Practitioner's guide for Neural Networks
EE425X - Machine Learning: A signal processing persepective

1 Main Architectures

1.1 Fully Connected Networks

The simplest model for a deep neural network is a Fully Connected Neural Network. This constitutes one input layer, one output layer and several hidden layers. Each layer consists of multiple perceptrons (interchangeably referred to as neurons in these notes) that has been discussed earlier. And thus this is also referred to as a multi layer perceptron (MLP). A schematic of this network is shown in Fig. 1.

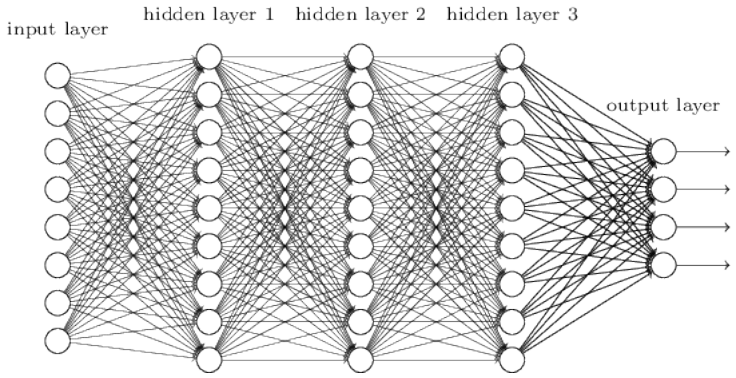

Figure 1: Image downloaded from `http://neuralnetworksanddeeplearning.com`

- **Mathematical Formulation:** Let the training data be $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ $\boldsymbol{x}_i$ are n-dimensional input vectors and $\boldsymbol{y}_i$ are "outputs" (labels/regression value etc.) for $i = 1, \cdots, m$. Let the number of neurons in the i-th layer be denoted by h_i and define $h_0 = n$ (for the input layer). The weights connecting the $i-1$-th and i-th layer is denoted by a matrix $\boldsymbol{W}_i \in \mathbb{R}^{h_{i-1} \times h_i}$. Now the output of the first (input) layer is given as $\sigma(\boldsymbol{x})$ where $\sigma(\cdot)$ denotes the element-wise activation function. In recent years, in practice the choice of σ is a rectified Linear Unit also referred to as ReLU. In this case, $\sigma(x) = \text{sign}(x)$ which is x if $x \geq 0$ and 0 otherwise. The output of first hidden layer thus becomes, $\boldsymbol{x}^{(1)} = \sigma(\boldsymbol{W}_1 \sigma(\boldsymbol{x}^{(0)}))$ and similarly, the output of the i-th hidden layer is $\boldsymbol{x}^{(i)} = \sigma(\boldsymbol{W}_i \sigma(\boldsymbol{x}^{(i-1)}))$.

- **How to decide the architecture?** The input layer consists of n neurons where n is the dimensionality of the input. The output layer can have varying number of neurons. Eg. (a) for a (linear) regression problem, the output layer will have one neuron; (b) for a classification problem, typically, the number of output neurons is equal to the number of classes and the output i corresponds to the probability of a particular example belonging to class i; (c) for inverse problems such as image denoising, the number of output layers will be equal to the number of input layers.

  Choosing the number of hidden layers is more involved. If the number of layers is too small, then the neural network cannot learn the underlying input-output mapping well enough, whereas if the number of layers is too large, then the network will tend to overfit on the training data and will be unable to

generalize to the test data. This is typically determined through observing the bias-variance trade-offs for various number of hidden layers and then picking the best architecture. Choosing the number of neurons on each hidden layer is also performed the same way. The empirical consensus within the deep learning community is that the number of hidden neurons must be sufficiently large so that (i) the neural network can be trained through a simple enough algorithm and (ii) the expressivity of the network is enough to capture the complex input-output mapping of the data.

- **When is this architecture used?** Fully connected neural networks were extensively used for tasks such as image classification and speech recognition in the early 1990s-2000s. However, recently, they have been replaced with more advanced architectures that leverage the structure of the input data to better perform tasks. These architectures will be discussed subsequently.

1.2 Convolutional Neural Networks

As the name suggests, a major building block of Convolutional Neural Networks (CNNs) are the convolution operation learnt in Signals and Systems. This is very useful when the task at hand includes images. This idea has been used in tasks such as image recognition, image classificiation, image captioning and so on. A simple schematic of one convolutional layer is shown in Fig. 2.

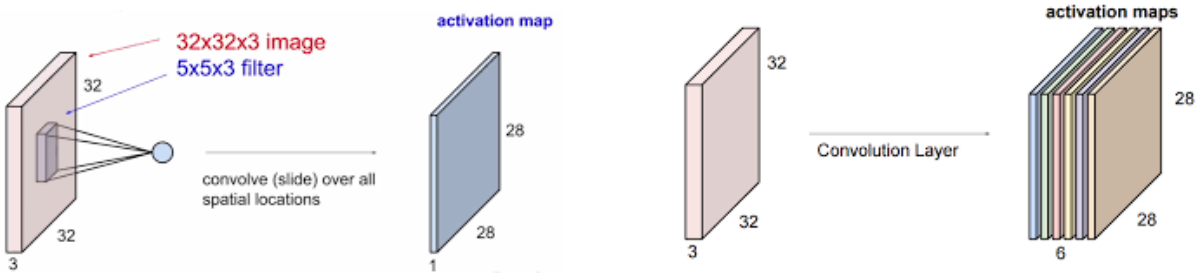

Figure 2: The figure on the left indicates one filter and the figure on right indicates the action of multiple (6) filters. The images are taken from `http://cs231n.stanford.edu/`.

Intuitively, these are used to model spatial correlations, i.e., in images, the location of a particular object is not as important (for simple tasks like image recognition) as the structure of the image itself. This imposes a "prior" on how a natural image looks like. This motivates the use of convolutional filters. In addition to the standard convolutional operation, deep networks also include a non-linear activation on the output of this operation. The output of this operation give rise to activation maps. Generally, to increase redundancy, multiple convolutional filters are learnt for each layer (image) which may correspond to learning features across channels, eg. blue vertical and horizontal edges, red diagonal edges and so on.

- **Mathematical Formulation:** Since we will typically deal with images while using convolutional networks, the inputs are now matrices (or tensors), i.e., each $\boldsymbol{x}_i \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. $n_1 \times n_2$ is the size of the image and n_3 is generally the number of channels, eg., for a RGB image, $n_3 = 3$. Consider the example of Fig. 2. For simplicity, we drop the subscript of the layer. Let the i-th filter be denoted as $\boldsymbol{h}^{(i)} \in \mathbb{R}^{h_1 \times h_2 \times n_3}$ and the output be denoted as $\boldsymbol{y}^{(i)}$ In this case, the (a, b)-th entry of the i-th activation map can be written as

$$\boldsymbol{y}^{(i)}_{a,b} = \sum_{u=1}^{h_1} \sum_{v=1}^{h_2} \sum_{v=1}^{n_3} \boldsymbol{x}_{a',b',w} \boldsymbol{h}^{(i)}_{u,v,w}$$

where, $a' = a + u$ and $b' = b + v$ for stride length one.

Notice that the number of parameters that need to be learnt in this increases significantly as one increases the number of activation maps ($\approx h_1 h_2 n_3$ per layer!). One method to alleviate this problem

is through parameter sharing. ~~Mathematically, this can be expressed as~~

$$\boldsymbol{h}^{(i)}_{a,b,c} = \boldsymbol{h}^{(i)}_{a',b',c} \quad \text{for all pairs } (a,a'),(b,b')$$

~~and thus in practice now we only need to learn n_3 parameters. This is also helpful for training the neural network (optimization sense).~~

- **Does this approach not overfit?** Generally, to ensure that the deep network is powerful enough, a lot of filters are learnt which makes the number of parameters in the order of 100s of millions even though the input dimension is only 1 million or so pixels. This tells us from previous experience that such a network will overfit. A simple method to avoid overfitting in this setting is to include a layer called max-pooling as shown in Fig. 3. This reduces the number of parameters in the network so as to prevent overfitting. Additionally, this operation ensures that the trained network is invariant to basic operations such as rotations, translations and scaling of input images.

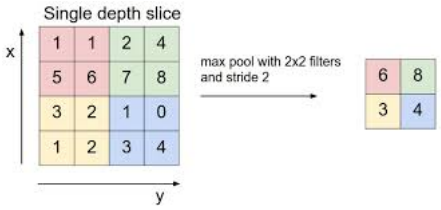

Figure 3: Image downloaded from `http://cs231n.stanford.edu/`.

- **How to decide Network Architecture?** The consensus in the community is that a deeper network is better in terms of performance as well as training[1]. The number of layers and the sizes of the filter depend on the particular task at hand. In general, the characteristic of a "good network architecture" is that the initial layers have smaller filter sizes and fewer number of output channels. The subsequent layers tend to increase the number of channel while reducing the output size of each convolution. Each convolutional layer is followed by a max-pooling layer to avoid redundancies and ensure that the network is invariant to the operations mentioned before. The final layers (typically 2-3) are fully connected (dense) layers which represent a trainable classifier. Thus, to summarize, the first layers learn simple strucutres such as edges and subsequent layers learn more sophisticated motifs and structures. A visualization for a well established deep network trained for image classification is shown in Fig. 4.

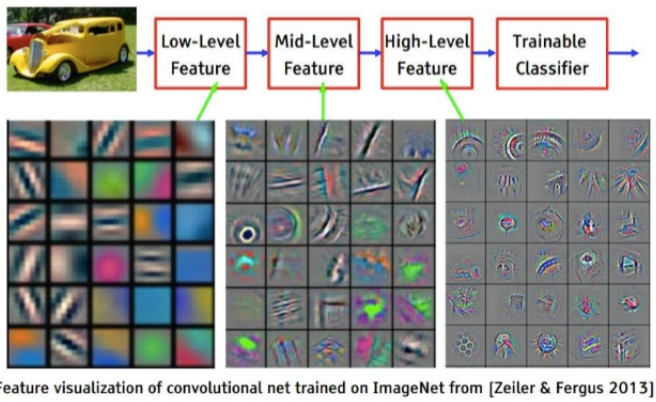

Figure 4: Image downloaded from `http://cs231n.stanford.edu/`.

- **An example Network Architecture:** This is the winner of the ImageNet competition in 2012. The network is known as AlexNet.

[1] This is related to the spinglass effect

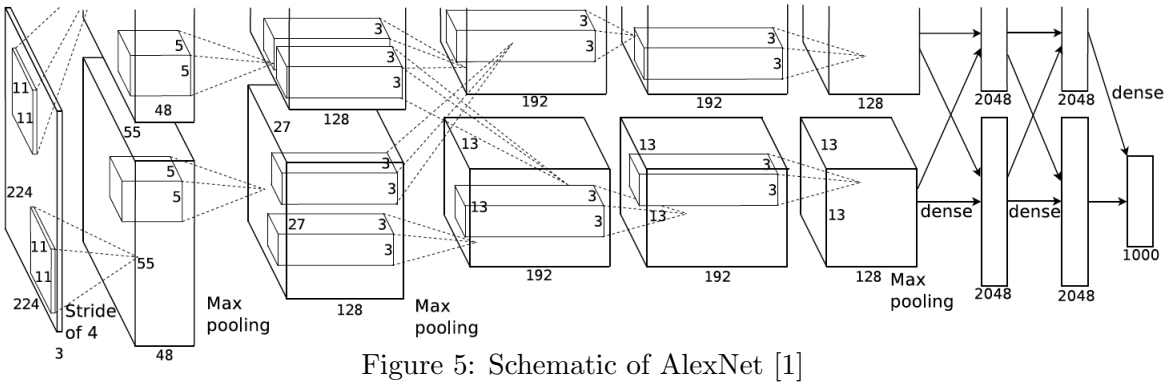

Figure 5: Schematic of AlexNet [1]

There are two flow graphs (top and bottom in Fig. 5) but this is only because the training was very difficult. We can ignore one of thse and consider the bottom one. This is the overview of AlexNet.

1. The inputs to the network are of size $224 \times 224 \times 3$.
2. The first layer considers 96 filters of size $11 \times 11 \times 3$ filters with a stride length of 4. Thus, the size of the first hidden layer is $55 \times 55 \times 96$. This is followed by pooling (overlapping type).
3. The second layer consists of 256 filters of size $5 \times 5 \times 48$. Thus the size of the second hidden layer is $27 \times 27 \times 256$.
4. The third layer consists of 384 filters of size $3 \times 3 \times 256$. Thus the size of the third layer is $13 \times 13 \times 384$.
5. The fourth layer consists of 384 filters of size $3 \times 3 \times 192$. Thus the size of the fourth layer is $13 \times 13 \times 384$.
6. The fifth layer consists of 256 filters of size $3 \times 3 \times 192$ and thus the size is 4096 (after flattening).
7. The sixth and seventh layers are fully connected with 4096 neurons each and ReLU activation.
8. The final layer produces a soft-max output, i.e., $\sigma(\boldsymbol{y})_j = \frac{e^{y_j}}{\sum_i e^{y_i}}$ provides a 1000-dimensional output.
9. This network has 60 million parameters and $2-3$ days to train with 2 (good) GPUs.

- **When is this architecture used?** As mentioned earlier, this architecture is used to exploit spatial correlations and thus is an integral component of deep networks that work on images and videos (for the spatial correlations).

1.3 Recurrent Neural Networks

Finally, we consider the classes of inputs where we want to exploit a temporal correlation. This is standard in tasks such as speech recognition, machine translation, text prediction and so on. This type of networks is known as recurrent neural networks. A schematic of a simple recurrent neural network is shown in Fig. 6. *These are not feed-forward neural networks.*

- **Mathematical Formulation** This architecture helps in inducing *memory* to a neural network. Thus, we can model functions of the form

$$\boldsymbol{y}_t = \boldsymbol{W}_x \boldsymbol{x}_{t-1} + \boldsymbol{W}_y \boldsymbol{y}_t$$

where $\boldsymbol{W}_x$ models the relation with time (memory) and $\boldsymbol{W}_y$ models the input-output mapping at each time.

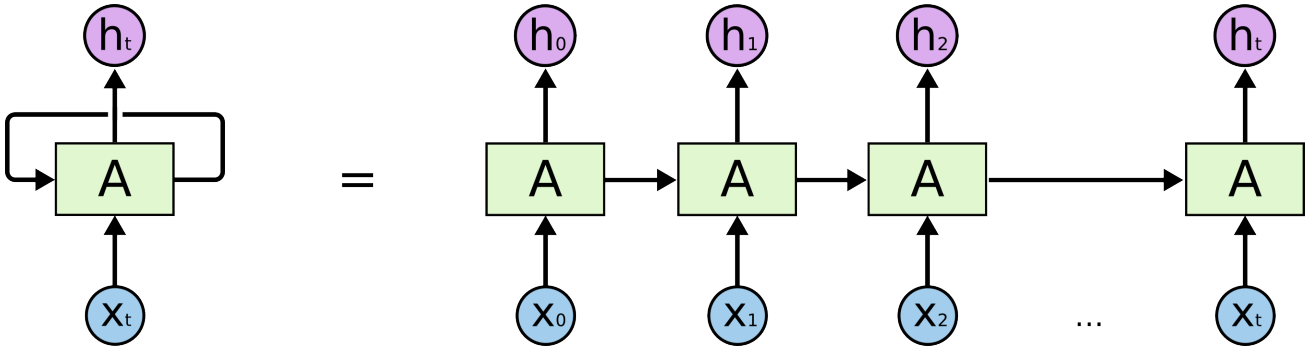

Figure 6: Image downloaded from `http://cs231n.stanford.edu/`.

- **How to decide architecture?** Typically, these networks are very deep, but not very wide, i.e., they have a large number of layers, but few number of neurons per layer. This aims to capture the dependence in time. For example, while predicting what the next word of a text message is, a phone will try to remember what the previous few words were and predict the next word. Two challenges of this approach (i) if one tries to remember too many of the previous words then training the network becomes a challenge due to the large number of parameters that need to be trained (ii) the network will overfit because there are too many parameters as compared to the dimensionality of the input. Thus, there is a trade-off. This is also often referred to as Long Short Term Memory (LSTM).
- **When is this architecture used?** As mentioned earlier, this architecture is used when there are temporal correlations that need to be exploited. Some of the major successes using RNN are (i) speech recognition: RNNs are now the state-of-the-art method that take a *wave file* as an input and provide a transcription; (ii) machine translation: RNNs take wave files of speech in one language as an input and provide a wave file of the translated speech into a language of choice; this is also the basis of Google translate (iii) text prediction: RNNs are used to predict the next letter of a sentence. *Most likely these are used in latest Gmail interfaces but a primitive version called Hidden Markov Models are used in phones.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.