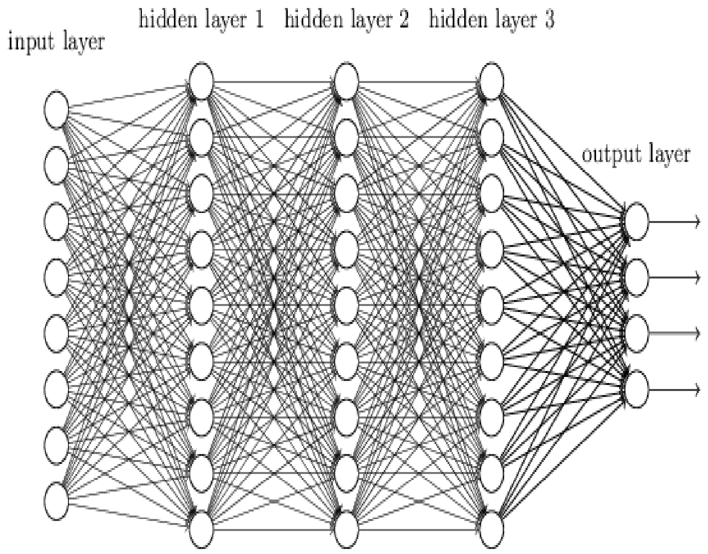
CS 383 – Machine Learning

Convolution Neural Networks

Slides adapted from material created by E. Alpaydin
Prof. Mordohai, Prof. Greenstadt, Pattern Classification (2nd Ed.),
Pattern Recognition and Machine Learning

Convolution Neural Networks

- Both shallow and deep neural networks are *fully connected*.
- What's wrong with this?
  - Take a while to train
  - Can easily over-fit due to large number of free parameters (weights).
- Furthermore, when we have input features that have spatial information, these are highly vulnerable to slight translations
  - What happens if everything is shifted by one pixel?



Convolution Neural Networks

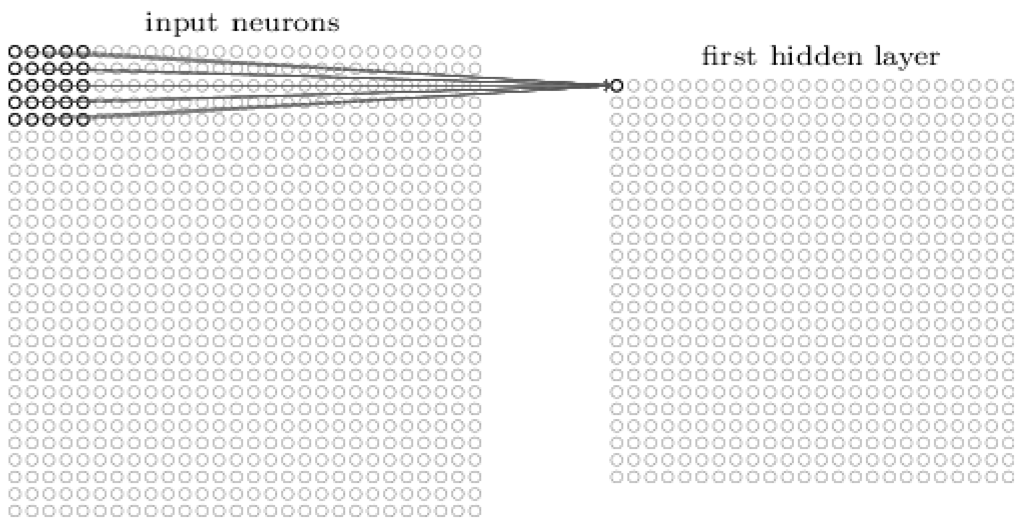
- Convolution Neural Networks (CNNs) are popular for image and audio classification due to their invariance to translations.
- Similar to deep networks, they learn intermediate concepts/features.
- But they aren’t fully connected
  - So spatial relationships can be taken into account.
  - And they’re faster and less prone to over-fitting than a deep network.

CNNs

- CNNs typically have
  - One or more convolution layer
  - A final fully connected shallow ANN.
- The convolution layer contains several parts
  - Feature Map Extraction
  - Pooling.
- Let’s look at each of these

CNNs

- The first stage of the convolution layer is the convolution process.
- Here we define some $M \times M$ region, and pass this over the inputs to get output values.



CNNs

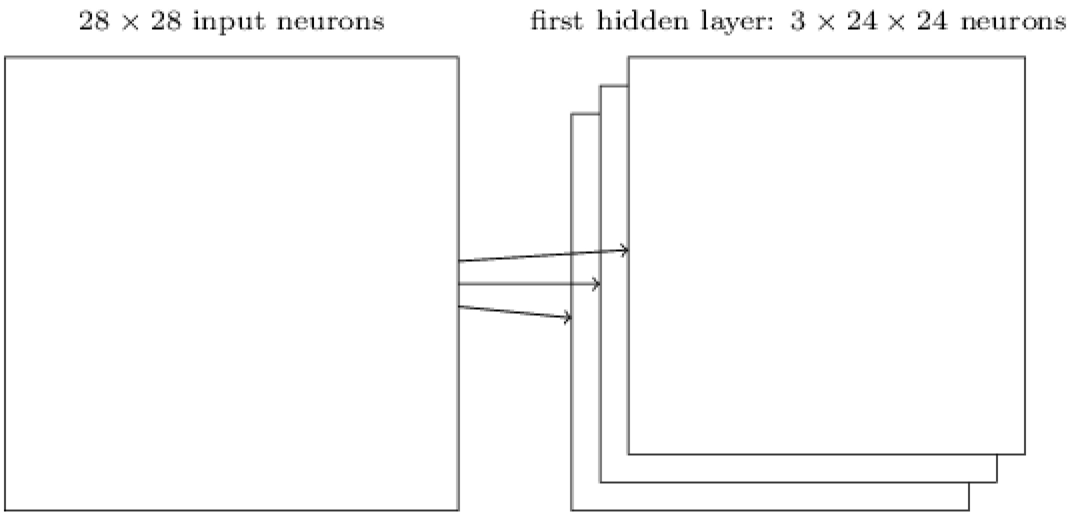
- The values from the current $M \times M$ region are multiplied by weights to get our value for the hidden layer
- HOWEVER, we have a single set of $M \times M$ weights used for all areas on the input image.
- Let $\sigma(x)$ be some activation function (like the sigmoid) and $a_{x,y}$ be the value from location x, y in the previous layer.
- We can then compute the value coming out of the j, k^{th} hidden neuron as:

$$h_{j,k} = \sigma\left(b + \sum_{l=0}^{M} \sum_{m=0}^{M} w_{l,m} a_{j+l,k+m}\right)$$

- This output is often referred to as a *feature map*.

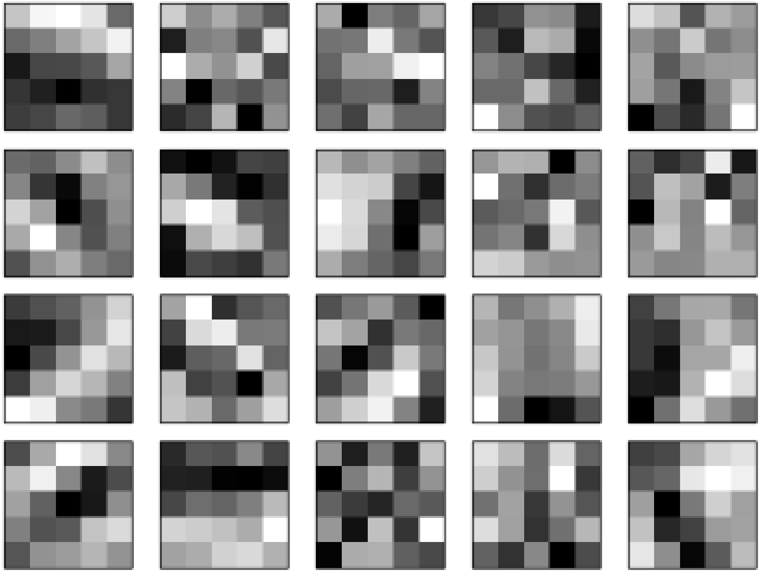
CNNs

- Most applications call for several feature maps.
- If we want P feature maps, then we need to initialize randomly $M \times M \times P$ weights.



CNNs

- Here are 20 trained $M \times M$ weights.
- They represent *filters/concepts*

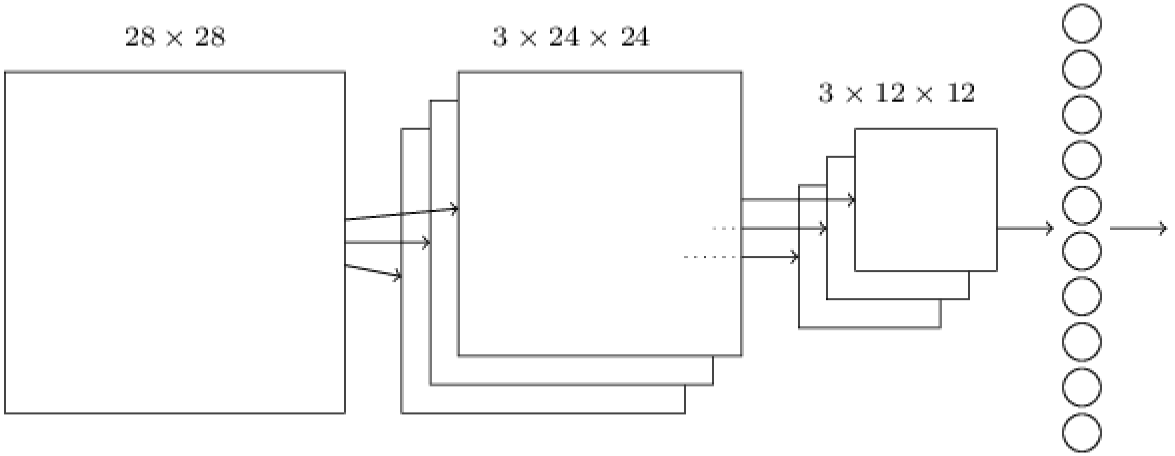


CNNs

- The next part of a CNN layer is *pooling*.
- Pooling is essentially *downsampling*: we're taking our feature map and making a new, smaller map by "summarizing" the original one.
- This is typically done by again moving around a non-overlapping $Q \times Q$ window and extracting some value from each locations
- Common pooling techniques include:
  - Max Pooling – Select the maximum value in the square
  - L2 Pooling – Compute the square root of the sum of the values in the square.

CNNs

- Now that our Convolution Layer is made, the output of the pooling process becomes the input of a fully-connected shallow ANN.



CNNs

- Training a CNN is similar to training a regular neural network; we use *forward-backwards propagation.*
- As part of the forward process:
  - Create the feature maps using the current convolution layer weights.
  - Create the pooled maps from the feature maps.
  - Feed these into a normal ANN.
- And in the backwards process…
  - We also need to update the convolution layer weights.
  - We just do this by
    - Propagating the errors to the pools
    - Propagate the errors to the feature map locations.
    - Update the convolution layer weights based on these propagations.

CNN References

- http://neuralnetworksanddeeplearning.com/chap6.html
- http://www.robots.ox.ac.uk/~vgg/practicals/cnn/