Convolutional models

Professor Marie Roch
Local connectivity

• Sometimes, it might make sense to have local connectivity in a network
Why local connectivity?

Local patterns might be seen in multiple parts of the input.

What if the same pattern might occur in different places?
Local patterns

• How can we learn to recognize a local pattern anywhere in the input?

• If we can learn a set of weights that are applied to a local area, this can serve as a feature extractor.

• We need tools to apply these weights across the input.
Convolution

• Operation that combines two signals (1D definitions):
  – Continuous
    \[ x(t) * w(t) = \int_{-\infty}^{\infty} x(a)w(t - a)da \]
  – Discrete
    \[ x[t] * w[t] = \sum_{-\infty}^{\infty} x[a]w[t - a] \]

Notes on notation:
Engineers denote continuous signals with ( ), discrete with [ ]
Goodfellow et al. use (x*w)(t) rather than x(t)*w(t)
Convolution

- Formula outputs sample at time $t$.
- By convention:

$$x[t] * w[t] = \sum_{a=-\infty}^{\infty} x[a]w[t - a]$$

- The kernel shapes the response of the convolution and is time reversed.
- Kernel usually has finite support
Convolution in 1D

\[ f(x) = \begin{cases} \frac{a_x}{2}x & \text{for } |x| \leq a_x \\ 0 & \text{otherwise} \end{cases} \]

\[ g(x) = \begin{cases} \frac{x-b_y}{b_y} & \text{for } x \in [b, b'] \\ 0 & \text{else} \end{cases} \]

Demonstration courtesy of Karlsruhe Institute of Technology
https://phiresky.github.io/convolution-demo/
Convolution in 2D

• For audio, typically used on spectrograms or other time/frequency representations:
  \[ x[t, \omega] \ast w[t, \omega] = \sum_{m} \sum_{n} x[m, n]w[t - m, n - \omega] \]

• Many libraries implement cross-correlation instead of convolution (kernel not reversed, not important in practice):
  \[ \sum_{m} \sum_{n} x[t + m, \omega + n]w[m, n] \]

• Convolution is applied for each time-frequency cell \([t, \omega]\)
Convolution in 2D

Animation credit: Irhum Shafkat
Convolution in 2D

\[ w = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix} \]
Multiple kernels

• Suppose > 1 kernel, e.g., 10
• In the previous example, we would have 10 3x3 feature maps
• If the next layer convolved another 10 kernels across each of the 10 feature maps, we would have 100 feature maps.
• To prevent combinatorial explosion, we fuse the feature maps between layers, usually by addition, possibly after using an activation function
Key ideas

• Kernel is a learned feature extractor

• Desirable properties
  – Sparse interactions
  – Parameter sharing
  – Equivalence under translation
Sparse interaction

• Output points only depend on a local neighborhood.
• Similar to cutting connections (weights to zero) in a dense layer

Fig. 9.2 Goodfellow et al.
Parameter sharing

• We slide the convolutional layer across the tensor.
• Each position produces a new output.
• We use the same kernel, thus the outputs from a single kernel have tied (shared) weights.
Receptive field

Subsequent convolutions increase the area that affects a layer output
Equivariance under translation

- When something in the input shifts position, its representation (convolution with the kernel) will also shift position.

Image: John Tlumacki, Boston Globe
Key ideas

**Convolutional layers learn a representation**

- **Input:** (40x40)
- **C1:** 32@,(36x36)
- **P1:** 32@,(18x18)
- **C2:** 64@,(14x14)
- **P2:** 64@,(7x7)
- **FC1:** 3136
- **FC2:** 1000
- **Output:** 2


---

North Atlantic right whale upcall

*Image: NOAA*
Key ideas

• For audio, convolutional layers frequently followed by
  – an optional RNN
  – flattening and a feed-forward network to perform the classification.
Common architecture

- Linear activation in first stages (not all networks use this)
- Nonlinear activations
- Pooling
Pooling layers

- Reduce dimensions
- Helps with translation invariance
- How it works:
  - specify mask size (2x2)
  - specify stride (2)
  - apply statistic to covered cells
  - Example is max-pooling

Yamashita et al. 2018
https://doi.org/10.1007/s13244-018-0639-9
Pooling

• Can be seen as an alternative to resizing an input to a fixed size

• Example:
  – Learn a 9x9 feature vector to send to classification layer
  – Can adjust the pooling step sizes so that end result is 9x9 for many input sizes.
Convolutional layer connections

• Each convolutional layer $l$ has $l_f$ filters $f_l(i), 1 \leq i \leq l_f$.
• In layer $l+1$, most deepnet libraries will combine the $l_f$ filter outputs with each of $l+1$’s convolutional filters:

$$f_{out_{l+1}}(i) = b_{l+1}(i) + \sum_{j=1}^{l_f} f_{out_{l}}(j) \ast f_{(l+1)}(i)$$
How are CNNs used in audio

• Two main paradigms
  – Image processing paradigm
    • Spectrogram
    • Multiple convolutional and pooling layers
  – Framed speech
    • 1D convolution on time domain or spectrum

• Possible RNN network on extracted feature matrix
• Feed forward layer
Are CNNs appropriate for audio?

Not always:

• Position has meaning

• Harmonic structure changes across the spectrum
Harmonic structure

Synthetic fundamental and first harmonic

$F_0$ 100 to 350 Hz

$F_0$ 200 to 700 Hz

- Time: 0.46
  - Frequency: 0.3438 kHz
  - Power: -18.5

- Time: 0.46
  - Frequency: 0.6875 kHz
  - Power: -18.99

- Time: 1.06
  - Frequency: 0.3644 kHz
  - Power: -18.4

- Time: 1.06
  - Frequency: 0.4844 kHz
  - Power: -19
Audio is additive
Positions within images are only meaningful in relation to other things
Shifting frequency changes meaning…

Roch et al. 2007
Harmonic structure

• Unless using constant-quality representations harmonics change differently

• A CNN filter may learn harmonic structure that is appropriate for one frequency, but not for a frequency that a little different:
  – 100 200 300 400 Hz vs
  – 120 240 360 440 Hz
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input, Conv2D, MaxPooling2D

model = Sequential()
model.add(Input(shape=(1000,1000,1)))  # 1000x1000 grayscale
model.add(Conv2D(32, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2))
model.add(Conv2D(64, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
# Convert to vector, then feed forward network
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dense(NumClasses, activation='softmax'))