Convolution

CISC 7026: Introduction to Deep Learning

University of Macau

Agenda

1. Review
2. Signal Processing
3. Convolution
4. Convolutional Neural Networks
5. Additional Dimensions
6. Coding

Agenda

1. **Review**
2. Signal Processing
3. Convolution
4. Convolutional Neural Networks
5. Additional Dimensions
6. Coding

Review

Agenda

1. **Review**
2. Signal Processing
3. Convolution
4. Convolutional Neural Networks
5. Additional Dimensions
6. Coding

Agenda

1. Review
2. **Signal Processing**
3. Convolution
4. Convolutional Neural Networks
5. Additional Dimensions
6. Coding

Signal Processing

So far, we have not considered the structure of inputs X

Signal Processing

So far, we have not considered the structure of inputs X

We treat images as a vector, with no relationship between neighboring pixels

Signal Processing

So far, we have not considered the structure of inputs X

We treat images as a vector, with no relationship between neighboring pixels

Neurons i in layer ℓ has no relationship to neuron j in layer ℓ

Signal Processing

So far, we have not considered the structure of inputs X

We treat images as a vector, with no relationship between neighboring pixels

Neurons i in layer ℓ has no relationship to neuron j in layer ℓ

However, there is structure inherent in the real world

Signal Processing

So far, we have not considered the structure of inputs X

We treat images as a vector, with no relationship between neighboring pixels

Neurons i in layer ℓ has no relationship to neuron j in layer ℓ

However, there is structure inherent in the real world

By representing this structure, we can make more efficient neural networks that generalize better

Signal Processing

So far, we have not considered the structure of inputs X

We treat images as a vector, with no relationship between neighboring pixels

Neurons i in layer ℓ has no relationship to neuron j in layer ℓ

However, there is structure inherent in the real world

By representing this structure, we can make more efficient neural networks that generalize better

To do so, we must think of the world as a collection of signals

Signal Processing

A **signal** represents information as a function of time, space or some other variable

Signal Processing

A **signal** represents information as a function of time, space or some other variable

$$x(t) = \ldots$$

$$x(u, v) = \ldots$$

Signal Processing

A **signal** represents information as a function of time, space or some other variable

$$x(t) = \ldots$$

$$x(u, v) = \ldots$$

$x(t), x(u, v)$ are some physical processes that we may or may not know

Signal Processing

A **signal** represents information as a function of time, space or some other variable

$$x(t) = \dots$$

$$x(u, v) = \dots$$

$x(t), x(u, v)$ are some physical processes that we may or may not know

Signal processing is a field of research that focuses on analyzing the meaning of signals

Signal Processing

A **signal** represents information as a function of time, space or some other variable

$$x(t) = ...$$

$$x(u, v) = ...$$

$x(t), x(u, v)$ are some physical processes that we may or may not know

Signal processing is a field of research that focuses on analyzing the meaning of signals

Knowing the meaning of signals is very useful

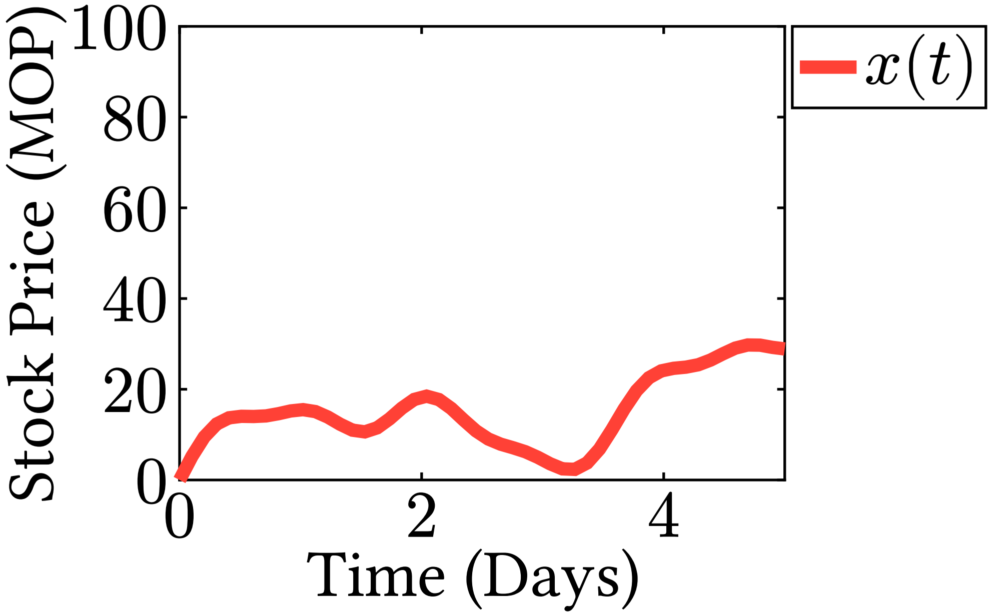
Signal Processing

Signal Processing

$$x(t) = \text{stock price}$$

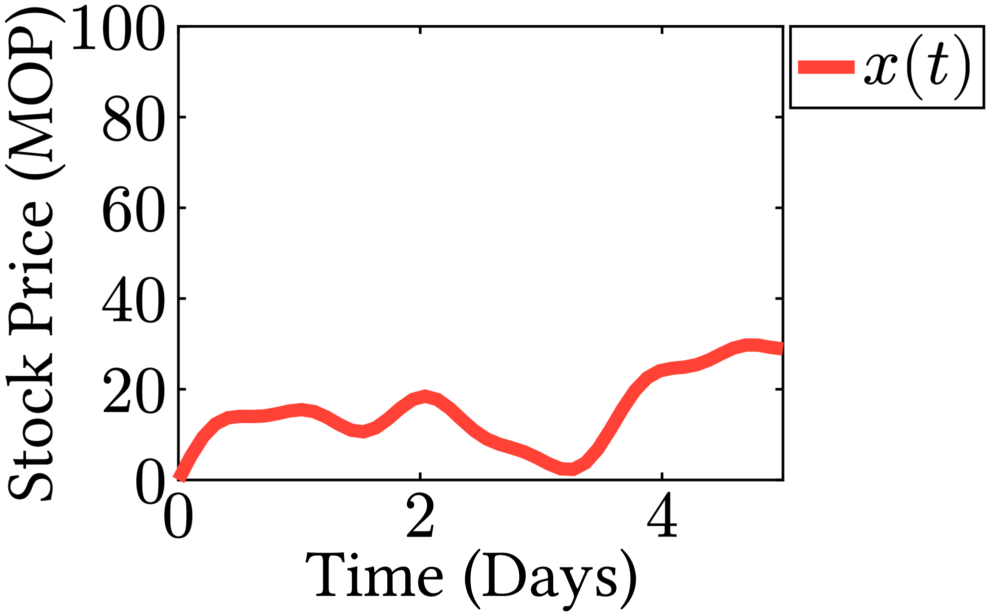
Signal Processing

$$x(t) = \text{stock price}$$



Signal Processing

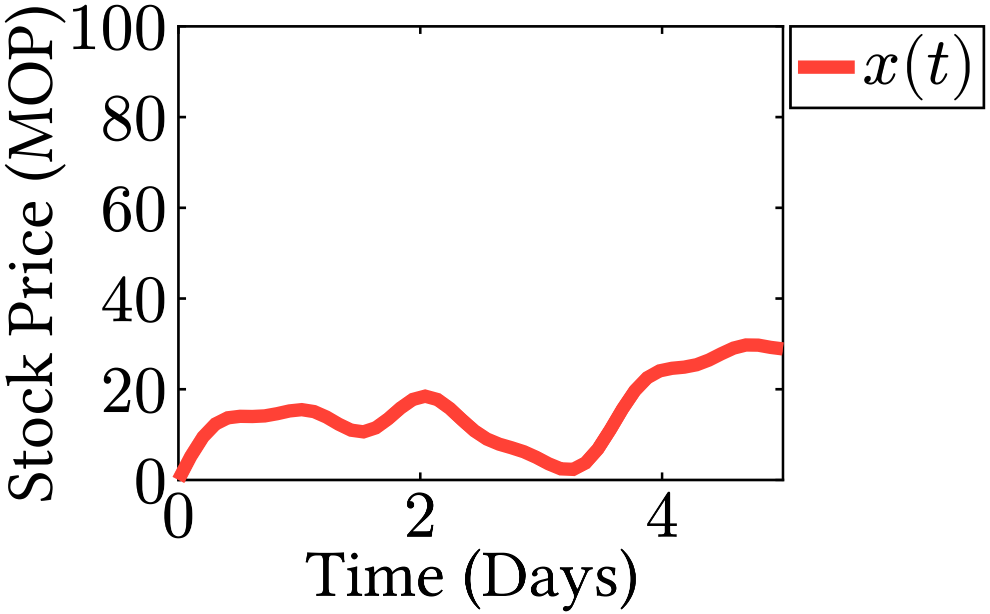
$$x(t) = \text{stock price}$$



There is an underlying structure that we do not fully understand

Signal Processing

$$x(t) = \text{stock price}$$



There is an underlying structure that we do not fully understand

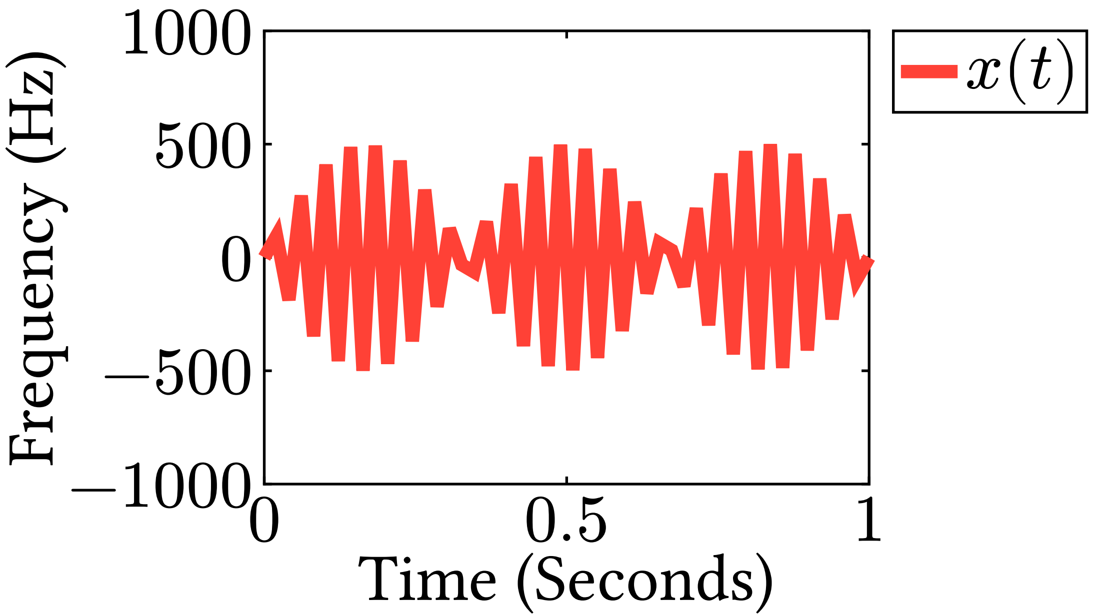
Structure: Tomorrow's stock price will be close to today's stock price

Signal Processing

$$x(t) = \text{audio}$$

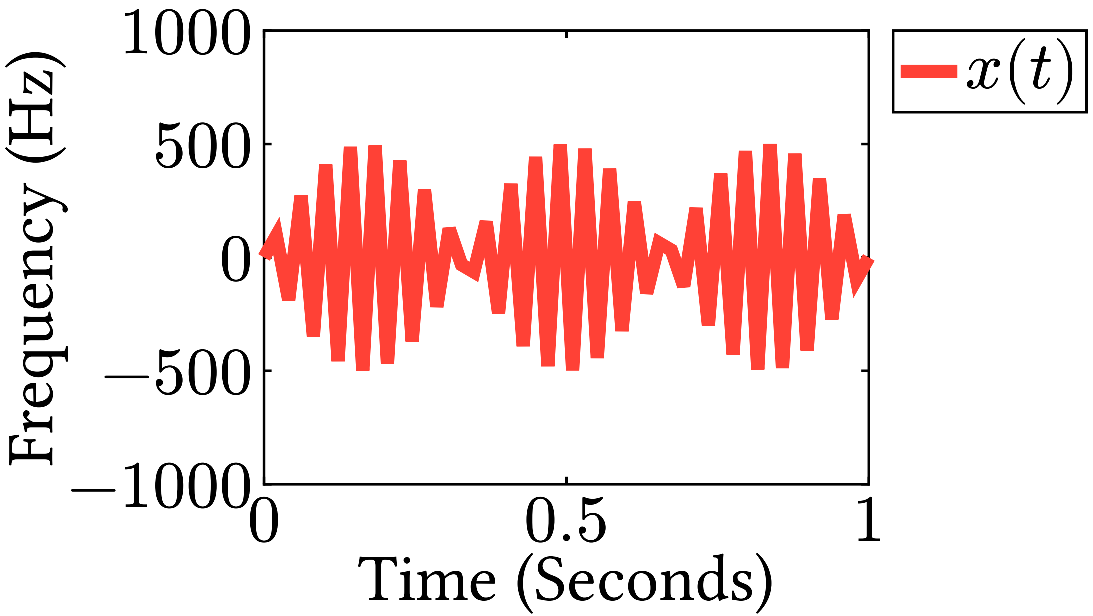
Signal Processing

$$x(t) = \text{audio}$$



Signal Processing

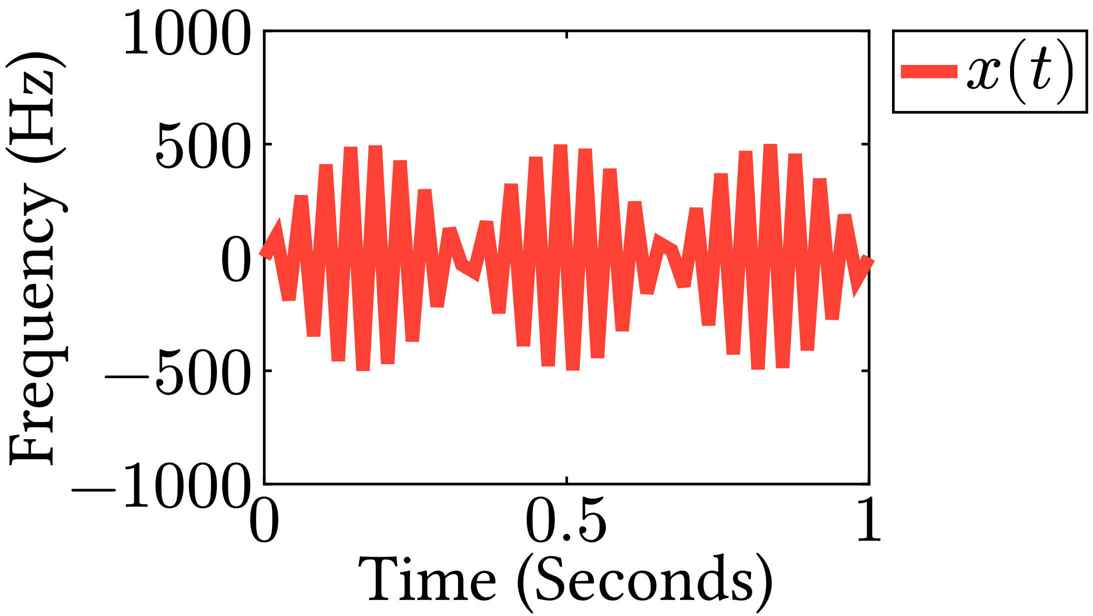
$$x(t) = \text{audio}$$



Structure: Nearby waves form syllables

Signal Processing

$$x(t) = \text{audio}$$



Structure: Nearby waves form syllables

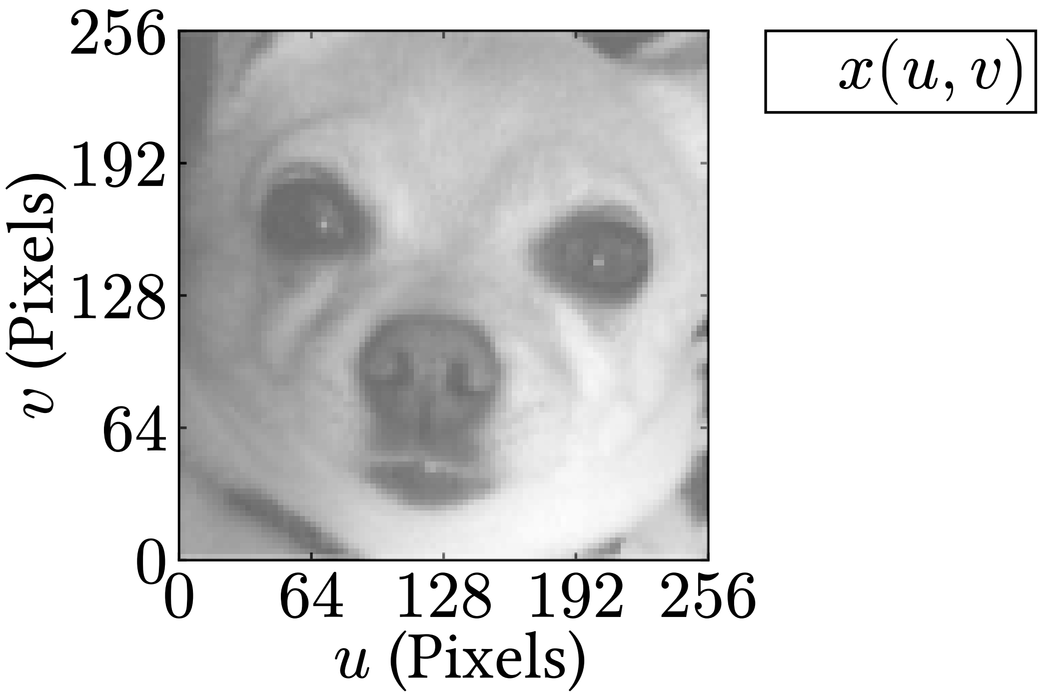
Structure: Nearby syllables combine to create meaning

Signal Processing

$$x(u, v) = \text{image}$$

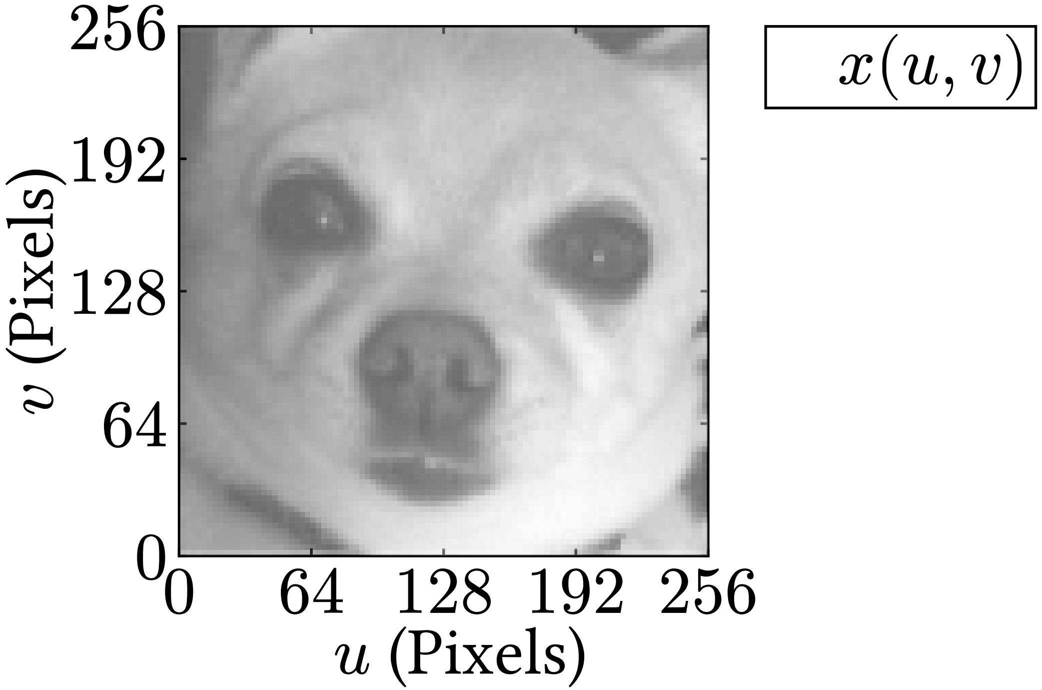
Signal Processing

$$x(u, v) = \text{image}$$



Signal Processing

$$x(u, v) = \text{image}$$



Structure: Repeated components (circles, symmetry, eyes, nostrils, etc)

Signal Processing

In signal processing, we often consider:

Signal Processing

In signal processing, we often consider:

- Locality

Signal Processing

In signal processing, we often consider:

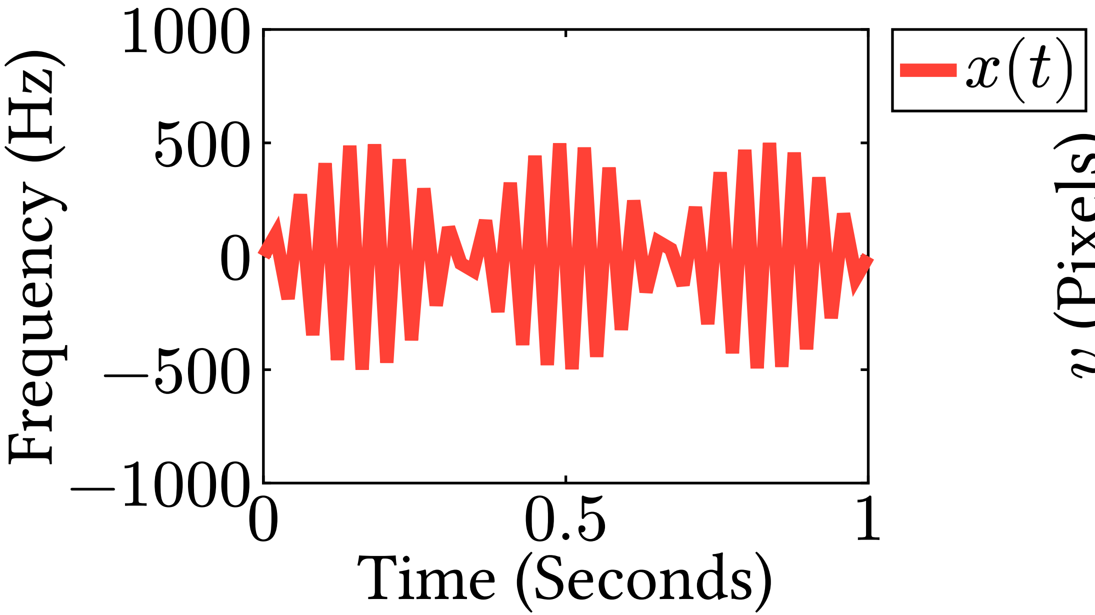
- Locality
- Translation equivariance

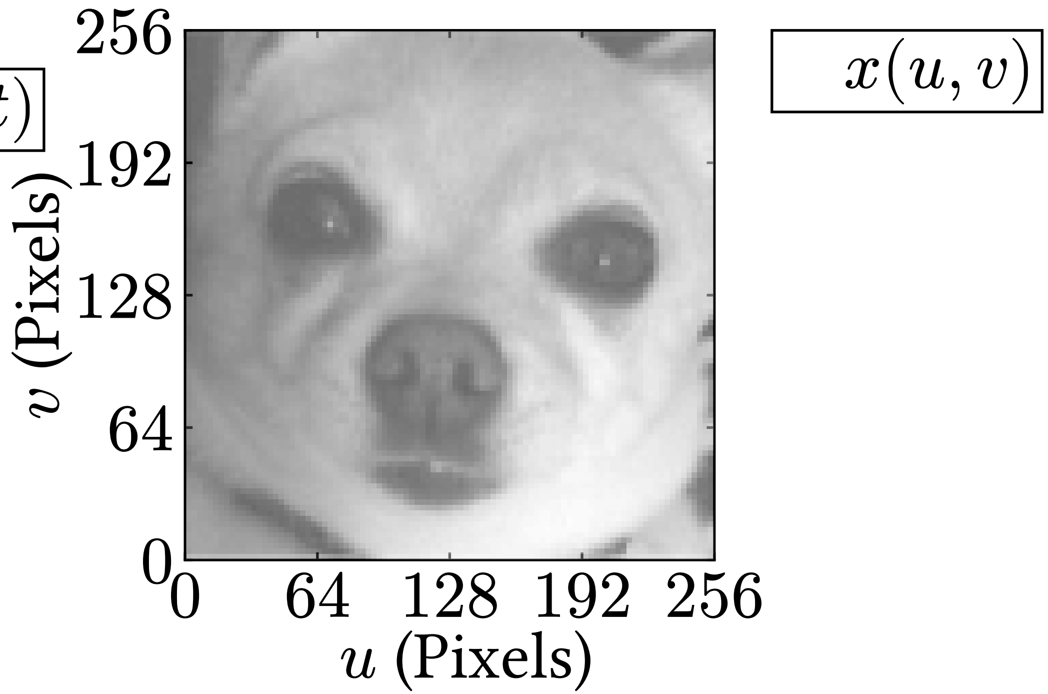
Signal Processing

Locality: Information concentrated over small regions of space/time

Signal Processing

Locality: Information concentrated over small regions of space/time



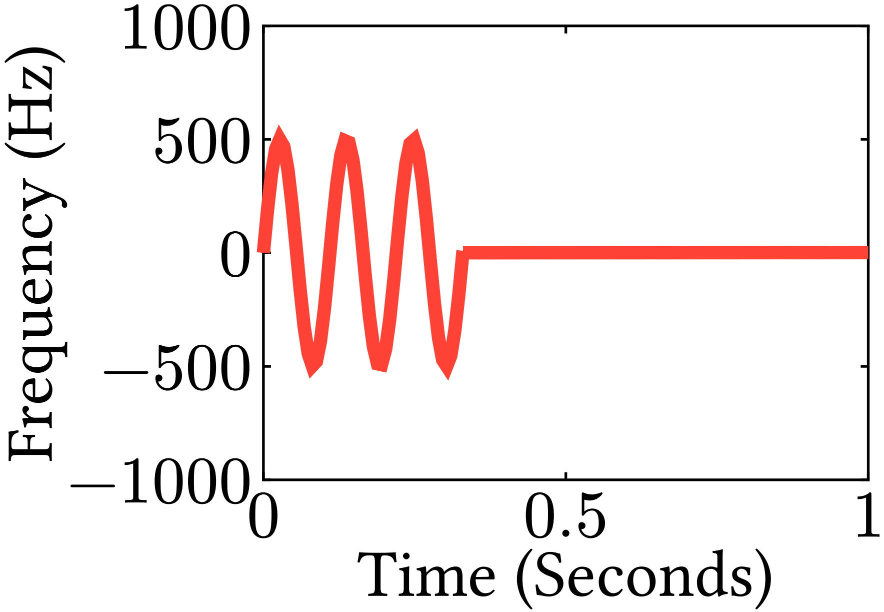


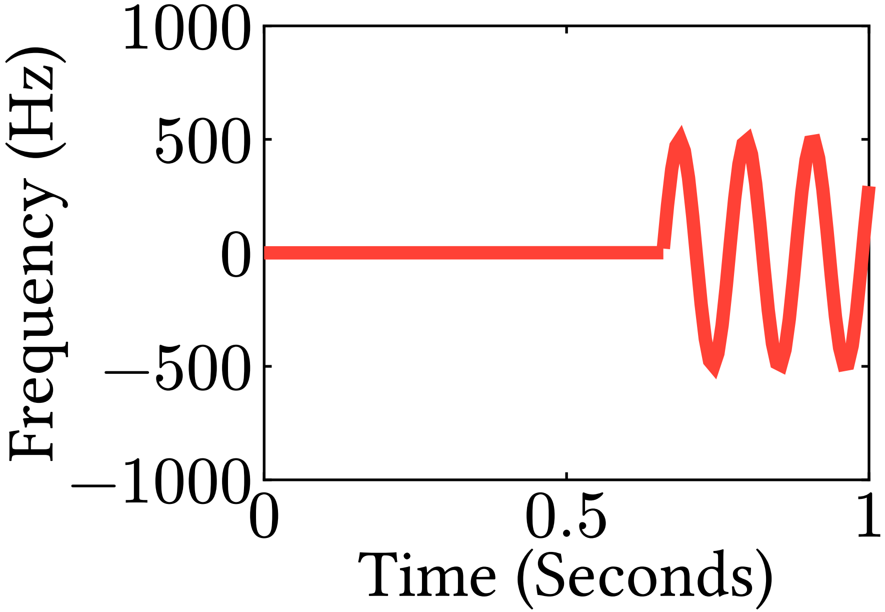
Signal Processing

Translation Equivariance: Shift in signal results in shift in output

Signal Processing

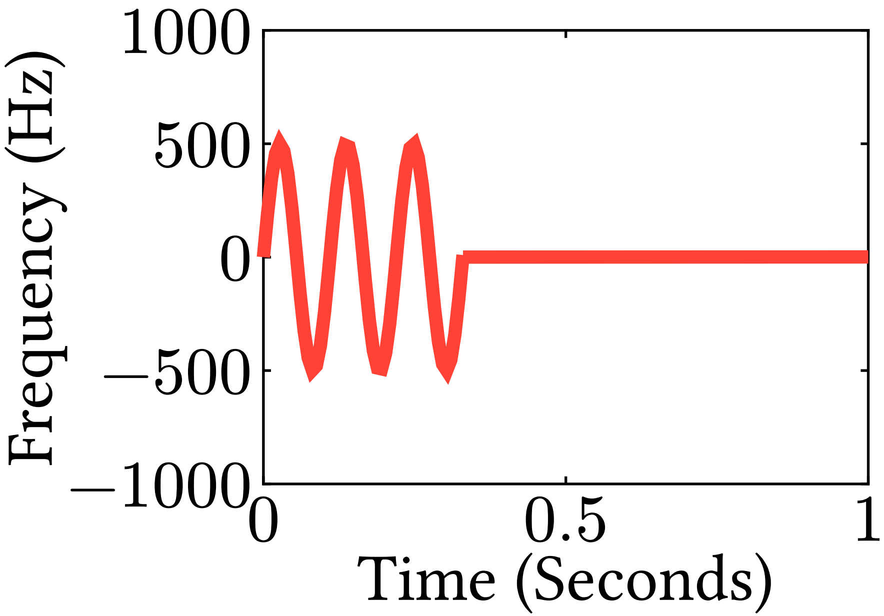
Translation Equivariance: Shift in signal results in shift in output

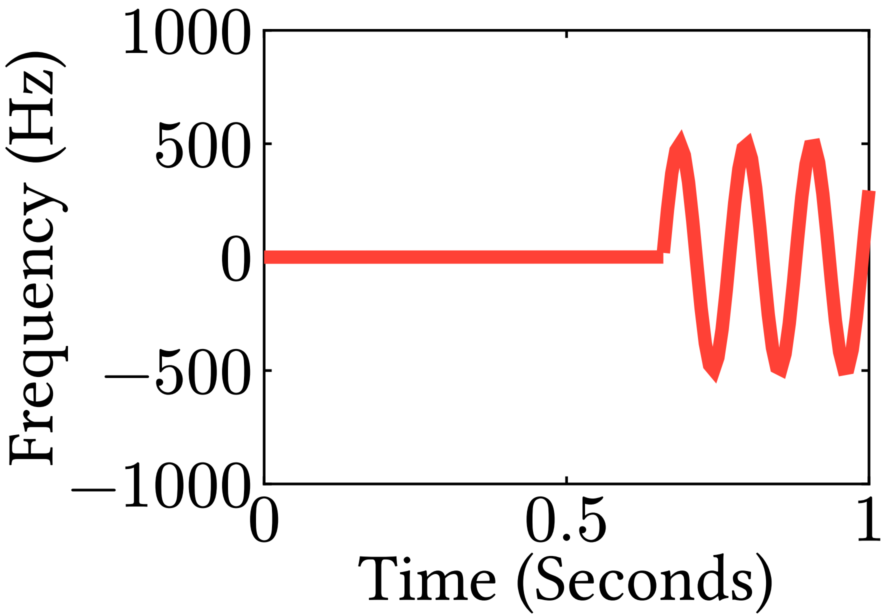




Signal Processing

Translation Equivariance: Shift in signal results in shift in output



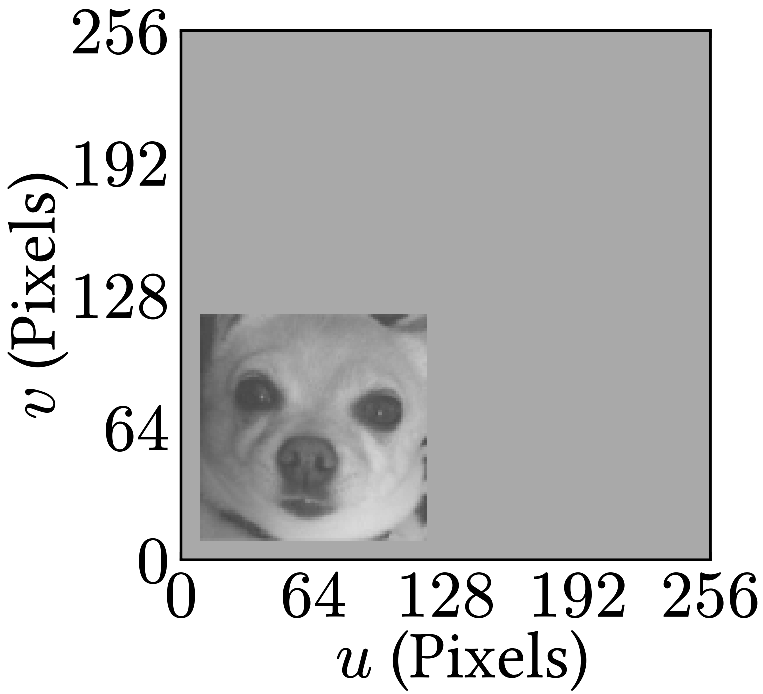


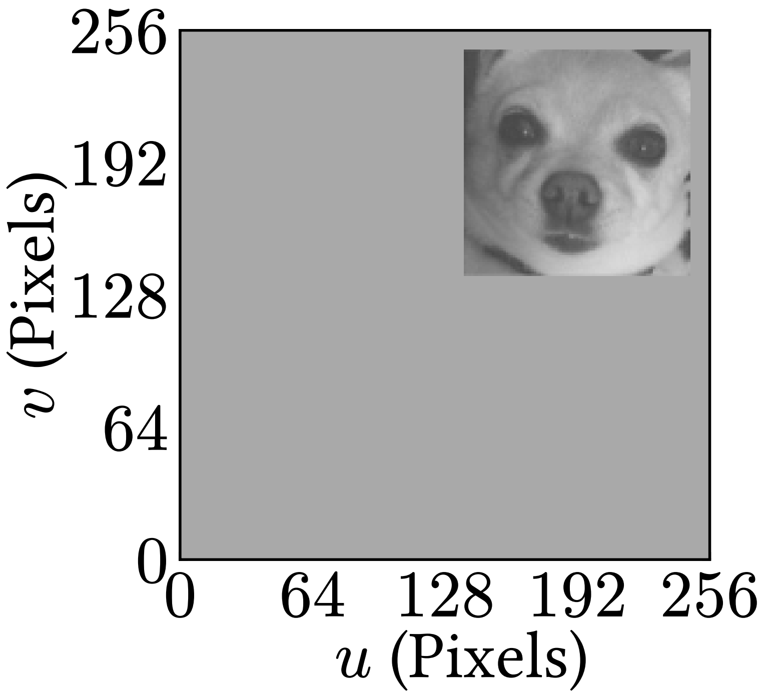
Both say “hello”

Signal Processing

Translation Equivariance: Shift in signal results in shift in output

Translation Equivariance: Shift in signal results in shift in output





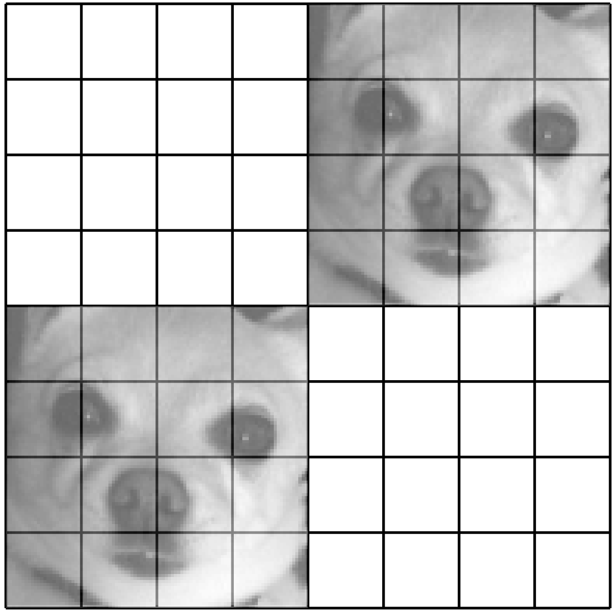
Both contain a dog

Signal Processing

Perceptrons are not local or translation equivariant, each pixel is an independent neuron

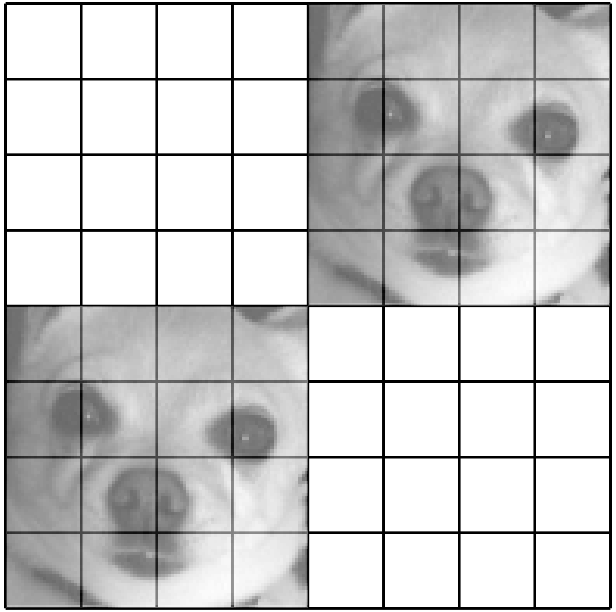
Signal Processing

Perceptrons are not local or translation equivariant, each pixel is an independent neuron



Signal Processing

Perceptrons are not local or translation equivariant, each pixel is an independent neuron

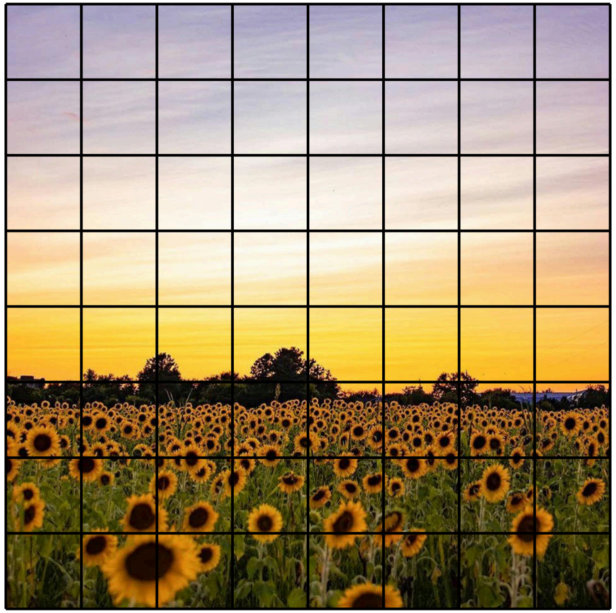


Signal Processing

A more realistic scenario of locality and translation equivariance

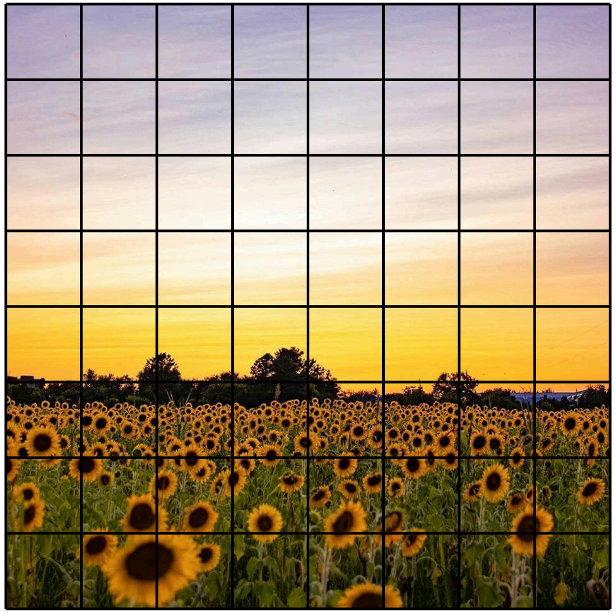
Signal Processing

A more realistic scenario of locality and translation equivariance



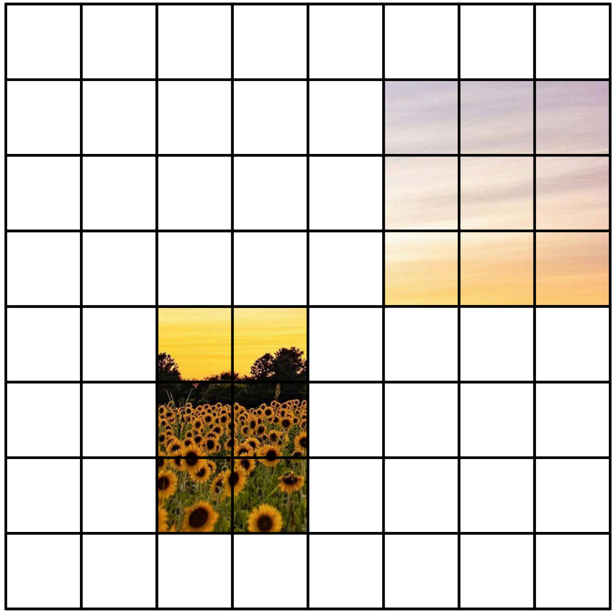
Signal Processing

A more realistic scenario of locality and translation equivariance



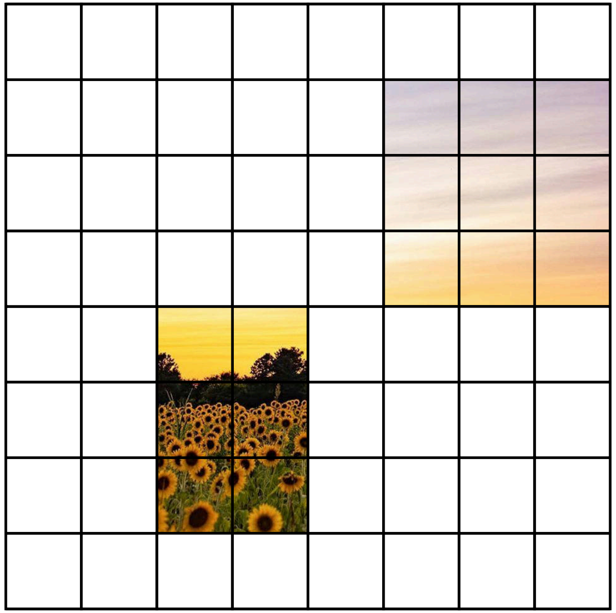
Signal Processing

A more realistic scenario of locality and translation equivariance



Signal Processing

A more realistic scenario of locality and translation equivariance



Agenda

1. Review
2. **Signal Processing**
3. Convolution
4. Convolutional Neural Networks
5. Additional Dimensions
6. Coding

Agenda

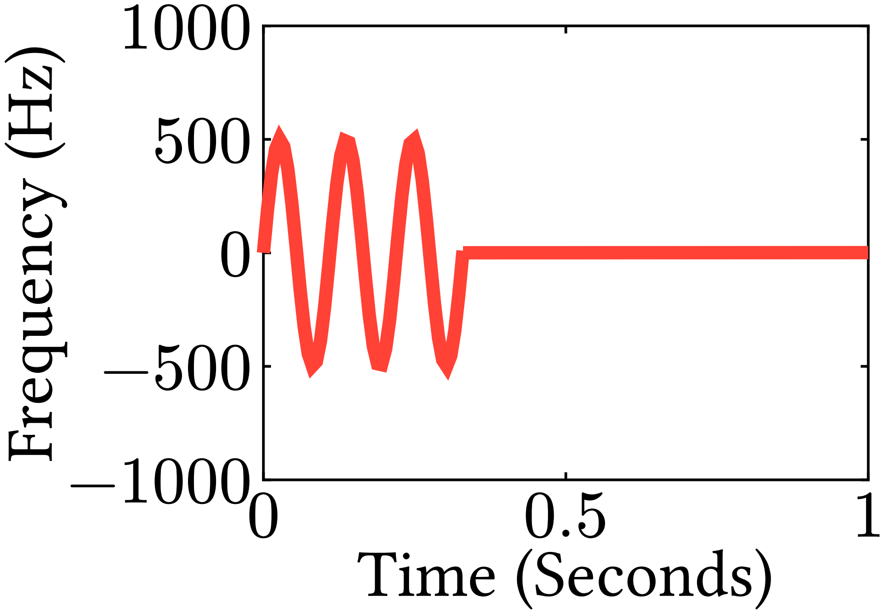
1. Review
2. Signal Processing
3. **Convolution**
4. Convolutional Neural Networks
5. Additional Dimensions
6. Coding

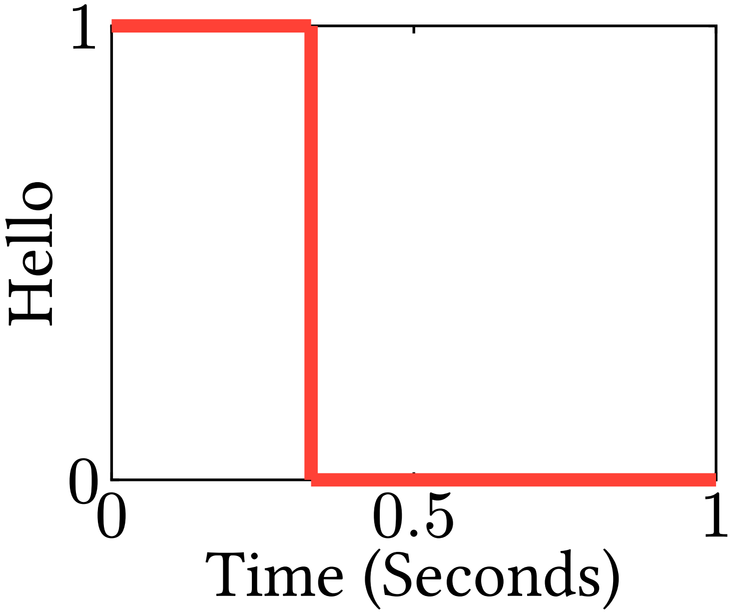
Convolution

In signal processing, we often turn signals into other signals

Convolution

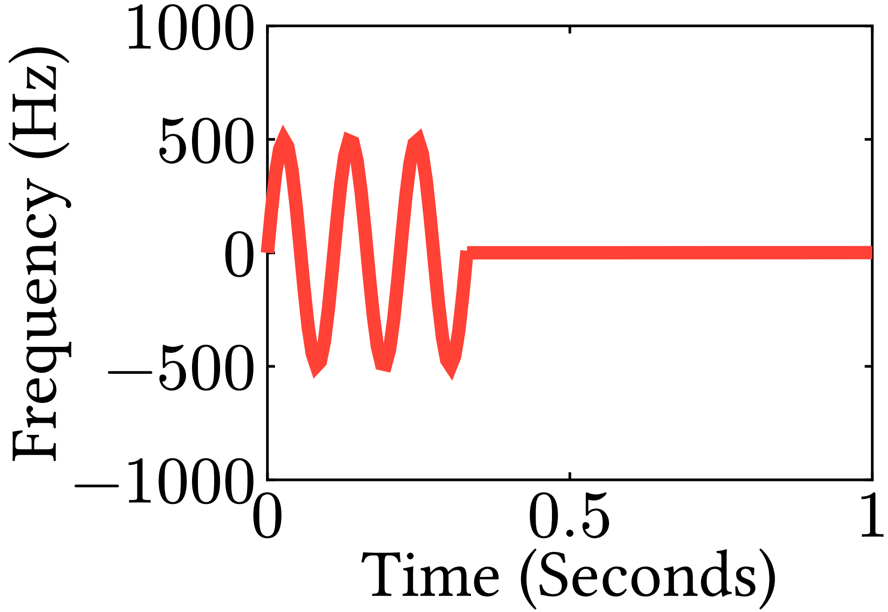
In signal processing, we often turn signals into other signals

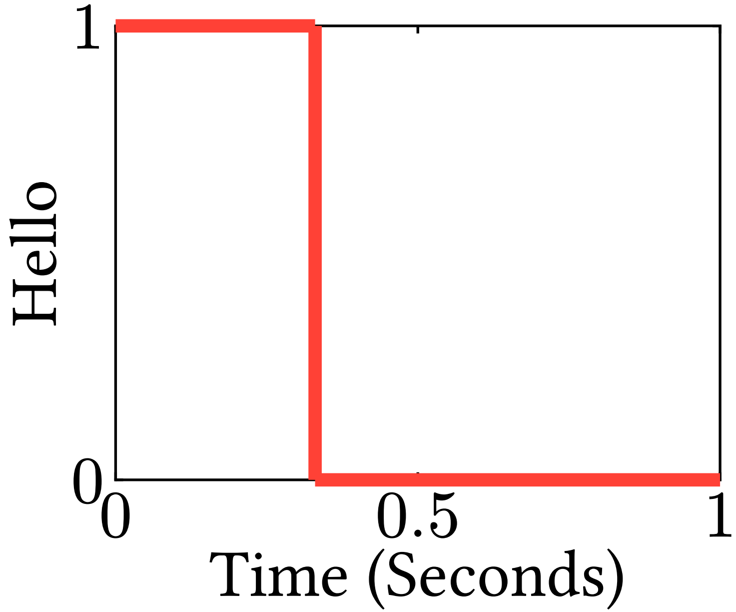




Convolution

In signal processing, we often turn signals into other signals

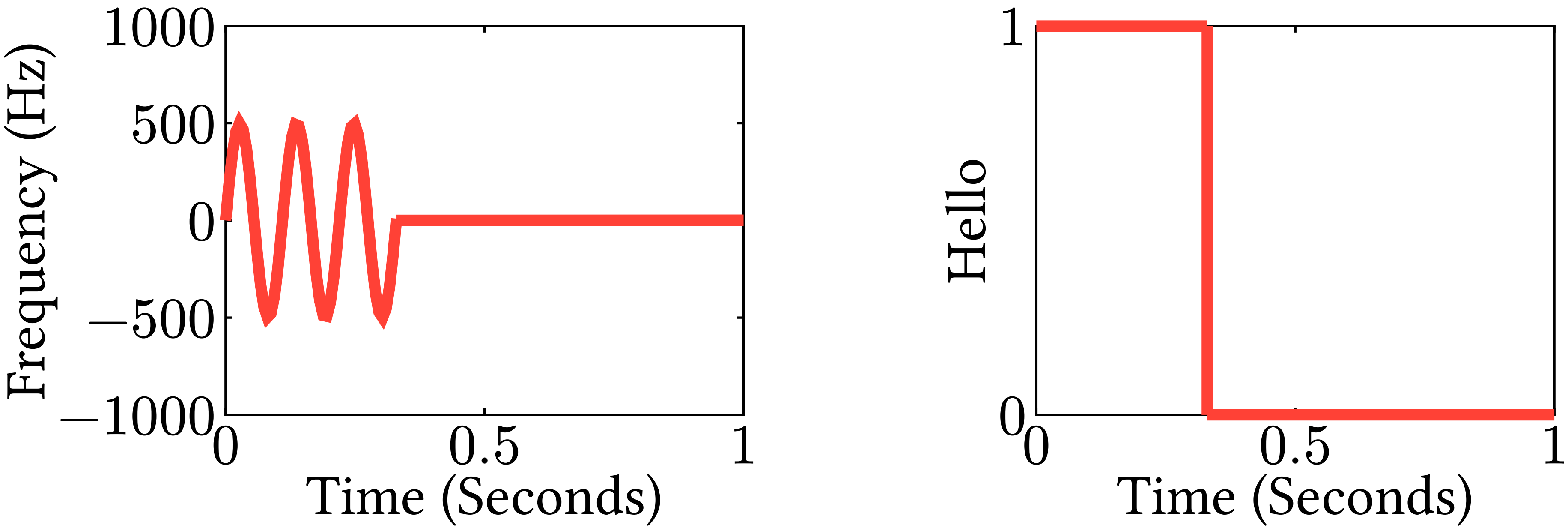




A standard way to transform signals is **convolution**

Convolution

In signal processing, we often turn signals into other signals



A standard way to transform signals is **convolution**

Convolution is translation equivariant and can be local

Convolution

Convolution is the sum of products of a signal $x(t)$ and a **filter** $g(t)$

Convolution

Convolution is the sum of products of a signal $x(t)$ and a **filter** $g(t)$

If the t is continuous in $x(t)$

$$x(t) * g(t) = \int_{-\infty}^{\infty} x(t - \tau) g(\tau) d\tau$$

Convolution

Convolution is the sum of products of a signal $x(t)$ and a **filter** $g(t)$

If the t is continuous in $x(t)$

$$x(t) * g(t) = \int_{-\infty}^{\infty} x(t - \tau) g(\tau) d\tau$$

If the t is discrete in $x(t)$

$$x(t) * g(t) = \sum_{\tau=-\infty}^{\infty} x(t - \tau) g(\tau)$$

Convolution

Convolution is the sum of products of a signal $x(t)$ and a **filter** $g(t)$

If the t is continuous in $x(t)$

$$x(t) * g(t) = \int_{-\infty}^{\infty} x(t - \tau) g(\tau) d\tau$$

If the t is discrete in $x(t)$

$$x(t) * g(t) = \sum_{\tau = -\infty}^{\infty} x(t - \tau) g(\tau)$$

We slide the filter $g(t)$ across the signal $x(t)$

Convolution

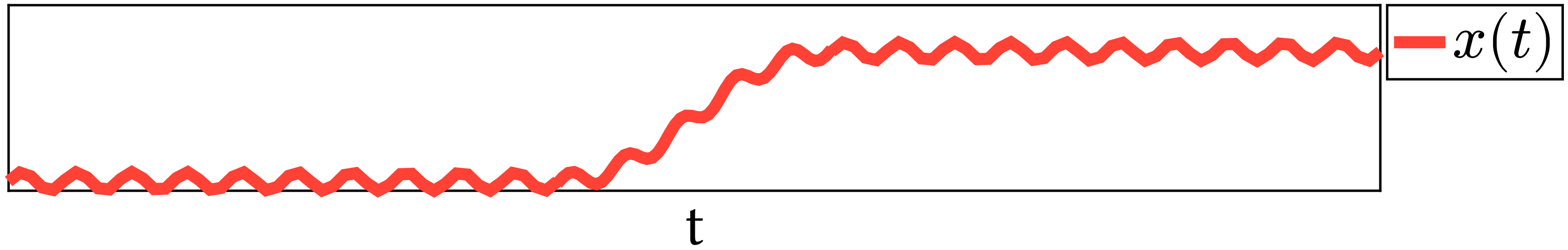
Example: Let us examine a low-pass filter

Convolution

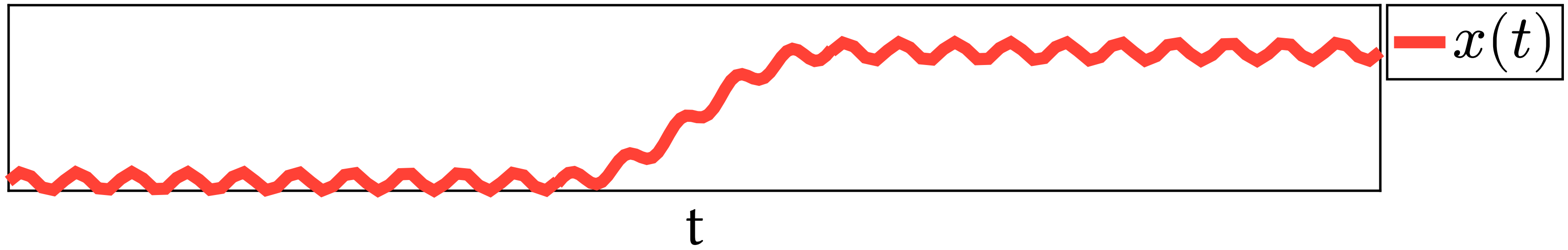
Example: Let us examine a low-pass filter

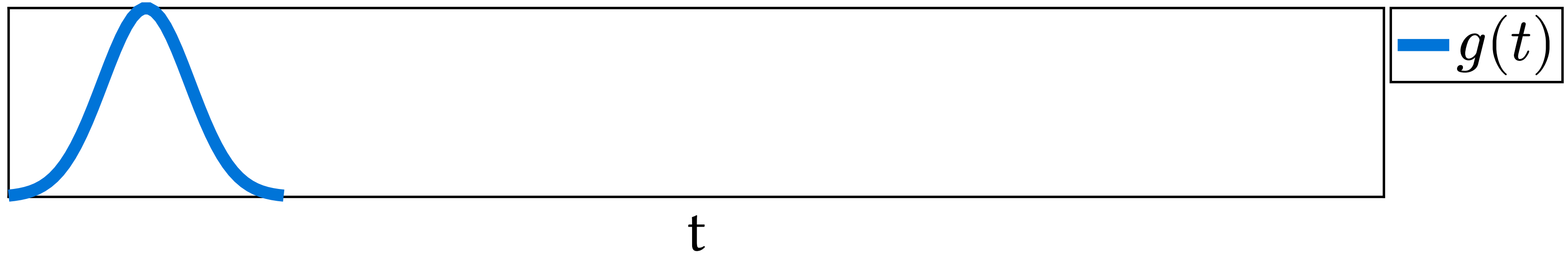
The filter will take a signal and remove noise, producing a cleaner signal

Convolution

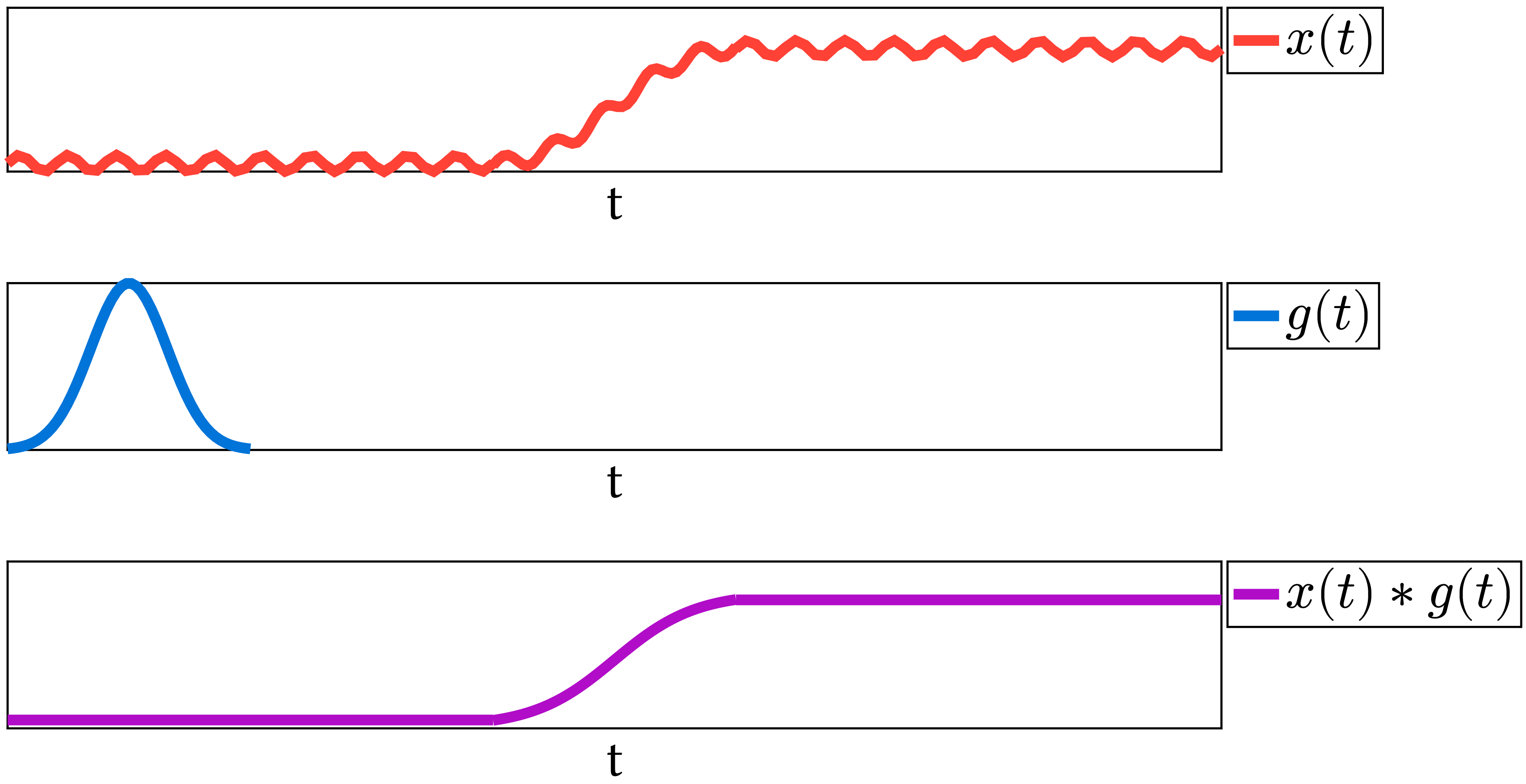


Convolution

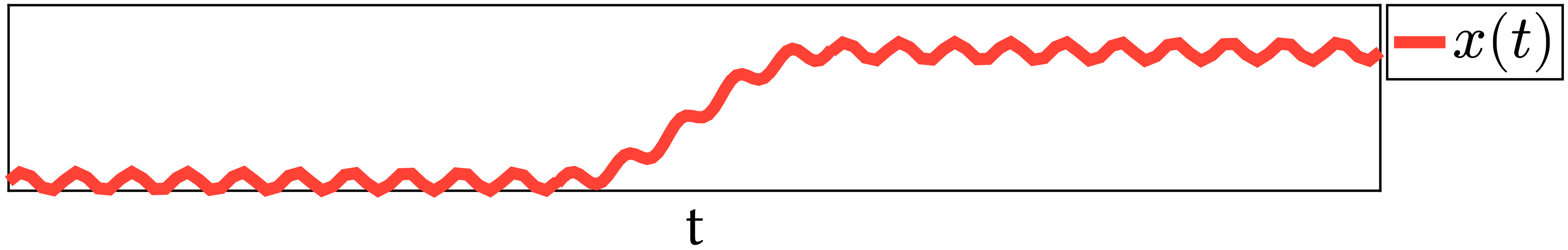


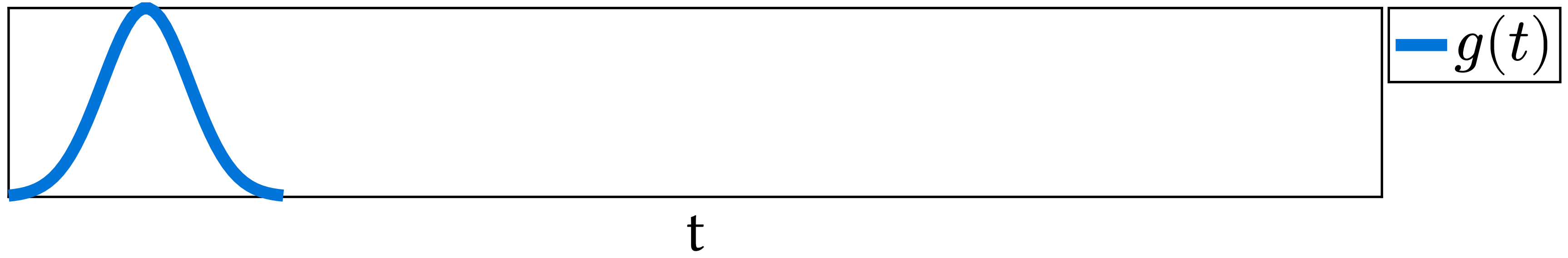


Convolution



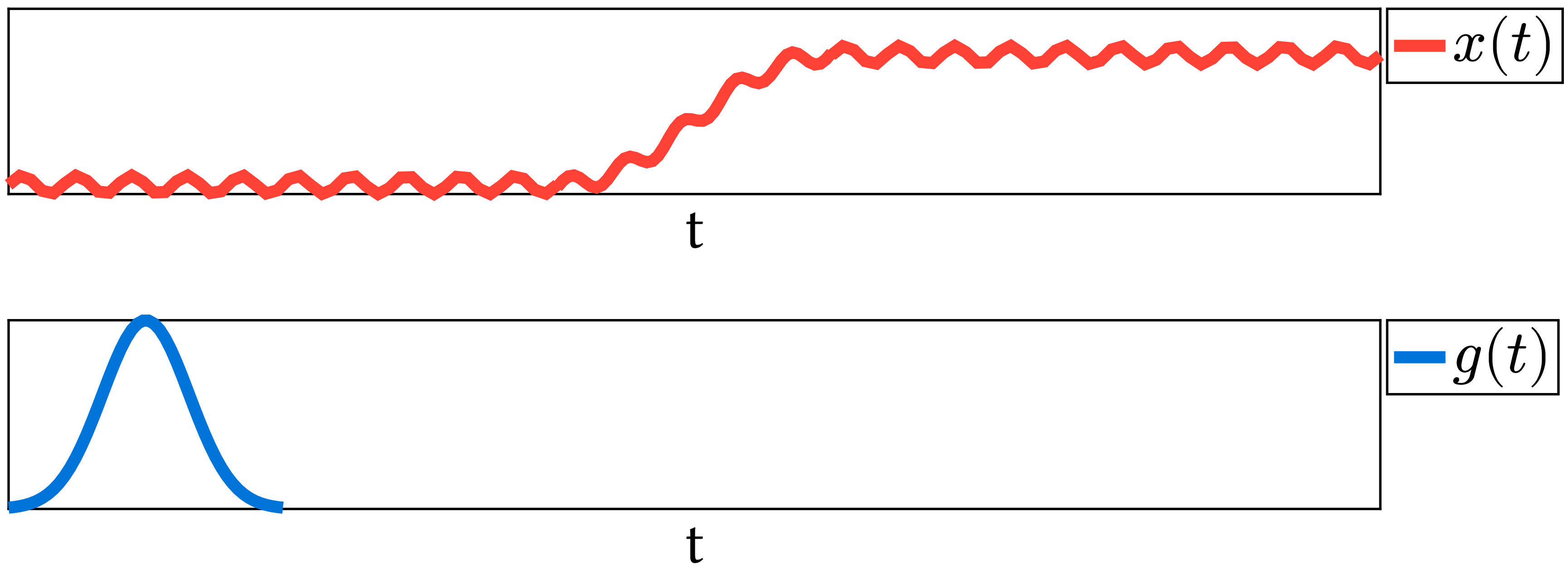
Convolution





Convolution is **local** to the filter $g(t)$

Convolution



Convolution is **local** to the filter $g(t)$

Convolution is also **equivariant** to time/space shifts

Convolution

Often, we use continuous time/space convolution for analog signals

Convolution

Often, we use continuous time/space convolution for analog signals

- Physics

Convolution

Often, we use continuous time/space convolution for analog signals

- Physics
- Control theory

Convolution

Often, we use continuous time/space convolution for analog signals

- Physics
- Control theory
- Electrical engineering

Convolution

Often, we use continuous time/space convolution for analog signals

- Physics
- Control theory
- Electrical engineering

Almost all deep learning occurs on digital hardware (discrete time/space)

- Images

Convolution

Often, we use continuous time/space convolution for analog signals

- Physics
- Control theory
- Electrical engineering

Almost all deep learning occurs on digital hardware (discrete time/space)

- Images
- Quantized audio

Convolution

Often, we use continuous time/space convolution for analog signals

- Physics
- Control theory
- Electrical engineering

Almost all deep learning occurs on digital hardware (discrete time/space)

- Images
- Quantized audio
- Anything stored as bits instead of a function

Convolution

Often, we use continuous time/space convolution for analog signals

- Physics
- Control theory
- Electrical engineering

Almost all deep learning occurs on digital hardware (discrete time/space)

- Images
- Quantized audio
- Anything stored as bits instead of a function

But it is good to know both! Continuous variables for theory. Discrete variables for software

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 2 & 1 & & & \\ 1 & 2 & 3 & 4 & 5 \\ & & & & \end{bmatrix}$$

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 2 & 1 & & & \\ 1 & 2 & 3 & 4 & 5 \\ 4 & & & & \end{bmatrix}$$

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} & \color{red}{2} & \color{red}{1} & & \\ 1 & \color{red}{2} & \color{red}{3} & 4 & 5 \\ 4 & \color{red}{5} & & & \end{bmatrix}$$

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} & & 2 & 1 & \\ 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 10 & & \end{bmatrix}$$

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} & & & \color{red}{2} & \color{red}{1} \\ 1 & 2 & 3 & \color{red}{4} & \color{red}{5} \\ 4 & 5 & 10 & \color{red}{13} & \end{bmatrix}$$

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 2 & 1 & & & \\ 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 10 & 13 & \end{bmatrix}$$

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 2 & 1 & & & \\ 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 10 & 13 & \end{bmatrix}$$

Question: Does anybody see a connection to neural networks?

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 2 & 1 & & & \\ 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 10 & 13 & \end{bmatrix}$$

Question: Does anybody see a connection to neural networks?

Hint: What if I rewrite the filter?

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 2 & 1 & & & \\ 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 10 & 13 & \end{bmatrix}$$

Question: Does anybody see a connection to neural networks?

Hint: What if I rewrite the filter?

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \theta_2 & \theta_1 & & & \\ 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 10 & 13 & \end{bmatrix}$$

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \theta_2 & \theta_1 & & & \\ 1 & 2 & 3 & 4 & 5 \\ \theta_2 + 2\theta_1 & 5 & 10 & 13 & \end{bmatrix}$$

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} & \theta_2 & \theta_1 & & \\ 1 & 2 & 3 & 4 & 5 \\ \theta_2 + 2\theta_1 & 2\theta_2 + 3\theta_1 & 10 & 13 & \end{bmatrix}$$

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} & \color{red}{\theta_2} & \color{red}{\theta_1} & & \\ 1 & \color{red}{2} & \color{red}{3} & 4 & 5 \\ \theta_2 + 2\theta_1 & \color{red}{2\theta_2 + 3\theta_1} & 10 & 13 & \end{bmatrix}$$

Just like neural networks, convolution is a linear operation

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} & \theta_2 & \theta_1 & & \\ 1 & 2 & 3 & 4 & 5 \\ \theta_2 + 2\theta_1 & 2\theta_2 + 3\theta_1 & 10 & 13 & \end{bmatrix}$$

Just like neural networks, convolution is a linear operation

It is a weighted sum of the inputs, just like a neuron

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} & {\color{red}\theta_2} & {\color{red}\theta_1} & & \\ 1 & {\color{red}2} & {\color{red}3} & 4 & 5 \\ \theta_2 + 2\theta_1 & {\color{red}2\theta_2 + 3\theta_1} & 10 & 13 & \end{bmatrix}$$

Just like neural networks, convolution is a linear operation

It is a weighted sum of the inputs, just like a neuron

Question: How does convolution differ from a neuron?

Convolution

$$\begin{bmatrix} g(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} & \theta_2 & \theta_1 & & \\ 1 & 2 & 3 & 4 & 5 \\ \theta_2 + 2\theta_1 & 2\theta_2 + 3\theta_1 & 10 & 13 & \end{bmatrix}$$

Just like neural networks, convolution is a linear operation

It is a weighted sum of the inputs, just like a neuron

Question: How does convolution differ from a neuron?

Answer: In a neuron, each input x_i has a different parameter θ_i. In convolution, we reuse (slide) θ_i over $x_1, x_2, ...$

Agenda

1. Review
2. Signal Processing
3. **Convolution**
4. Convolutional Neural Networks
5. Additional Dimensions
6. Coding

Agenda

1. Review
2. Signal Processing
3. Convolution
4. **Convolutional Neural Networks**
5. Additional Dimensions
6. Coding

Convolutional Neural Networks

Convolutional neural networks (CNNs) use convolutional layers

Convolutional Neural Networks

Convolutional neural networks (CNNs) use convolutional layers

Their translation equivariance and locality make them very efficient

Convolutional Neural Networks

Convolutional neural networks (CNNs) use convolutional layers

Their translation equivariance and locality make them very efficient

They also scale to variable-length sequences

Convolutional Neural Networks

Convolutional neural networks (CNNs) use convolutional layers

Their translation equivariance and locality make them very efficient

They also scale to variable-length sequences

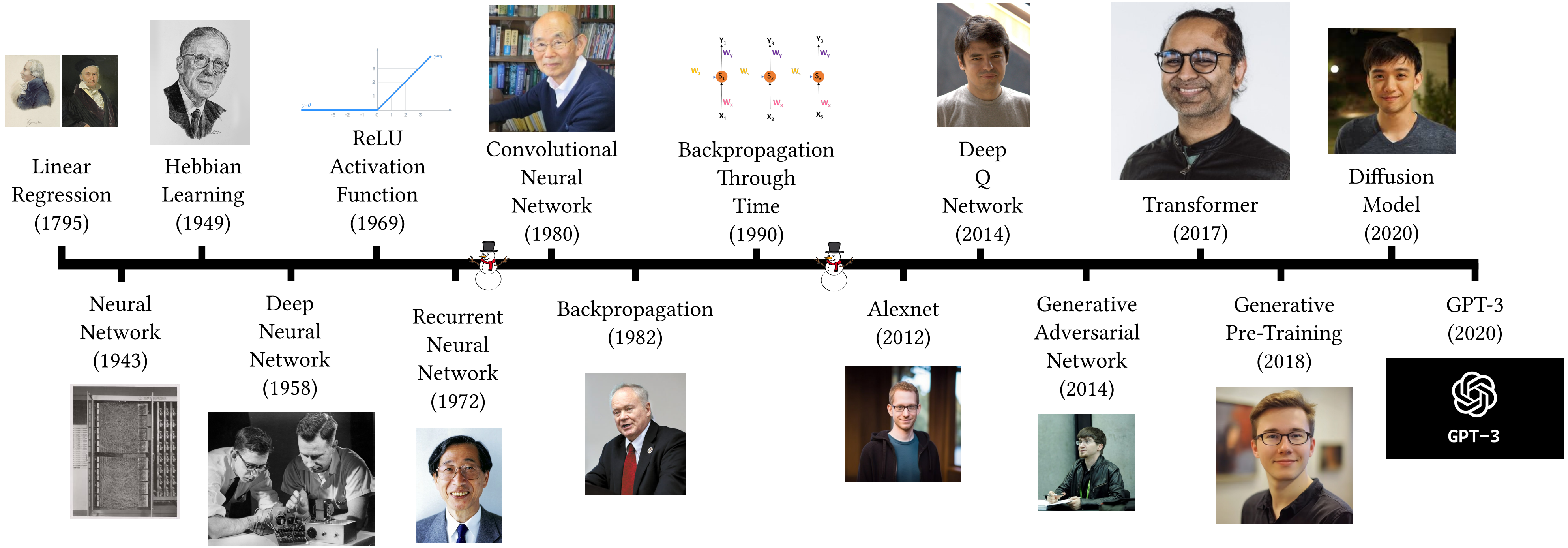
Efficiently expands neural networks to images, videos, sounds, etc

Convolutional Neural Networks

CNNs have been around since the 1970′s

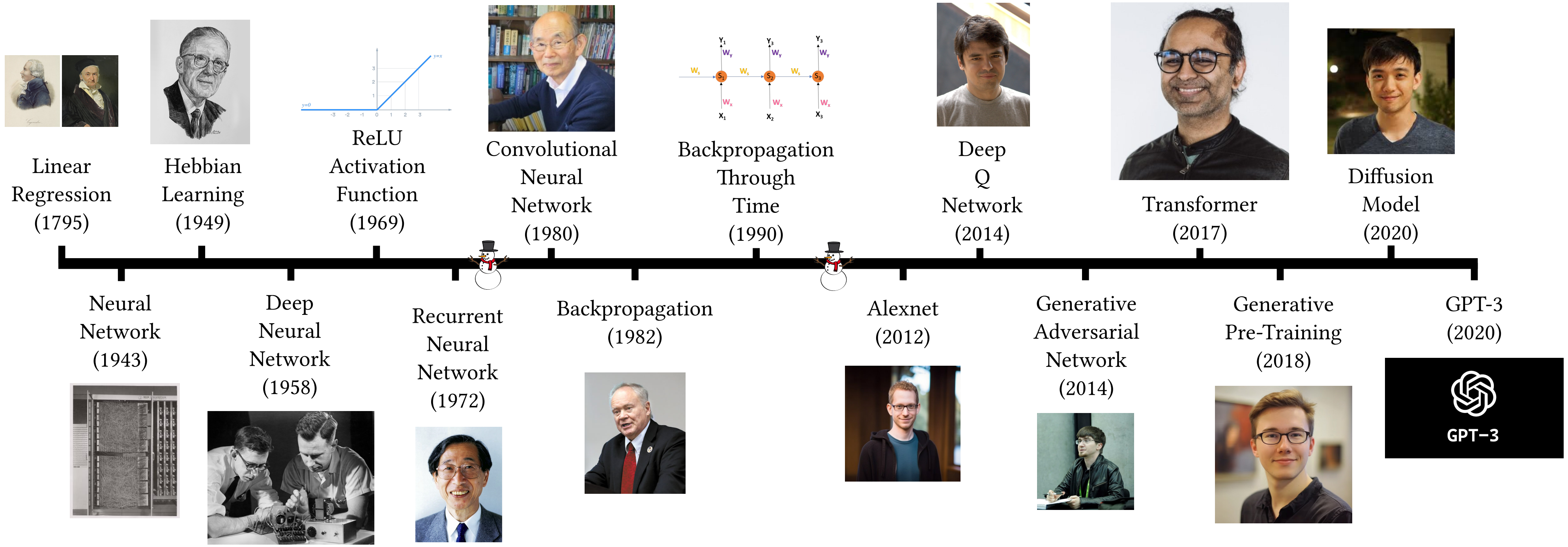
Convolutional Neural Networks

CNNs have been around since the 1970′s



Convolutional Neural Networks

CNNs have been around since the 1970's



2012: GPU and CNN efficiency resulted in breakthroughs

Convolutional Neural Networks

So how does a convolutional neural network work?

Convolutional Neural Networks

So how does a convolutional neural network work?

Like before, we will start with linear functions and derive a convolutional layer

Convolutional Neural Networks

Recall the neuron

Convolutional Neural Networks

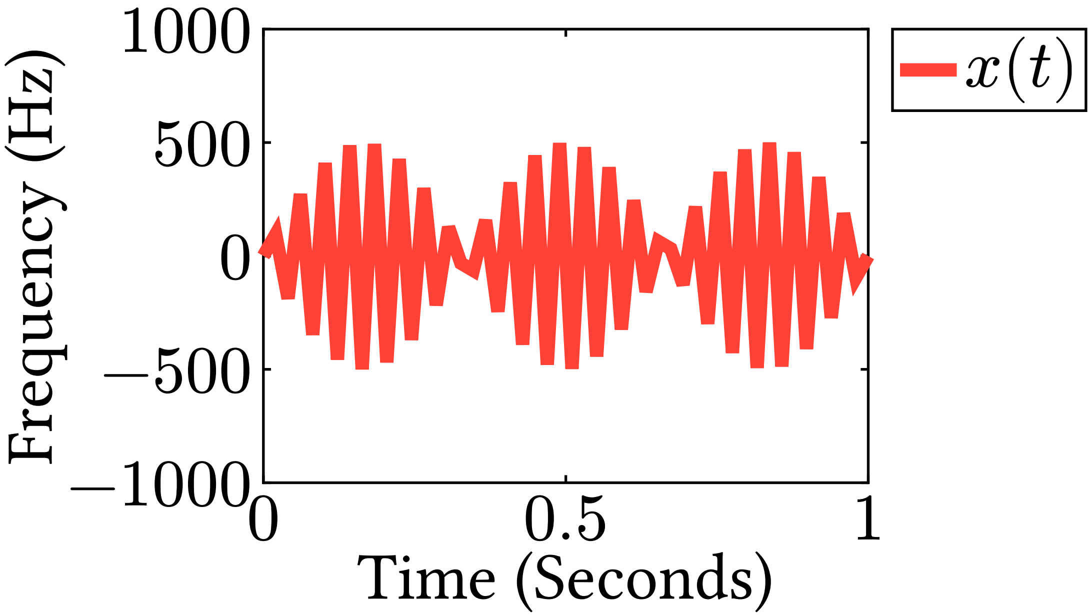
Recall the neuron

Neuron for single x: $$f(x, \boldsymbol{\theta}) = \sigma(\theta_1 x + \theta_0)$$

Convolutional Neural Networks

Recall the neuron

Neuron for single x: $$f(x, \boldsymbol{\theta}) = \sigma(\theta_1 x + \theta_0)$$

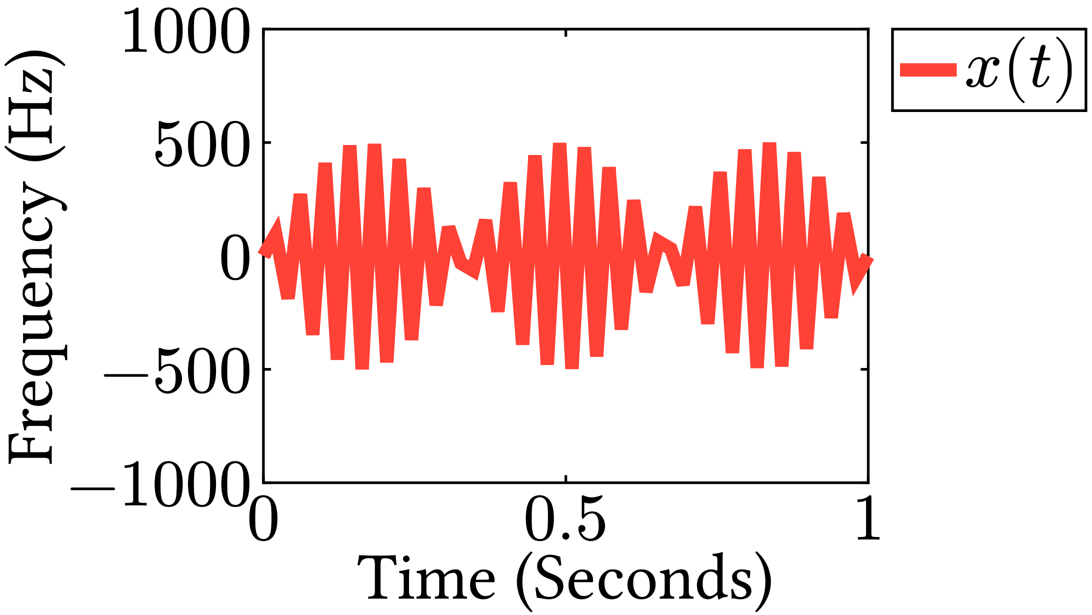


Convolutional Neural Networks

Recall the neuron

Neuron for single x:

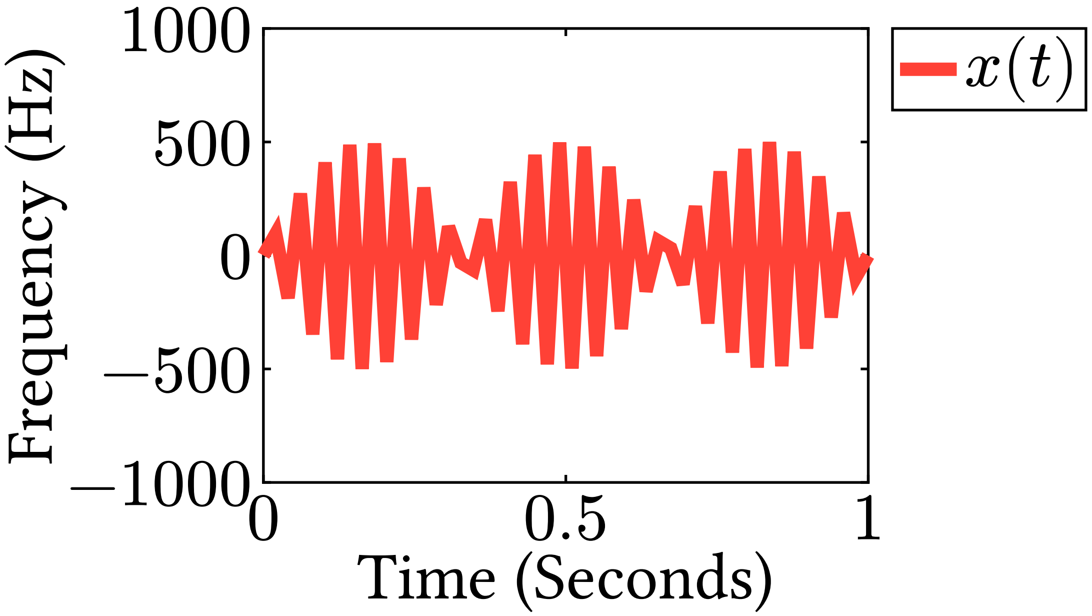
$$f(x, \boldsymbol{\theta}) = \sigma(\theta_1 x + \theta_0)$$



$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \boldsymbol{\theta}\right) =$$

$$\sigma(\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2) + \ldots)$$

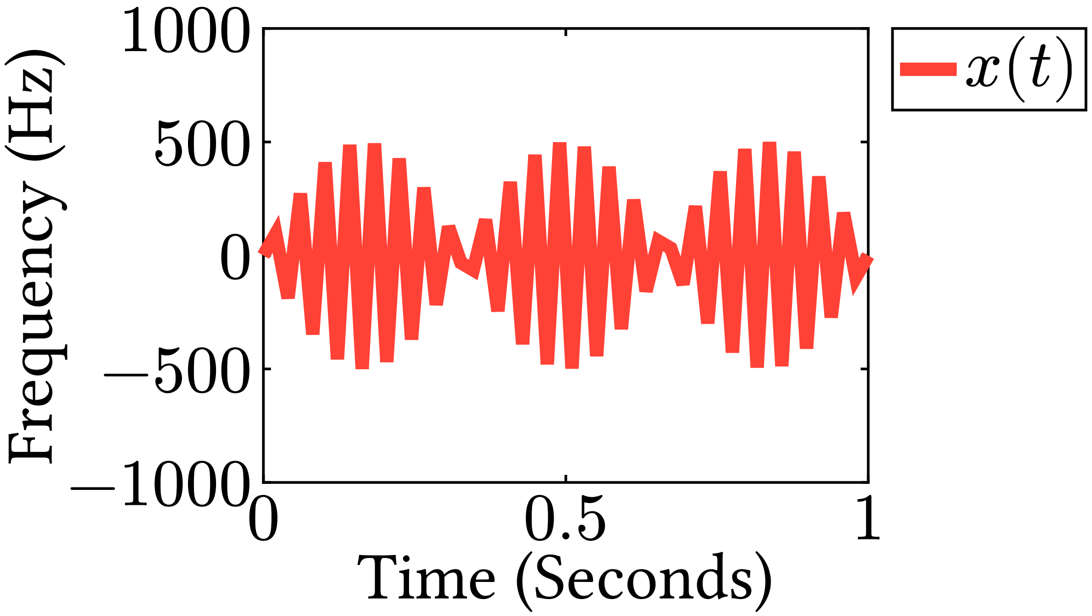
Convolutional Neural Networks



$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \boldsymbol{\theta}\right) =$$

$$\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2) + \ldots$$

Convolutional Neural Networks

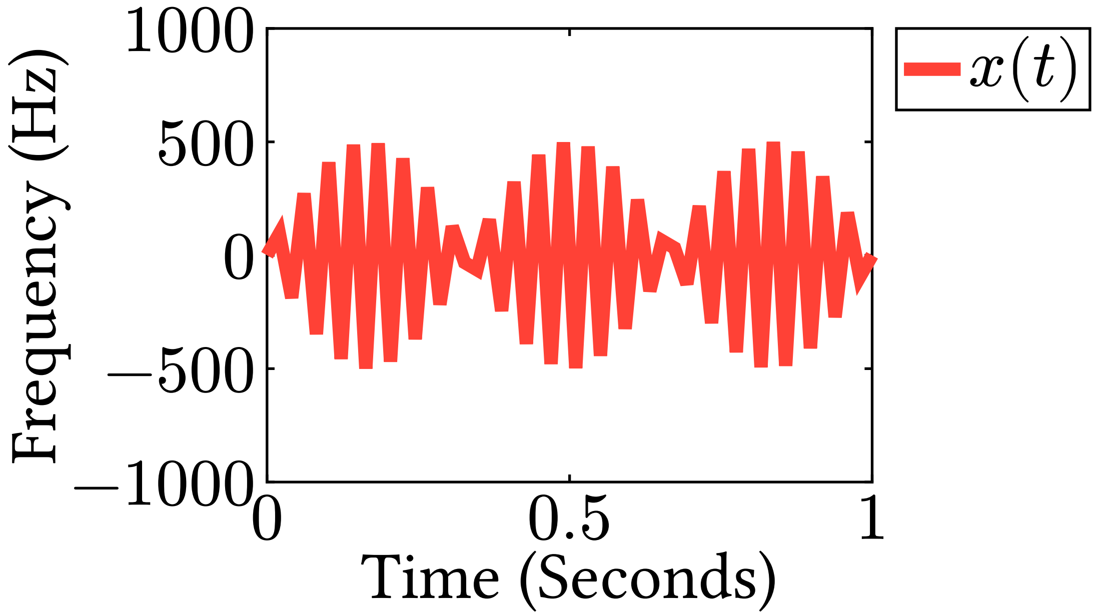


$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \boldsymbol{\theta}\right) =$$

$$\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2) + \ldots$$

Question: Any problems besides locality/equivariance?

Convolutional Neural Networks



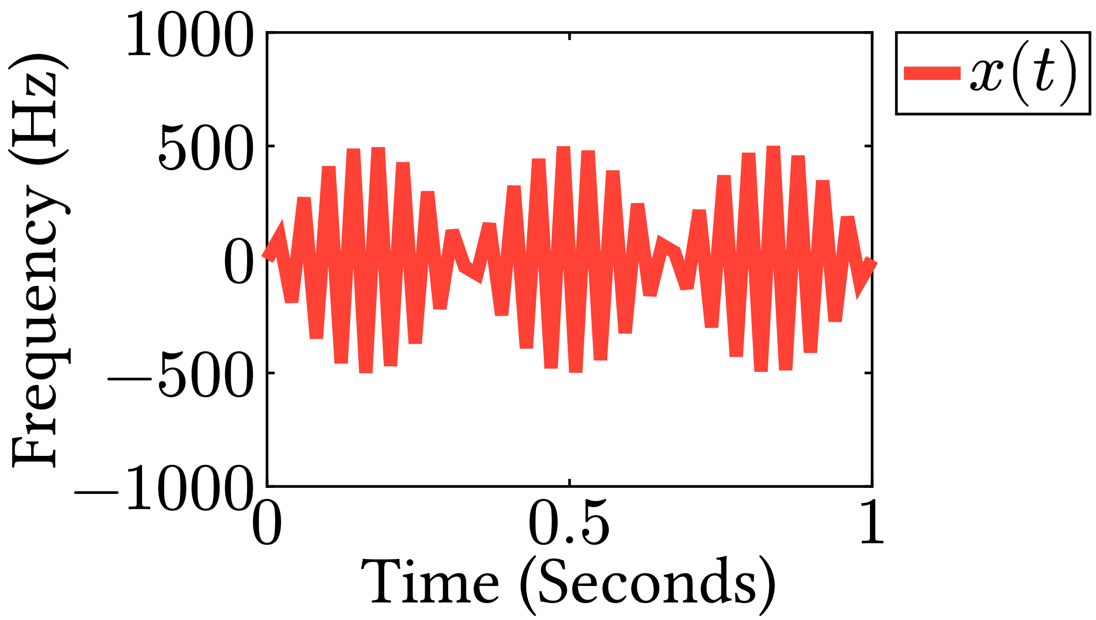
$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \boldsymbol{\theta}\right) =$$

$$\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2) + \ldots$$

Question: Any problems besides locality/equivariance?

Answer 1: Parameters scale with sequence length

Convolutional Neural Networks



$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \boldsymbol{\theta}\right) =$$

$$\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2) + \ldots$$

Question: Any problems besides locality/equivariance?

Answer 1: Parameters scale with sequence length

Answer 2: Parameters only for exactly 1 second waveforms

Convolutional Neural Networks

To fix problems, each timestep cannot use different parameters

Convolutional Neural Networks

To fix problems, each timestep cannot use different parameters

$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \end{bmatrix} \right)$$

$$= \sigma(\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2) + \theta_3 x(0.3) + \theta_4 x(0.4) + \theta_5 x(0.5) + ...)$$

Convolutional Neural Networks

To fix problems, each timestep cannot use different parameters

$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \end{bmatrix} \right)$$

$$= \sigma(\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2) + \theta_3 x(0.3) + \theta_4 x(0.4) + \theta_5 x(0.5) + \ldots)$$

$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \right) = \begin{bmatrix} \sigma(\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2)) \\ \sigma(\theta_0 + \theta_1 x(0.2) + \theta_2 x(0.3)) \\ \sigma(\theta_0 + \theta_1 x(0.3) + \theta_2 x(0.4)) \\ \vdots \end{bmatrix}$$

Convolutional Neural Networks

$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}\right) = \begin{bmatrix} \sigma(\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2)) \\ \sigma(\theta_0 + \theta_1 x(0.2) + \theta_2 x(0.3)) \\ \sigma(\theta_0 + \theta_1 x(0.3) + \theta_2 x(0.4)) \\ \vdots \end{bmatrix}$$

This is a convolutional layer!

Convolutional Neural Networks

$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}\right) = \begin{bmatrix} \sigma(\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2)) \\ \sigma(\theta_0 + \theta_1 x(0.2) + \theta_2 x(0.3)) \\ \sigma(\theta_0 + \theta_1 x(0.3) + \theta_2 x(0.4)) \\ \vdots \end{bmatrix}$$

This is a convolutional layer!

- Local, only considers two inputs at a time

Convolutional Neural Networks

$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \right) = \begin{bmatrix} \sigma(\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2)) \\ \sigma(\theta_0 + \theta_1 x(0.2) + \theta_2 x(0.3)) \\ \sigma(\theta_0 + \theta_1 x(0.3) + \theta_2 x(0.4)) \\ \vdots \end{bmatrix}$$

This is a convolutional layer!

- Local, only considers two inputs at a time
- Translation equivariant, each output corresponds to input

Convolutional Neural Networks

$$f\left(\begin{bmatrix} x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \right) = \begin{bmatrix} \sigma(\theta_0 + \theta_1 x(0.1) + \theta_2 x(0.2)) \\ \sigma(\theta_0 + \theta_1 x(0.2) + \theta_2 x(0.3)) \\ \sigma(\theta_0 + \theta_1 x(0.3) + \theta_2 x(0.4)) \\ \vdots \end{bmatrix}$$

This is a convolutional layer!

- Local, only considers two inputs at a time
- Translation equivariant, each output corresponds to input

Convolutional Neural Networks

We can write both the neuron and convolution in vector form

$$f(x(t), \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix} \right)$$

Convolutional Neural Networks

We can write both the neuron and convolution in vector form

$$f(x(t), \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}\right) \qquad f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) \\ \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) \\ \vdots \end{bmatrix}$$

A convolution layer applies a "mini" perceptron to every few timesteps

Convolutional Neural Networks

$$f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix} \right) \\ \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix} \right) \\ \vdots \end{bmatrix}$$

Convolutional Neural Networks

$$f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) \\ \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) \\ \vdots \end{bmatrix}$$

Question: What is the shape of the results?

Convolutional Neural Networks

$$f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) \\ \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) \\ \vdots \end{bmatrix}$$

Question: What is the shape of the results?

Answer 1: Depends on sampling rate and filter size!

Convolutional Neural Networks

$$f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) \\ \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) \\ \vdots \end{bmatrix}$$

Question: What is the shape of the results?

Answer 1: Depends on sampling rate and filter size!

Answer 2: $T - k$, where T is sequence length and k filter length

Convolutional Neural Networks

If we want a single output, we should **pool**

Convolutional Neural Networks

If we want a single output, we should **pool**

$$z(t) = f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix} \right) & \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix} \right) & \dots \end{bmatrix}^\top$$

Convolutional Neural Networks

If we want a single output, we should **pool**

$$z(t) = f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix} \right) & \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix} \right) & \dots \end{bmatrix}^\top$$

$$\text{SumPool}(z(t)) = \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix} \right) + \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix} \right) + \dots$$

Convolutional Neural Networks

If we want a single output, we should **pool**

$$z(t) = f(x(t), \boldsymbol{\theta}) = \left[\sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix} \right) \ \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix} \right) \ \ldots \right]^\top$$

$$\mathrm{SumPool}(z(t)) = \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix} \right) + \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix} \right) + \ldots$$

$$\mathrm{MeanPool}(z(t)) = \frac{1}{T} \, \mathrm{SumPool}(z(t))$$

Convolutional Neural Networks

Question: $x(t)$ is a function, what is the function signature?

Convolutional Neural Networks

Question: $x(t)$ is a function, what is the function signature?

Answer:

$$x : \mathbb{R}_+ \mapsto \mathbb{R}$$

Convolutional Neural Networks

Question: $x(t)$ is a function, what is the function signature?

Answer:

$$x : \mathbb{R}_+ \mapsto \mathbb{R}$$

So far, we have considered:

Convolutional Neural Networks

Question: $x(t)$ is a function, what is the function signature?

Answer:

$$x : \mathbb{R}_+ \mapsto \mathbb{R}$$

So far, we have considered:

- 1 dimensional variable t

Convolutional Neural Networks

Question: $x(t)$ is a function, what is the function signature?

Answer:

$$x : \mathbb{R}_+ \mapsto \mathbb{R}$$

So far, we have considered:

- 1 dimensional variable t
- 1 dimensional input/channel $x(t)$

Convolutional Neural Networks

Question: $x(t)$ is a function, what is the function signature?

Answer:

$$x : \mathbb{R}_+ \mapsto \mathbb{R}$$

So far, we have considered:

- 1 dimensional variable t
- 1 dimensional input/channel $x(t)$
- 1 filter

Convolutional Neural Networks

Question: $x(t)$ is a function, what is the function signature?

Answer:

$$x : \mathbb{R}_+ \mapsto \mathbb{R}$$

So far, we have considered:

- 1 dimensional variable t
- 1 dimensional input/channel $x(t)$
- 1 filter

We must consider a more general case

Convolutional Neural Networks

Question: $x(t)$ is a function, what is the function signature?

Answer:

$$x : \mathbb{R}_+ \mapsto \mathbb{R}$$

So far, we have considered:

- 1 dimensional variable t
- 1 dimensional input/channel $x(t)$
- 1 filter

We must consider a more general case

Things will get more complicated, but the core idea is exactly the same

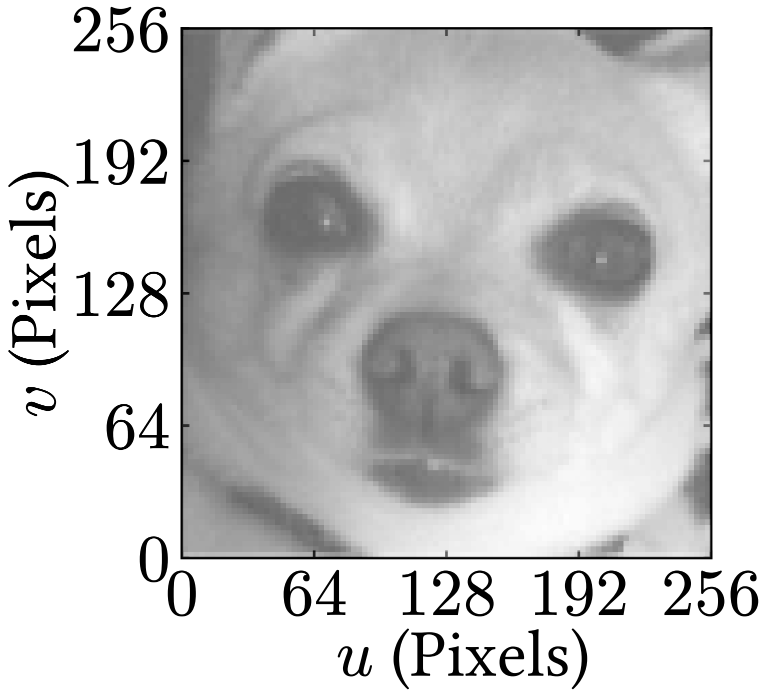
Agenda

1. Review
2. Signal Processing
3. Convolution
4. **Convolutional Neural Networks**
5. Additional Dimensions
6. Coding

Agenda

1. Review
2. Signal Processing
3. Convolution
4. Convolutional Neural Networks
5. **Additional Dimensions**
6. Coding

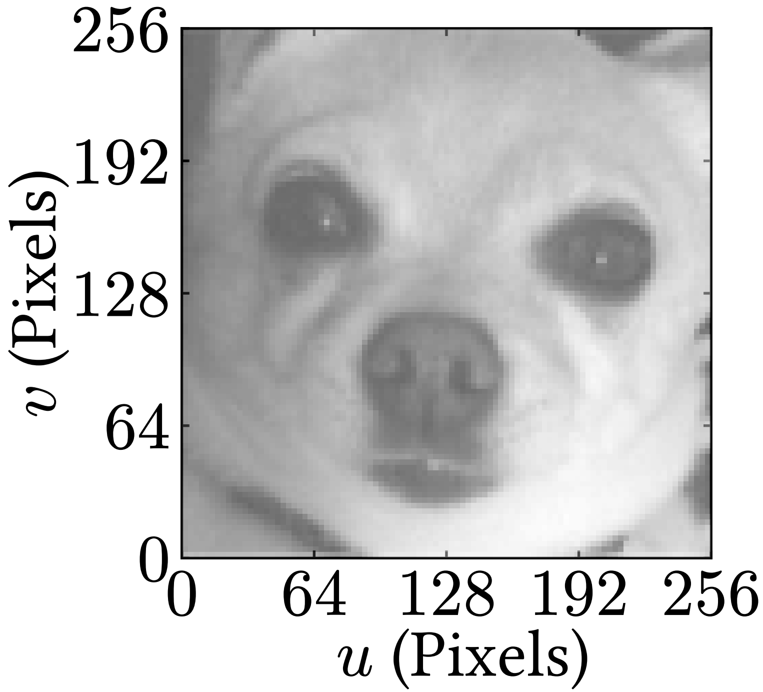
Additional Dimensions



$$\boxed{x(u, v)}$$

Question: How many input dimensions for x?

Additional Dimensions



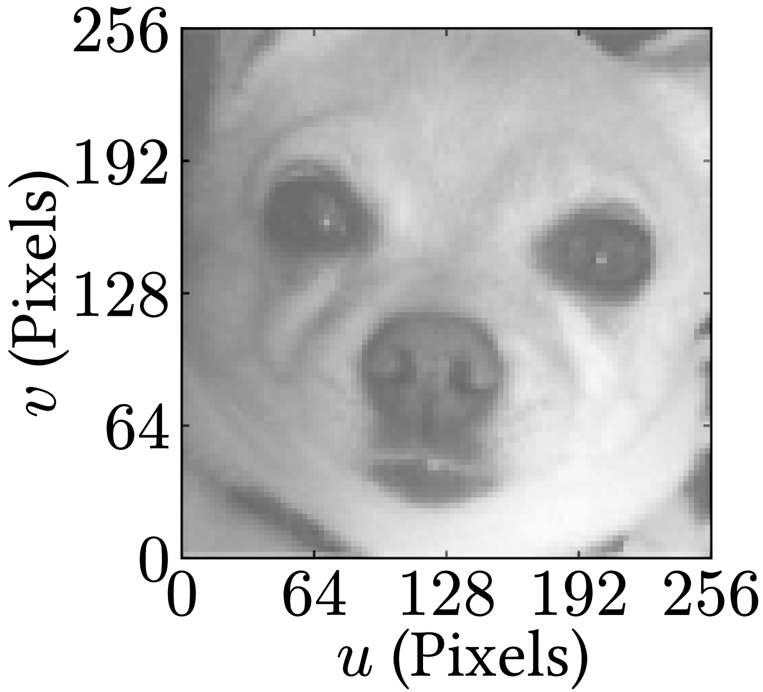
$$\boxed{x(u, v)}$$

Question: How many input dimensions for x?

Answer: 2, u, v

Question: How many output dimensions for x?

Additional Dimensions



$\boxed{x(u, v)}$

Question: How many input dimensions for x?

Answer: 2, u, v

Question: How many output dimensions for x?

Answer: 1, black/white value

$$x : \underbrace{\mathbb{Z}_{0,255}}_{\text{width}} \times \underbrace{\mathbb{Z}_{0,255}}_{\text{height}} \mapsto \underbrace{\mathbb{Z}_{0,255}}_{\text{Color values}}$$

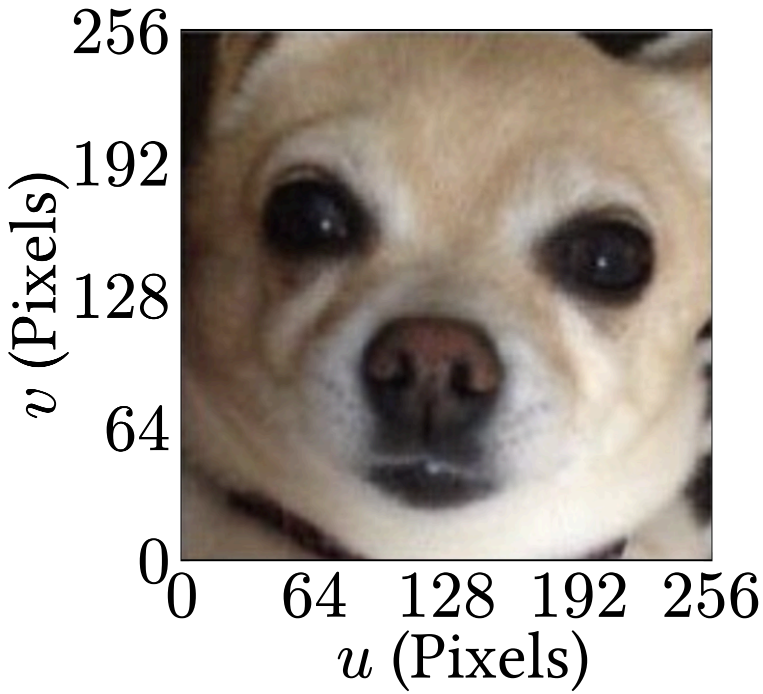
Additional Dimensions

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$*$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

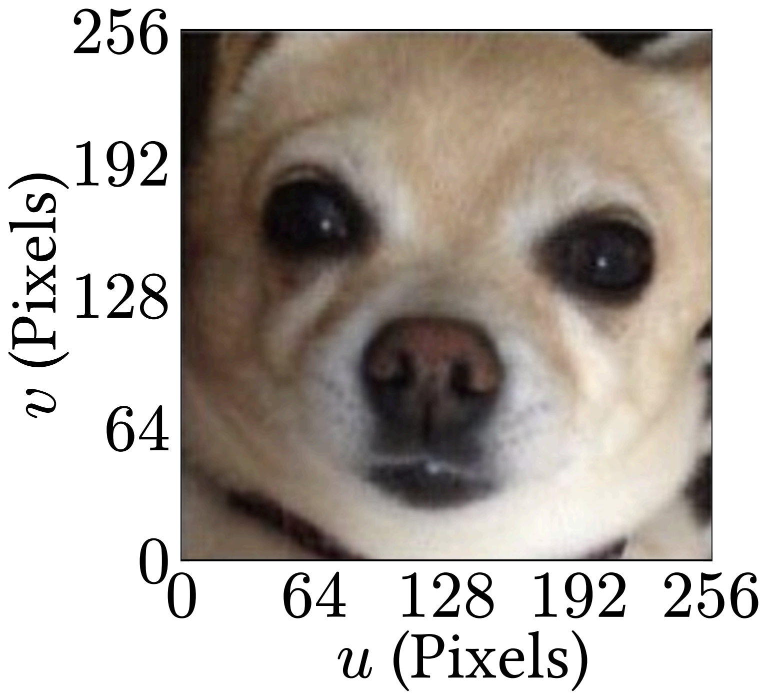
Additional Dimensions



$x(u, v)$

Question: How many input dimensions for x?

Additional Dimensions



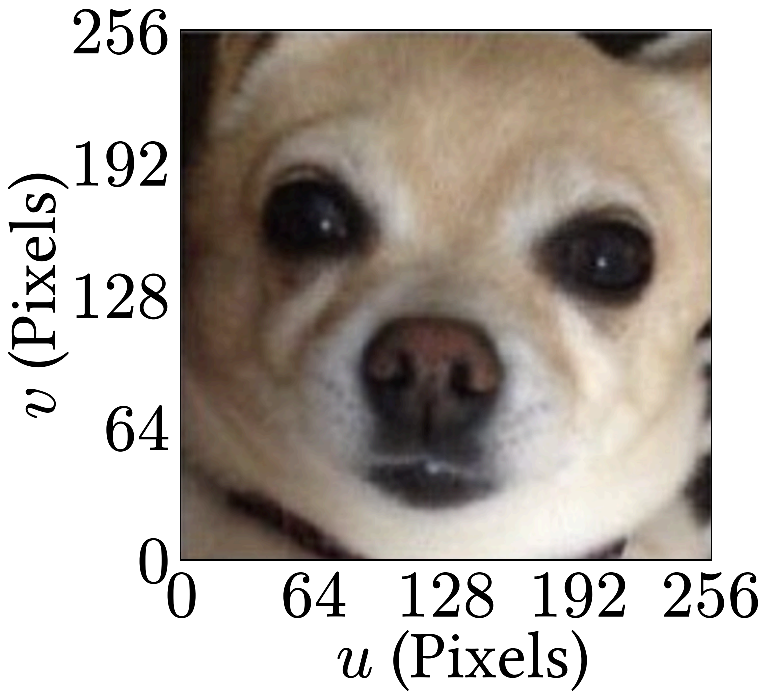
$x(u, v)$

Question: How many input dimensions for x?

Answer: 2, u, v

Question: How many output dimensions for x?

Additional Dimensions



$\boxed{x(u, v)}$

Question: How many input dimensions for x?

Answer: 2, u, v

Question: How many output dimensions for x?

Answer: 3 – red, green, and blue channels

$$x : \underbrace{\mathbb{Z}_{0,255}}_{\text{width}} \times \underbrace{\mathbb{Z}_{0,255}}_{\text{height}} \mapsto \underbrace{[0, 1]^3}_{\text{Color values}}$$

Additional Dimensions

Additional Dimensions

Computers represent 3 color channels each with 256 integer values

Additional Dimensions

Computers represent 3 color channels each with 256 integer values

But we usually convert the colors to be in $[0, 1]$ for scale reasons

Additional Dimensions

Computers represent 3 color channels each with 256 integer values

But we usually convert the colors to be in $[0, 1]$ for scale reasons

$$\begin{bmatrix} \frac{R}{255} & \frac{G}{255} & \frac{B}{255} \end{bmatrix}$$

Additional Dimensions

Each pixel contains 3 colors (channels)

Additional Dimensions

Each pixel contains 3 colors (channels)

And the pixels extend in 2 directions (variable)

Additional Dimensions

Each pixel contains 3 colors (channels)

And the pixels extend in 2 directions (variable)

$$\boldsymbol{x}(u, v) = \begin{bmatrix} \underbrace{\begin{bmatrix} 130 & 140 & 120 & 103 \\ 80 & 140 & 120 & 105 \\ 130 & 140 & 75 & 165 \\ 210 & 140 & 90 & 150 \end{bmatrix}}_{\text{red}} & \underbrace{\begin{bmatrix} 130 & 140 & 75 & 165 \\ 210 & 140 & 90 & 150 \\ 130 & 140 & 120 & 103 \\ 80 & 140 & 120 & 105 \end{bmatrix}}_{\text{green}} & \underbrace{\begin{bmatrix} 210 & 140 & 90 & 150 \\ 130 & 140 & 75 & 165 \\ 110 & 140 & 120 & 103 \\ 80 & 140 & 120 & 105 \end{bmatrix}}_{\text{blue}} \end{bmatrix}^\top$$

Additional Dimensions

Each pixel contains 3 colors (channels)

And the pixels extend in 2 directions (variable)

$$\boldsymbol{x}(u, v) = \begin{bmatrix} \underbrace{\begin{bmatrix} 130 & 140 & 120 & 103 \\ 80 & 140 & 120 & 105 \\ 130 & 140 & 75 & 165 \\ 210 & 140 & 90 & 150 \end{bmatrix}}_{\text{red}} & \underbrace{\begin{bmatrix} 130 & 140 & 75 & 165 \\ 210 & 140 & 90 & 150 \\ 130 & 140 & 120 & 103 \\ 80 & 140 & 120 & 105 \end{bmatrix}}_{\text{green}} & \underbrace{\begin{bmatrix} 210 & 140 & 90 & 150 \\ 130 & 140 & 75 & 165 \\ 110 & 140 & 120 & 103 \\ 80 & 140 & 120 & 105 \end{bmatrix}}_{\text{blue}} \end{bmatrix}^\top$$

This form is called CHW (channel, height, width) format

Convolutional filter must process this data!

Additional Dimensions

$$
\begin{bmatrix}
0&1&1&1&1&1&0&1\\
0&1&1&1&1&1&1&1\\
1&0&0&1&1&0&0&1\\
1&0&0&1&1&0&0&1\\
1&1&0&0&0&1&1&1\\
1&1&0&0&0&1&1&1\\
0&1&0&0&0&1&1&0\\
1&0&1&1&1&1&1&1
\end{bmatrix}
*
\begin{bmatrix}1&0\\0&1\end{bmatrix}
+
\begin{bmatrix}
0&1&1&1&1&1&0&1\\
0&1&1&1&1&1&1&1\\
1&0&0&1&1&0&0&1\\
1&0&0&1&1&0&0&1\\
1&1&0&0&0&1&1&1\\
1&1&0&0&0&1&1&1\\
0&1&0&0&0&1&1&0\\
1&0&1&1&1&1&1&1
\end{bmatrix}
*
\begin{bmatrix}1&1\\0&1\end{bmatrix}
+
\begin{bmatrix}
0&1&1&1&1&1&0&1\\
0&1&1&1&1&1&1&1\\
1&0&0&1&1&0&0&1\\
1&0&0&1&1&0&0&1\\
1&1&0&0&0&1&1&1\\
1&1&0&0&0&1&1&1\\
0&1&0&0&0&1&1&0\\
1&0&1&1&1&1&1&1
\end{bmatrix}
*
\begin{bmatrix}1&0\\0&0\end{bmatrix}
$$

Additional Dimensions

I will not bore you with the full equations

Additional Dimensions

I will not bore you with the full equations

Question: What is the shape of $\boldsymbol{\theta}$ for a single layer?

Additional Dimensions

I will not bore you with the full equations

Question: What is the shape of $\boldsymbol{\theta}$ for a single layer?

Answer:

$$\boldsymbol{\theta} \in \mathbb{R}^{c_x \times c_y \times (k+1) \times k}$$

- Input channels: c_x
- Output channels: c_y
- Filter u (height): $k + 1$
- Filter v (width): k

Additional Dimensions

```
import torch
c_x = 3 # Number of colors
c_y = 32
k = 2 # Filter size
h, w = 128, 128 # Image size

conv1 = torch.nn.Conv2d(
  in_channels=c_x,
  out_channels=c_y,
  kernel_size=2
)
image = torch.rand((1, c_x, h, w)) # Torch requires BCHW
out = conv1(image) # Shape(1, c_y, h - k, w - k)
```

Additional Dimensions

One last thing, stride allows you to "skip" cells during convolution

Additional Dimensions

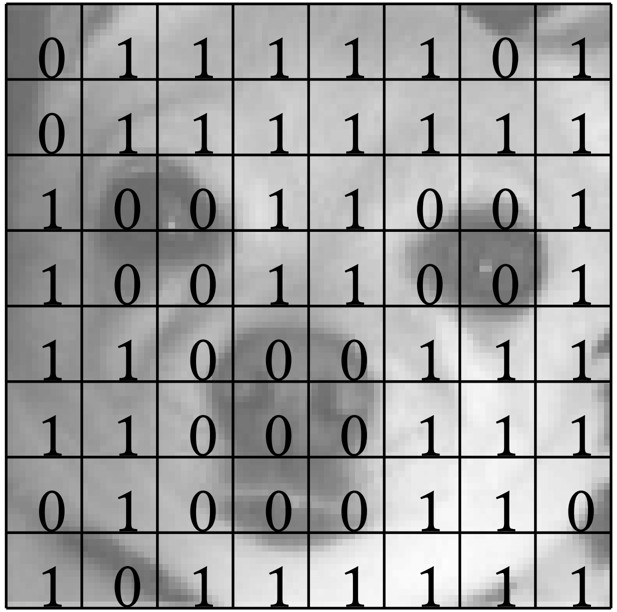
One last thing, stride allows you to "skip" cells during convolution

This can decrease the size of image without pooling

Additional Dimensions

One last thing, stride allows you to "skip" cells during convolution

This can decrease the size of image without pooling



$$*$$

1	0
0	1

Agenda

1. Review
2. Signal Processing
3. Convolution
4. Convolutional Neural Networks
5. **Additional Dimensions**
6. Coding

Agenda

1. Review
2. Signal Processing
3. Convolution
4. Convolutional Neural Networks
5. Additional Dimensions
6. **Coding**

Coding

```
import jax, equinox
c_x = 3 # Number of colors
c_y = 32
k = 2 # Filter size
h, w = 128, 128 # Image size
conv1 = equinox.nn.Conv2d(
  in_channels=c_x,
  out_channels=c_y,
  kernel_size=2,
  key=jax.random.key(0)
)
image = jax.random.uniform(jax.random.key(1), (c_x, h, w))
out = conv1(image) # Shape(c_y, h - k, w - k)
```

Coding

```
import torch
conv1 = torch.nn.Conv2d(3, c_h, 2)
pool1 = torch.nn.AdaptivePool2d((a, a))
conv2 = torch.nn.Conv2d(c_h, c_y, 2)
pool2 = torch.nn.AdaptivePool2d((b, b))
linear = torch.nn.Linear(c_y * b * b)
z_1 = conv1(image)
z_1 = torch.nn.functional.leaky_relu(z_1)
z_1 = pool(z_1) # Shape(1, c_h, a, a)
z_2 = conv1(z1)
z_2 = torch.nn.functional.leaky_relu(z_2)
z_2 = pool(z_2) # Shape(1, c_y, b, b)
z_3 = linear(z_2.flatten())
```

Coding

```
import jax, equinox
conv1 = equinox.nn.Conv2d(3, c_h, 2)
pool1 = equinox.nn.AdaptivePool2d((a, a))
conv2 = equinox.nn.Conv2d(c_h, c_y, 2)
pool2 = equinox.nn.AdaptivePool2d((b, b))
linear = equinox.nn.Linear(c_y * b * b)
z_1 = conv1(image,(3, h, w))
z_1 = jax.nn.leaky_relu(z_1)
z_1 = pool(z_1) # Shape(c_h, a, a)
z_2 = conv1(z1)
z_2 = jax.nn.leaky_relu(z_2)
z_2 = pool(z_2) # Shape(c_y, b, b)
z_3 = linear(z_2.flatten())
```

Coding

Single channel, single filter, single variable, $\boldsymbol{\theta} \in \mathbb{R}^{k+1}, k = 2$

Coding

Single channel, single filter, single variable, $\boldsymbol{\theta} \in \mathbb{R}^{k+1}, k = 2$

$$f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(1) \\ x(2) \end{bmatrix}\right) & \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(2) \\ x(3) \end{bmatrix}\right) & \dots \end{bmatrix}^\top$$

Single channel, single filter, **two variables**, $\boldsymbol{\theta} \in \mathbb{R}^{2k+1}, k = 2$

Coding

Single channel, single filter, single variable, $\boldsymbol{\theta} \in \mathbb{R}^{k+1}, k = 2$

$$f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(1) \\ x(2) \end{bmatrix}\right) & \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(2) \\ x(3) \end{bmatrix}\right) & \ldots \end{bmatrix}^\top$$

Single channel, single filter, **two variables**, $\boldsymbol{\theta} \in \mathbb{R}^{2k+1}, k = 2$

$$f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 & 0 & 0 \\ x(1,1) & x(1,2) & x(1,3) \\ x(2,1) & x(2,2) & x(2,3) \end{bmatrix}\right) & \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 & 0 & 0 \\ x(2,1) & x(2,2) & x(2,3) \\ x(3,1) & x(3,2) & x(3,3) \end{bmatrix}\right) \end{bmatrix}^\top$$

Coding

Three channels, single filter, two variables, $\boldsymbol{\theta} \in \mathbb{R}^{2k+1}, k = 2$

$$f_r(x(t), \boldsymbol{\theta}) = \left[\sigma\left(\boldsymbol{\theta}_r^\top \begin{bmatrix} 1 & 0 & 0 \\ x(1,1) & x(1,2) & x(1,3) \\ x(2,1) & x(2,2) & x(2,3) \end{bmatrix} \right) \; \sigma\left(\boldsymbol{\theta}_r^\top \begin{bmatrix} 1 & 0 & 0 \\ x(2,1) & x(2,2) & x(2,3) \\ x(3,1) & x(3,2) & x(3,3) \end{bmatrix} \right) \right]^\top$$

Coding

Three channels, single filter, two variables, $\boldsymbol{\theta} \in \mathbb{R}^{2k+1}, k = 2$

$$f_r(x(t), \boldsymbol{\theta}) = \left[\sigma\left(\boldsymbol{\theta}_r^\top \begin{bmatrix} 1 & 0 & 0 \\ x(1,1) & x(1,2) & x(1,3) \\ x(2,1) & x(2,2) & x(2,3) \end{bmatrix} \right) \sigma\left(\boldsymbol{\theta}_r^\top \begin{bmatrix} 1 & 0 & 0 \\ x(2,1) & x(2,2) & x(2,3) \\ x(3,1) & x(3,2) & x(3,3) \end{bmatrix} \right) \right]^\top$$

Coding

We only considered one filter

Coding

We only considered one filter

In a perceptron, many parallel neurons makes a wide neural network

Coding

We only considered one filter

In a perceptron, many parallel neurons makes a wide neural network

In a CNN, many parallel filters makes a wide CNN

Coding

We only considered one filter

In a perceptron, many parallel neurons makes a wide neural network

In a CNN, many parallel filters makes a wide CNN

$$\boldsymbol{z}(t) = f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) & \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) & \dots \end{bmatrix}^\top$$

Coding

We only considered one filter

In a perceptron, many parallel neurons makes a wide neural network

In a CNN, many parallel filters makes a wide CNN

$$\boldsymbol{z}(t) = f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) & \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) & \dots \end{bmatrix}^\top$$

Convolution

Neuron:

$$\sigma(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2) + \ldots) = \sigma\left(\sum_{i=0}^{d_x} \theta_{1,i} \overline{x}_i + \sum_{i=0}^{d_x} \theta_{2,i} \overline{x}_i + \ldots\right)$$

Convolution

Convolution

Neuron:

$$\sigma(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2) + \ldots) = \sigma\left(\sum_{i=0}^{d_x} \theta_{1,i} \overline{x}_i + \sum_{i=0}^{d_x} \theta_{2,i} \overline{x}_i + \ldots\right)$$

Convolution:

$$\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(t) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(t+1) = \left(\sum_{i=0}^{d_x} \theta_{1,i} \overline{x}_i(t)\right) + \left(\sum_{i=0}^{d_x} \theta_{2,i} \overline{x}_i(t+1)\right)$$

Convolution

Neuron:

$$\sigma(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2) + \ldots) = \sigma\left(\sum_{i=0}^{d_x} \theta_{1,i} \overline{x}_i + \sum_{i=0}^{d_x} \theta_{2,i} \overline{x}_i + \ldots \right)$$

Convolution:

$$\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(t) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(t+1) = \left(\sum_{i=0}^{d_x} \theta_{1,i} \overline{x}_i(t) \right) + \left(\sum_{i=0}^{d_x} \theta_{2,i} \overline{x}_i(t+1) \right)$$

Convolution

$$\begin{bmatrix} \boldsymbol{\theta_1} \\ \boldsymbol{\theta_2} \end{bmatrix} * \overline{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(0) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(1) & \theta_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2) & \dots \end{bmatrix}$$

Convolution

$$\begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{bmatrix} * \overline{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(0) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(1) & \theta_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2) & \dots \end{bmatrix}$$

Convolution

$$\begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{bmatrix} * \overline{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(0) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(1) & \theta_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2) & \ldots \end{bmatrix}$$

We call this a **convolutional layer**

Convolution

$$\begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{bmatrix} * \overline{\boldsymbol{x}}(t) = [\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(0) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(1) \quad \theta_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2) \quad ...]$$

We call this a **convolutional layer**

Question: Anything missing?

Convolution

$$\begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{bmatrix} * \overline{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(0) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(1) & \theta_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2) & \dots \end{bmatrix}$$

We call this a **convolutional layer**

Question: Anything missing?

Answer: Activation function!

Convolution

$$\begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{bmatrix} * \overline{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(0) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(1) & \theta_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2) & \dots \end{bmatrix}$$

We call this a **convolutional layer**

Question: Anything missing?

Answer: Activation function!

$$\begin{bmatrix} \sigma(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(0) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(1)) & \sigma(\theta_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2)) & \dots \end{bmatrix}$$

Convolution

$$\begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{bmatrix} * \overline{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(0) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(1) & \theta_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2) & \ldots \end{bmatrix}$$

We call this a **convolutional layer**

Question: Anything missing?

Answer: Activation function!

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}(0) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(1)\right) & \sigma\left(\theta_1^\top \overline{\boldsymbol{x}}(1) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}(2)\right) & \ldots \end{bmatrix}$$

Much better

Convolution

Convolution is **local**, in this example, we only consider two consecutive timesteps

Convolution

Convolution is **local**, in this example, we only consider two consecutive timesteps

Convolution is **shift equivariant**, if $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ detect "hello", it does not matter whether "hello" occurs at $x(0), x(1)$ or $x(100), x(101)$

Convolution

Convolution

```python
import jax, equinox
# Assume a sequence of length m
# Each timestep has dimension d_x
x = stock_data # Shape (d_x, time)
conv_layer = equinox.nn.Conv1d(
  in_channels=d_x,
  out_channels=d_y,
  kernel_size=k # Size of filter in timesteps/parameters,
  key=jax.random.key(0)
)

z = jax.nn.leaky_relu(conv_layer(x))
```

Convolution

```
import torch
# Assume a sequence of length m
# Each timestep has dimension d_x
# Torch requires 3 dims! Be careful!
x = stock_data # Shape (batch, d_x, time)
conv_layer = torch.nn.Conv1d(
  in_channels=d_x,
  out_channels=d_y,
  kernel_size=k # Size of filter in timesteps/parameters,
)

z = jax.nn.leaky_relu(conv_layer(x))
```

Agenda

1. Review
2. Signal Processing
3. **Convolution**
4. Convolutional Neural Networks
5. Additional Dimensions
6. Coding

Agenda

1. Review
2. Signal Processing
3. Convolution
4. **Convolutional Neural Networks**
5. Additional Dimensions
6. Coding

2D Convolution

We defined convolution over one variable t

2D Convolution

We defined convolution over one variable t

For images, we often have two variables denoting width and height u, v

$$x(u, v)$$

2D Convolution

We defined convolution over one variable t

For images, we often have two variables denoting width and height u, v

$$x(u, v)$$

We can also do convolutions over two dimensions

2D Convolution

We defined convolution over one variable t

For images, we often have two variables denoting width and height u, v

$$x(u, v)$$

We can also do convolutions over two dimensions

Most image-based neural networks use convolutions