Deep Learning Notes 9.2 & 9.3

Key Words : Sparse Interaction; Parameter Sharing; Equivariant Representation; Pooling

1. Sparse Interaction

1. Traditional neural network layers use matrix multiplication describing the interaction between each input unit and each output unit. This means every output unit interacts with every input unit.

2. In order to reduce the memory requirements of the model and improve its statistical efficiency, **Sparse Interactions** can be used in CNN. This is accomplished by making the kernel smaller than the input. For example, if we have m inputs and n outputs, then matrix multiplication requires $m \times n$ parameters and the algorithms used in practice have $\mathcal{O}(m \times n)$ runtime. If we limit the number of connections each output may have to k, then the sparsely connected approach requires only $k \times n$ parameters and $\mathcal{O}(k \times n)$ runtime. Typically we let k **several orders of magnitude smaller than** m to obtain good performance. In a deep convolutional network, units in the deeper layers may indirectly interact with **a larger portion of** the input.

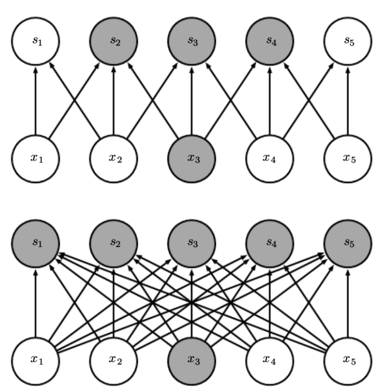

Figure 1: (Top) **Sparse Interactions**: only three outputs are affected by x_3.(Bottom) **Fully Interactions**: all of the outputs are affected by x_3.

2. Parameter Sharing

3. **Parameter Sharing** refers to **using the same parameter for more than one function in a model**.

4. In a convolutional neural net, each member of the kernel is used at every position of the input. The parameter sharing used by the convolution operation means that rather than learning a separate set of parameters for every location, we learn only one set.

5. So it does further reduce the storage requirements of the model to k parameters. Convolution is thus dramatically more efficient than dense matrix multiplication in terms of the memory requirements and statistical efficiency.

3. Equivariant Representation

6. To say a function is equivariant means that **if the input changes, the output changes in the same way**. For example, if pattern $[0, 3, 2, 0, 0]$ on the input results in $[0, 1, 0, 0]$ in the output, then the pattern $[0, 0, 3, 2, 0]$ might lead to $[0, 0, 1, 0]$. It is also noteworthy that **Invariant** to translation means that a translation of input features doe not change the outputs at all. So if your pattern $[0, 3, 2, 0, 0]$ on the input results in $[0, 1, 0]$ in the output, then the pattern $[0, 0, 3, 2, 0]$ would also lead to $[0, 1, 0]$.

7. A function $f(x)$ is equivariant to a function g if

$$f(g(x)) = g(f(x)) \tag{1}$$

8. In the case of convolution, if we let g be any function that translates the input, i.e., shifts it, then the convolution function is equivariant to g.

4. Pooling

9. A **Pooling** function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. A typical layer of a convolutional network consists of three stages as follows

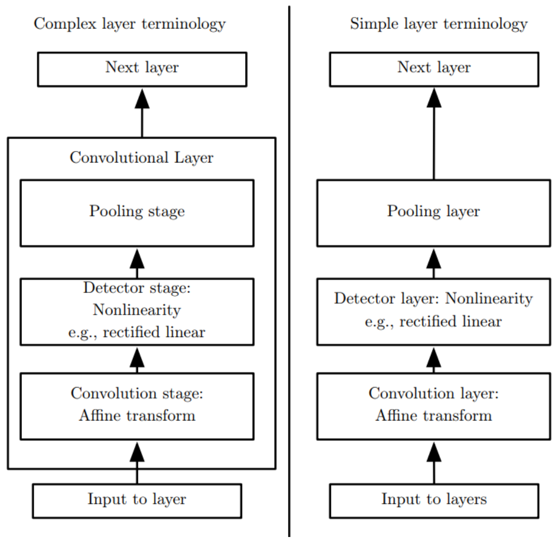

Figure 2: There are two commonly used sets of terminology for describing these layers, we often the left one.

10. Pooling helps to make the representation become approximately **invariant** to small translations of the input. Invariance to translation means that **if we translate the input by a small amount, the values of most of the pooled outputs do not change**.

11. The max pooling and average pooling method as follows

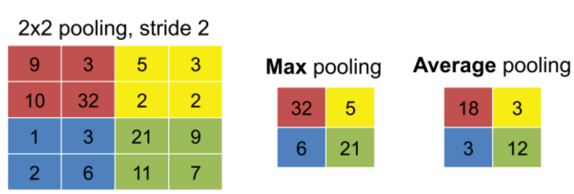

Figure 3: The max pooling and average pooling method.

12. Fig.3 is from V. Sze, Y. Chen, T. Yang and J. S. Emer, 'Efficient Processing of Deep Neural Networks: A Tutorial and Survey,' in *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295-2329, Dec. 2017.