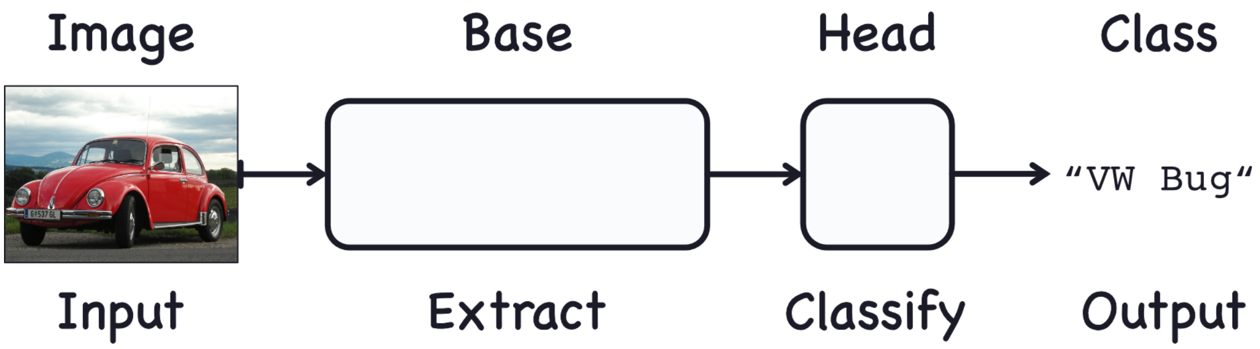
Chapter 5: Deep learning for computer vision

Computer vision task involving deep learning can be carried out with keras using convnets (Convolutional Neural networks CNN). Though Densely connected layers can also be used to carry out computer vision but the accuracy is low and it cost more than a convnets because a Dense layer learns global patterns of input space while a convent learns a local patterns of the input space.

Two basics characteristics makes convnets of choice in the image processing world.

1) Translation invariant: Convnets learns local patterns, the local patterns they learn in a location can be recognized in another location. This prevents them from relearning it in another location since they can recognize it. therefore, reducing the cost of learning. This characteristic sync with the problem characteristic.
2) Spatially hierarchical: Convnet learns features in hierarchical way.from low-level features(edge, darkspot etc) to mid-level feature(eye,ears,nose etc) to high level features(facial structure).This is because once it learns edges feature in one layer,it will recognize it else where and does not need to learn it again, rather it learns more complex features.

A convolutional network used for image classification has 2 parts. convolution base and dense layer as depicted below.



photograph by Dnalor 1 (Wikimedia Commons) CC-BY-SA 3.0

The convolutional base extract features while the dense head classify image. The convolutional base does automatic feature extraction of the image by three operation.

1. **Filter** an image for a particular feature (convolution)
2. **Detect** that feature within the filtered image (ReLU)
3. **Condense** the image to enhance the features (maximum pooling)

Convolution: convolution refers to the mathematical combination of two functions to produce a third function. The Convolution take in an input of 3D tensor (image_height, image_width, image_channels) .The convolution operation work by applying a filter of n*n (n^2 different weights) to the same size portion of input(patch) i.e n*n. The operation is carried out on all the patches by sliding the filter over the input. This produces an output feature map which is a 3D tensor having a width and height but a depth which is a parameter of the layer. The output depth is the number of filters applied to the input.

Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1))

For the layer above the 32 means the number of filters to apply to the input, while (3,3) mean the kernel size or filter size which is the same the size of each patch of the input it is applied on.

Feature map and padding

The feature map is the output of one filter applied to the previous layer. A given filter is drawn across the entire previous layer, moved one pixel at a time. Each position results in an activation of the neuron and the output is collected in the feature map. The number of pixels moved at a time is called Stride. The output width and height may differ from the input width and height. In other to make the spatial dimension of the input to be the same as the output we use padding. Padding consists of adding an appropriate number of rows and columns on each side of the input feature map so as to make it possible to fit center convolution windows around every input tile.It does so by a process of adding layers of zeros to our input images

Max-pooling operation

Max pooling takes a patch of activations in the original feature map and replaces them with the maximum activation in that patch. Max pooling consists of extracting windows (pool size) from the input feature maps and outputting the max value of each channel. The Max pooling does not have any trainable parameter. The max pooling helps to downsize the feature map so a to 1) reduce dimensionality 2) reduce the number of pixels 3) reduce overfitting

WORKFLOW FOR AN IMAGE CLASSIFICATION

1)Data processing

2)Build the CNN network

3) Perform Naïve training (we are sure we will experience overfitting)

4)Data augmentation (NB: don’t augment validation and test data set) and Dropout to regularize then retrain

5)Using Pretrained network for feature extraction with data augmentation and Dropout then retrain (Make sure to freeze the trainable weight of the pre trained network)

6)Fine-tuning the pretrain network: This is done by unfreezing a few top layers of the pre trained network so that the weight in the layers can be trainable along with the part of the model (in this case, the fully connected classifier).Fine tuning slightly adjusts the more abstract representations of the model being reused, in order to make them more relevant for the problem at hand

NB:An important universal characteristic of the representations
learned by deep neural networks: the features extracted by a layer become increasingly abstract
with the depth of the layer. The activations of higher layers carry less
and less information about the specific input being seen, and more and more information about
the target (in this case, the class of the image: cat or dog)