classification

Non-parametric methods

K-Nearest Neighbors (KNN)

1. process:
   i. memorize all data and labels
   ii. for a test sample, calculate the distance between it and all the training samples
      - L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$
      - L2 distance: $d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$
   iii. choose the k nearest neighbors based on the distance
   iv. assign the label of the majority of the k nearest neighbors as the label of the test sample
2. KNN with pixel distance is never used.
   - Pixel distance as a metric is too sensitive to change in background/illumination/pose/viewpoint/occlusion/… that are not essential to the semantics.
   - Very slow at test time.
3. However, nearest neighbor-based techniques are still useful when the metrics is learned via a deep neural network and widely used in image retrieval, metric learning, 3D vision, and etc.

Parametric methods

CNN

Simply speaking, the previous network predicts the unnormalized log-probability(logits) of each class, then we use softmax as activation to get the normalized probability distribution. The cross-entropy loss is used to train the network.

1. softmax:

$$\sigma(z_i) = \frac{\beta e^{z_i}}{\sum_{j=1}^{K} e^{\beta z_j}}$$

   β is set to 1 by default.
2. cross-entropy loss:

$$L = -\sum_{i}^{num\_class} P(x_i) log(Q(x_i))$$

if the target is a one-hot vector, then cross-entropy loss is the NLL loss.

VGGNet

1. use smaller filters (3x3)
   - slow down the growing of receptive fields. stack 3 layers of 3x3 filters have the same receptive field as a single 7x7 filter but it is deeper and has more activation layers.
   - also reduces the number of parameters and makes the model more efficient.
2. pooling layers
   - consistently aggregate knowledge from the previous layers and reduce the spatial dimensionality of the feature maps.

ResNet

1. residual block: 3x3 convolution + batch normalization + ReLU + 3x3 convolution + batch normalization + skip connection.
2. periodically, double # of filters and downsample spatially using stride 2(spatial halving)
   - just like VGG, but don't use POOL layers, just use stride 2 in conv layer.
   - double the number of filters to avoid losing too much information.
3. for deeper networks, use 'bottleneck' layer: 1x1 convolution + batch normalization + ReLU + 3x3 convolution + batch normalization + ReLU + 1x1 convolution + batch normalization + skip connection. (hxwxc->hxwxc/4->hxwxc/4->hxwxc)
   - shrink the memory size and # of parameters.
   - in practice, use 6 layers of bottleneck to substitude a residual block. (now we actually have 6 3x3 conv! but we still have fewer parameters than a residual block)
   - why bottleneck structure? First, it reduce # of channel to give up redandant features, than it restore the channel number to synthesize more useful features.

Learning to Search for Network Architectures

- Neural Architecture Search (NAS) is a popular technique for automatically searching for the best architecture for a given task.

Segmentation

1. Semantic segmentation
2. Instance segmentation

3. Semantic + Instance segmentation

Auto-Encoder

- Information bottleneck: the dimension of z space is much smaller than that of x
- Get rid of redundant information via dimension reduction
- The first step to all advanced segmentation networks

Uppooling layer

output contains copies of the filter weighted by the input, summing at where overlaps in the output.

Convolution transpose multiply by the transpose of the same matrix. Example:

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Bottleneck Structure

- Large receptive field and provides global context
- Get rid of redundant information
- Lower the computation cost

Skip link:

- Assist final segmentation
- Avoid memorization

Evaluation Metrics

1. accuracy

- Pixel Accuracy: the number of pixels that are correctly labeled
  - rarely used, it easily biased towards the majority class. Like if I have 80% pixels of grass, I just need to predict all pixels as grass, then the model will get 80% accuracy.
- Mean Accuracy: the average accuracy of each class

2. IoU (Intersection over Union)

- Intersection: the number of pixels that are both correctly labeled and predicted as the same class
- Union: the number of pixels that are correctly labeled or predicted as the same class
- IoU = Intersection / Union