

TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2020

Convolutional Neural Networks (CNNs)

Imagenet Classification

1000 kinds of objects.

2016 is 3.0%, is 2017 2.25%

SOTA as of January 2020 is 1.3%

What is a CNN?

VGG, Zisserman, 2014

Davi Frossard

A Convolution Layer

Each box is a tensor $L_\ell[b, x, y, i]$

Each value $L_\ell[b, x, y, i]$ (for $\ell > 0$) is the output of a single linear threshold unit.

A Convolution Layer

$$W[\Delta x, \Delta y, i, j] \quad L_\ell[b, x, y, i] \quad L_{\ell+1}[b, x, y, j]$$

River Trail Documentation

$$L_{\ell+1}[b, x, y, j]$$

$$= \sigma \left(\left(\sum_{\Delta x, \Delta y, i} W[\Delta x, \Delta y, i, j] L_\ell[b, x + \Delta x, y + \Delta y, i] \right) - B[j] \right)$$

Convolution Layer in Einstein Notation

$$L_{\ell+1}[b, x, y, j]$$

$$= \sigma \left(\left(\sum_{\Delta x, \Delta y, i} W[\Delta x, \Delta y, i, j] L_{\ell}[b, x + \Delta x, y + \Delta y, i] \right) - B[j] \right)$$

$$= \sigma (W[\Delta X, \Delta Y, I, j] L_{\ell}[b, x + \Delta X, y + \Delta Y, I] - B[j])$$

Many “Neurons” (Linear Threshold Units)

Each $L_{\ell+1}[b, x, y, j]$ is the output of a single linear threshold unit.

$$\begin{aligned} & L_{\ell+1}[b, x, y, j] \\ &= \sigma (W[\Delta X, \Delta Y, I, j] L_{\ell}[b, x + \Delta X, y + \Delta Y, I] - B[j]) \end{aligned}$$

In this equation the input to the activation function σ is a scalar value — this is a single linear threshold unit.

2D CNN in PyTorch

`conv2d(input, weight, bias, stride, padding, dilation, groups)`

input – tensor (minibatch,in-channels,iH,iW)

weight – filters (out-channels, in-channels/groups,kH,kW)

bias – tensor (out-channels) . Default: None

stride – Single number or (sH, sW). Default: 1

padding – Single number or (padH, padW). Default: 0

dilation – Single number or (dH, dW). Default: 1

groups – split input into groups. Default: 1

Padding

Jonathan Hui

If we pad the input with zeros then the input and output can have the same spatial dimensions.

Zero Padding in NumPy

In NumPy we can add a zero padding of width p to an image as follows:

```
padded = np.zeros(W + 2*p, H + 2*p)  
  
padded[p:W+p, p:H+p] = x
```

Padding

$$L'_\ell = \text{Padd}(L_\ell, p)$$

$$L_{\ell+1}[b, x, y, j] =$$

$$\sigma \left(W[\Delta X, \Delta Y, I, j] L'_\ell[b, x + \Delta X, y + \Delta Y, I] - B[j] \right)$$

If the input is padded but the output is not padded then Δx and Δy are non-negative.

Reducing Spatial Dimention

Reducing Spatial Dimensions: Max Pooling

$$L_{\ell+1}[b, x, y, i] = \max_{\Delta x, \Delta y} L_{\ell}[b, s * x + \Delta x, s * y + \Delta y, i]$$

This is typically done with a stride greater than one so that the image dimension is reduced.

Reducing Spatial Dimensions: Strided Convolution

We can move the filter by a “stride” s for each spatial step.

$$L_{\ell+1}[b, \textcolor{red}{x}, \textcolor{red}{y}, j] =$$

$$\sigma(W[\Delta X, \Delta Y, I, j]L_{\ell}[b, \textcolor{red}{s} * \textcolor{red}{x} + \Delta X, \textcolor{red}{s} * \textcolor{red}{y} + \Delta Y, I] - B[j])$$

For strides greater than 1 the spatial dimension is reduced.

Fully Connected (FC) Layers

Fully Connected (FC) Layers

We reshape $L_\ell[b, x, y, i]$ to $L_\ell[b, i']$ and then

$$L_{\ell+1}[b, j] = \sigma (W[j, I] L_\ell[b, I] - B[j])$$

Dilation

A CNN for image classification typically reduces an $N \times N$ image to a single feature vector.

Dilation is a trick for treating the whole CNN as a “filter” that can be passed over an $M \times M$ image with $M > N$.

An output tensor with full spatial dimension can be useful in, for example, image segmentation.

Dilation

This is called a “fully convolutional” CNN.

Dilation

To implement a fully convolutional CNN we can “dilate” the filters by a dilation parameter d .

$$\begin{aligned} & L_{\ell+1}[b, x, y, j] \\ &= \sigma(W[\Delta X, \Delta Y, I, j]L_{\ell}[b, x + \textcolor{red}{d} * \textcolor{red}{\Delta X}, y + \textcolor{red}{d} * \textcolor{red}{\Delta Y}, I] + B[j]) \end{aligned}$$

Vector Concatenation

We will write

$$L[b, x, y, J_1 + J_2] = L_1[b, x, y, J_1] ; L[b, x, y, J_2]$$

To mean that the vector $L[b, x, y, J_1 + J_2]$ is the concatenation of the vectors $L_1[b, x, y, J_1]$ and $L_2[b, x, y, J_2]$.

Hypercolumns

For a given image location $\langle x, y \rangle$ we concatenate all the feature vectors of all layers above the point $\langle x, y \rangle$.

$$\begin{aligned} & L \left[b, x, y, \sum_{\ell} J_{\ell} \right] \\ = & L_0 [b, x, y, J_0] \\ & \vdots \\ & ; L_{\ell} \left[b, \left\lfloor x \left(\frac{X_{\ell}}{X_1} \right) \right\rfloor, \left\lfloor y \left(\frac{Y_{\ell}}{Y_0} \right) \right\rfloor, J_{\ell} \right] \\ & \vdots \\ & ; L_{\mathcal{L}-1} [b, J_{\mathcal{L}-1}] \end{aligned}$$

Grouping

The input features and the output features are each divided into G groups.

$$L_{\ell+1}[b, x, y, J] = L_{\ell+1}^0[b, x, y, J/G]; \cdots ; L_{\ell+1}^{G-1}[b, x, y, J/G]$$

where we have G filters $W^g[\Delta X, \Delta Y, I/G, J/G]$ with

$$L_{\ell+1}^g[b, x, y, j]$$

$$= \sigma(W^g[\Delta X, \Delta Y, I/G, j]L_{\ell}^g[x + \Delta X, y + \Delta Y, I/G, j] - B^g[j])$$

This uses a factor of G fewer weights.

2D CNN in PyTorch

`conv2d(input, weight, bias, stride, padding, dilation, groups)`

input – tensor (minibatch,in-channels,iH,iW)

weight – filters (out-channels, in-channels/groups,kH,kW)

bias – tensor (out-channels) . Default: None

stride – Single number or (sH, sW). Default: 1

padding – Single number or (padH, padW). Default: 0

dilation – Single number or (dH, dW). Default: 1

groups – split input into groups. Default: 1

Modern Trends

Modern Convolutions use 3X3 filters. This is faster and has fewer parameters. Expressive power is preserved by increasing depth with many stride 1 layers.

Max pooling and dilation seem to have disappeared.

ResNet and resnet-like architectures are now dominant.

Alexnet, 2012

Given Input[227, 227, 3]

$$L_1[55 \times 55 \times 96] = \text{ReLU}(\text{CONV}(\text{Input}, \Phi_1, \text{width } 11, \text{pad } 0, \text{stride } 4))$$

$$L_2[27 \times 27 \times 96] = \text{MaxPool}(L_1, \text{width } 3, \text{stride } 2))$$

$$L_3[27 \times 27 \times 256] = \text{ReLU}(\text{CONV}(L_2, \Phi_3, \text{width } 5, \text{pad } 2, \text{stride } 1))$$

$$L_4[13 \times 13 \times 256] = \text{MaxPool}(L_3, \text{width } 3, \text{stride } 2))$$

$$L_5[13 \times 13 \times 384] = \text{ReLU}(\text{CONV}(L_4, \Phi_5, \text{width } 3, \text{pad } 1, \text{stride } 1))$$

$$L_6[13 \times 13 \times 384] = \text{ReLU}(\text{CONV}(L_5, \Phi_6, \text{width } 3, \text{pad } 1, \text{stride } 1))$$

$$L_7[13 \times 13 \times 256] = \text{ReLU}(\text{CONV}(L_6, \Phi_7, \text{width } 3, \text{pad } 1, \text{stride } 1))$$

$$L_8[6 \times 6 \times 256] = \text{MaxPool}(L_7, \text{width } 3, \text{stride } 2))$$

$$L_9[4096] = \text{ReLU}(\text{FC}(L_8, \Phi_9))$$

$$L_{10}[4096] = \text{ReLU}(\text{FC}(L_9, \Phi_{10}))$$

$$s[1000] = \text{ReLU}(\text{FC}(L_{10}, \Phi_s)) \quad \text{class scores}$$

VGG, 2014

Stanford CS231

Inception, Google, 2014

ResNet, 2015

[Kaiming He]

Imagenet Classification

1000 kinds of objects.

2016 is 3.0%, is 2017 2.25%

SOTA as of January 2020 is 1.3%

END