Convolutional neural network vs traditional multilayer perceptron (MLP)

Following the concept of receptive fields, CNNs exploit spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers

Each filter is replicated across the entire visual field.

Three hyperparameters control the size of the output volume of the layer: Depth Stride Zero-padding

Width and height are determined by image size, stride and zero-padding

The depth of the output volume controls the number of neurons in a layer that connect to the same region of the input volume.

These neurons learn to activate for different features in the input.

E.g., if the first convolutional layer takes the raw image as input, then different neurons along the depth dimension may activate in the presence of various oriented edges, or blobs of color.

For example, consider a convolutional layer with 5 × 5 filters, outputting 200 feature maps of size 150 × 100, with stride 1 and "same" padding. If the input is a 150 × 100

RGB image (three channels), then the number of parameters is (5 × 5 × 3 + 1) × 200 = 15,200 (the + 1 corresponds to the bias terms), which is fairly small compared to a

fully connected layer.

However, each of the 200 feature maps contains 150 × 100 neurons, and each of these neurons needs to compute a weighted sum of its 5 × 5 × 3 = 75 inputs: that’s a total of 225 million float multiplications.

Localizing an object in a picture can be expressed as a regression task.

to predict a bounding box around the object, a common approach is to horizontal and vertical coordinates of the object’s center, as well as its height and width.

The bounding boxes should be normalized so that the horizontal and vertical coordinates, as well as the height and width, all range from 0 to 1.

Also, it is common to predict the square root of the height and width rather than the height and width directly: this way, a 10-pixel error for a large bounding box will not be penalized as much as a 10-pixel error for a small bounding box.

IoU = the area of overlap between the predicted bounding box and the target bounding box, divided by the area of their union.

Take a CNN that was trained to classify and locate a single object, then slide it across the image (also at multiple scales)

It will detect the same object multiple times, at slightly different positions

Post-processing needed to get rid of all the unnecessary bounding boxes

E.g. find the bounding box with the highest objectness score, and get rid of all the other bounding boxes that overlap a lot with it (e.g., with an IoU > 60%).

Quite slow, since it requires running the CNN many times.

If we use the Sliding Window technique like the way we classify and localize images, we need to apply a CNN to many different crops of the image.

Because CNN classifies each crop as object or background, we need to apply CNN to huge numbers of locations and scales, which is very computationally expensive!

1. Find ROIs, “blobby” image regions that are likely to contain objects.
2. Then we run a CNN on top of each of these region proposals.
3. We take the output of each CNN and feed it into an SVM to classify the region and a linear regression to tighten the bounding box of the object.

Essentially, we turned object detection into an image classification problem.

However, there are some problems — the training is slow, a lot of disk space is required, and inference is also slow.

1. Performing feature extraction before proposing regions, thus only running one CNN over the entire image.
2. Replace the slow selective search algorithm with a fast neural network by inserting a Region Proposal Network (RPN) to predict proposals from features.

Faster R-CNN may not be the simplest or fastest method for object detection, but it’s still one of the best performing

Tracking-by-Detection, Following a specific object of interest, or multiple objects, in a given scene.

A large-scale CNN can be trained both as a classifier and as a tracker

2 representative CNN-based tracking algorithms are fully-convolutional network tracker (FCNT) and multi-domain CNN (MD Net).

FCNT analyzes and takes advantage of the feature maps of the VGG model.

CNN feature maps can be used for localization and tracking.

Many CNN feature maps are noisy or un-related for the task of discriminating a particular object from its background.

Higher layers capture semantic concepts on object categories, whereas lower layers encode more discriminative features to capture intra-class variation.

The **GNet** captures the category information of the object, while the **SNet** discriminates the object from a background with a similar appearance.

Fast RCNN performs detection on various region proposals and thus end up performing prediction multiple times for various regions in a image

Yolo architecture is more like FCNN (fully convolutional neural network) and passes the image once through the network.

Main rationale is to avoid having separate algorithms focus on their respective subproblems in isolation, as this typically increases training time and can lower network accuracy.

Splitting the input image in mxm grid and for each grid generation bounding boxes and class probabilities for those bounding boxes

Changes to loss functions for better results: Differential weight for confidence predictions from boxes that contain object and boxes that don't contain object during training. Predict the square root of the bounding box width and height to penalize error in small object and large object differently.

Segmentation:

Each pixel is classified and labeled

Different objects of the same class are **not**distinguished!

In a regular CNN, images gradually lose their spatial resolution (due to the layers with strides greater than 1)

One solution is to use a fully convolutional network