More on convolutional neural networks
Convolution in 3-D

- It is possible to perform convolution in any number of dimensions.

- With neural networks, the usual formula of 3-D convolution is given by:

\[
TM(r, c) = \sum_{c=1}^{t_c} \sum_{i=-t_h/2}^{+t_h/2} \sum_{j=-t_w/2}^{+t_w/2} \text{im}[r+i, c+j, c]t[i, j, c]
\]

where \( t_c \) is the number of **channels** in the convolution kernel (and also the input image).
Convolution: multiple output channels

- By applying multiple convolution kernels to each input image, we can produce multiple feature maps (recall the MNIST example):
Convolution: multiple output channels

- Hence, convolution can take a multi-channel image as \textit{input}, and also produce a multi-channel image as \textit{output}. 
Convolutional neural networks
Convolutional neural networks

• Based on this infrastructure, we can construct **convolutional neural networks (CNNs)** — networks that consist of 1+ convolutional layers (and usually some non-convolutional layers too).

• CNNs have revolutionized computer vision, especially in object detection, object recognition, semantic segmentation, activity recognition, and other tasks.
Convolutional neural networks

• While the kernels in the examples so far were built by hand, this is almost never done in practice.

• Instead, we train the weights using back-propagation, just like with feed-forward neural networks.

• With CNNs, the learned weights represent the elements of the convolution kernels.
Convolutional neural networks

• CNNs are a special case of feed-forward (FF) NNs.

• In the FF NN below, each of the 36 hidden units is associated with 64 weights.
Convolutional neural networks

- CNNs are a special case of feed-forward (FF) NNs.

- In the FF NN below, each of the 36 hidden units is associated with 64 weights.
Convolutional neural networks

• CNNs are a special case of feed-forward (FF) NNs.

• In the FF NN below, each of the 36 hidden units is associated with 64 weights.
Convolutional neural networks

- We can re-arrange the hidden units into a grid (this just affects the visualization, not the computation).
Convolutional neural networks

- We can also require that most of the weights for each hidden unit be 0.
Convolutional neural networks

- Since the image pixels corresponding to the 0-weights contribute nothing to each hidden unit, they can be removed, resulting in just 9 weights per hidden unit.
Convolutional neural networks

- Can further constrain weight matrix $W^{(1)}$ by requiring that each 3x3 set of non-zero weights be the same for each of the hidden units.
Convolutional neural networks

- Can can further constrain weight matrix $W^{(1)}$ by requiring that each 3x3 set of non-zero weights be the same for each of the hidden units.
Convolutional neural networks

- Can further constrain weight matrix $W^{(1)}$ by requiring that each 3x3 set of non-zero weights be the same for each of the hidden units.
Convolutional neural networks

- This is now completely equivalent to a convolutional layer instead of a general feed-forward layer.
CNN architecture

- CNNs (since ~2012) typically employ:
  - Multiple convolutional layers
  - Pooling layers (e.g., max-pool) between some of the convolutional layers.
  - Feed-forward layers at the end.
  - Non-linear activation functions between each layer.

- Example:
CNN architecture

- Trend (~2016+):
  - Less pooling
  - More convolutional layers
Recurrent neural networks
Fixed length versus variable length

• Up till now, all the ML models we have considered have processed inputs $x$ of some \textbf{fixed length} $m$ to predict some target value $y$.

• But what if $m$ varies from example to example?
Fixed length versus variable length

- Examples of variable-length inputs:
  - Text passages: number of words.
  - Videos: number of frames.
  - Time series: number of events.
Handling variable-length inputs

Possible strategies for handling variable-length inputs:

1. Decide what the maximum length should be, and “pad” the input to always reach that length.

E.g., if max-length $M=6$, but actual length of some example $m=4$, then pad with 2 zeros.
Handling variable-length inputs

Possible strategies for handling variable-length inputs:

1. Decide what the maximum length should be, and “pad” the input to always reach that length.

   • Advantages:
     • Simple to implement.

   • Disadvantages:
     • Wasteful of memory
     • Hard to learn weights from input units close to $M$. 
Handling variable-length inputs

- Possible strategies for handling variable-length inputs:

  2. Decide what the minimum length should be, and discard any features beyond this length.

  E.g., if min-length $M=6$, but actual length of some example $m=16$, then ignore the last 12 features.
Handling variable-length inputs

• Possible strategies for handling variable-length inputs:

  2. Decide what the minimum length should be, and discard any features beyond this length.

• Advantages:
  • Simple to implement.

• Disadvantages:
  • Throws away potentially valuable information.
Handling variable-length inputs

- Possible strategies for handling variable-length inputs:

  3. Extract summary features from the sequence that do not depend on the input length.

  E.g., extract aggregate color information, number of detected objects of fixed set of classes, etc.
Handling variable-length inputs

- Possible strategies for handling variable-length inputs:
  
  3. Extract summary features from the sequence that do not depend on the input length.

Judge the type of video based on the fixed-length feature representation.
Handling variable-length inputs

- Possible strategies for handling variable-length inputs:

  3. Extract summary features from the sequence that do not depend on the input length.

- Advantages:
  - Simple to implement; can give high accuracy.

- Disadvantages:
  - Requires manual feature engineering.
Handling variable-length inputs

- Possible strategies for handling variable-length inputs:

  4. Scan the input in blocks of $k$ features at a time, and update a “hidden state” to store most important information.
Handling variable-length inputs

• Possible strategies for handling variable-length inputs:

  4. Scan the input in blocks of $k$ features at a time, and update a “hidden state” to store most important information.

![Recurrent neural network diagram]
Handling variable-length inputs

- Possible strategies for handling variable-length inputs:

  4. Scan the input in blocks of $k$ features at a time, and update a “hidden state” to store most important information.
Handling variable-length inputs

Possible strategies for handling variable-length inputs:

3. Scan the input one feature at a time, and update a “hidden state” to store most important information.

Advantages:

• More powerful, often resulting in higher accuracy.

Disadvantages:

• More complicated to implement & train.
We can construct a minimal recurrent neural network (RNN) as follows:

\[ \hat{y}_t = g(x_t; U, V, w) = h_t^T w \]
\[ h_t = \sigma(Uh_{t-1} + Vx_t) \]