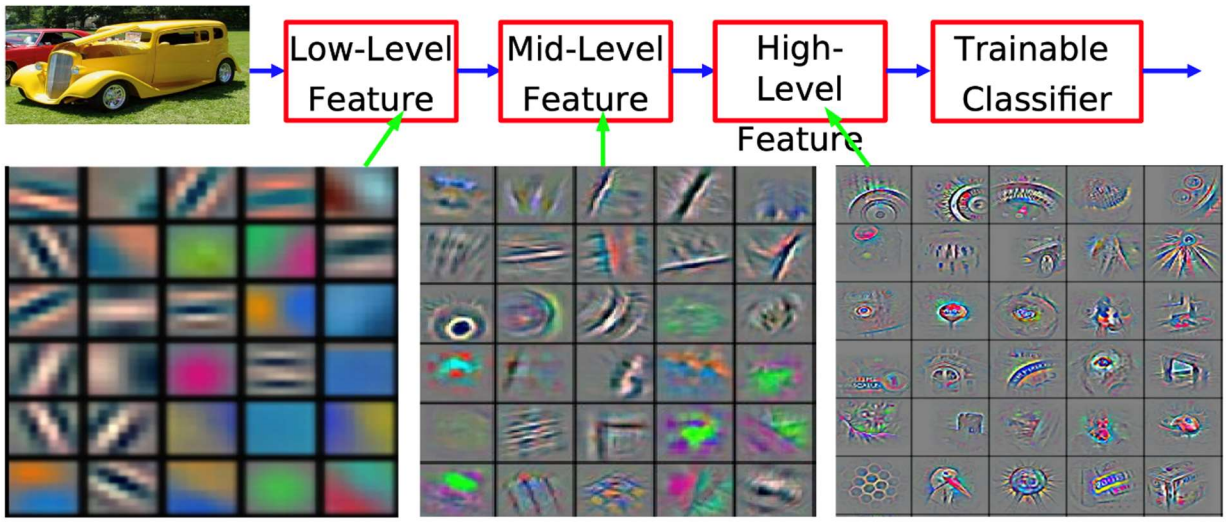
Lecture 3 Convolutional Neural Networks

Natural data is hierarchical, and can therefore be interpreted by characterizing low level features and compounding them into higher level features.



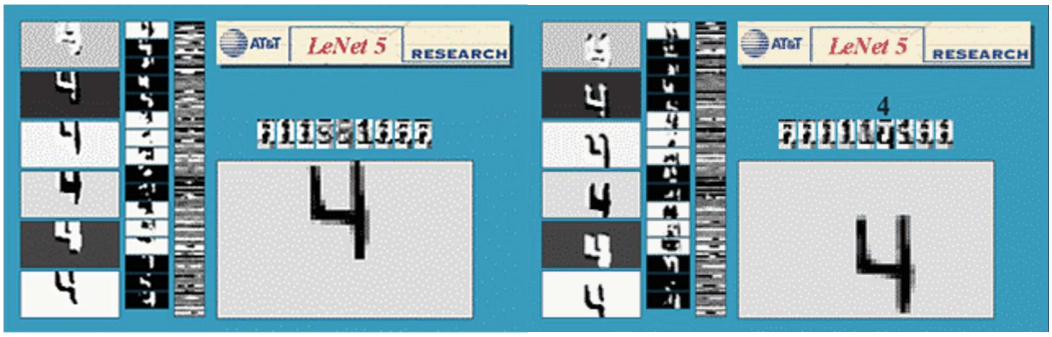
Pieces of a Convolutional Net

1. Layers
    a. Convolution (linear part)
    b. Non-linearity (e.g. ReLU, tanh)
    c. Pooling (decreasing dimensionality of the data)
    d. Normalization (adjust magnitude of data so that features are comparable)
    e. Final fully connected layer (pass the output of the convolutional net to a fully connected classifier)
2. Residual bypass connection (help avoid vanishing gradient) see https://theaisummer.com/skip-connections/

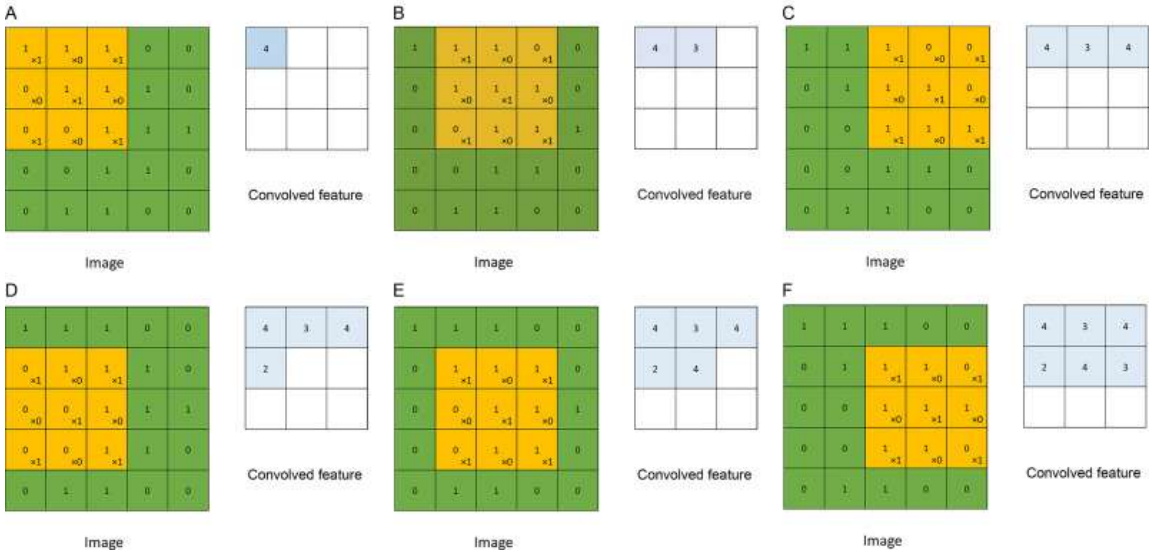
Performing a convolution

Overlay a convolution kernel with the data, multiply element-wise, and take the sum (this equation is technically a cross-correlation rather than a convolution, which would replace the indices on x with $i - k$ and $j - l$). This is an example of weight sharing, where the kernel keeps the same parameters as it is moved over the whole of the input, and it allows us to detect a motif wherever it appears in the data. In a convolutional neural network, this convolution is the linear operator.

$$y_{ij} = \sum_{kl} w_{kl} x_{i+k,j+l}$$



We can see how weight sharing comes into play above. With this network, shifting the input simply shifts the output by the same amount, rather than failing to recognize any motifs.



Adjustments can be made to a convolution:

1. Striding: In the above image, the kernel is moved by one pixel at a time, meaning it has a stride of one. If we increase the stride to two, we'll move the edge of the kernel by two pixels on a move instead, allowing us to either significantly decrease the number of calculations in a layer.
2. Padding: Adding 0-valued pixels to the edges of the input can let us keep the size of the image the same. Can cause some strange edge effects.
3. Pooling: Takes a window of the data in the same manner as a convolution, and compresses it to one value, either the average of the window, or the maximum value in the window. This allows downsampling of the data by compressing the data, which can help avoid overfitting, and can extract dominant features.

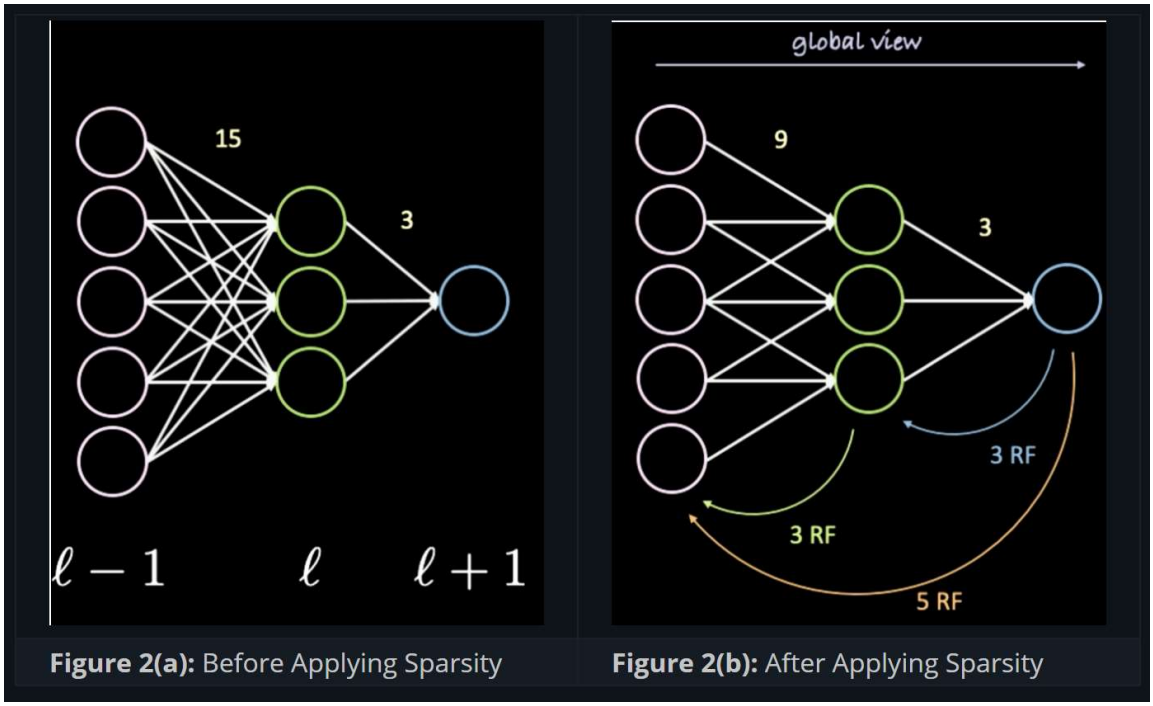


Figure 2(a): Before Applying Sparsity

Figure 2(b): After Applying Sparsity

Rather than connecting all the inputs to every neuron, we take advantage of the locality of the data. This is what the convolution kernel is doing; here we have 1-D data, and a convolution kernel of size 1X3. This cuts down on the number of calculations.

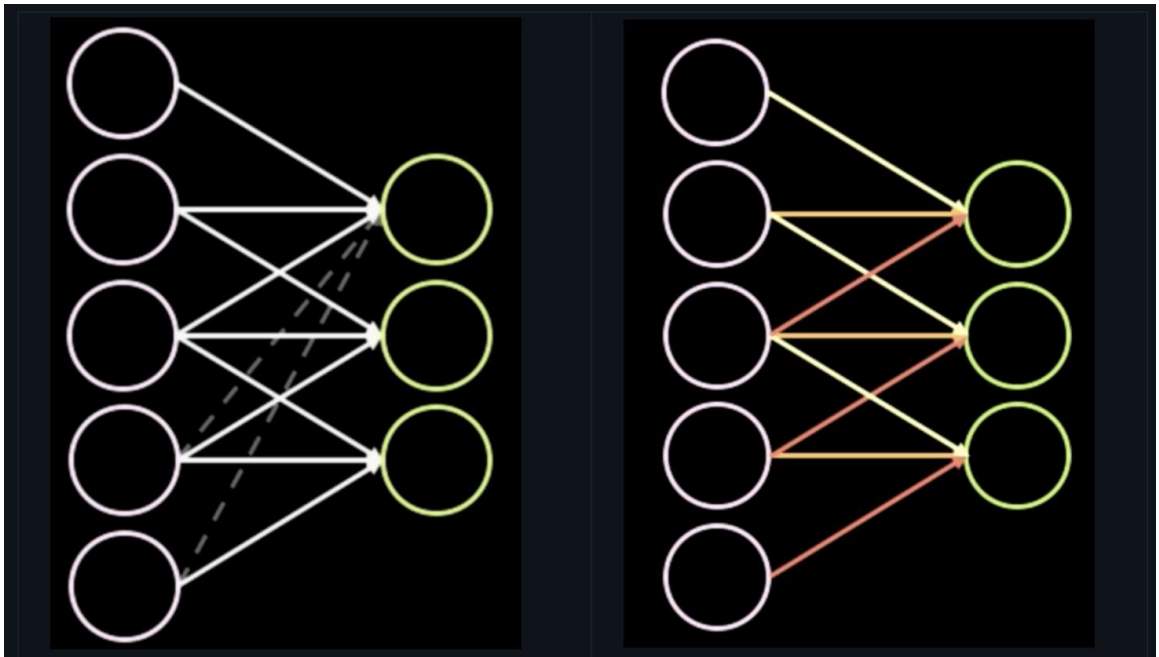


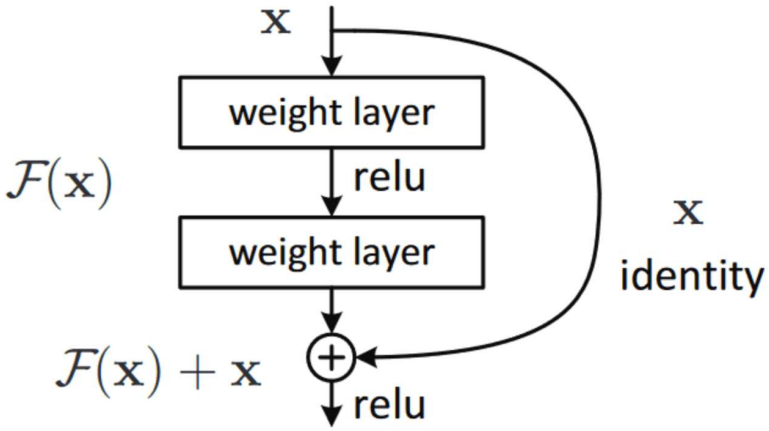
Figure 3(a): Before Applying Parameter Sharing

Figure 3(b): After Applying Parameter Sharing

Uses

- Data stored in arrays
- Strong locality
- Audio, time series, image recognition, video recognition

Residual bypass



By allowing this kind of bypass we can avoid backpropagating through every layer, which means multiplying by the derivative of every layer (these derivatives are often less than one, which results in a network gradient that approaches zero).

Relevant Papers

Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 2414–2423. https://doi.org/10.1109/CVPR.2016.265

Van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Senior, A., & Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. arXiv.org. http://search.proquest.com/docview/2080352104/