Convolution

CISC 7026: Introduction to Deep Learning

University of Macau

Agenda

1. Review
2. Signal Processing
3. Convolution
4. 2D Convolution
5. Downsampling
6. Coding

Agenda

1. **Review**
2. Signal Processing
3. Convolution
4. 2D Convolution
5. Downsampling
6. Coding

Review

Agenda

1. **Review**
2. Signal Processing
3. Convolution
4. 2D Convolution
5. Downsampling
6. Coding

Agenda

1. Review
2. **Signal Processing**
3. Convolution
4. 2D Convolution
5. Downsampling
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

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

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

However, there is structure inherent in the real world

By representing this structure within neural networks, we can make neural networks that are more efficient and 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

However, there is structure inherent in the real world

By representing this structure within neural networks, we can make neural networks that are more efficient and 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

$$x(t) = \dots$$

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

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

$$x(t) = \ldots$$

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

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

Knowing the meaning of signals is very useful

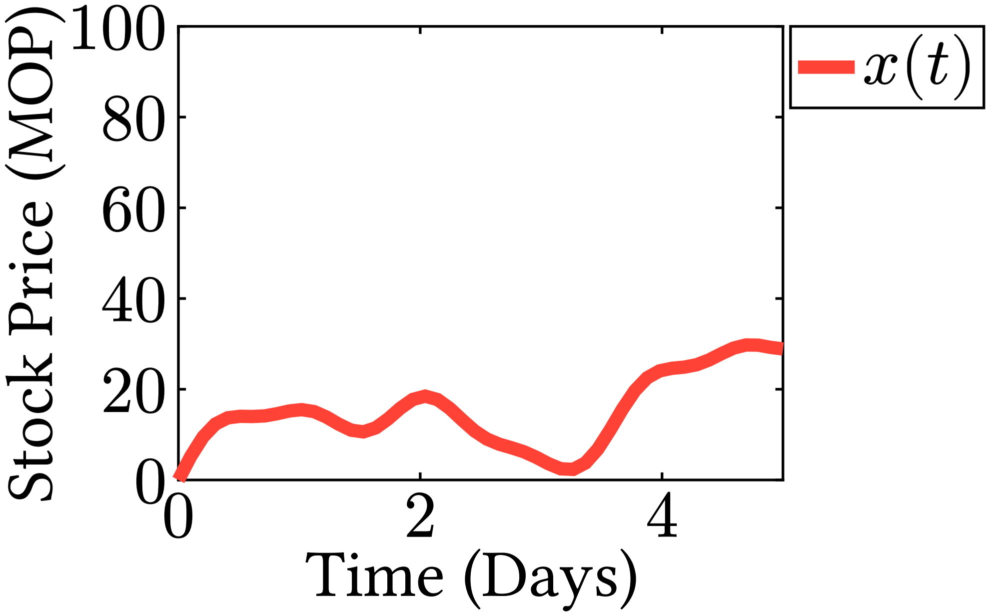
Signal Processing

Signal Processing

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

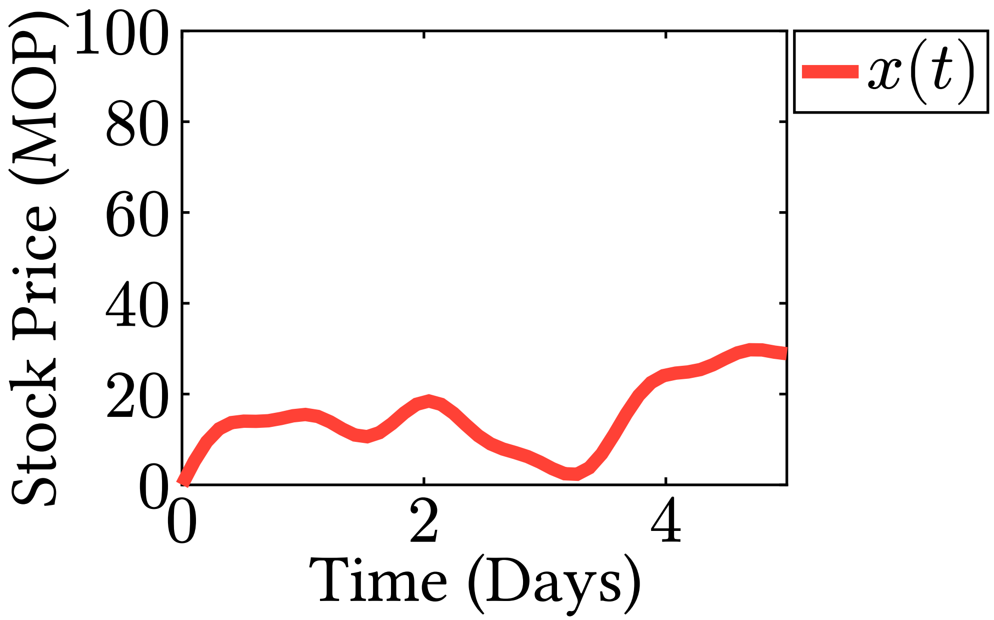
Signal Processing

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



Signal Processing

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



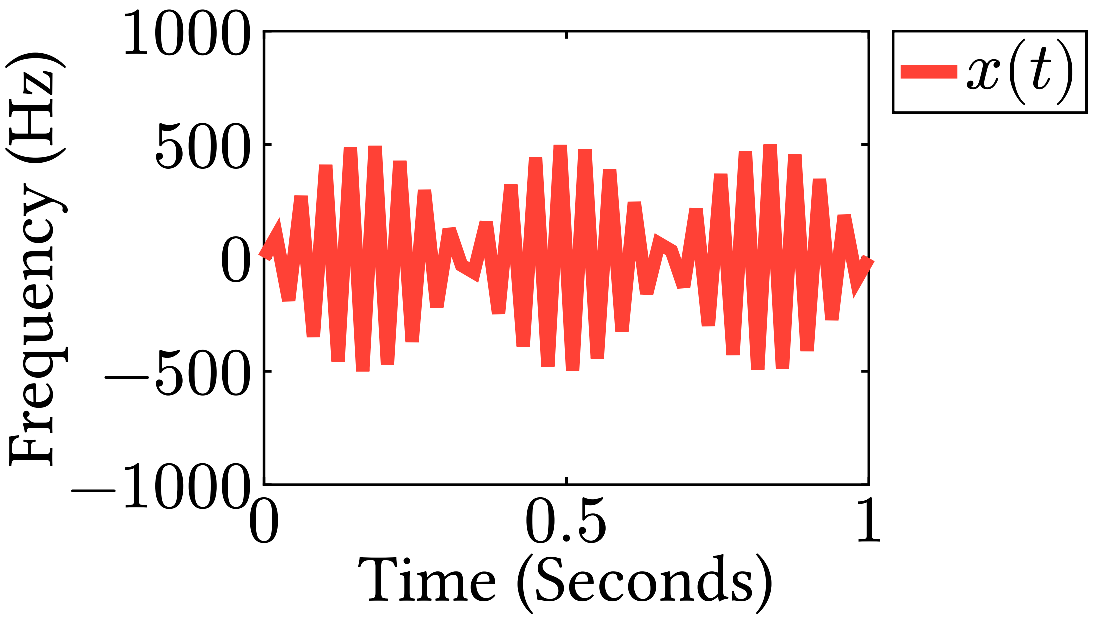
Structure: Tomorrow's stock price will be close to today's stock price

Signal Processing

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

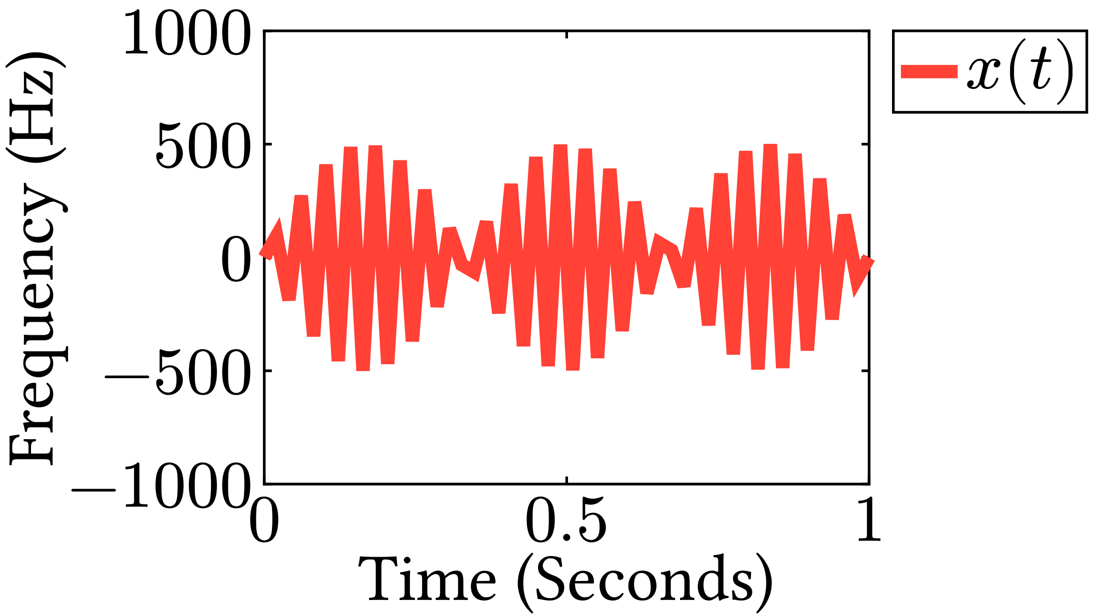
Signal Processing

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



Signal Processing

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



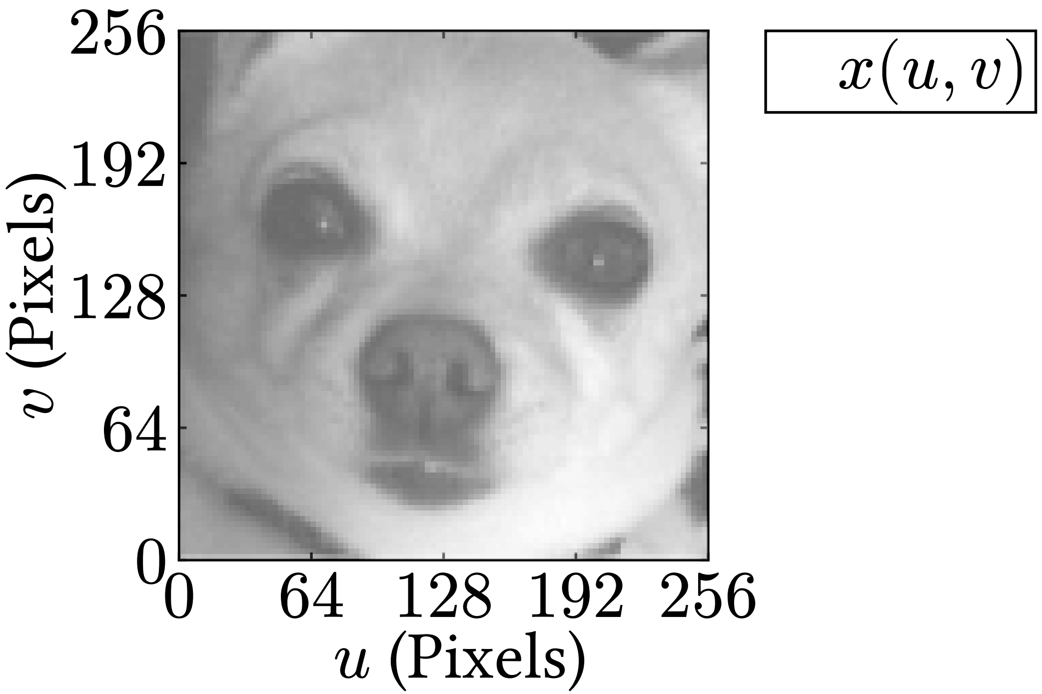
Structure: Nearby waves form syllables

Signal Processing

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

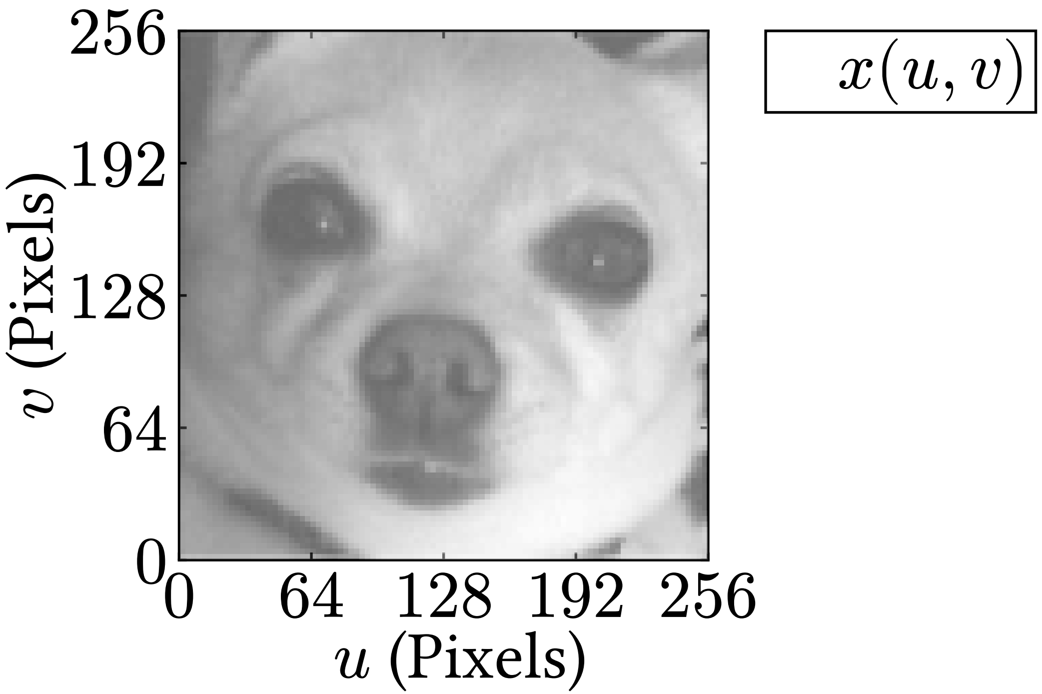
Signal Processing

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



Signal Processing

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



Structure: Repeated components (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:

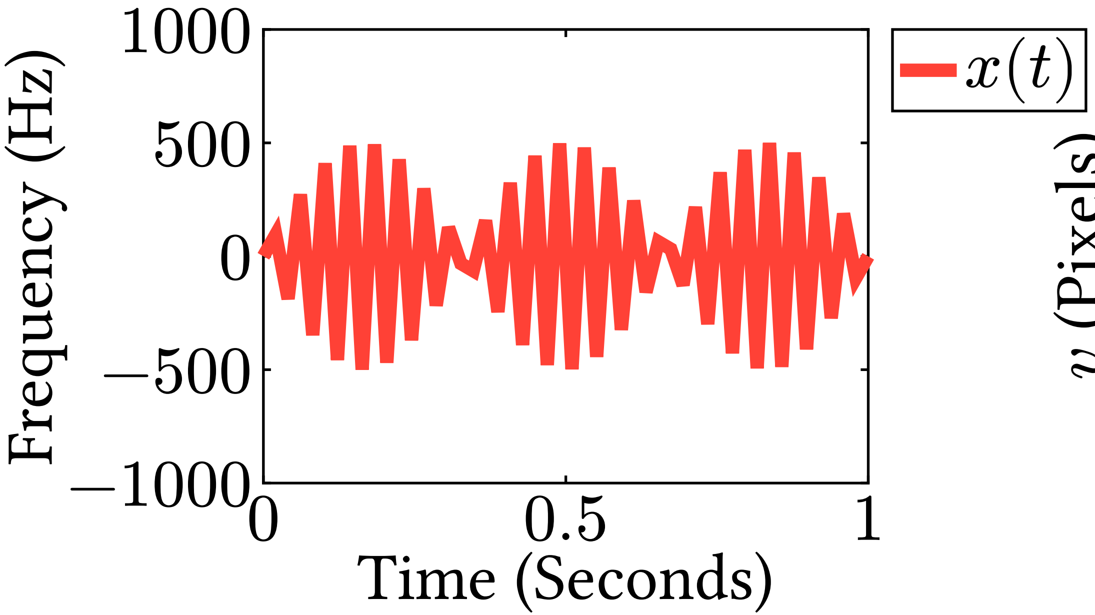
- Locality
- Translation invariance

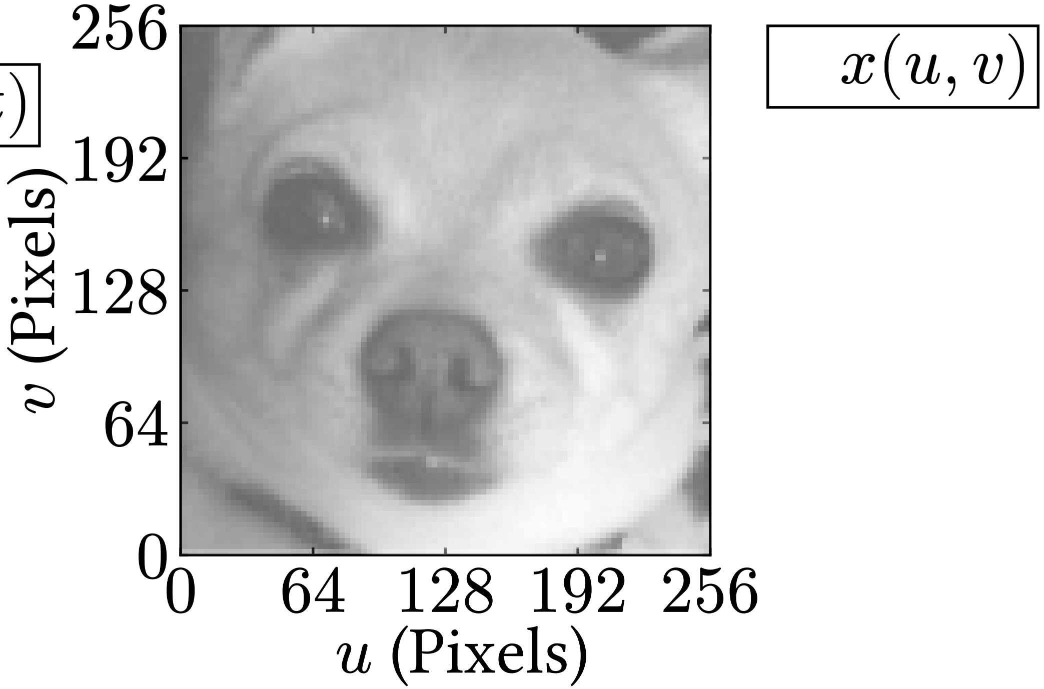
Signal Processing

Locality: Information concentrated over small regions of space/time

Signal Processing

Locality: Information concentrated over small regions of space/time



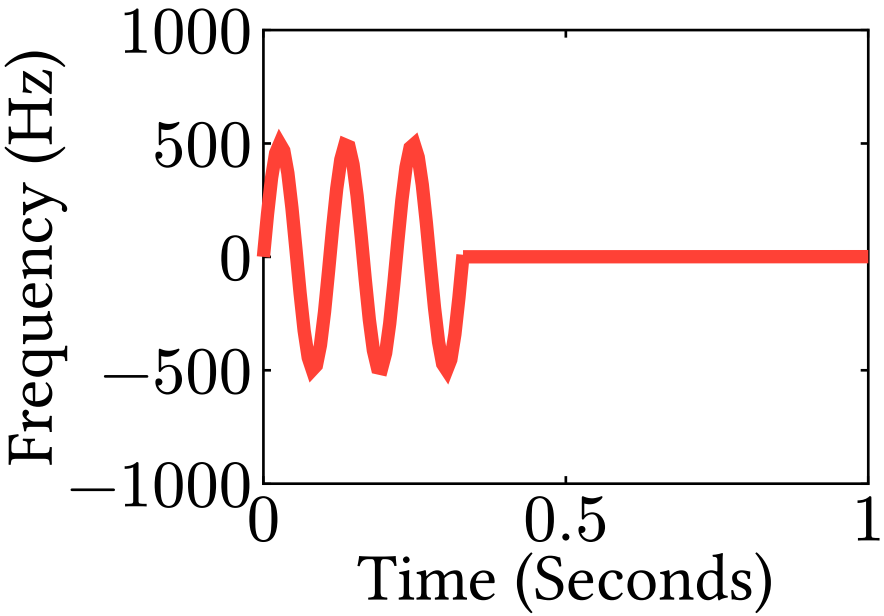


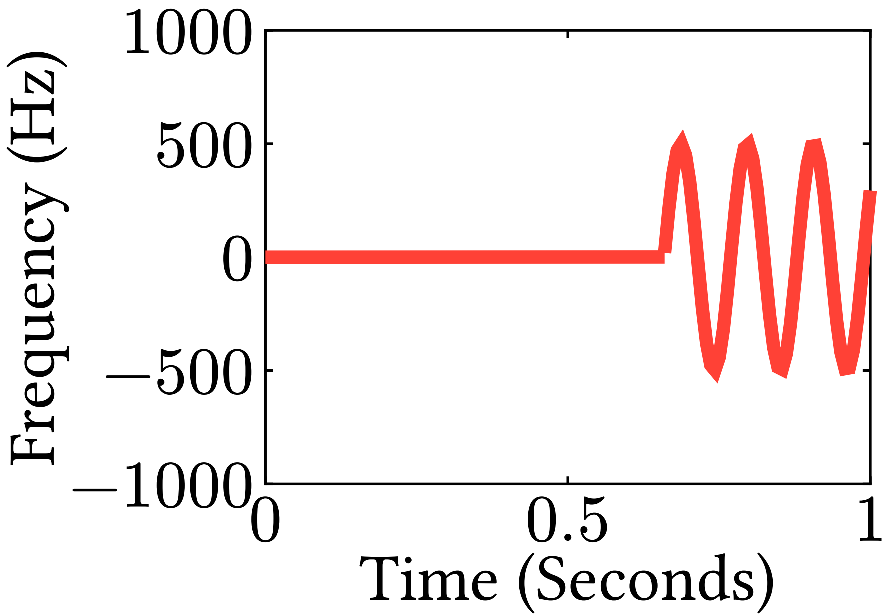
Signal Processing

Translation Invariance: Signal does not change when shifted in space/ time

Signal Processing

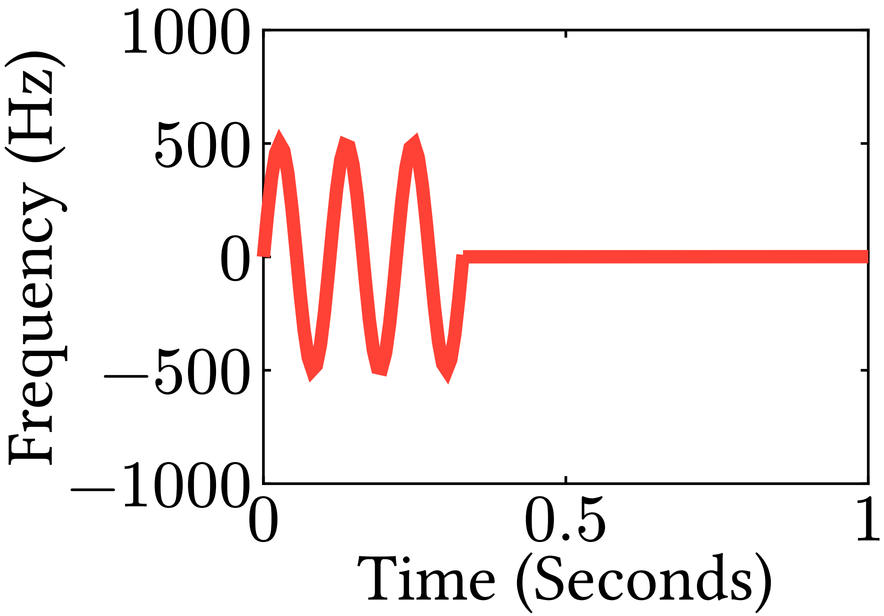
Translation Invariance: Signal does not change when shifted in space/time

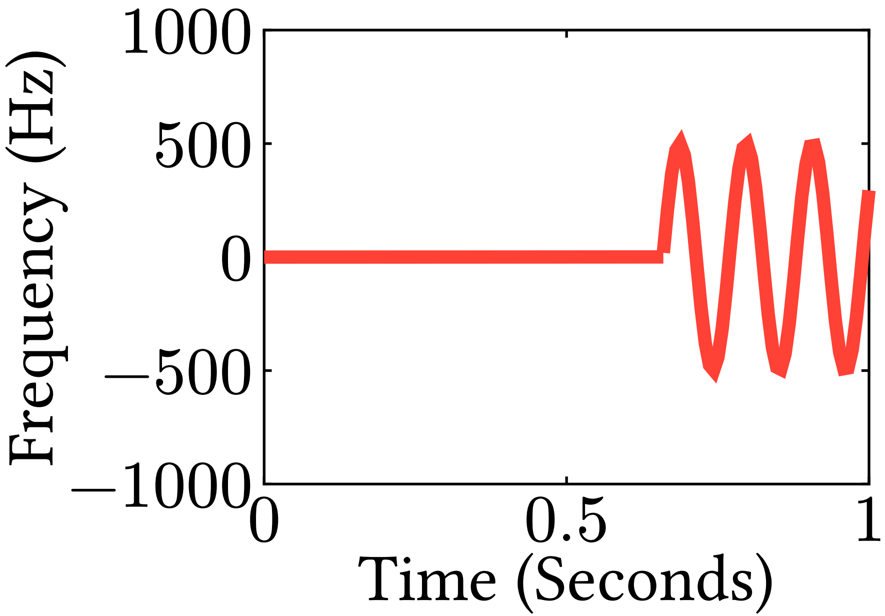




Signal Processing

Translation Invariance: Signal does not change when shifted in space/ time



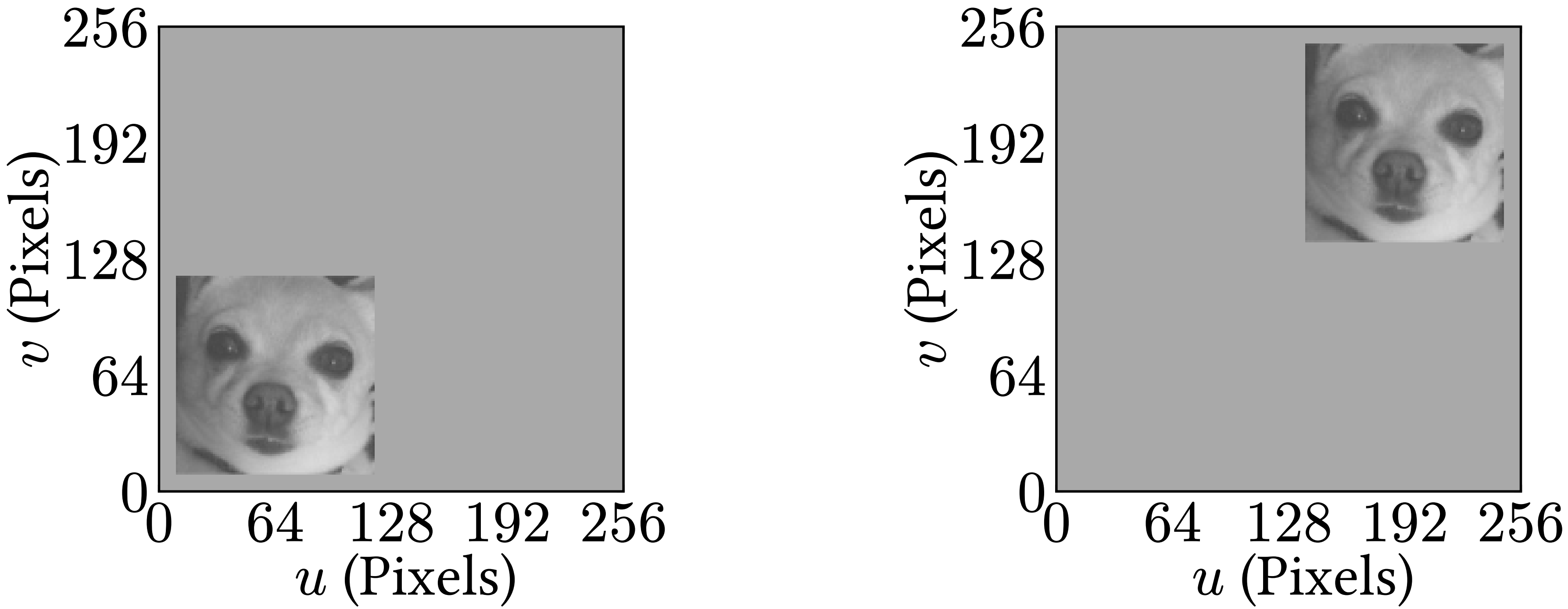


Both say "hello"

Signal Processing

Translation Invariance: Signal does not change when shifted

Translation Invariance: Signal does not change when shifted



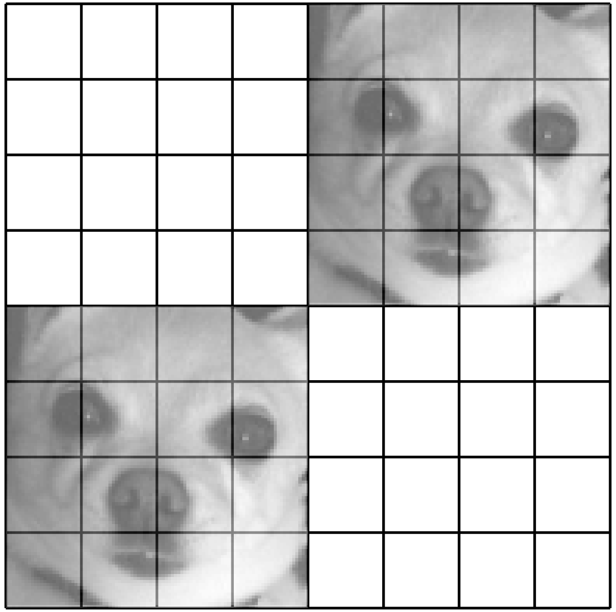
Both contain a dog

Signal Processing

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

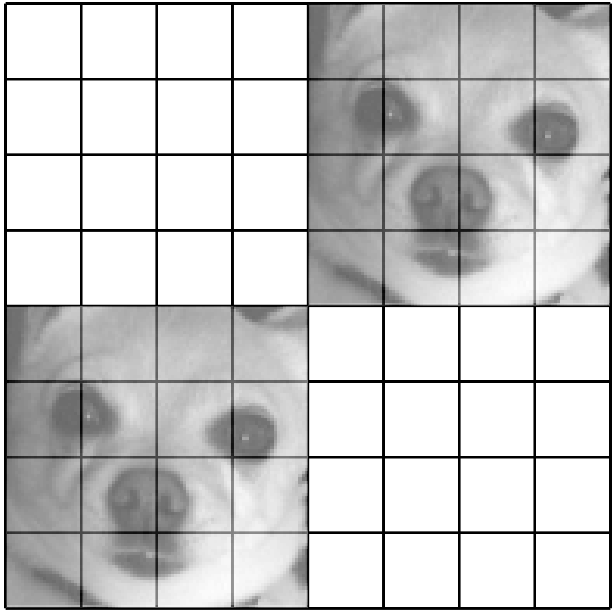
Signal Processing

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



Signal Processing

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



How can we get these properties in neural networks?

Agenda

1. Review
2. **Signal Processing**
3. Convolution
4. 2D Convolution
5. Downsampling
6. Coding

Agenda

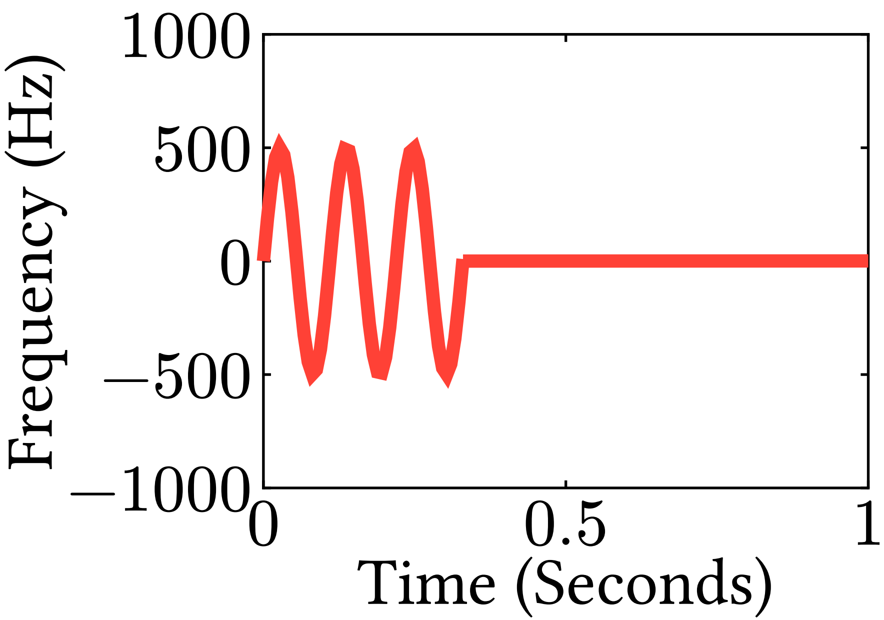
1. Review
2. Signal Processing
3. **Convolution**
4. 2D Convolution
5. Downsampling
6. Coding

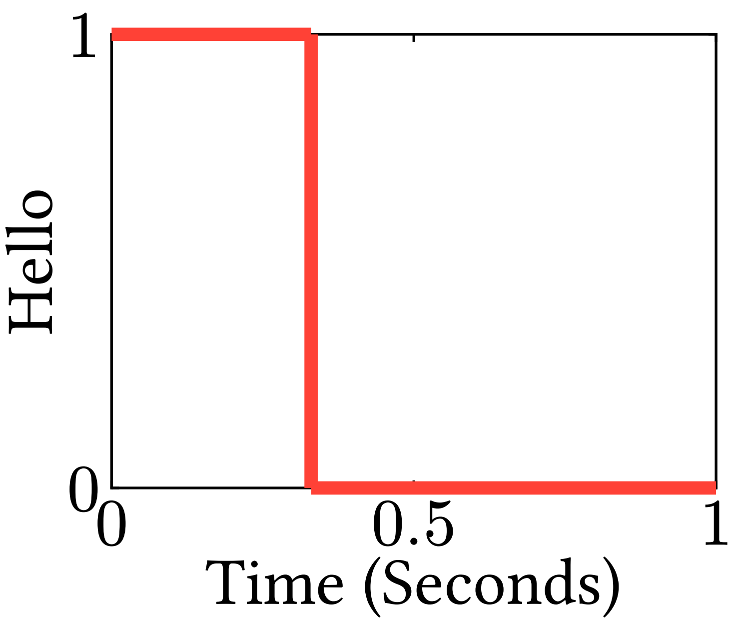
Convolution

In signal processing, we often turn signals into other signals

Convolution

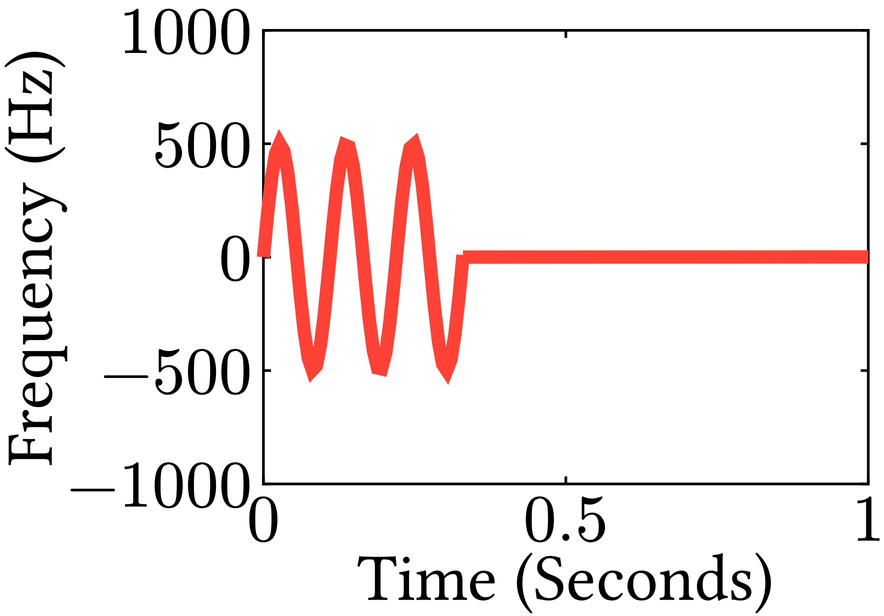
In signal processing, we often turn signals into other signals

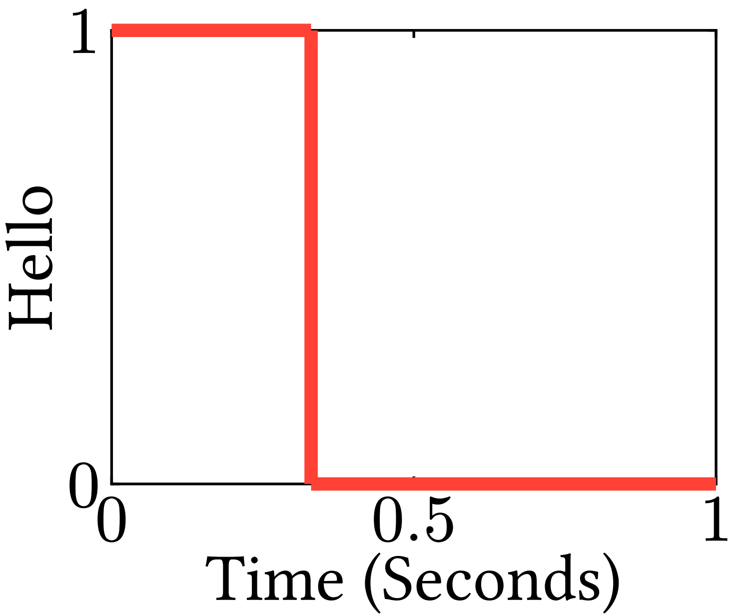




Convolution

In signal processing, we often turn signals into other signals





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

Convolution

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 time and space is continuous, we write convolution as

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

Convolution

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

If time and space is continuous, we write convolution as

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

If x, g are discrete time or space, we use a sum instead of integral

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

Convolution

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

If time and space is continuous, we write convolution as

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

If x, g are discrete time or space, we use a sum instead of integral

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

We slide the filter across the signal, taking the product as we go

Convolution

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

If time and space is continuous, we write convolution as

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

If x, g are discrete time or space, we use a sum instead of integral

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

We slide the filter across the signal, taking the product as we go

Convolution

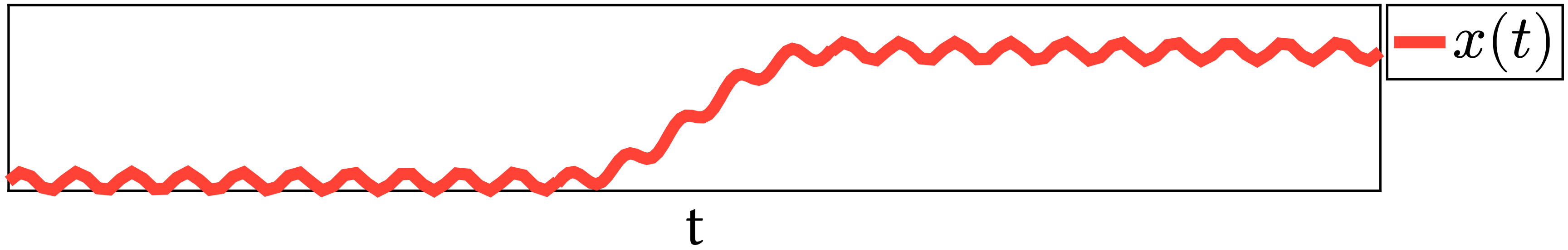
Example: Let us examine a low-pass filter

Convolution

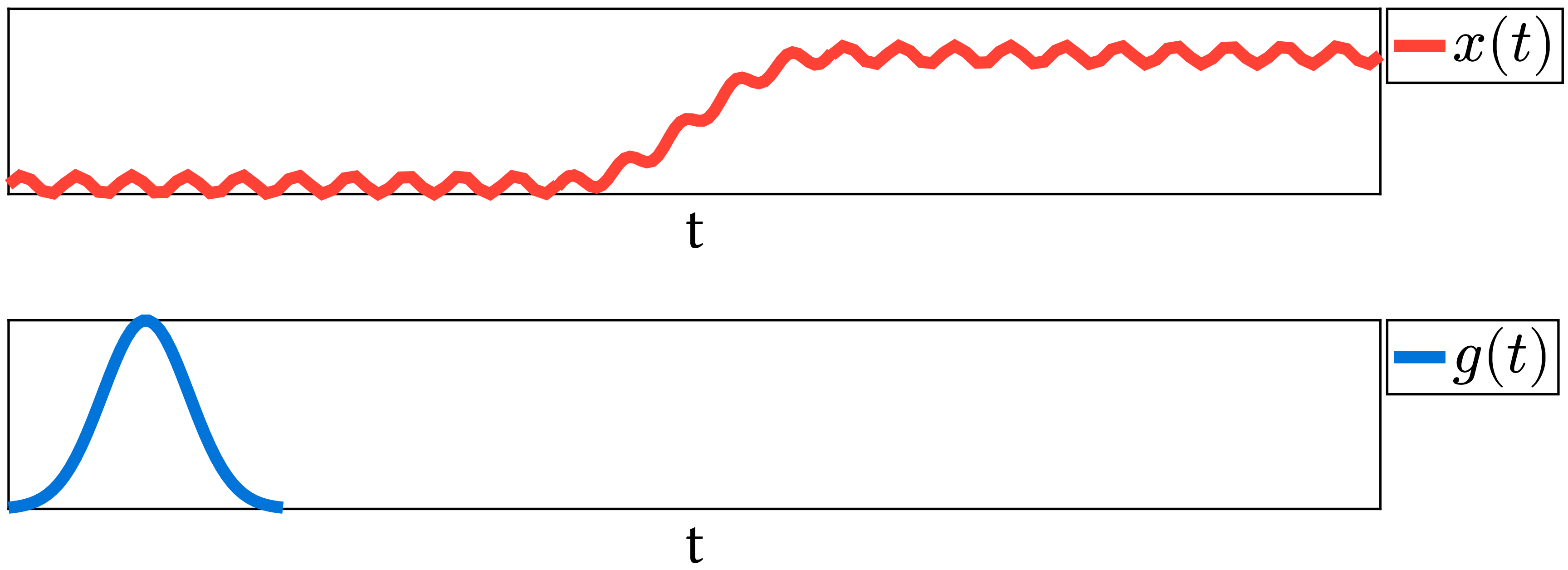
Example: Let us examine a low-pass filter

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

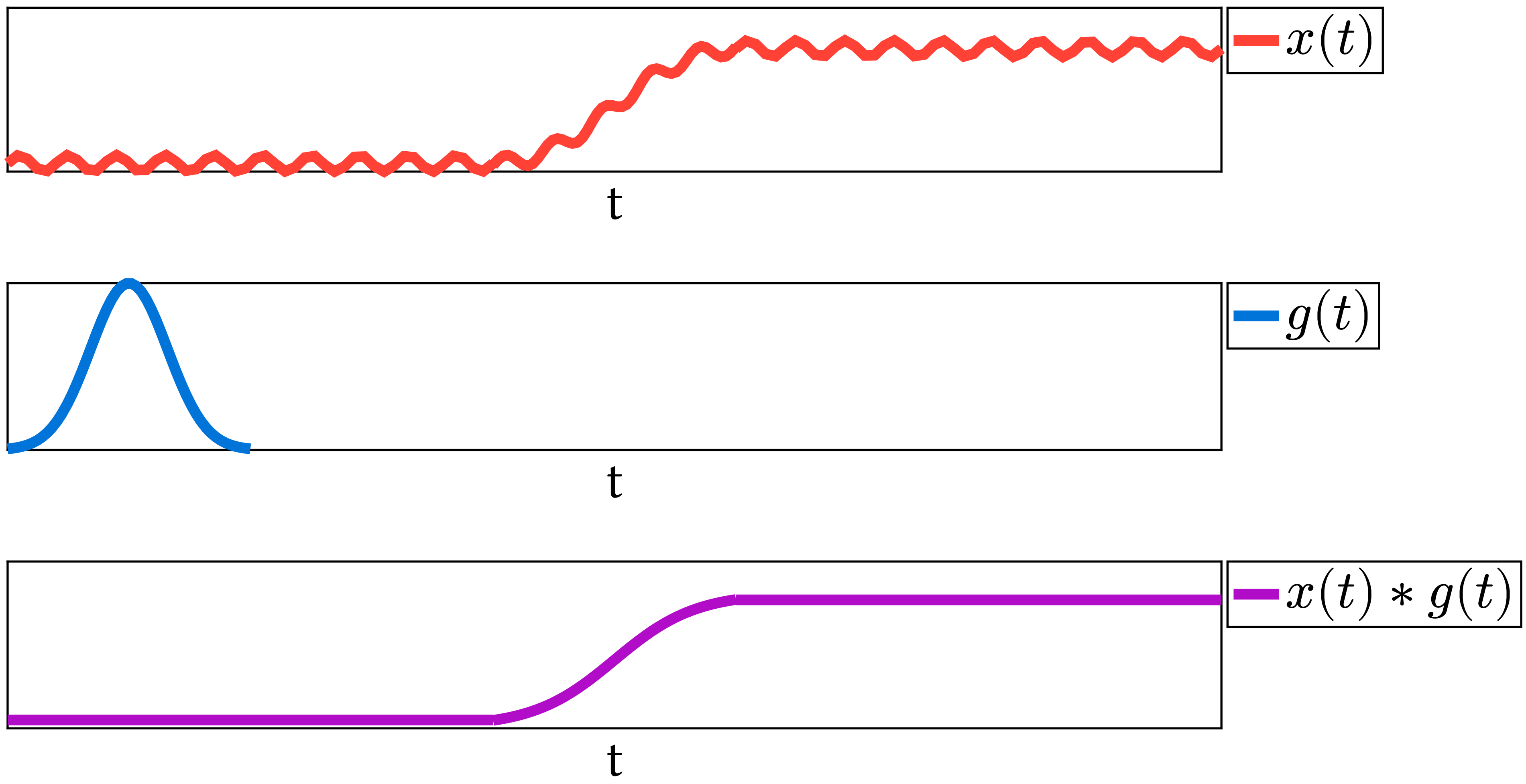
Convolution



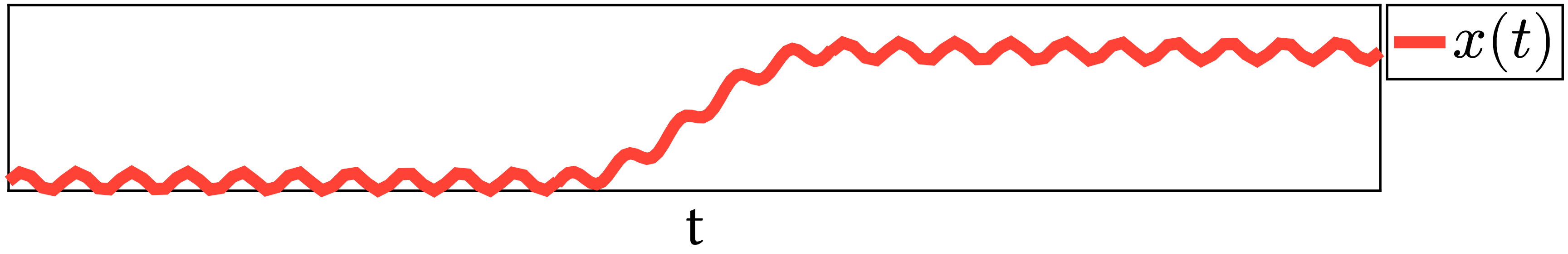
Convolution

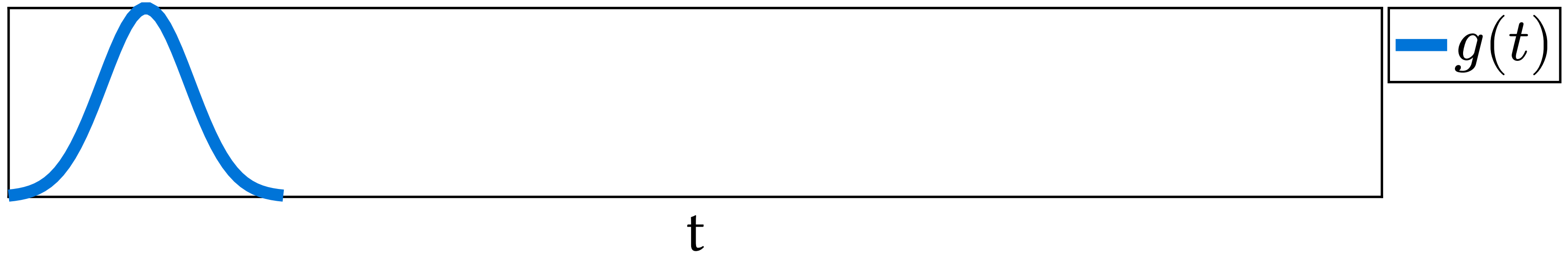


Convolution



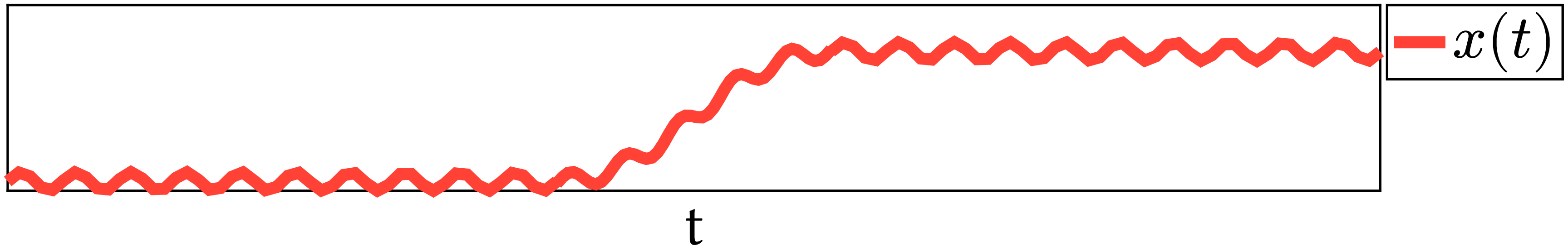
Convolution

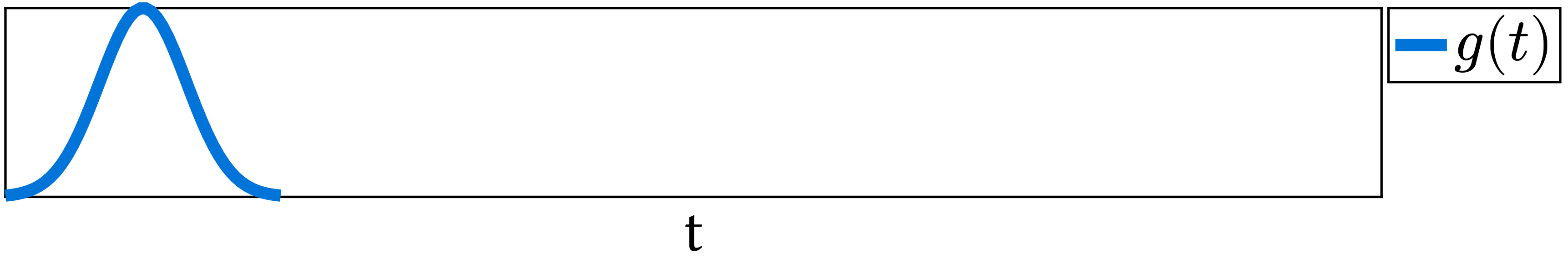




Convolution is **local** to the filter $g(t)$

Convolution





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

Convolution is also **invariant** 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

For digital signals, we use discrete time/space

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

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

Convolution

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

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

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

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

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

Just like neural networks, convolution is a linear operation

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

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

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

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?

Answer: In a neuron, each input x_i has a different parameter θ_i. In convolution, we reuse θ_i on $x_j, x_k, ...$

Convolution

Neuron:

$$\boldsymbol{\theta}^\top \overline{\boldsymbol{x}} = \sum_{i=0}^{d_x} \theta_i \overline{x}_i$$

Convolution

Neuron: $$\boldsymbol{\theta}^\top \overline{\boldsymbol{x}} = \sum_{i=0}^{d_x} \theta_i \overline{x}_i$$

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:

$$\boldsymbol{\theta}^\top \overline{\boldsymbol{x}} = \sum_{i=0}^{d_x} \theta_i \overline{x}_i$$

Convolution:

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

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

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

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

We call this a **convolutional layer**

Convolution

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

We call this a **convolutional layer**

Question: Anything missing?

Convolution

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

We call this a **convolutional layer**

Question: Anything missing?

Answer: Activation function!

Convolution

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

We call this a **convolutional layer**

Question: Anything missing?

Answer: Activation function!

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

Convolution

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

We call this a **convolutional layer**

Question: Anything missing?

Answer: Activation function!

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

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 invariant**, 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)
)

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

Convolution

```python
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,
)

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

Agenda

1. Review
2. Signal Processing
3. **Convolution**
4. 2D Convolution
5. Downsampling
6. Coding

Agenda

1. Review
2. Signal Processing
3. Convolution
4. **2D Convolution**
5. Downsampling
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