COMPSCI 689
Lecture 13: Neural Network Architectures

Benjamin M. Marlin

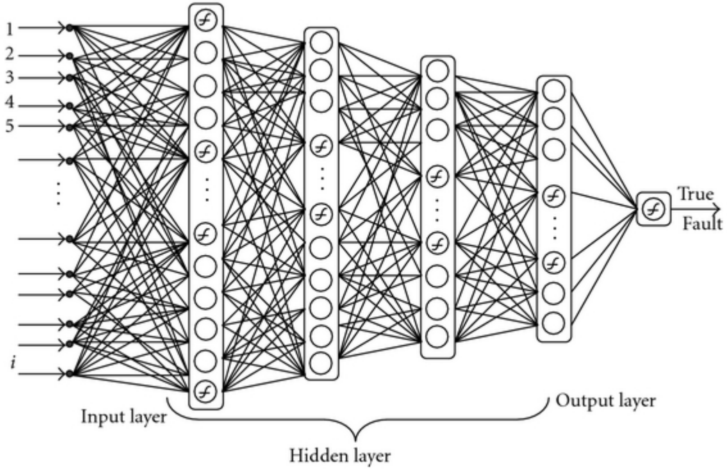
College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

Introduction

- Basic MLPs have multiple layers of hidden units that are fully connected with distinct weights on each connection.
- Such an architecture does not encode any domain knowledge about the relationship between the inputs and outputs.

Multi-Layer Perceptron



Introduction

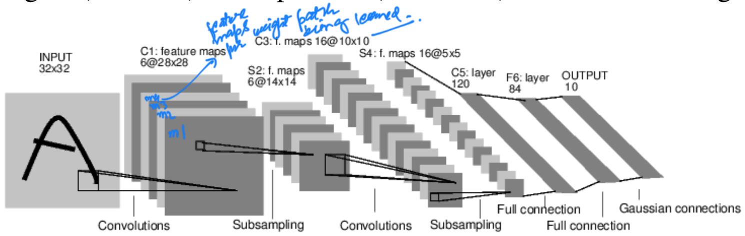
- For specialized types of inputs like images or sequences, it's possible to encode domain knowledge about the structure of the inputs or input-output relationships into the architecture of a neural network.
- This can help make learning more efficient by using notions like sparsity and invariance to reduce the number of parameters that must be learned.

Images and Sequences

- The two most common types of structures encountered in the neural networks literature are images and sequences.
- In object classification, an object in an image can occur at any scale, orientation, location, etc., and the information about whether an image contains an object is typically spatially localized.
- In sequence data like text, the underlying grammar and vocabulary create regularities in sequences in terms of both short and long range dependencies.

Architectures for Images

Deep learning for images uses a modified deep architecture called a *convolutional network* or convnet or CNN that learns small patches of weights (or filters) and replicates (convolves) them over an image.

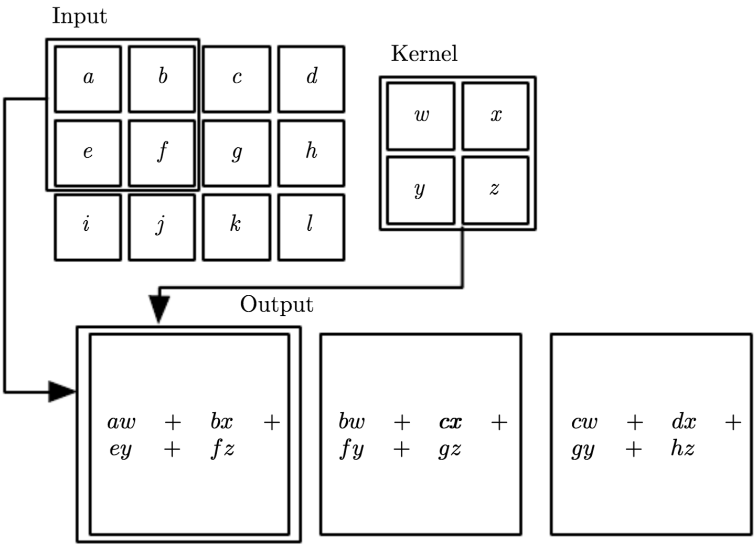


This architecture was popularized the 1980s and was directly inspired by the structure of the visual areas of the brain.

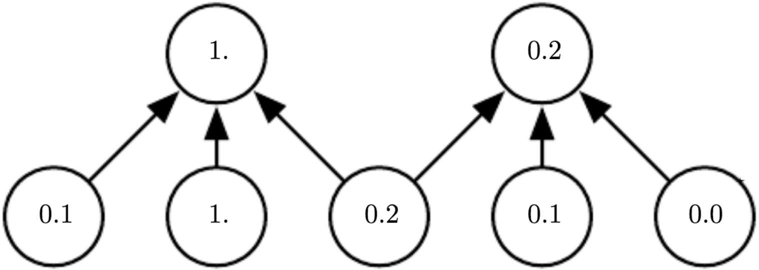
Convolutional Network Invariances

- The convolutional network architecture encodes two primary pieces of domain knowledge: translational invariance and spatial locality.
- Sub-sampling using max pooling (taking the max of the inputs in a spatial window) makes individual hidden units more invariant to local translations in an image.
- Translational invariance is achieved more globally by using the same set of weights in every image patch via convolution.
- The use of compact convolution kernels also results in a massive reduction in the number of weights and creates hidden units that have spatially localized inputs.

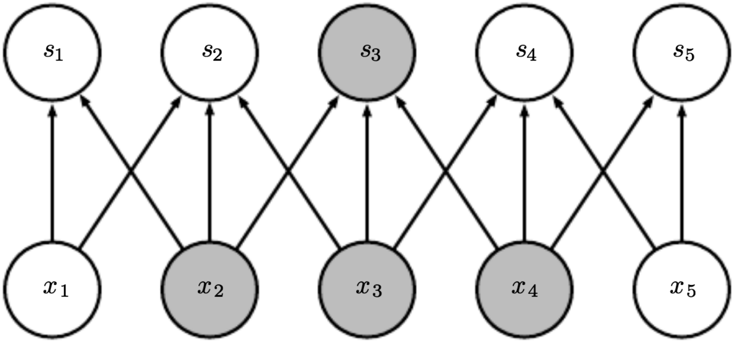
Example: 2D Cross Correlation



Example: 1D Pooling

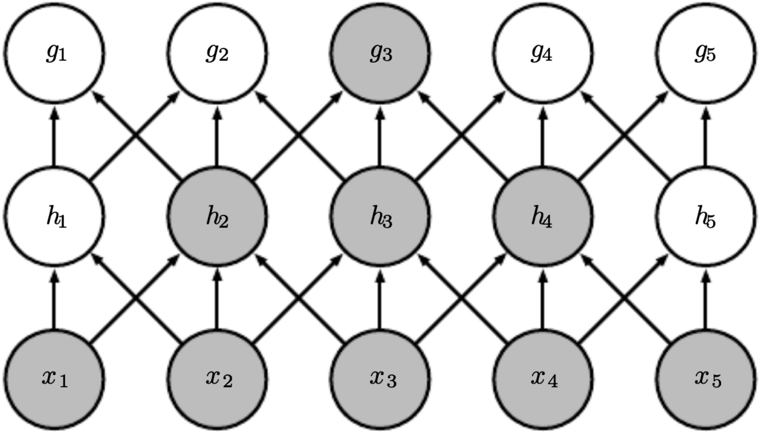


Example: 1D Receptive Field

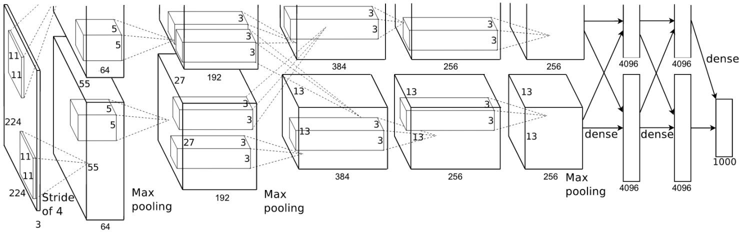


Via analogy to animal visual systems, the set of inputs that feed into a given hidden unit in a given feature map are referred to as that unit's *receptive field*.

Example: 1D Receptive Field



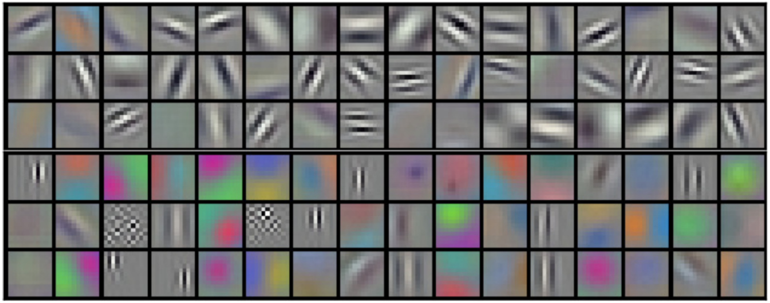
AlexNet (2012)



```
self.features = nn.Sequential(
    nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3, stride=2),
    nn.Conv2d(64, 192, kernel_size=5, padding=2),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3, stride=2),
    nn.Conv2d(192, 384, kernel_size=3, padding=1),
    nn.ReLU(inplace=True),
    nn.Conv2d(384, 256, kernel_size=3, padding=1),
    nn.ReLU(inplace=True),
    nn.Conv2d(256, 256, kernel_size=3, padding=1),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3, stride=2),
)
```

```
self.classifier = nn.Sequential(
    nn.Dropout(),
    nn.Linear(256 * 6 * 6, 4096),
    nn.ReLU(inplace=True),
    nn.Dropout(),
    nn.Linear(4096, 4096),
    nn.ReLU(inplace=True),
    nn.Linear(4096, num_classes),
)
```

Example Layer 1 Filters/Kernels (Natural Images)



Example Filtermaps (Natural Images)

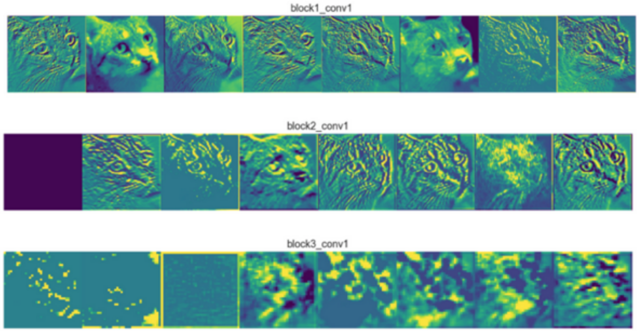
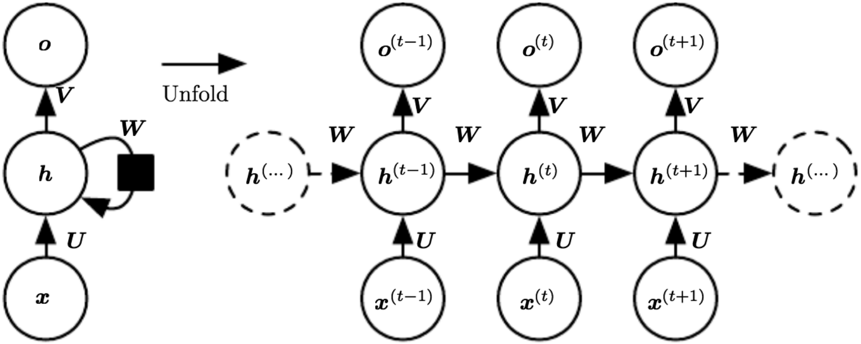


Image credit: Arden Dertat

Sequences

- Like with images, sequences have special structure that can be exploited by custom network architectures.
- The fundamental idea is that of a *recurrent* architecture that applies the same function to the data (and hidden state) for every position in a sequence.
- This allows for a model to be applied to sequences of any length using a fixed number of parameters.
- To learn a recurrent model, it suffices to “unroll" it for a number of steps equal to the length of sequence, and then backprop through the unrolled network.

Recurrent Neural Network (Dense Output)



Recurrent Neural Network Family

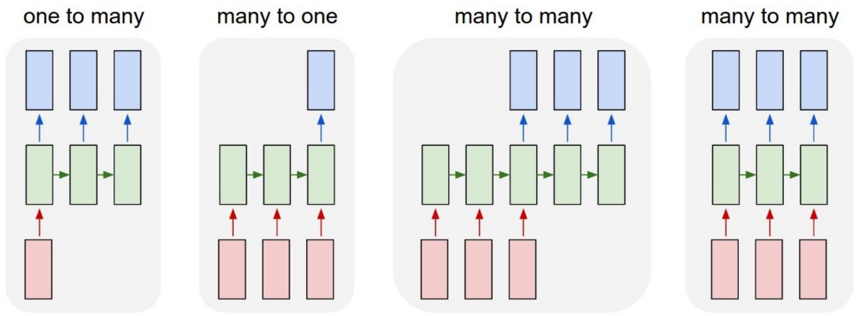


Image Credit: Andrej Karpathy

Learning with Long Sequences

- Learning recurrent networks for long sequences, particularly for data with long-range dependencies between inputs and outputs, can be very challenging.
- One major problem is that gradients can explode or vanish depending on the non-linearities used since a recurrent network applied to a long sequence is equivalent to a very deep feed-forward network.
- To deal with this problem, specialized architectures have been developed to improve the propagation of gradient information backward through sequences during learning, and to make it easier to hold on to useful information for long periods of time.

The Long Short-Term Memory Cell

$$\bar{\mathbf{z}}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1} + \mathbf{b}_z$$

$$\mathbf{z}^t = g(\bar{\mathbf{z}}^t) \qquad \textit{block input}$$

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t) \qquad \textit{input gate}$$

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f$$

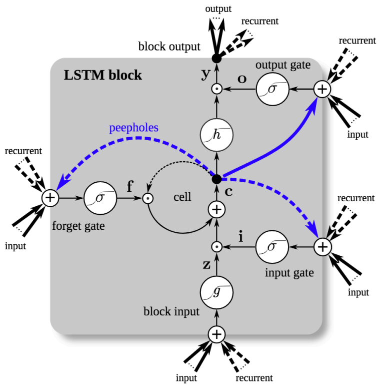
$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t) \qquad \textit{forget gate}$$

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t \qquad \textit{cell}$$

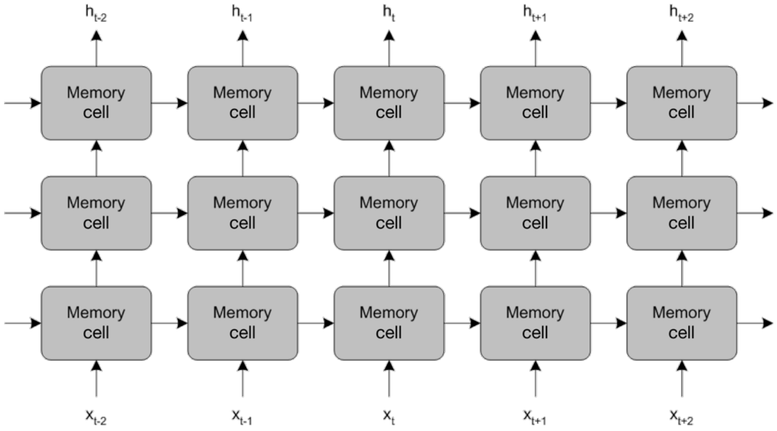
$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t) \qquad \textit{output gate}$$

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t \qquad \textit{block output}$$



Deep LSTM Models



Transformers

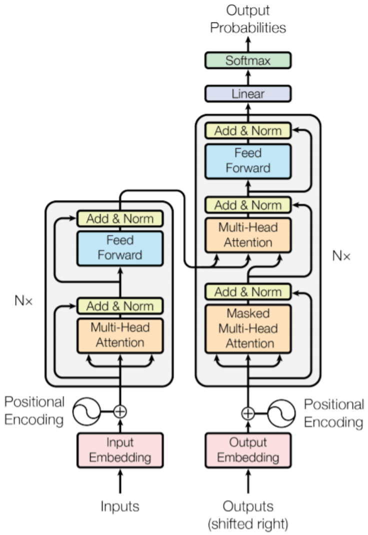


Figure 1: The Transformer - model architecture.