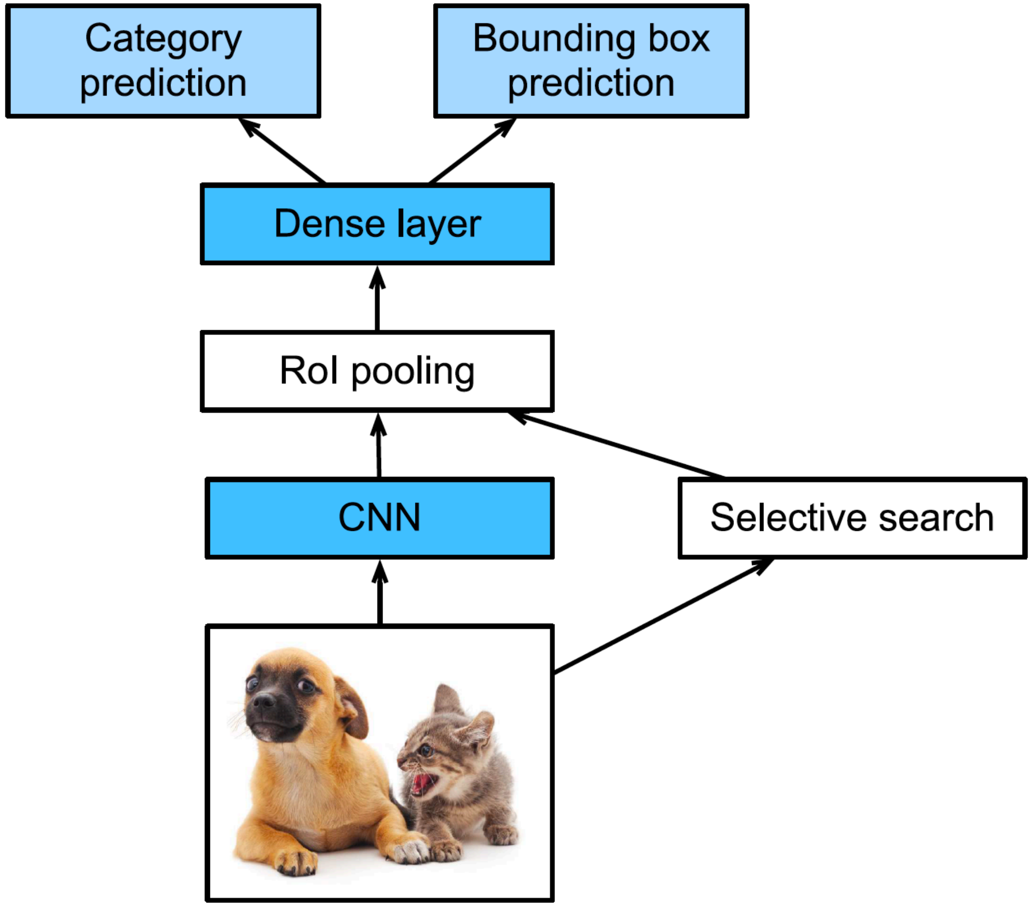
Fast R-CNN

- The main downside of R-CNN is the slow speed.
- The main performance bottleneck of an R-CNN model is the need to independently extract features for each proposed region.
- As these regions have a high degree of overlap, independent feature extraction results in a high volume of repetitive computations.



Fast R-CNN

- The main downside of R-CNN is the slow speed.
- The main performance bottleneck of an R-CNN model is the need to independently extract features for each proposed region.
- As these regions have a high degree of overlap, independent feature extraction results in a high volume of repetitive computations.
- Fast R-CNN improves on the R-CNN by only performing CNN forward computation on the image as a whole.



Fast R-CNN

- Fast R-CNN model. It’s primary computation steps are described below:

Fast R-CNN

- Fast R-CNN model. It’s primary computation steps are described below:
  1. Compared to an R-CNN model, a Fast R-CNN model uses the entire image as the CNN input for feature extraction, rather than each proposed region.

Fast R-CNN

- Fast R-CNN model. It’s primary computation steps are described below:
  1. Compared to an R-CNN model, a Fast R-CNN model uses the entire image as the CNN input for feature extraction, rather than each proposed region.
  2. Assuming selective search generates n proposed regions, their different shapes indicate regions of interests (RoIs) of different shapes on the CNN output.

Fast R-CNN

- Fast R-CNN model. It’s primary computation steps are described below:
    1. Compared to an R-CNN model, a Fast R-CNN model uses the entire image as the CNN input for feature extraction, rather than each proposed region.
    2. Assuming selective search generates n proposed regions, their different shapes indicate regions of interests (RoIs) of different shapes on the CNN output.
    3. Fast R-CNN introduces RoI pooling, which uses the CNN output and RoIs as input to output a concatenation of the features extracted from each proposed region.

Fast R-CNN

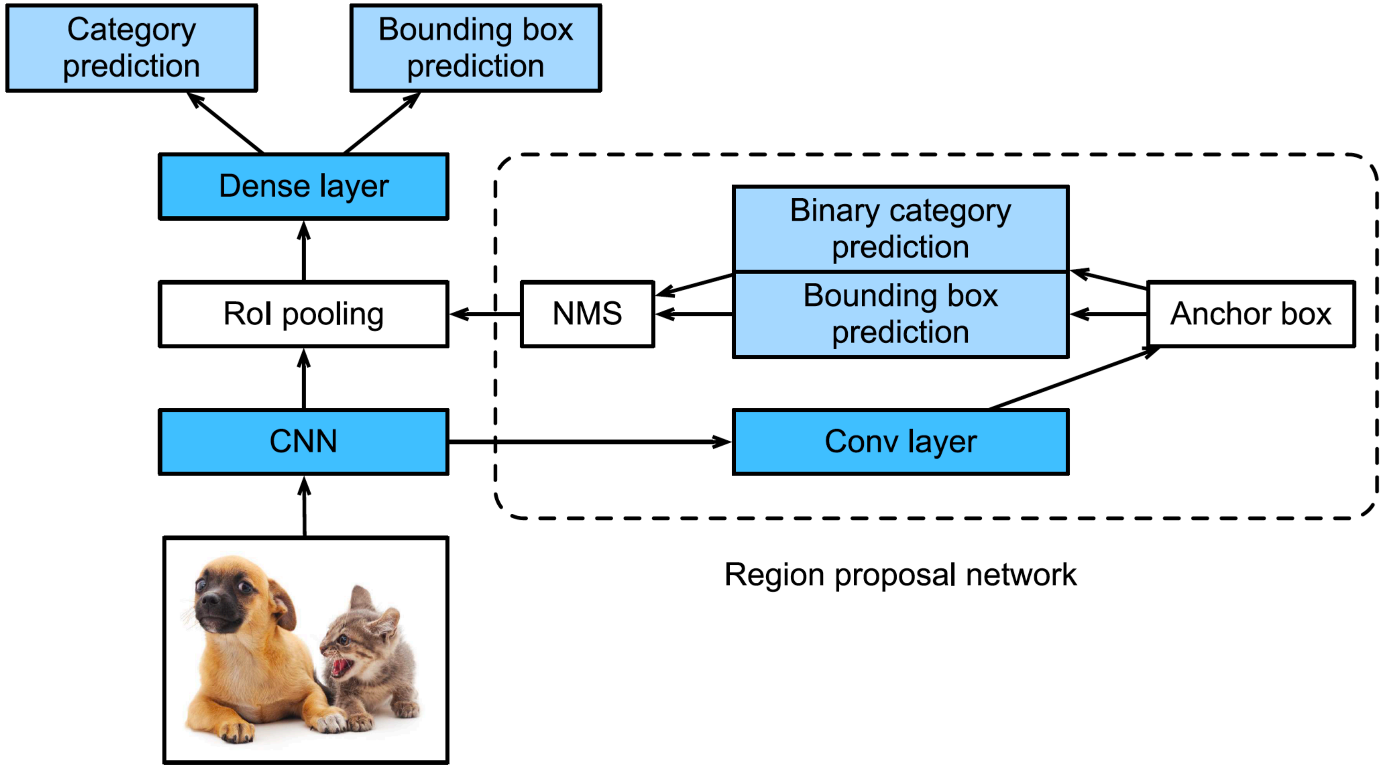
- Fast R-CNN model. It's primary computation steps are described below:
  1. Compared to an R-CNN model, a Fast R-CNN model uses the entire image as the CNN input for feature extraction, rather than each proposed region.
  2. Assuming selective search generates n proposed regions, their different shapes indicate regions of interests (RoIs) of different shapes on the CNN output.
  3. Fast R-CNN introduces RoI pooling, which uses the CNN output and RoIs as input to output a concatenation of the features extracted from each proposed region.
  4. A fully connected layer is used to transform the output shape to n x d, where d is determined by the model design.

Fast R-CNN

- Fast R-CNN model. It’s primary computation steps are described below:
  1. Compared to an R-CNN model, a Fast R-CNN model uses the entire image as the CNN input for feature extraction, rather than each proposed region.
  2. Assuming selective search generates n proposed regions, their different shapes indicate regions of interests (RoIs) of different shapes on the CNN output.
  3. Fast R-CNN introduces RoI pooling, which uses the CNN output and RoIs as input to output a concatenation of the features extracted from each proposed region.
  4. A fully connected layer is used to transform the output shape to n x d, where d is determined by the model design.
  5. During category prediction, the shape of the fully connected layer output is again transformed to n x q and we use softmax regression (q is the number of categories). During bounding box prediction, the shape of the fully connected layer output is again transformed to n x 4.

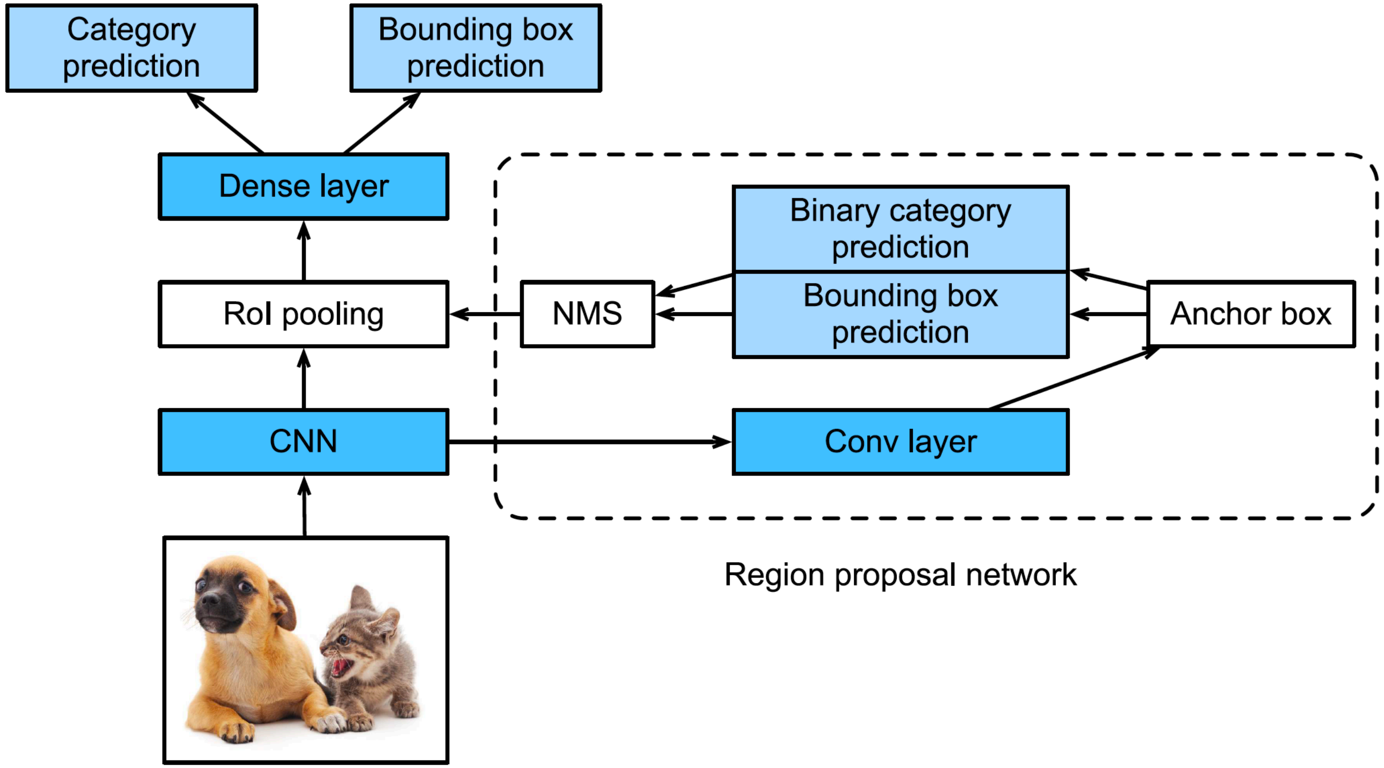
Faster R-CNN

- Fast R-CNN generally requires that many proposed regions be generated in selective search.



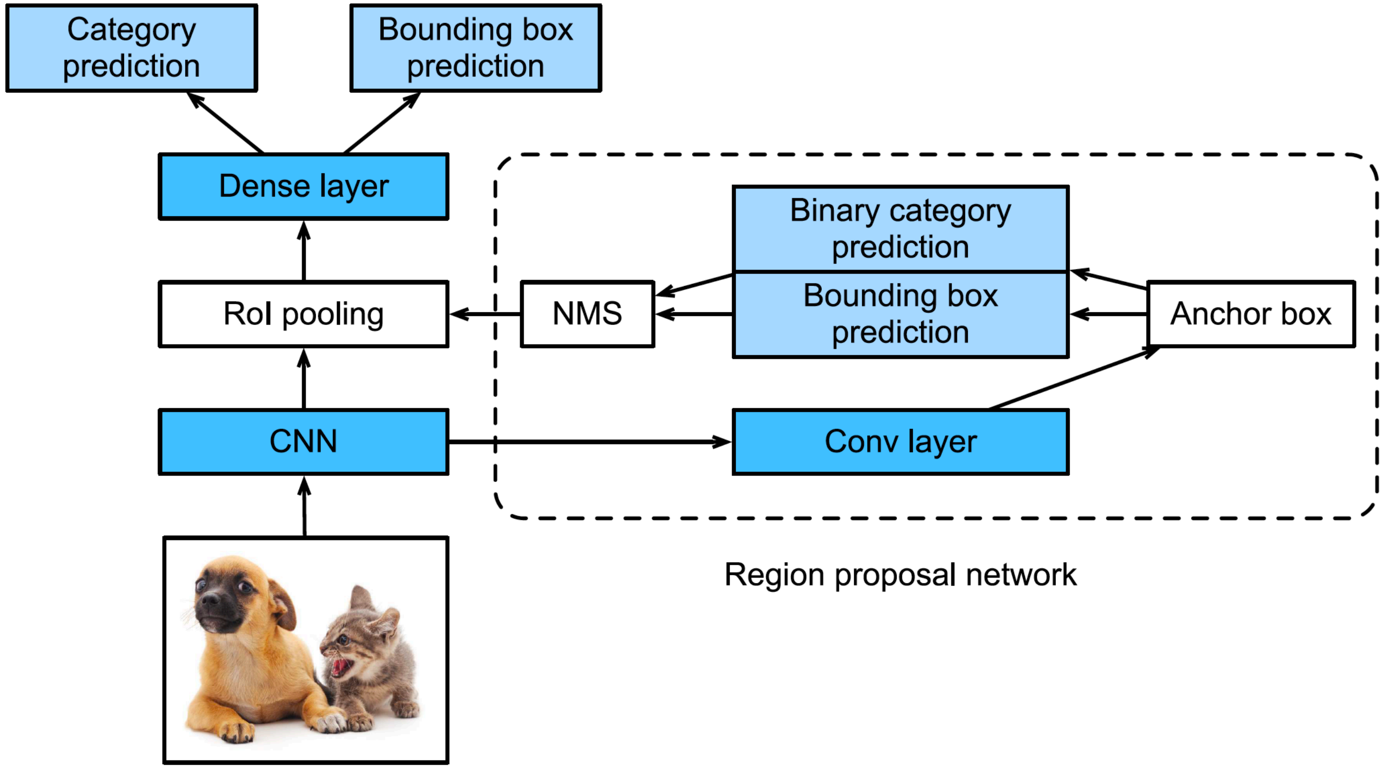
Faster R-CNN

- Fast R-CNN generally requires that many proposed regions be generated in selective search.
- Faster R-CNN replaces selective search with a region proposal network.



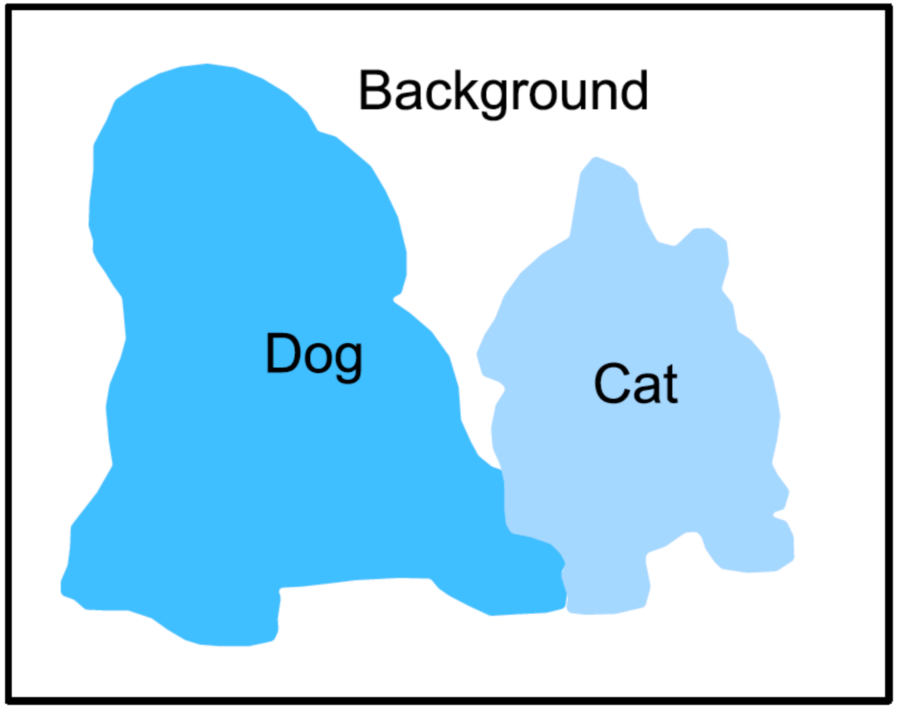
Faster R-CNN

- Fast R-CNN generally requires that many proposed regions be generated in selective search.
- Faster R-CNN replaces selective search with a region proposal network.
- This reduces the number of proposed regions generated, while ensuring precise object detection.



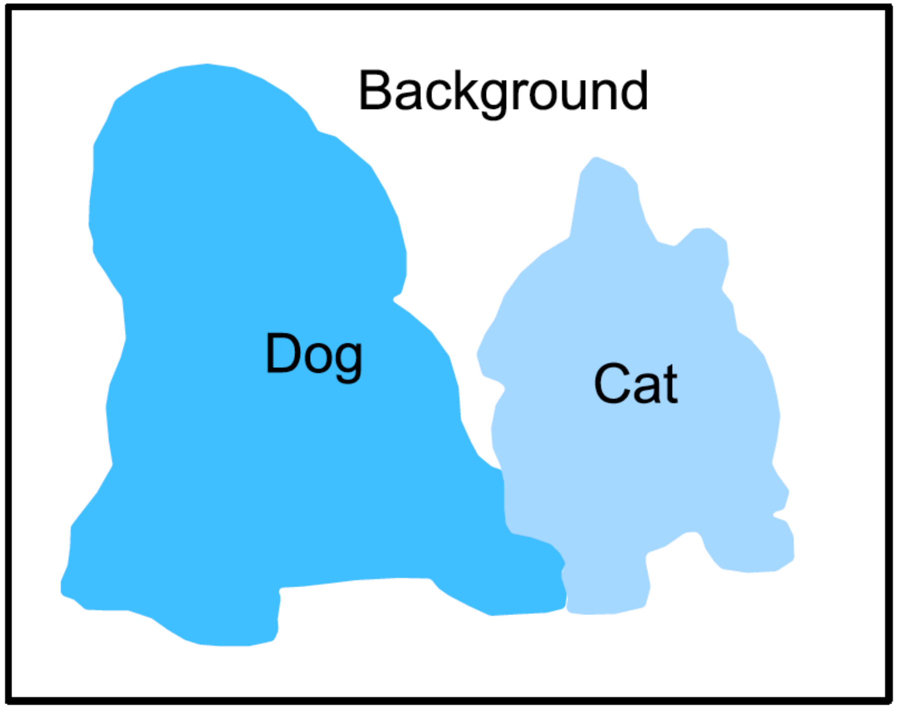
Semantic Segmentation

- So far, we only used rectangular bounding boxes to label and predict objects in images.



Semantic Segmentation and Data Sets

- So far, we only used rectangular bounding boxes to label and predict objects in images.
- Now, we will look at semantic segmentation, which attempts to segment images into regions with different semantic categories.



Transposed Convolution

- The layers we introduced so far for convolutional neural networks, including convolutional layers and pooling layers, often reduce the input width and height, or keep them unchanged.

Transposed Convolution

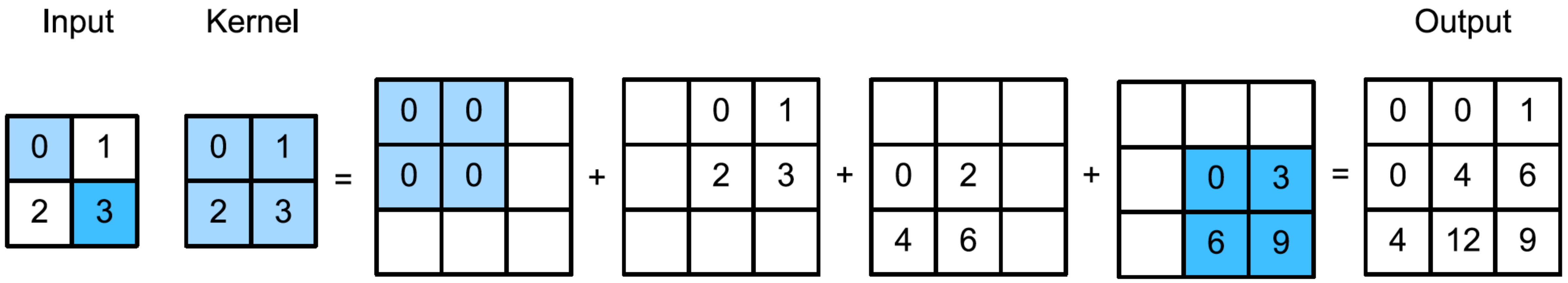
- The layers we introduced so far for convolutional neural networks, including convolutional layers and pooling layers, often reduce the input width and height, or keep them unchanged.
- Applications such as semantic segmentation and generative adversarial networks however, require to predict values for each pixel and therefore needs to increase input width and height.

Transposed Convolution

- The layers we introduced so far for convolutional neural networks, including convolutional layers and pooling layers, often reduce the input width and height, or keep them unchanged.
- Applications such as semantic segmentation and generative adversarial networks however, require to predict values for each pixel and therefore needs to increase input width and height.
- Transposed convolution, also named fractionally-strided convolution or deconvolution, serves this purpose.

Basic 2D Transposed Convolution

- Let’s consider a basic case that both input and output channels are 1, with 0 padding and 1 stride.
- Below figure illustrates how transposed convolution with a 2 x 2 kernel is computed on the 2 x 2 input matrix.



Fully Convolutional Networks (FCN)

- Semantic segmentation uses each pixel in an image for category prediction.

Fully Convolutional Networks (FCN)

- Semantic segmentation uses each pixel in an image for category prediction.
- An FCN uses convolutional neural network to transform image pixels to image categories.

Fully Convolutional Networks (FCN)

- Semantic segmentation uses each pixel in an image for category prediction.
- An FCN uses convolutional neural network to transform image pixels to image categories.
- An FCN transforms the height and width of the intermediate layer feature map back to the size of input image through the transposed convolution layer.

Fully Convolutional Networks (FCN)

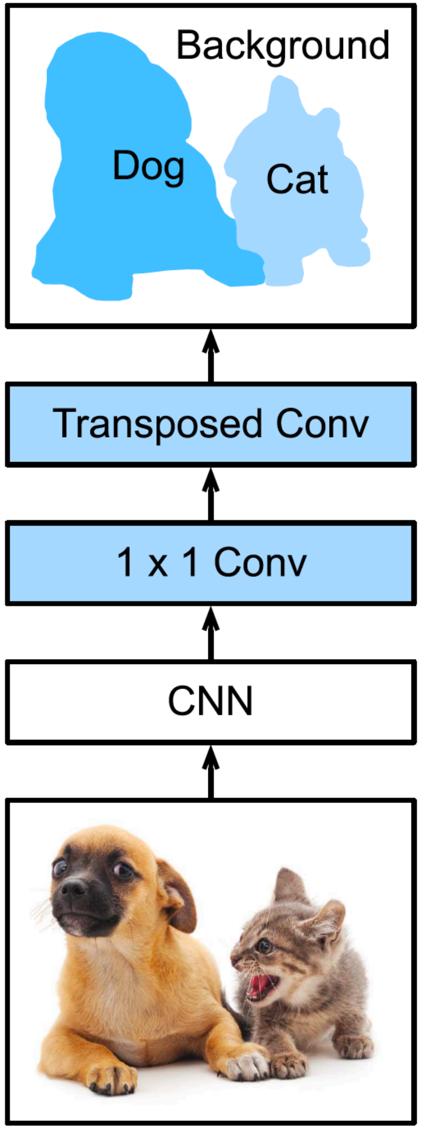
- Semantic segmentation uses each pixel in an image for category prediction.
- An FCN uses convolutional neural network to transform image pixels to image categories.
- An FCN transforms the height and width of the intermediate layer feature map back to the size of input image through the transposed convolution layer.
- This way the predictions have a one-to-one correspondence with input image in spatial dimension (height and width).

Fully Convolutional Networks (FCN)

- Semantic segmentation uses each pixel in an image for category prediction.
- An FCN uses convolutional neural network to transform image pixels to image categories.
- An FCN transforms the height and width of the intermediate layer feature map back to the size of input image through the transposed convolution layer.
- This way the predictions have a one-to-one correspondence with input image in spatial dimension (height and width).
- Given a position on the spatial dimension, the output of the channel dimension will be a category prediction of the pixel corresponding to the location.

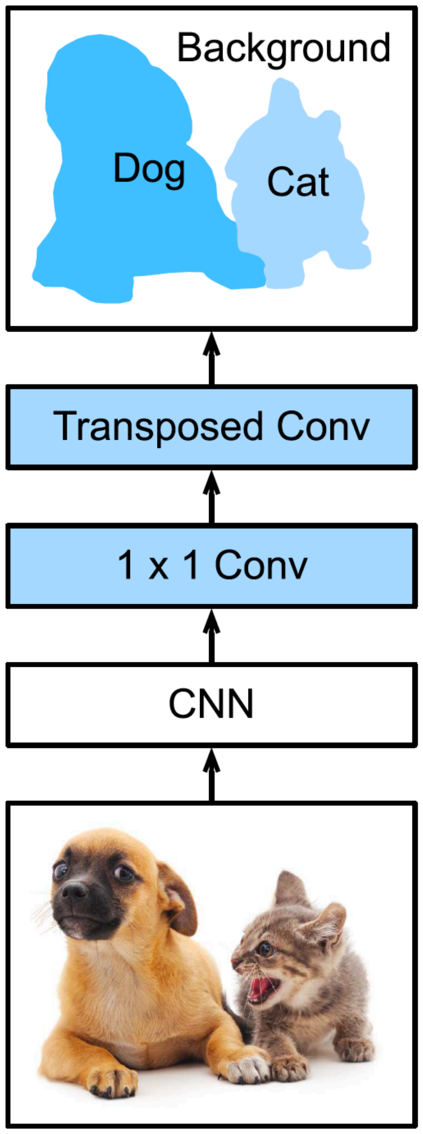
Fully Convolutional Neural Networks

- The fully convolutional network first uses the convolutional neural network to extract image features.



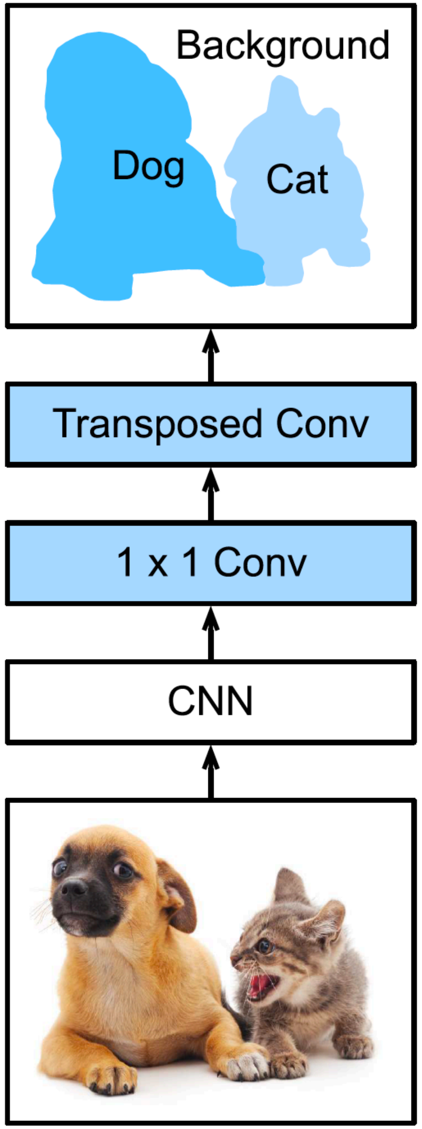
Fully Convolutional Neural Networks

- The fully convolutional network first uses the convolutional neural network to extract image features.
- Then transforms the number of channels into the number of categories through the 1 x 1 convolution layer.



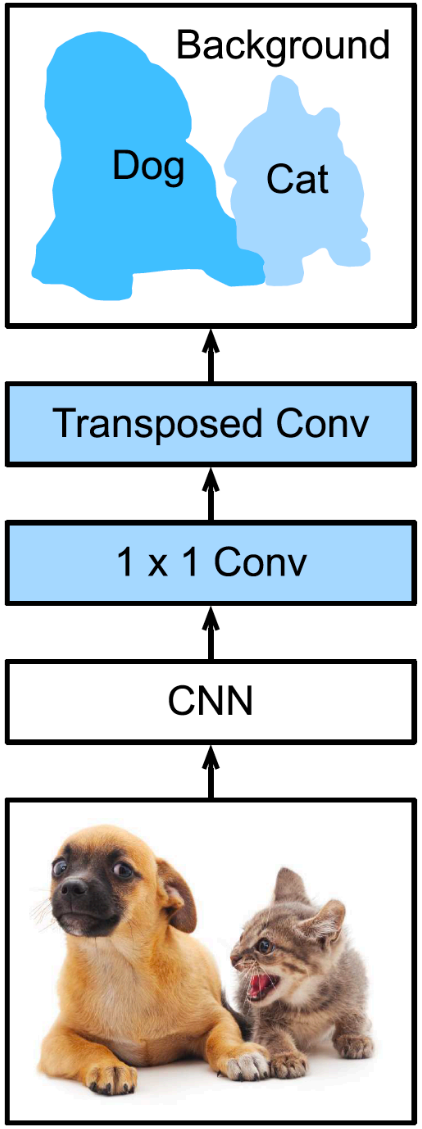
Fully Convolutional Neural Networks

- The fully convolutional network first uses the convolutional neural network to extract image features.
- Then transforms the number of channels into the number of categories through the 1 x 1 convolution layer.
- And finally transforms the height and width of the feature map to the size of the input image by using the transposed convolution layer.



Fully Convolutional Neural Networks

- The fully convolutional network first uses the convolutional neural network to extract image features.
- Then transforms the number of channels into the number of categories through the 1 x 1 convolution layer.
- And finally 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.



Fully Convolutional Neural Networks

- The fully convolutional network first uses the convolutional neural network to extract image features.
- Then transforms the number of channels into the number of categories through the 1 x 1 convolution layer.
- And finally 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.