Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Assignment-III

Akash Tadwai - ES18BTECH11019

March 11, 2021

1 ResNet

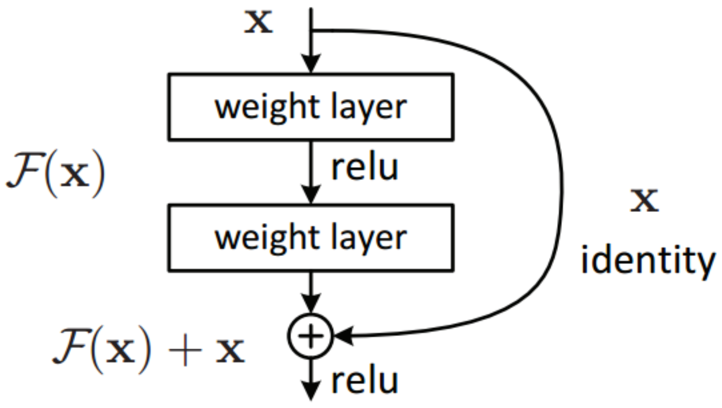
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 1: Basic Resnet Block

- In forward pass,
  - Let the input be x
  - The output from the first layer be z_1
  - The output from the second layer be z_2
- Clearly, $\mathcal{F}(x) = z_2$, let $z_3 = z_2 + x$ and final output is $y = ReLU(\mathcal{F}(x) + x) = ReLU(z_3)$.
- In backward pass, We assume we have $\frac{\partial L}{\partial y}$ from previous layer.

$$\begin{aligned} \frac{\partial L}{\partial z_3} &= \frac{\partial L}{\partial y} * \frac{\partial y}{\partial z_3} \\ &= \frac{\partial L}{\partial y} * (z_3 > 0) \end{aligned} \tag{1}$$

$$
\begin{aligned}
\frac{\partial L}{\partial z_2} &= \frac{\partial L}{\partial z_3} * \frac{\partial z_3}{\partial z_2} \\
&= \frac{\partial L}{\partial y} * (z_3 > 0) \\
\frac{\partial L}{\partial z_1} &= \frac{\partial L}{\partial z_2} * \frac{\partial z_2}{\partial z_1} \\
&= \frac{\partial L}{\partial y} * (z_3 > 0) * \frac{\partial z_2}{\partial z_1} \\
\frac{\partial L}{\partial x} &= \frac{\partial L}{\partial y} * \frac{\partial y}{\partial x} \\
&= \frac{\partial L}{\partial y} * \frac{\partial y}{\partial z_3} * \frac{\partial z_3}{\partial x} \\
&= \frac{\partial L}{\partial y} * (z_3 > 0) * \frac{\partial (z_2 + x)}{\partial x} \\
&= \frac{\partial L}{\partial y} * (z_3 > 0) * (1 + \frac{\partial z_2}{\partial x}) \\
&= \boxed{\frac{\partial L}{\partial y} * (z_3 > 0) * (1 + \frac{\partial z_2}{\partial z_1} * \frac{\partial z_1}{\partial x})}
\end{aligned}
$$

2 Receptive Field

- Let the input to the first 3×3 convolutional layer with stride 1 and no pooling have dimension $n \times n \times c$. Then the resultant dimension on convolving with one filter is:

$$\lfloor \frac{n - f + 2p}{s} \rfloor + 1 = \frac{n - 3 + 0}{1} + 1 = n - 2$$

  Hence, with each convolution the dimension changes as follows:

$$(n \times n) \rightarrow (n - 2 \times n - 2)$$

- Going from the 4th layer of the network, let it be of size 1×1. Then the size of the third layer is $(1 + 2) \times (1 + 2) = 3 \times 3$. The third layer is obtained by convolving the second layer of size $(3 + 2) \times (3 + 2) = 5 \times 5$. Doing this four times, we get:

$$1 \times 1 \leftarrow 3 \times 3 \leftarrow 5 \times 5 \leftarrow 7 \times 7 \leftarrow 9 \times 9$$

  Hence, the support is $9 \times 9 = \mathbf{81}\ \textit{pixels}$.

Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

3 Effect on Bias and Variance

- Adding more hidden units will *decrease bias and increase variance.*
- Increasing the number of hidden units will make the model more complex and the model will be able to approximate complex functions. A complex model may lead to **overfitting** the train data and the bias of the model will **decrease**.

4 Sigmoid and Tanh

Based on definition of tanh(·), we can obtain:

$$\begin{aligned}\tanh(a) &= \frac{e^a - e^{-a}}{e^a + e^{-a}} \\ &= -1 + \frac{2e^a}{e^a + e^{-a}} \\ &= -1 + 2\frac{1}{1 + e^{-2a}} \\ &= 2\sigma(2a) - 1\end{aligned}$$

The original Neural network is given by,

$$y_k(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{M} w_{kj}^{(2)} \sigma \left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \tag{2}$$

Lets say we have $w_{ji}^{(1s)}, w_{j0}^{(1s)}$ and $w_{kj}^{(2s)}, w_{k0}^{(2s)}$ as parameters for a network whose hidden units use logistic sigmoid function as activation and $w_{ji}^{(1t)}, w_{j0}^{(1t)}$ and $w_{kj}^{(2t)}, w_{k0}^{(2t)}$ for another one using tanh(·), for the network using tanh(·) as activation, we can write down the following expression for the *kth* neuron in the last layer by using (2):

$$\begin{aligned}a_k^{(t)} &= \sum_{j=1}^{M} w_{kj}^{(2t)} \tanh\left(a_j^{(t)}\right) + w_{k0}^{(2t)} \\ &= \sum_{j=1}^{M} w_{kj}^{(2t)} \left[2\sigma\left(2a_j^{(t)}\right) - 1\right] + w_{k0}^{(2t)} \\ &= \sum_{j=1}^{M} 2w_{kj}^{(2t)} \sigma\left(2a_j^{(t)}\right) + \left[-\sum_{j=1}^{M} w_{kj}^{(2t)} + w_{k0}^{(2t)}\right]\end{aligned}$$

What's more, we also have :

$$a_k^{(s)} = \sum_{j=1}^{M} w_{kj}^{(2s)} \sigma\left(a_j^{(s)}\right) + w_{k0}^{(2s)}$$

To make the two networks equivalent, i.e., $a_k^{(s)} = a_k^{(t)}$, we should make sure:

$$\begin{cases} a_j^{(s)} = 2a_j^{(t)} \\ w_{kj}^{(2s)} = 2w_{kj}^{(2t)} \\ w_{k0}^{(2s)} = -\sum_{j=1}^{M} w_{kj}^{(2t)} + w_{k0}^{(2t)} \end{cases}$$

Note that the first condition can be achieved by simply enforcing:

$$w_{ji}^{(1s)} = 2w_{ji}^{(1t)}, \quad \text{and} \quad w_{j0}^{(1s)} = 2w_{j0}^{(1t)}$$

Therefore, these two networks are equivalent under a linear transformation and she is correct.

5 Quadratic Error Function and Hessian

$$E(\mathbf{w}) = E\left(\mathbf{w}^{\star}\right) + \frac{1}{2}\left(\mathbf{w} - \mathbf{w}^{\star}\right)^{\mathrm{T}} \mathbf{H}\left(\mathbf{w} - \mathbf{w}^{\star}\right) \tag{1}$$

where the Hessian $\mathbf{H}$ is evaluated at $\mathbf{w}^{\star}$. Lets consider the eigenvalue equation for the Hessian matrix

$$\mathbf{H}\mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{2}$$

where the eigenvectors $\mathbf{u}_i$ form a complete orthonormal set so that

$$\mathbf{u}_i^{\mathrm{T}} \mathbf{u}_j = \delta_{ij} \tag{3}$$

We now expand $(\mathbf{w} - \mathbf{w}^{\star})$ as a linear combination of the eigenvectors in the form

$$\mathbf{w} - \mathbf{w}^{\star} = \sum_i \alpha_i \mathbf{u}_i \tag{4}$$

This can be regarded as a transformation of the coordinate system in which the origin is translated to the point $\mathbf{w}^{\star}$, and the axes are rotated to align with the eigenvectors (through the orthogonal matrix whose columns are the $\mathbf{u}_i$), Substituting (4) into (1) , and using (2) and (3), allows the error function to be written in the form

$$E(\mathbf{w}) = E\left(\mathbf{w}^{\star}\right) + \frac{1}{2}\sum_i \lambda_i \alpha_i^2$$

To obtain the contour, we enforce $E(\mathbf{w})$ to equal to a constant C

$$E(\mathbf{w}) = E\left(\mathbf{w}^{*}\right) + \frac{1}{2}\sum_i \lambda_i \alpha_i^2 = C$$

We rearrange the equation above, and then obtain:

$$\sum_i \lambda_i \alpha_i^2 = B$$

Where $B = 2C - 2E(\mathbf{w}^*)$ is a constant. Therefore, the contours of constant error are ellipses whose axes are aligned with the eigenvector $\mathbf{u_i}$ of the Hessian Matrix $\mathbf{H}$. The length for the j th axis is given by setting all $\alpha_i = 0, \mid \mathrm{i} \neq j$

$$\alpha_j = \sqrt{\frac{B}{\lambda_j}}$$

In other words, the length is inversely proportional to the square root of the corresponding eigenvalue λ_j

6 Developing DeepLearning Model

- I would suggest them to use **Transfer Learning**.
- Since pretrained model (deep learning model deployed at Olympic National Park, Washington trained on 1 million images from 1000 classes) with good results is readily available, they can reduce the time spent on training, hyperparameter tuning and thus need for high-end computing hardware.
- Since the Dataset is small; target and source datasets are similar:
  - Specialized features likely remain same for source and target datasets. Hence we only need to change the final classification layer.
  - As there are totally 200 species, we remove the last layer of the Olympic National Park model and we insert the 200 neurons in the last layer and train only the last layer by keeping other weights **frozen**.

For the LaTeXtyped version of this Report visit : https://www.overleaf.com/read/tngzhnsrrnjf

******THE END******