TTIC 31230, Fundamentals of Deep Learning

David McAllester

Architectural Elements That Improve SGD

ReLu

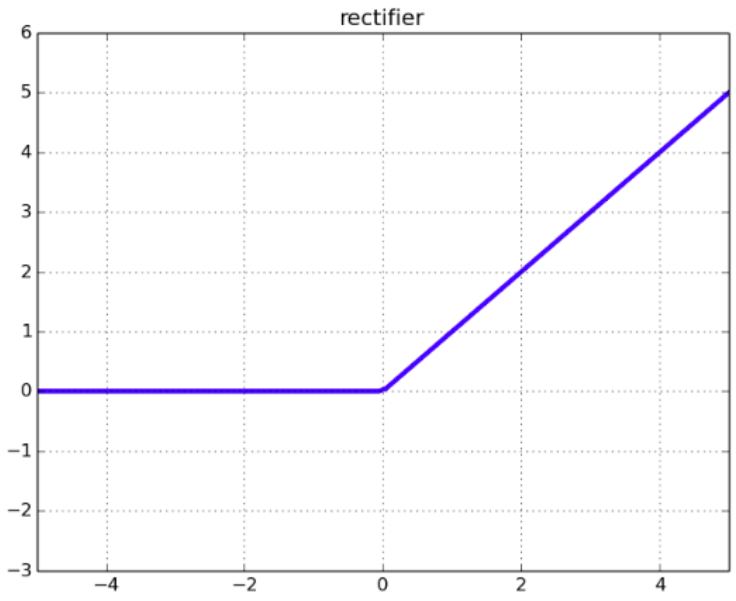
Initialization

Normalization

Residual Connections

ReLu

$\mathrm{ReLu}(x) = \max(x, 0)$



The ReLu does not saturate at positive inputs.

Problems with Depth: Repeated Multiplication by Network Weights

Consider

$$y = \sum_i w[i]x[i] \quad = \quad W[I]x[I]$$

If the weights are large the activations will grow exponentially in network depth.

If the weights are small the actvations will become exonentially small.

Problems with Depth:
Repeated Multiplication by Network Weights

Exploding activations cause exploding gradients.

$$y \mathrel{\texttt{+=}} w[i]x[i]$$

$$w.grad \mathrel{\texttt{+=}} y.grad\,x[i]$$

The size of $w[i]$.grad is proportional to $x[i]$

He Initialization

Initialize weights to preserve zero-mean unit variance value distributions.

$$y = \sum_i w[i]x[i]$$

If we assume x_i has zero mean and unit variance then we want y to have zero mean and unit variance (over random training points).

He initialization randomly draws $w[i]$ from

$$\mathcal{N}(0, \sigma^2) \quad \sigma = \sqrt{1/N}$$

He Initialization

$$y = \sum_i w[i]x[i]$$

$$w[i] \sim \mathcal{N}(0, \sigma^2) \quad \sigma = \sqrt{1/N}$$

Assuming independence we have that y has zero mean and unit variance.

Batch Normalization

For CNNs we convert a tensor $L[b,x,y,n]$ to $\tilde{L}[b,x,y,n]$ as follows.

$$\hat{L}[x,y,n] = \frac{1}{B}\sum_b L[b,x,y,n]$$

$$\hat{\sigma}[x,y,n] = \sqrt{\frac{1}{B-1}\sum_b (L[b,x,y,n] - \hat{L}[x,y,n])^2}$$

$$\tilde{L}[b,x,y,n] = \frac{L[b,x,y,n] - \hat{L}[x,y,n]}{\hat{\sigma}[x,y,n]}$$

Spatial Batch Normalization

$$\hat{L}[n] = \frac{1}{BXY} \sum_{b,x,y} L[b,x,y,n]$$

$$\hat{\sigma}[n] = \sqrt{\frac{1}{BXY-1} \sum_{b,x,y} (L[b,x,y,n] - \hat{L}[n])^2}$$

$$\tilde{L}[b,x,y,n] = \frac{L[b,x,y,n] - \hat{L}[n]}{\hat{\sigma}[n]}$$

Layer Normalization

$$\hat{L}[b,n] = \frac{1}{XY}\sum_{x,y} L[b,x,y,n]$$

$$\hat{\sigma}[b,n] = \sqrt{\frac{1}{XY-1}\sum_{x,y}(L[b,x,y,n]-\hat{L}[b,n])^2}$$

$$\tilde{L}[b,x,y,n] = \frac{L[b,x,y,n]-\hat{L}[b,n]}{\hat{\sigma}[b,n]}$$

Adding an Affine Transformation

$$\breve{L}[b,x,y,n] = \gamma[n]\tilde{L}[b,x,y,n] + \beta[n]$$

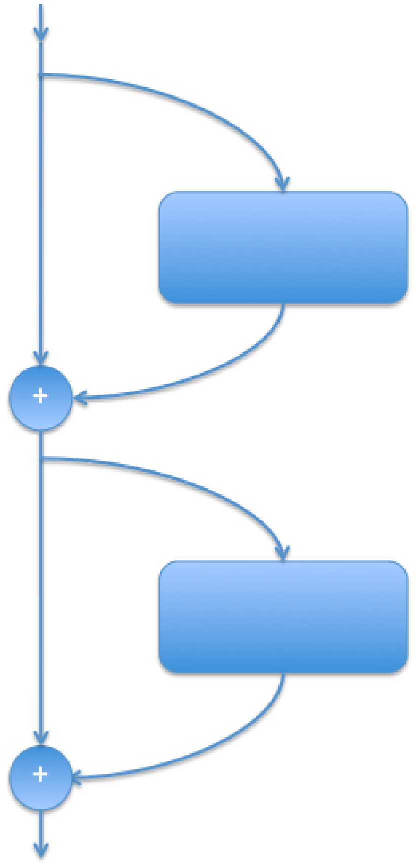
Here $\gamma[n]$ and $\beta[n]$ are parameters of the batch normalization.

This allows the batch normlization to learn an arbitrary affine transformation (offset and scaling).

The affine transformation can undo the normaliztion but using ReLu activations the normalized value remains independent of scaling the weights and bias terms (thresholds) of the layer.

Residual Connections

Deep Residual Networks (ResNets) by Kaiming He 2015



A residual connections connects input to output directly and hence preserves gradients.

ResNets were introduced in late 2015 (Kaiming He et al.) and revolutionized computer vision.

Residual Connections in CNNs

$$\tilde{L}_{\ell+1}[B,X,Y,N_{\mathrm{out}}]$$

$$= \mathrm{Conv}(K_{\ell+1}[N_{\mathrm{out}},\Delta X,\Delta Y,N_{\mathrm{in}}],B_{\ell+1}[N_{\mathrm{out}}],L_{\ell}[B,X,Y,N_{\mathrm{in}}])$$

$$L_{\ell+1}[B,X,Y,N_{\mathrm{out}}] = L_{\ell}[B,X,Y,N_{\mathrm{in}}] + \tilde{L}_{\ell+1}[B,X,Y,N_{\mathrm{out}}]$$

Capital letters indicate that complete tensors.

These equations require that the spacial dimension remains the same (stide 1) and $N_{\mathrm{out}} = N_{\mathrm{in}}$.

Residual Connections in CNNs

The residual connection typically skips over several layers, or in transformers, a complex multi-level network.

$$\begin{aligned}&\tilde{L}_{\ell+1}[B, X, Y, N_{\mathrm{out}}]\\ =\;&\mathrm{Conv}(K_{\ell+1}[N, \Delta X, \Delta Y, N], B_{\ell+1}[N], L_{\ell}[B, X, Y, N])\end{aligned}$$

$$\begin{aligned}&\tilde{L}_{\ell+2}[B, X, Y, N]\\ =\;&\mathrm{Conv}(K_{\ell+1}[N, \Delta X, \Delta Y, N], B_{\ell+1}[N], L_{\ell+1}[B, X, Y, N])\end{aligned}$$

$$L_{\ell+2}[B, X, Y, N] = L_{\ell}[B, X, Y, N] + \tilde{L}_{\ell+2}[B, X, Y, N]$$

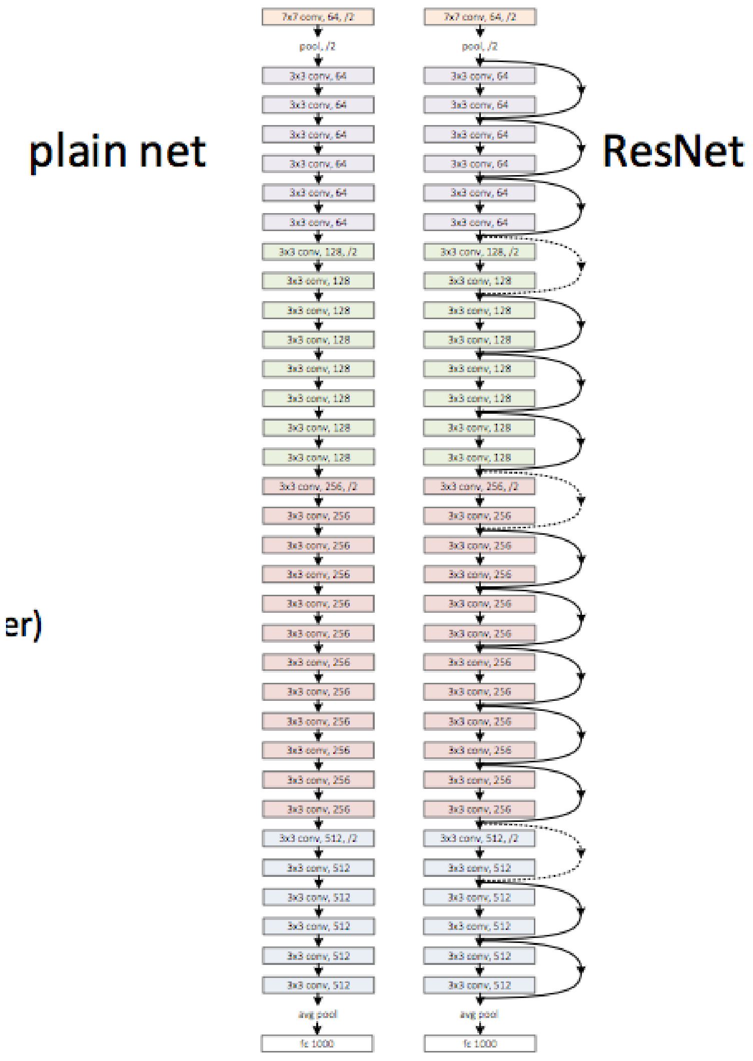
Handling Spacial Reduction

Spacial reduction and neuron expansion is done without convolution.

$$L_{\ell+1}[b,x,y,j] = \begin{cases} L_{\ell}[b, s*x, s*y, j] & \text{for } j < N_{\ell} \\ 0 & \text{otherwise} \end{cases}$$

Residual connections are still placed around all convolutions.

Resnet32



[Kaiming He]

Deeper Versions use Bottleneck Residual Paths

We reduce the number of neurons to $N_{\mathrm{bottle}} < N$ before doing the convolution.

$$\begin{aligned}
U[B,X,Y,N_{\mathrm{bottle}}] &= \mathrm{Conv}(K^U[N_{\mathrm{bottle}},1,1,N_{\mathrm{in}}],L_\ell[B,X,Y,N]) \\
V[B,X,Y,N_{\mathrm{bottle}}] &= \mathrm{Conv}(K^V[N_{\mathrm{bottle}},3,3,N_{\mathrm{bottle}}],U[B,X,Y,N_{\mathrm{bottle}}]) \\
R[B,X,Y,N] &= \mathrm{Conv}(K^R[N,1,1,N_{\mathrm{bottle}}],V[B,X,Y,N]) \\
L_{\ell+1} &= L_\ell + R
\end{aligned}$$

A General Residual Connection

$$y = x + R(x)$$

where $R(x)$ has the same shape as x.

Improving Trainability

ReLu

Initialization

Normalization

Residual Connections

END