CNN and Image Classification

Introduction

**Important Concepts**

CNN Feature Building

Adding Layers

Receptive Field

Pooling

Flattening and fully connected layers

A screenshot of a computer

Description automatically generated

A CNN is an NN with special layers.

Process

The model classifies an image by taking a part of the image, each input passes this chunk of image through a series of convolution layers:

* Convolution
* Filtering
* Pooling
* Fully Connected

Convolution and Pooling layers are the first layers used to extract features from an input. These can be seen as the learning layers. the fully connected layers are analogous to the normal hidden layers in an NN. Both are learned simultaneously by minimizing cross-entropy loss.

**How CNNs Build Features**

H.O.G. uses Sobel kernels to detect vertical and horizontal edges.

A screenshot of a video

Description automatically generated

H.O.G can be represented with a diagram similar to an NN.

A screenshot of a computer

Description automatically generated

The linear function is replaced with a convolution, and squaring and squareroot operations are analgous to activation functions.

A screenshot of a computer

Description automatically generated

In a CNN there are still neurons but the kernels are learnable parameters. Activation functions are applied to each pixel (ReLu). Instead of an activation, the output is an activation map or ‘feature map’ similar to a 1 channel img.

Like the HOGs Sobel kernel, each kernel of a CNN will detect a different property of the img, ie. the mouth, nose, outline, eyes.

There are M Kernels for M Features and M Feature Maps.

For each map, convolution and ReLu is applied.

A grayscale image can be regarded as a one-channel input. If there are M kernels, each feature map will be a channel, so there will be M outputs.

Convolutional layers can also be stacked. The input of one layer will be equal to the output of the last.

Its helpful to look at kernels to understand what each layer does.

A screenshot of a computer

Description automatically generated

The first layer in this cnn is looking to identify edges, (a Sobel kernel) the second looks like facial features and the third, looks like faces themselves.

Essentially adding more layers allows for the detection of more complex features.

Receptive Field

Is the size of the region in the input that produces a pixel value in the activation map.

A screenshot of a video

Description automatically generated

The left contains the image, the right contains the activation map. The larger the receptive field the more information (pixels) the activation map contains.

The receptive field can be increased by adding more layers. The requires less parameters than increasing the size of the kernel.

A screenshot of a video

Description automatically generated

Here we can see that adding a layer increases the receptive field.

**Pooling**

Pooling helps to reduce the number of parameters, increases the receptive fieldwhile preserving important features. It can be thought of as resizing the image.

Max pooling is commonly used for this.

A screenshot of a video

Description automatically generated

Max pooling takes the largest value from a matrix of pixels as the value to preserve when resizing.

It also makes CNNs more immutable to small changes in the image such as shifts.

**Flattening and Fully Connected Neural Networks**

This simply refers to the process of flattening or reshaping the outputs of the feature learning layers and using those as input for the fully connected layers.

A screenshot of a video

Description automatically generated

Ie. if the output of the pooling layer is a 7x7 img, then this is reshaped to a 1d array or feature vector.

Each neuron will have the input dimension (in this case 49) as a flattened output.

This process is the same for multiple output channels. So if there are 32 output channels each of 4x4 then feature vector will be 32x16 -> 1x512. So each neuron will have 512 input dimensions.

**CNN Architectures**

Popular CNN Architectures include:

* LeNet-5
* AlexNet
* VGGNet
* ResNet

LeNet is most notable for identifying handwritten digits.

LeNet receives an img output, normally grayscale and uses a 5x5 filter with a 1 stride, resulting in 28 x 28 outputs.

The next layer is a pooling layer with 14 x 14 outputs.

This layer repeats with a filter until the fully connected layers where it flattens to create 120 neurons and then another with 84 neurons using a sigmoid activation function to produce an output.

A screenshot of a video

Description automatically generated

To quasi standardize the accuracy of image classification, the ImageNet dataset is seen as the benchmark.

A graph showing a growth of a company

Description automatically generated with medium confidence

AlexNet after 2012 achieved 63.3% accuracy, prior to this a HOG like method was used to achieve around 51% accuracy. The jump was so large that people started using CNNs for classification as opposed to SVMs.

A screenshot of a video game

Description automatically generated

Here is the ALexNet diagram.

A screenshot of a video game

Description automatically generated

The VGG Network is a deep CNN that developed from the need to reduce the number of parameters and improve training time. It showed that deeper networks performed better.

The key insight gained from VGG was that larger kernels could be replaced by 3x3 kernels and stacking convolution layers, keeping th same receptive field but reducing parameters.

As models became deeper, vanishing gradient became a problem. This was solved by ResNet, which introduced residual layers. Residual layers or “Skip Connections” are bypassed by the gradient.



**Transfer Learning**

Transfer Learning is using a pretrained CNN to classify an img instead of building one. Pretrianed CNs