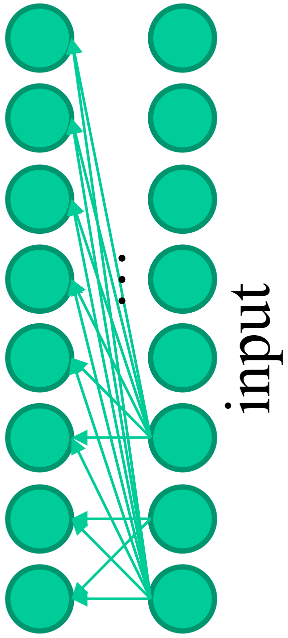
Convolutional models

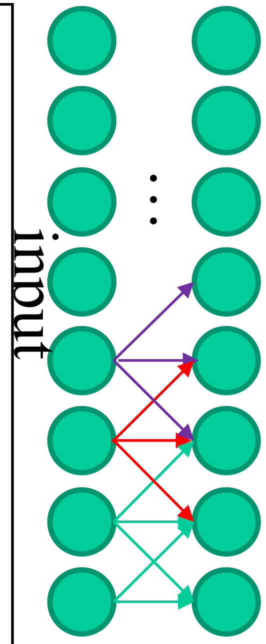
Professor Marie Roch

SAN DIEGO STATE UNIVERSITY

Local connectivity

- Sometimes, it might make sense to have local connectivity in a network





Why local connectivity?

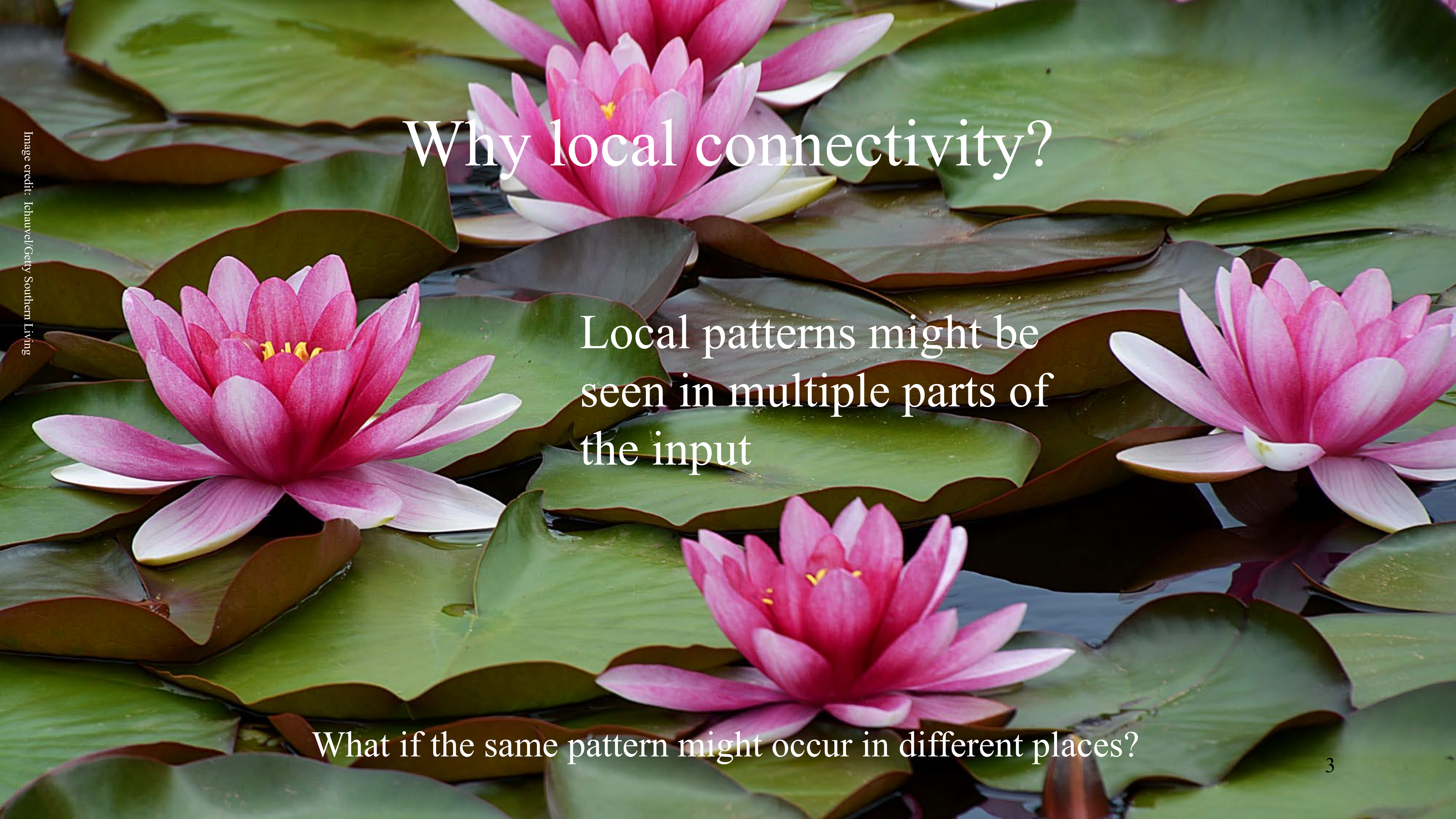


Image credit: lchauvel/Getty Southern Living

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)

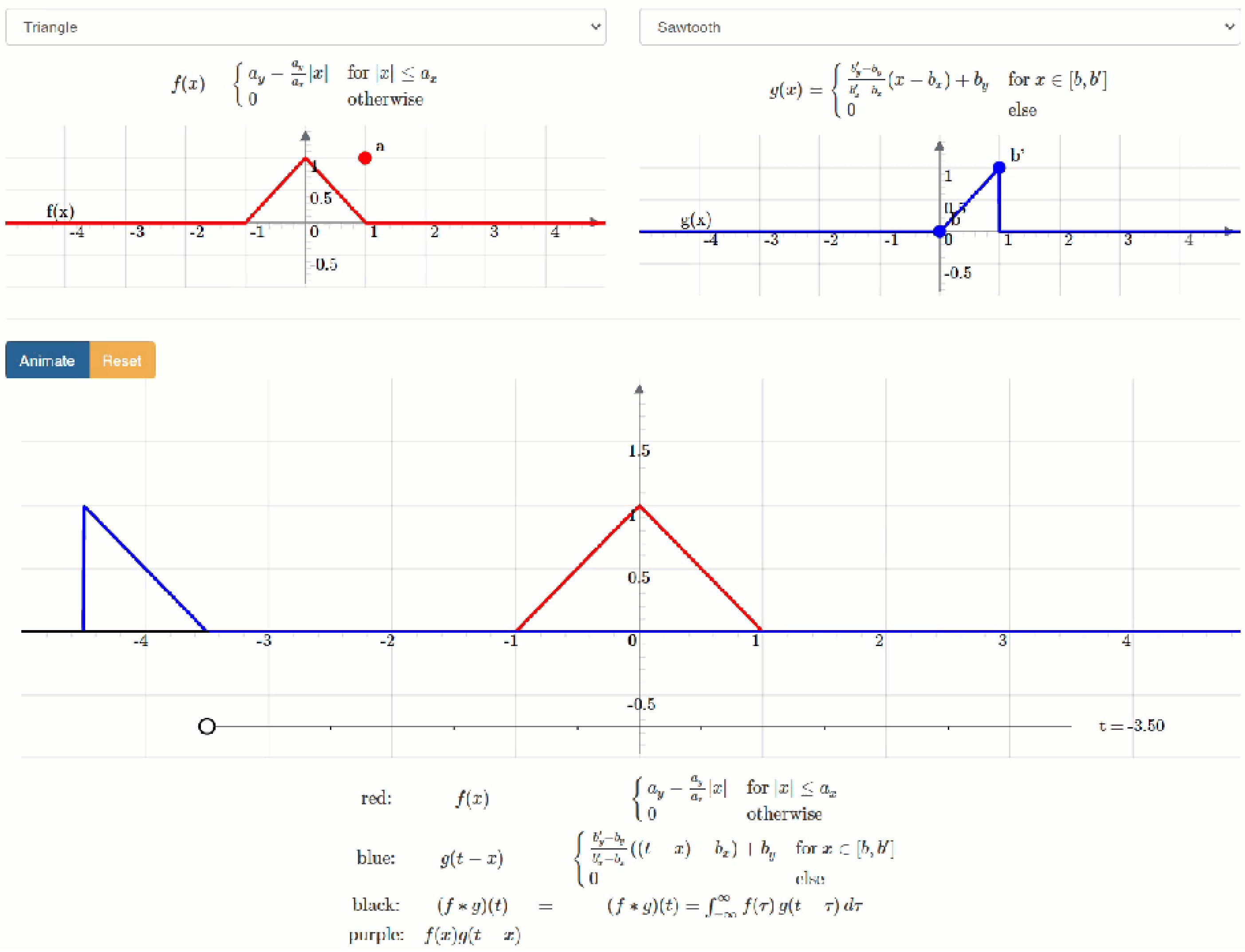
Convolution

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

$$\underbrace{x[t]}_{input} * \underbrace{w[t]}_{\substack{kernel \\ (filter)}} = \sum_{a=-\infty}^{\infty} x[a]w[t-a]$$

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

Convolution in 1D



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:

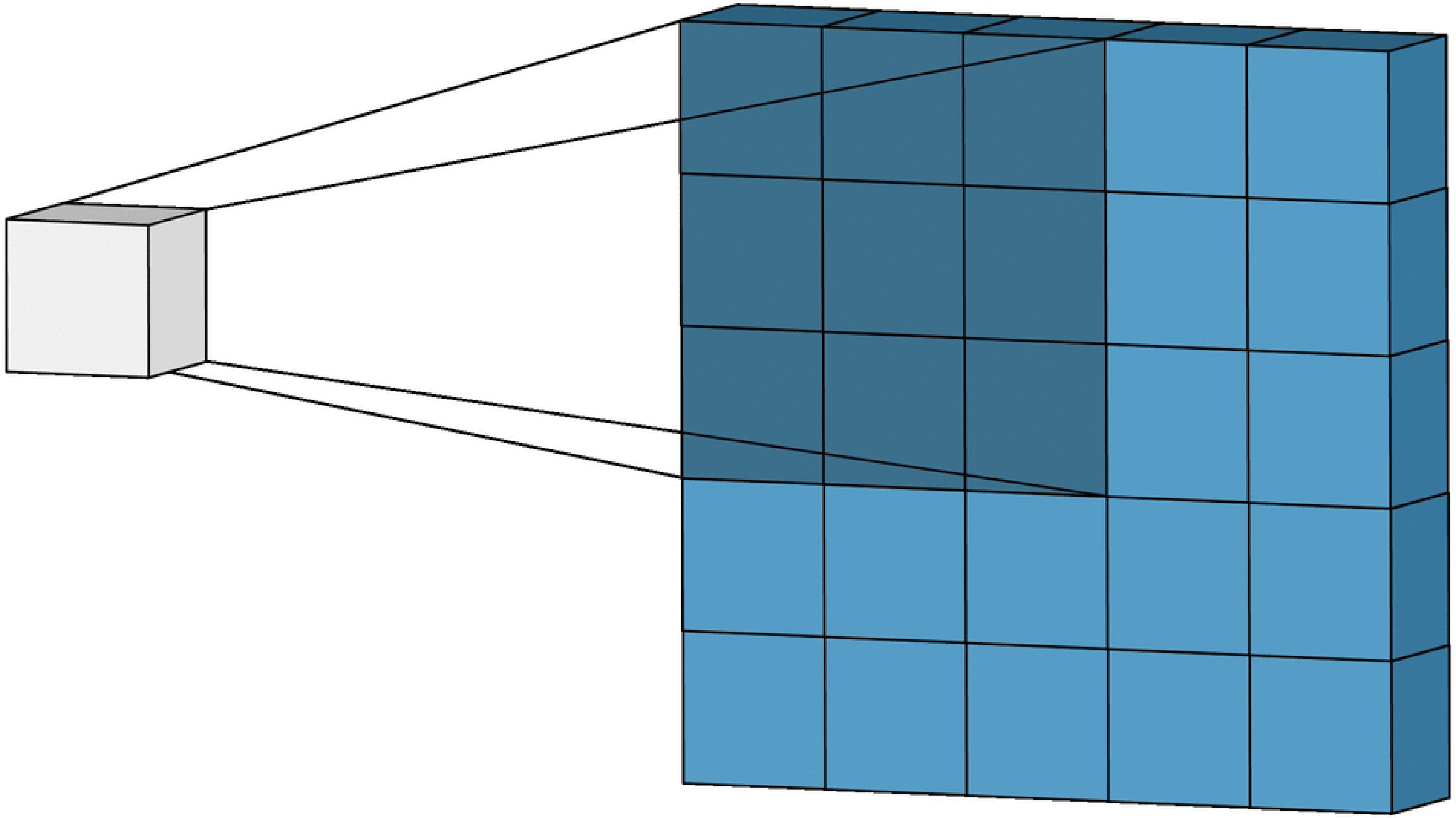
$$x[t,\omega] * w[t,\omega] = \sum_m \sum_n x[m,n]w[t-m,n-\omega]$$

- Many libraries implement cross-correlation instead of convolution (kernel not reversed, not important in practice):

$$\sum_m \sum_n x[t+m,\omega+n]w[m,n]$$

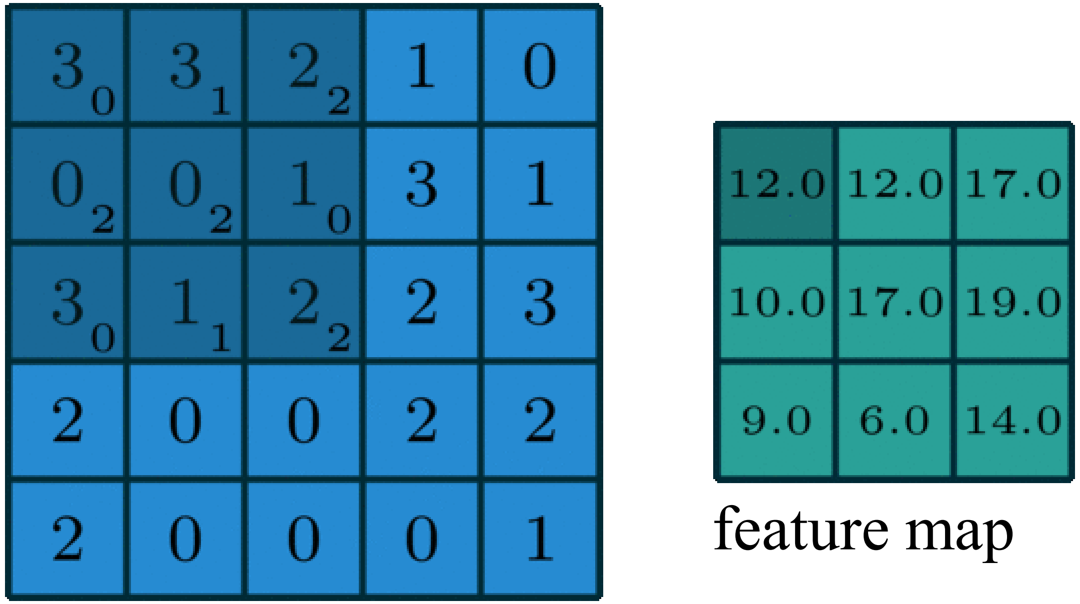
- Convolution is applied for each time-frequency cell $[t,\omega]$

Convolution in 2D



Animation credit: Irhum Shafkat

Convolution in 2D



feature map

$$w = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$

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

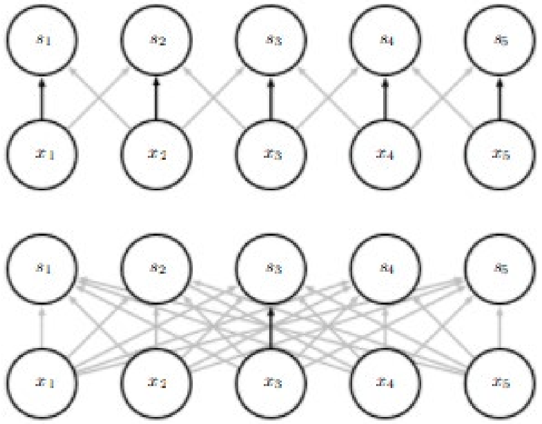


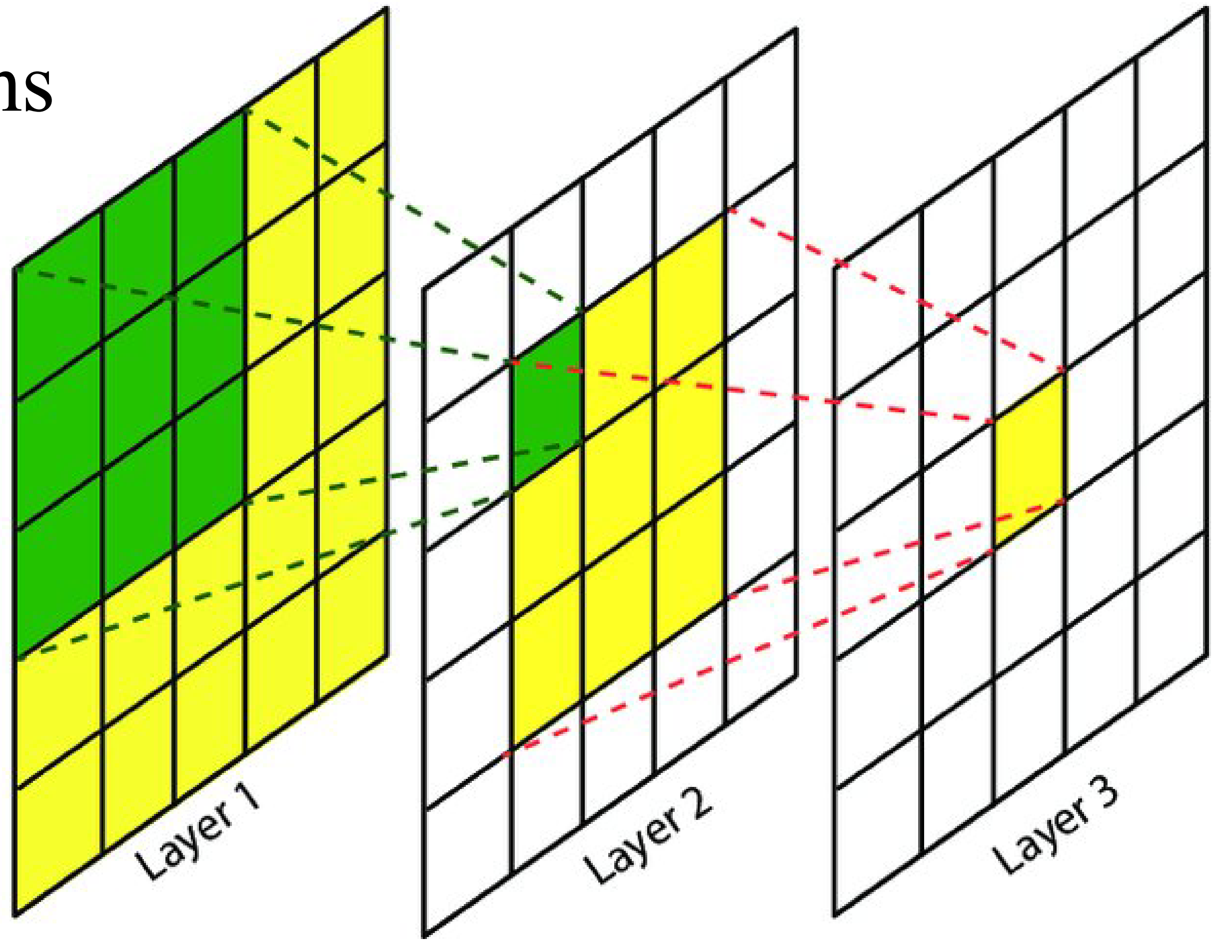
Fig. 9.2 Goodfellow et al.

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



Lin et al 2017 doi:10.3390/rs9050480

Equivariance under translation

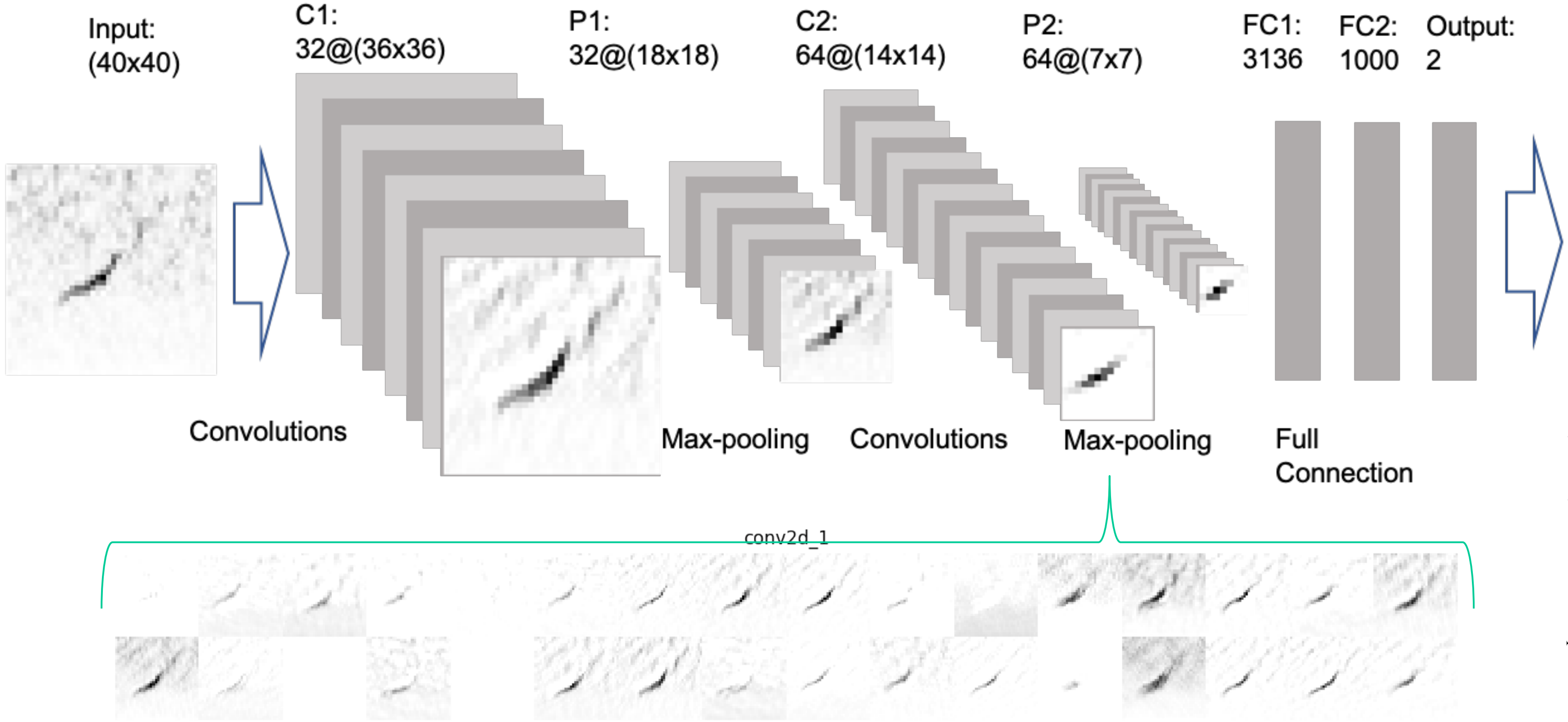
- When something in the input shifts position, its representation (convolution with the kernel) will also shift position.

Image: John Tlumacki, Boston Globe

image: NOAA

Key ideas

Convolutional layers learn a representation



Shiu, Palmer, et al. 2019 *Sci Rep*

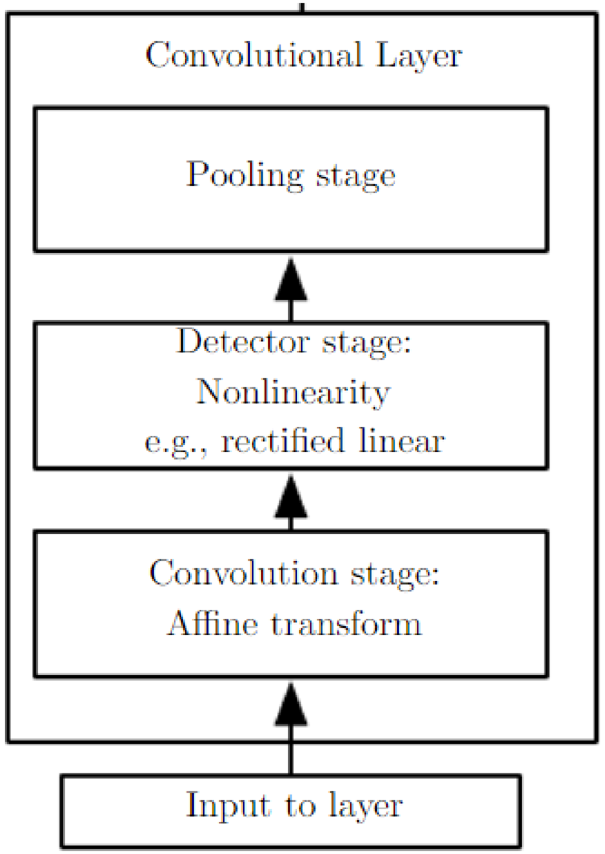
North Atlantic right whale upcall

Key ideas

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

Common architecture

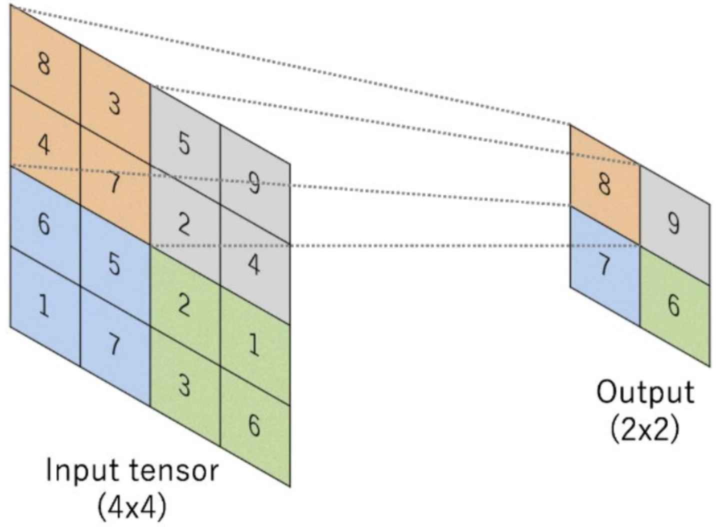
- Linear activation in first stages (not all networks use this)
- Nonlinear activations
- Pooling



Goodfellow et al 9.7

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 $f_l(i), 1 \le i \le l_f$.
- In layer $l+1$, most deepnet libraries will combine the l_f filter outputs with each of $l+1$'s convolutional filters:

$$fout_{l+1}(i) = b_{l+1}(i) + \sum_{j=1}^{l_f} fout_l(j) * f_{(l+1)}(i)$$

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

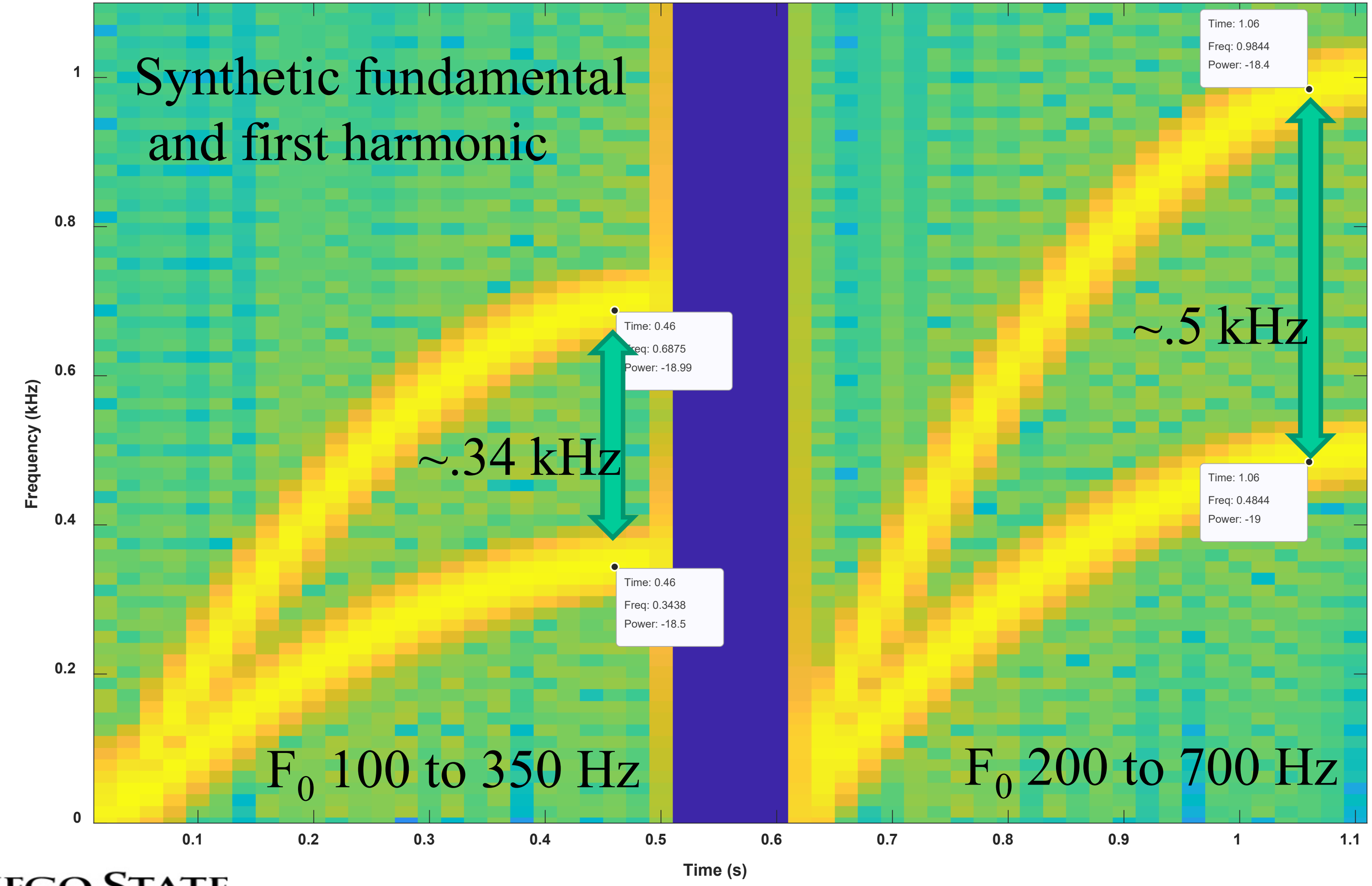
Are CNNs appropriate for audio?

Not always:

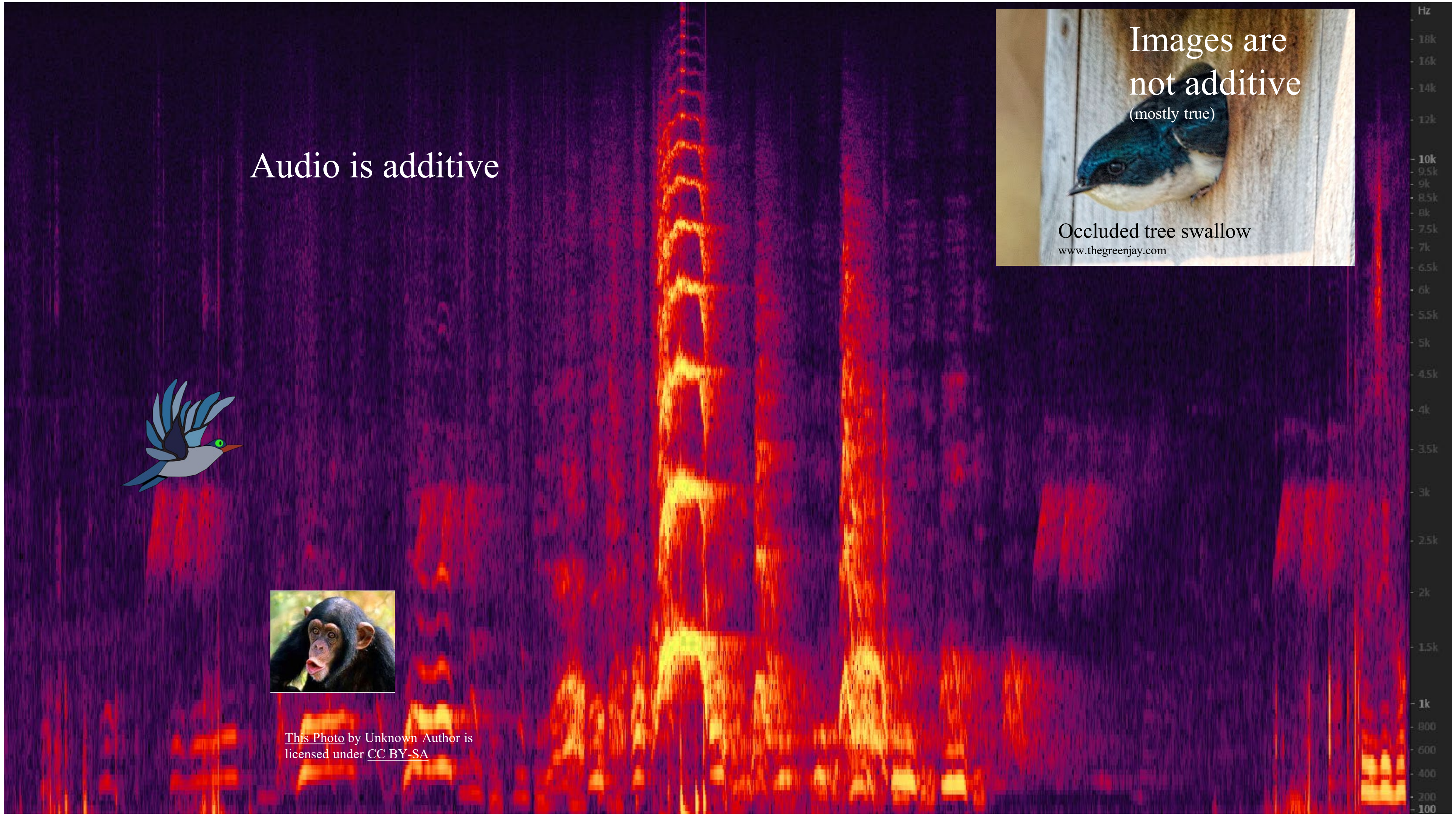
- Position has meaning

- Harmonic structure changes across the spectrum

Harmonic structure



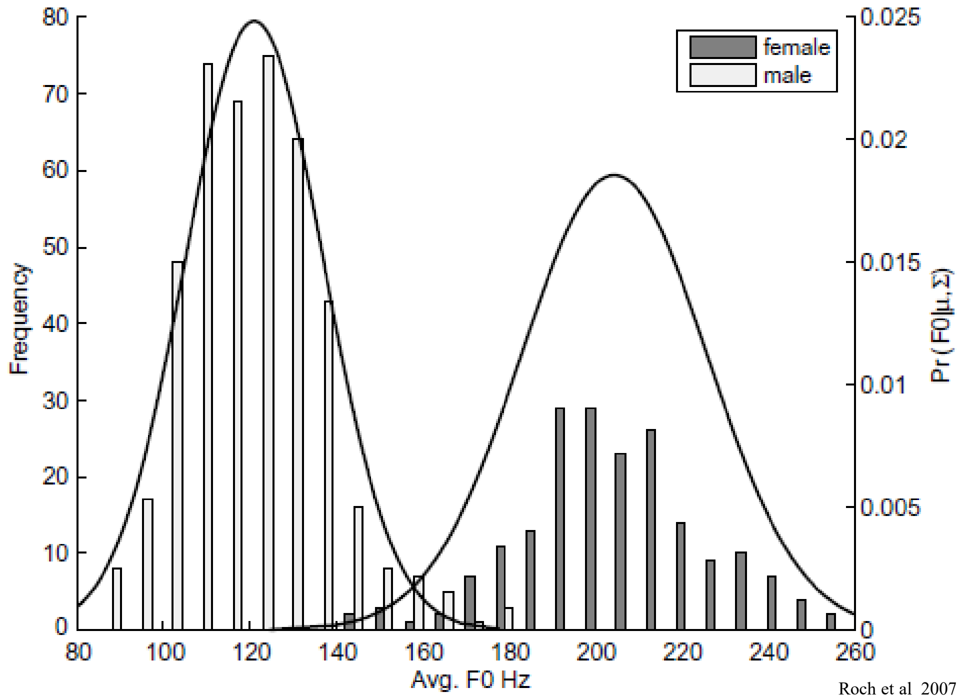
San Diego State University



SAN DIEGO STATE UNIVERSITY


Positions within images are only
meaningful in relation to other things

Shifting frequency changes meaning…



Roch et al 2007

SAN DIEGO STATE UNIVERSITY

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

Keras convolutional layers

```
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'))
```


Use Conv1D for 1D convolution