Gradient Descent

AND CNN

Error

- To perform back propagation on a neural network through gradient descent, some metric for the error of the approximation must be obtained. The most common error metric is the mean squared error, which is the average error squared. It is calculated by subtracting the theoretical value from every approximated value in the model, squaring the result and adding them up together to then divide by the amount of results. The squaring term is used to get rid of the negative errors and have a r e RMSE or root mean square emove the scaling.

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \tilde{y}_i\right)^2$$

Other Error Functions

- RMSE – Root of the MSE
- MAE - Mean absolute error – absolute value instead of squaring
- Cross Entropy loss – Used for classification
- Log loss – the log of the cross entropy loss

- Derive the derivative for each of these error functions
- Have each function next to their derivative and their surface
- Explain how each error function uses the information of distance between two points
- Create a graph of the sub operations of each error function
- Toeplitz matrix
- Hessian and jacobian

Linear Regression

- The simplest form of gradient descent can be demonstrated with linear regression.
- First the error is calculated which is the sum of every points distance to the line.
- Then the derivative of that error function is found with respect to the output function, which is in this case the line function y=mx+c.
- Finding the partial derivative of the error with respect to the slope (m) and the y intercept (c).

$$D_m = \frac{1}{n}\sum_{i=0}^{n} 2(y_i - (mx_i + c))(-x_i)$$

Partial Derivative wrt m, $$D_m = \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - \bar{y}_i)$$

Partial Derivative wrt c, $$D_c = \frac{-2}{n}\sum_{i=0}^{n} (y_i - \bar{y}_i)$$

Gradient Descent

- Gradient descent consists of taking partial derivatives of an error function with respect to parametrizable variables in the model. These derivatives tell us how we should change the variables to decrease the error.
- https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

Linear Regression

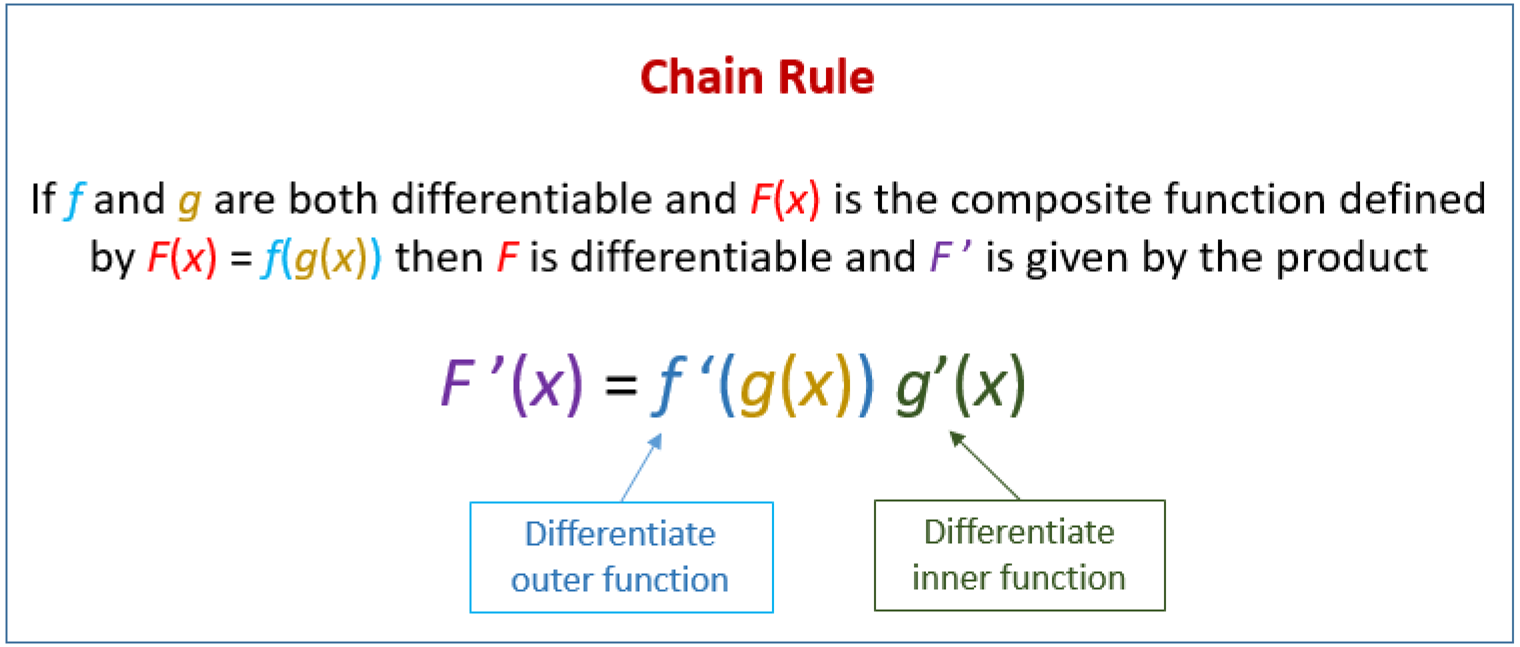
- To improve the line estimation, the derivatives are used to change the values of m and c; depending on the sign and magnitude of the derivative the changes in the variables vary.
- The derivative represents how the error changes with respect to local changes in the variable, so the new parametrization values are chosen towards the direction in which the derivative is negative.
- Calculate the new values of m and c which are the learning rate multiplied by the respective calculated derivatives.

$$m = m - L \times D_m$$

$$c = c - L \times D_c$$

Chain Rule

- The chain rule is used to calculate the gradient of a function inside another function.
- This gradient represents how the inner function is affecting the result produced by the outer function.
- It specifically denotes how the output changes in response to that variable or function.



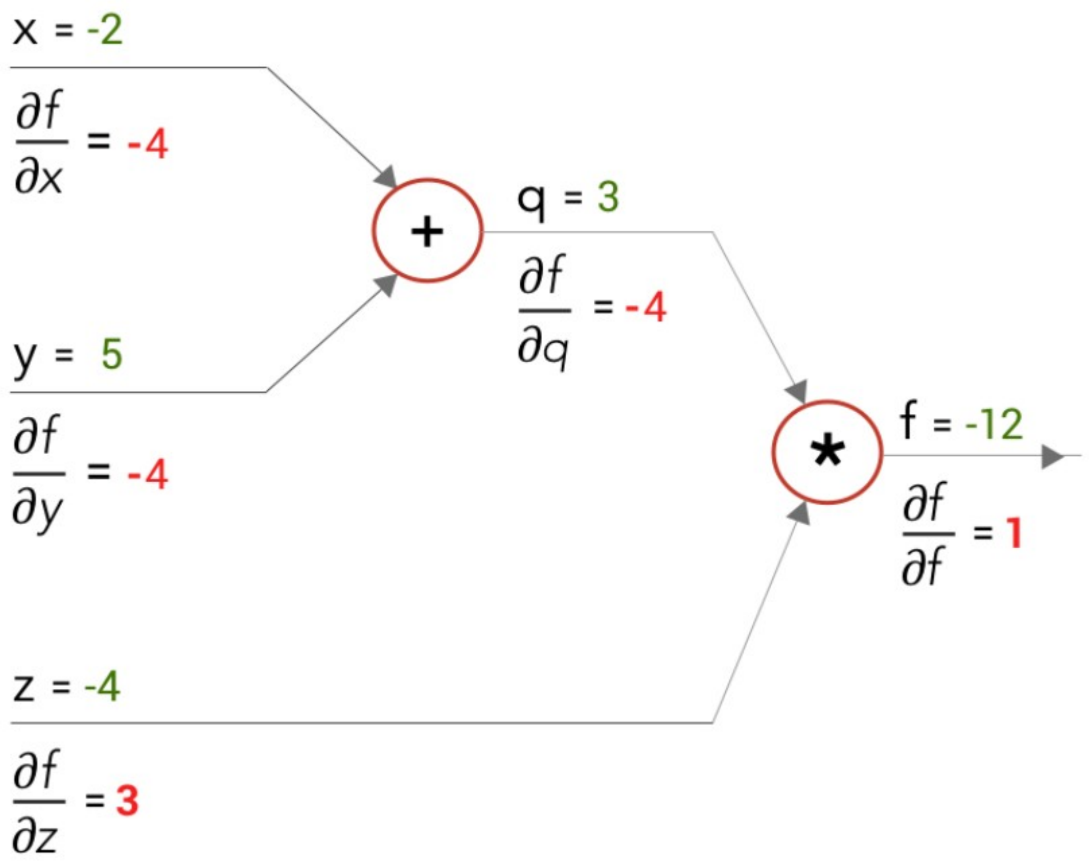
Chain Rule

If f and g are both differentiable and $F(x)$ is the composite function defined by $F(x) = f(g(x))$ then F is differentiable and F' is given by the product

$$F'(x) = \underbrace{f'(g(x))}_{\text{Differentiate outer function}}\ \underbrace{g'(x)}_{\text{Differentiate inner function}}$$

Partial Derivatives in a Network

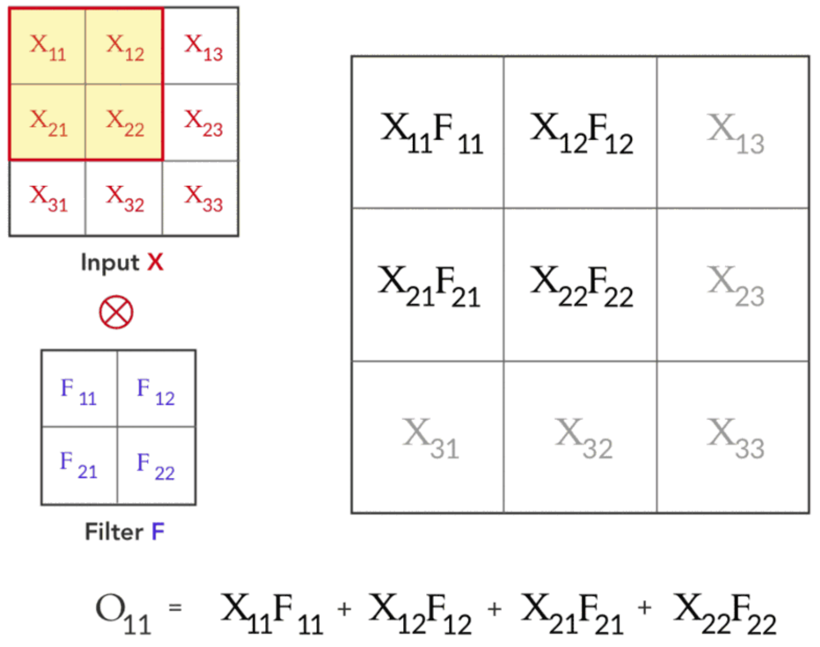
- Finding the partial derivatives of functions in a multi-layered network involves using the chain rule. The objective is to find how changes in each of the individual variables x, y, and z affect the output.
- The network on the right is composed of an addition gate followed by a multiplication gate.
- This is represented as an inner function which is adding terms nested inside another function which is multiplying that result by something else.
- The derivative of the outer function with respect to the addition gate q is -4, this is because the multiplier of that function is z = -4. Because of this, every unit change in q represents a change of that times 4 to the output. The same happens with z, its derivative is 3 because the value of q was 3.
- To calculate the partial derivatives of x and y the chain rule is applied. Since they are a part of an addition gate then their local derivatives are both one, as every unit change in their input produces a unit change locally. However, since these variables are inside q which has a derivative of -4, unitary changes in x or y will produce changes of -4 in the outer function. This is the effect that is represented with the chain rule, the derivatives outer functions are propagated into nested functions through multiplication.



$x = -2$
$\frac{\partial f}{\partial x} = -4$
$y = 5$
$\frac{\partial f}{\partial y} = -4$
$z = -4$
$\frac{\partial f}{\partial z} = 3$
+
$q = 3$
$\frac{\partial f}{\partial q} = -4$
*
$f = -12$
$\frac{\partial f}{\partial f} = 1$

Image Processing

- Convolutional neural networks use filters to fine tune the data that enters the model.
- Images are convoluted with filters to magnify the most relevant features.
- The filter is slid through the image pixel by pixel and the value of each output pixel is computed as the addition of the direct multiplications of each filter pixel with the input image pixel that it is being overlapped with. Each pixel in the output represents a linear combination of the filter with each one of the pixels it overlapped with in every position.



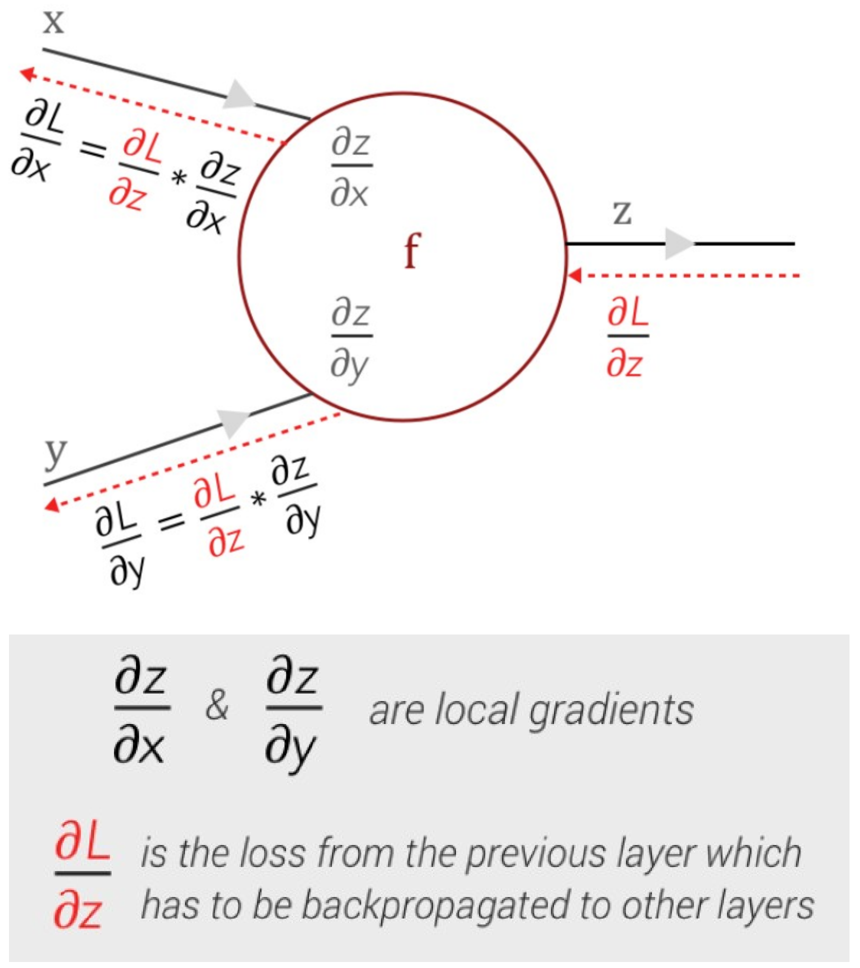
X_{11} X_{12} X_{13}
X_{21} X_{22} X_{23}
X_{31} X_{32} X_{33}
Input X
$\otimes$
F_{11} F_{12}
F_{21} F_{22}
Filter F
$X_{11}F_{11}$ $X_{12}F_{12}$ X_{13}
$X_{21}F_{21}$ $X_{22}F_{22}$ X_{23}
X_{31} X_{32} X_{33}
$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$

2D convolution Equation

$$f(x,y) * g(x,y) \quad = \quad \int_{\tau_1=-\infty}^{\infty} \int_{\tau_2=-\infty}^{\infty} f(\tau_1,\tau_2) \cdot g(x-\tau_1, y-\tau_2)\, d\tau_1\, d\tau_2$$

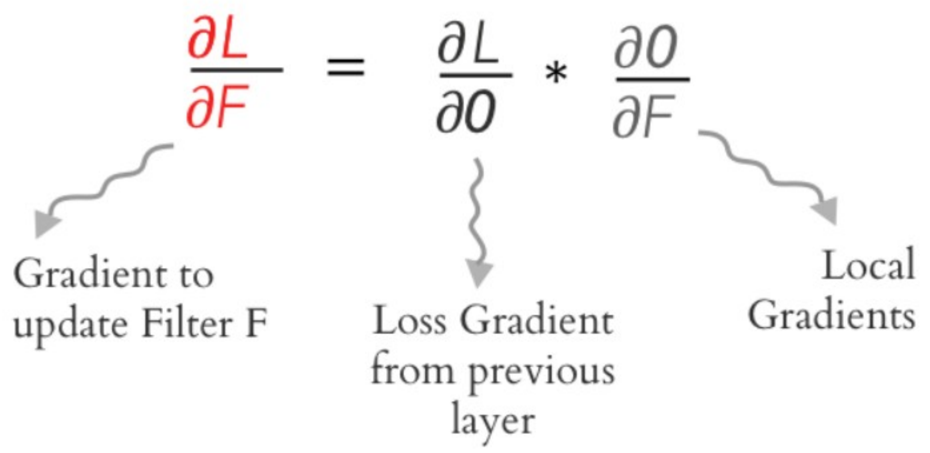
Gradient Descent In a CNN

- After computing the loss with the forward pass, we get an error function. In back propagation, we take the derivative of that error with respect to the outermost layer, and then we take the derivative of that layer with respect to the next. In this way we use the chain rule to figure out each function of how each variable in each layer changes the error


x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} * \frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z} * \frac{\partial z}{\partial y}$
$\frac{\partial z}{\partial x}$ & $\frac{\partial z}{\partial y}$ are local gradients
$\frac{\partial L}{\partial z}$ is the loss from the previous layer which has to be backpropagated to other layers

Partial Derivatives

- To minimize the error function, the partial derivatives of error are found with respect to each pixel in the filter.
- The chain rule expansion shows that this derivative is the derivative of the error (MSE) with respect to the output, multiplied by the derivative with respect to each local filter pixel.


$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial O} * \frac{\partial O}{\partial F}$
Gradient to update Filter F
Loss Gradient from previous layer
Local Gradients

For every element of F

$$\frac{\partial L}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial F_i}$$

Local Gradients

- As seen in the first equation, each output pixel is affected by different input and filter pixel values.
- The derivative of an output pixel with respect each pixel in the filter individually is the pixel that the filter pixel multiplied in the linear combination. If there are N filter pixels then there will be N partial derivatives each corresponding to one of the input pixels that was multiplied with its filter pixel to produce the output pixel.
- The derivative of the loss function with respect to each filter pixel (by the chain rule seen in previous slides) would correspond to the linear combination of the derivatives of the loss function with respect to each of the output pixels that the filter pixel affected, multiplied with the input pixels that were multiplied by that specific filter pixel to produce that same output pixel.

Local Gradients $\longrightarrow \textcircled{A}$

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

Finding derivatives with respect to F_{11}, F_{12}, F_{21} and F_{22}

$$\frac{\partial O_{11}}{\partial F_{11}} = X_{11} \quad \frac{\partial O_{11}}{\partial F_{12}} = X_{12} \quad \frac{\partial O_{11}}{\partial F_{21}} = X_{21} \quad \frac{\partial O_{11}}{\partial F_{22}} = X_{22}$$

Similarly, we can find the local gradients for O_{12}, O_{21} and O_{22}

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * X_{11} + \frac{\partial L}{\partial O_{12}} * X_{12} + \frac{\partial L}{\partial O_{21}} * X_{21} + \frac{\partial L}{\partial O_{22}} * X_{22}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * X_{12} + \frac{\partial L}{\partial O_{12}} * X_{13} + \frac{\partial L}{\partial O_{21}} * X_{22} + \frac{\partial L}{\partial O_{22}} * X_{23}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * X_{21} + \frac{\partial L}{\partial O_{12}} * X_{22} + \frac{\partial L}{\partial O_{21}} * X_{31} + \frac{\partial L}{\partial O_{22}} * X_{32}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * X_{22} + \frac{\partial L}{\partial O_{12}} * X_{23} + \frac{\partial L}{\partial O_{21}} * X_{32} + \frac{\partial L}{\partial O_{22}} * X_{33}$$

Convolution Perspective

- Calculating the partial derivatives of the error with respect to each of the filter pixels as mentioned previously is in fact a convolution between the input image and the derivatives of the loss with respect to each output pixel.
- The convolution is multiplying the loss derivatives with respect to each output pixel with the same values in the same order as the filter multiplied the original image.
- This operation can be extended to as many layers as needed because the chain rule carries the previous layer’s gradient to the next layer to obtain the partial derivatives of each variable in each layer with respect to the overall error function!

$$\begin{bmatrix} \frac{\partial L}{\partial F_{11}} & \frac{\partial L}{\partial F_{12}} \\ \frac{\partial L}{\partial F_{21}} & \frac{\partial L}{\partial F_{22}} \end{bmatrix} = \text{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} \right)$$

where

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} = \text{Input X} \qquad \begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} = \frac{\partial L}{\partial O}$$

Loss gradient from previous layer

Image Processing Gradient Descent

https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

- To find how the error changes with respect to the filter, by the chain rule we must find how the error changes with respect to the output and multiply that by how the output changes with respect to the filter.
- The change of the output with respect to the filter is the input, in an image this would be each pixel value. So the partial derivative of the error with respect to each pixel in the filter would be the sum of the partial derivatives of the error with respect to each output pixel times each input pixel for each pixel in the filter. That yields this linear combination:

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * X_{11} + \frac{\partial L}{\partial O_{12}} * X_{12} + \frac{\partial L}{\partial O_{21}} * X_{21} + \frac{\partial L}{\partial O_{22}} * X_{22}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * X_{12} + \frac{\partial L}{\partial O_{12}} * X_{13} + \frac{\partial L}{\partial O_{21}} * X_{22} + \frac{\partial L}{\partial O_{22}} * X_{23}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * X_{21} + \frac{\partial L}{\partial O_{12}} * X_{22} + \frac{\partial L}{\partial O_{21}} * X_{31} + \frac{\partial L}{\partial O_{22}} * X_{32}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * X_{22} + \frac{\partial L}{\partial O_{12}} * X_{23} + \frac{\partial L}{\partial O_{21}} * X_{32} + \frac{\partial L}{\partial O_{22}} * X_{33}$$

Simple Convolution Example

- In a simple case, the input vector x would be convolved with the kernel vector w.
- The weight vector strides across the input vector one row at a time, producing a total of three output values seen in the vector y.
- The values of y are line of weights and input va

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$\begin{aligned} y_1 &= w_1 x_1 + w_2 x_2 \\ y_2 &= w_1 x_2 + w_2 x_3 \\ y_3 &= w_1 x_3 + w_2 x_4 \end{aligned}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Simple Convolution Backpropagation

- To find the derivative of the error function with respect to the weights, the chain rule must be applied.
- Since error depends on the output and the outp depends on the weights the derivative with respect to the weights will be the derivative of the error with respect to the output multiplied by the derivative of the output with respect to the weights.
- In this simple example the derivative of the output with respect to the weights would be a 3x2 matrix, containing in each row the derivative of each value (pixel for imgs) of the output with respect to each weight.

$$dw = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w} = dy \cdot \frac{\partial y}{\partial w}$$

$$\frac{\partial y}{\partial w} = \begin{bmatrix} \frac{\partial y_1}{\partial w_1} & \frac{\partial y_1}{\partial w_2} \\ \frac{\partial y_2}{\partial w_1} & \frac{\partial y_2}{\partial w_2} \\ \frac{\partial y_3}{\partial w_1} & \frac{\partial y_3}{\partial w_2} \end{bmatrix}$$

Simple Convolution Backpropagation

- The derivative of each output value (pixel) with respect to the each weight is simply each input pixel.
- To find the derivative of the error with respect to each weight the matrix is multiplied by the derivatives of the error with respect to each output value (pixel).
- That operation yields an equation for the derivative of each weight which is the linear combination of each input value (pixel) with the derivative of the error with respect to each output value (pixel).
- **Notice that dw is a convolution of the input x with a filter dy (the derivative of the error with respect to the each output value).**

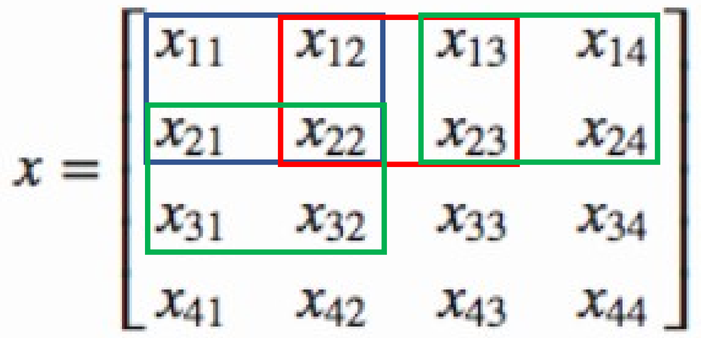
$$\frac{\partial y}{\partial w} = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \\ x_3 & x_4 \end{bmatrix}$$

$$\begin{bmatrix} dy_1 & dy_2 & dy_3 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \\ x_3 & x_4 \end{bmatrix}$$

$$dw_1 = x_1 dy_1 + x_2 dy_2 + x_3 dy_3$$
$$dw_2 = x_2 dy_1 + x_3 dy_2 + x_4 dy_3$$

Image Convolution Example

- Extending the last case to two dimensions, the case seen is the exact same as the convolution of an image with a filter (kernel).
- If no padding is applied the output will be a 3x3 matrix, as the filter is applied three times with horizontal striding and the same vertically.
- As the same case with the previous example, the resulting output pixels are a linear combination of the filter pixels with the input pixels, the amount of terms in each output value depends on the number of pixels there are in the filter.



$$x = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}$$

$$w = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$y = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix}$$

$$\begin{aligned} y_{11} &= w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22} \\ y_{12} &= w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23} \\ &\ldots \end{aligned}$$

Image Convolution Forward Propagation

- Representing the output with subscripts would look like two sums, one for each axis in the image. The sums would be the multiplication of each input pixel with each weight pixel, the weight is simply each index location from the sum count and the input pixel subscript consists of the sum of those index locations plus the location in the filter depicted by ij, the -1s are needed because the summations start at 1.

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}$$

$$w = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$y = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix}$$

$$y_{ij} = \left(\sum_{k=1}^{2} \sum_{l=1}^{2} w_{kl} x_{i+k-1,j+l-1} \right)$$

Image Convolution Backpropagation

- Applying the same chain rule as the last example, the derivative of the weight at each index is the derivative of the error with respect to the output at that index multiplied by the derivative of the output at that index with respect to the weight at that index.
- Considering the formula applied previously: the derivative of each output pixel with respect to each filter pixel will be the sums of the derivative of each weight pixel times the corresponding input pixel (the output) with respect to each weight pixel.

$$dw = \frac{\partial L}{\partial y_{ij}} \cdot \frac{\partial y_{ij}}{\partial w} = dy \cdot \frac{\partial y}{\partial w}$$

$$dw_{mn} = dy_{ij} \cdot \frac{\partial y_{ij}}{\partial w_{mn}}$$

$$y_{ij} = \left(\sum_{k=1}^{2} \sum_{l=1}^{2} w_{kl} x_{i+k-1,j+l-1} \right)$$

$$\frac{\partial y_{ij}}{\partial w_{mn}} = \sum_{k=1}^{2} \sum_{l=1}^{2} \frac{\partial w_{kl}}{\partial w_{mn}} x_{i+k-1,j+l-1}$$

Image Convolution Backpropagation

- Applying the same chain rule as the last example, the derivative of the weight at each index is the derivative of the error with respect to the output at that index multiplied by the derivative of the output at that index with respect to the weight at that index.

$$dw = \frac{\partial L}{\partial y_{ij}} \cdot \frac{\partial y_{ij}}{\partial w} = dy \cdot \frac{\partial y}{\partial w}$$

$$dw_{mn} = dy_{ij} \cdot \frac{\partial y_{ij}}{\partial w_{mn}}$$

- Considering the formula applied previously: The derivative of each output pixel with respect to each filter pixel will be the sums of the derivative of each weight pixel times the corresponding input pixel (the output) with respect to each weight pixel. In this case the sum corresponds to a 2x2 filter, it can be NxN.

$$y_{ij} = \left(\sum_{k=1}^{2} \sum_{l=1}^{2} w_{kl} x_{i+k-1,j+l-1} \right)$$

$$\frac{\partial y_{ij}}{\partial w_{mn}} = \sum_{k=1}^{2} \sum_{l=1}^{2} \frac{\partial w_{kl}}{\partial w_{mn}} x_{i+k-1,j+l-1}$$

Image Convolution Backpropagation

- Notice that the sums are only counting the k and l terms, and the bottom w term inside the sum has indexes m and n. This means that that term is constant throughout all the sum. In other words, this sum is occurring once for every single pixel in the filter. Basically, it is iterating through each output pixel deriving with respect to one weight pixel.

$$\frac{\partial y_{ij}}{\partial w_{mn}} = \sum_{k=1}^{2} \sum_{l=1}^{2} \frac{\partial w_{kl}}{\partial w_{mn}} x_{i+k-1,j+l-1}$$

- The result is that $\frac{\partial w_{kl}}{\partial w_{mn}}$ will be 0 because the w_{kl} will be a different pixel than w_{mn}. Except where (k,l)=(m,n) in which case $\frac{\partial w_{kl}}{\partial w_{mn}}$ will be 1.

Image Convolution Backpropagation

- In each case where the $\frac{\partial w_{kl}}{\partial w_{mn}}$ term is equal to one, the formula on the right is what is the result. This is the remaining term from each double sum.
- Considering the chain rule formula derived previously, the dw$_{mn}$ will be the double sum (representing the size of the output) of the derivative of the error with respect to each output pixel multiplied by the corresponding pixels that it affected during the forward pass. The k and l terms correspond to the indices where there was a match in the last sum, these were the indices of the input pixels that were affected by that filter pixel.

$$\frac{\partial y_{ij}}{\partial w_{mn}} = x_{i+k-1,j+l-1}$$

$$dw = \frac{\partial L}{\partial y_{ij}} \cdot \frac{\partial y_{ij}}{\partial w} = dy \cdot \frac{\partial y}{\partial w}$$

$$dw_{mn} = dy_{ij} \cdot \frac{\partial y_{ij}}{\partial w_{mn}}$$

$$dw_{mn} = dy_{ij} \cdot x_{i+k-1,j+l-1}$$

$$\Rightarrow dw_{mn} = \sum_{i=1}^{3} \sum_{j=1}^{3} dy_{ij} \cdot x_{i+k-1,j+l-1}$$

Image Convolution Backpropagation

- In the same way as the simpler case, the derivative of the error with respect to each filter pixel ends up being the convolution between the input image and the derivatives of the error with respect to each output pixel. In this case the input image was a 4x4 and the filter was a 2x2, which were convolved with no padding to produce a 3x3 output image, therefore the resulting matrix of the partial derivatives of the error with respect to each output pixel is also a 3x3 matrix, as there is one derivative per pixel.

$$dw = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} * \begin{bmatrix} dy_{11} & dy_{12} & dy_{13} \\ dy_{21} & dy_{22} & dy_{23} \\ dy_{31} & dy_{32} & dy_{33} \end{bmatrix}$$

$$dw = x * dy$$

Algorithm Implementations with mathematical derivations and explanations

- https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199
- https://towardsdatascience.com/backpropagation-in-a-convolutional-layer-24c8d64d8509

Convolution

Mathematical Meaning

- Convolution is the sum of the dot products of two functions.
- For discrete functions, a simple summation of the multiplication is taken on each timestamp.
- For continuous functions the same dot product is taken over an integral with infinitesimal steps dτ, evaluated from -∞< τ < ∞.
- The t- τ in g represents a shift in the equation by t and the -τ represents a flip along the y axis.

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k].$$

$$(f^{*}g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\tau'$$

$(f^{*}g)(t)$	= functions that are being convoluted
t	= real number variable of functions f and g
$g(\tau)$	= convolution of the function f(t)
τ'	= first derivative of g(tau) function

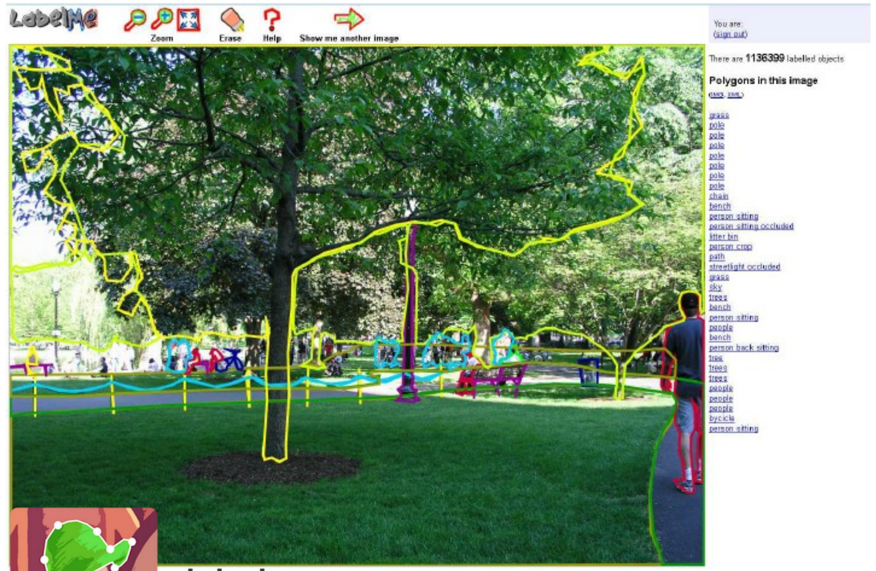
CV Datasets

- Before 2012 datasets were in the order of tens of thousands of images.
- New Datasets

LabelMe: Hundreds of thousands of fully-segmented images

ImageNet: Over 15 million labeled high-resolution images in over 22,000 categories.

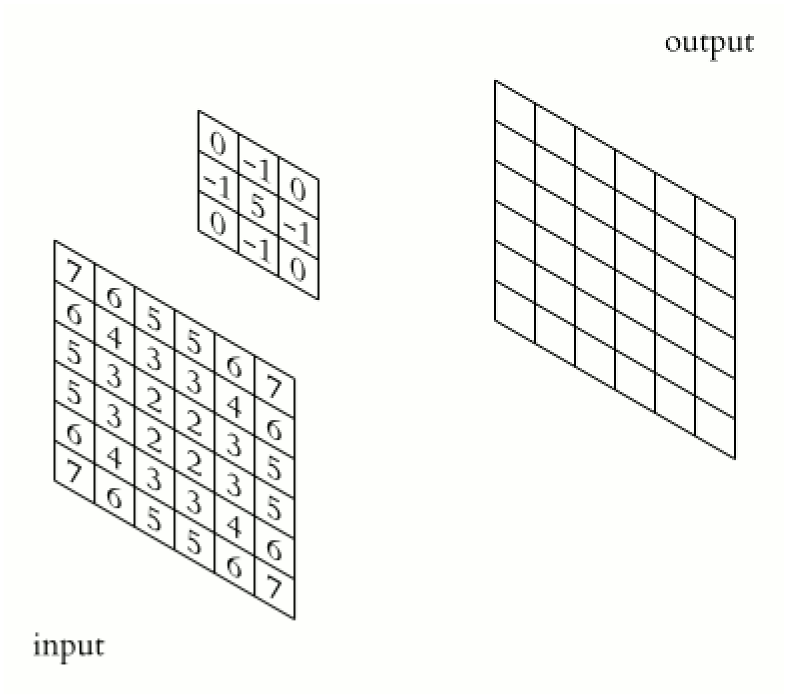
Imagenet

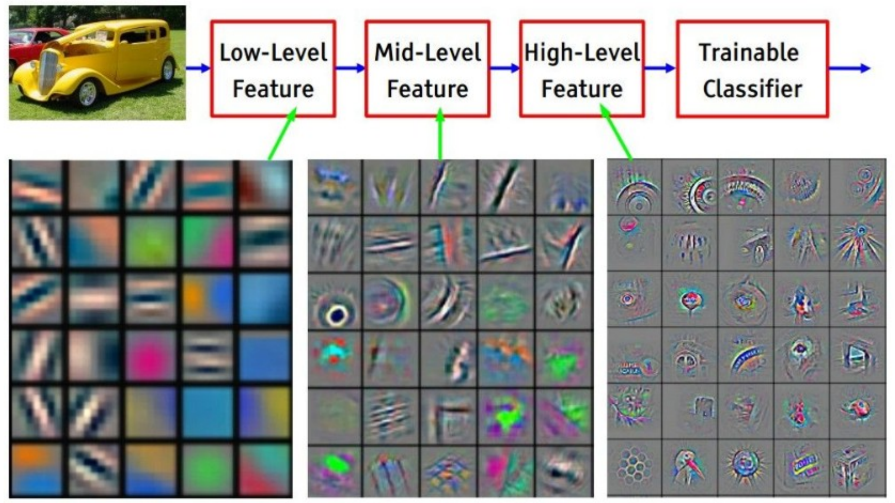


labelme

CNN Layers

- Deep training with the number of parameters in a fully connected network is unhandleable.
- The Dimensionality of every weight vector must be equal to the number of neurons in the previous layer, or the input dimensions.
- The convolution allowed networks to reuse weights along the image so more operations can be done on less weights, enabling more layers to be added to networks, this allows for hierarchical pattern recognition.





Manifold Theory

- One's observed data lie on a low-dimensional manifold embedded in a higher-dimensional space.