

## Surprises & Paradoxes I

1 Simpson Paradox

2 Data Morph: A Guided Tour

3 How to Fit Any Dataset with a Single Parameter

## Simpson Paradox

	Women			Men		
	applied	accepted	%	applied	accepted	%
Computer Science	26	7	27	228	58	25
Economics	240	63	26	512	112	22
Engineering	164	52	32	972	252	26
Medicine	416	99	24	578	140	24
Veterinary medicine	338	53	16	180	22	12
Total	1184	274	23	2470	584	24

Table: Cambridge University Admission Data, 1996.

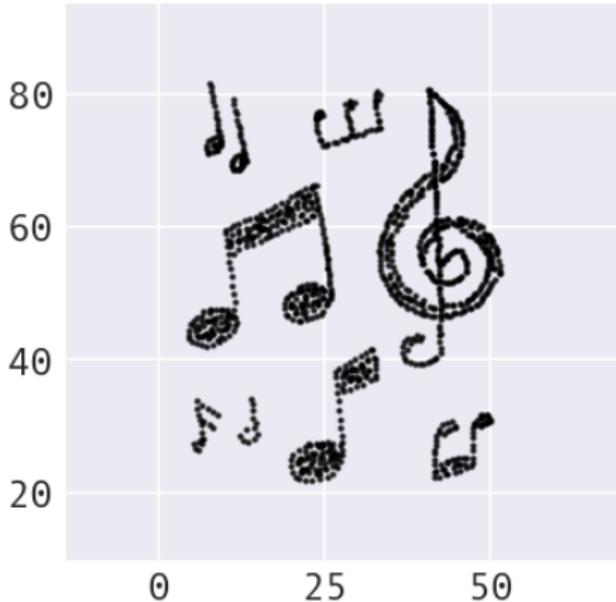
## Data Morph: A Guided Tour

# Milestones

- Anscombe, F., 1973. Anscombe's Quartet.
- Cairo, A., 2016. Datasaurus Dozen
- Matejka, J., Fitzmaurice, G., 2017. Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing. Website, Paper, Code, YouTube
- Molin, S., 2024. Data Morph: Moving Beyond the Datasaurus Dozen. Website, Code.

Let's play a game. I'm thinking of a distribution with the following summary statistics. Can you picture what a scatter plot of the data would look like?

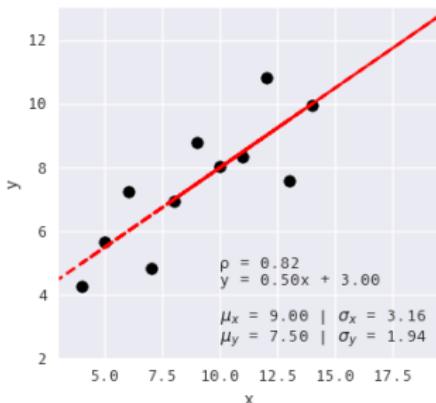
- $X$  mean = 30.37
- $Y$  mean = 53.01
- $X$  standard deviation = 13.44
- $Y$  standard deviation = 15.53
- Correlation coefficient = 0.04



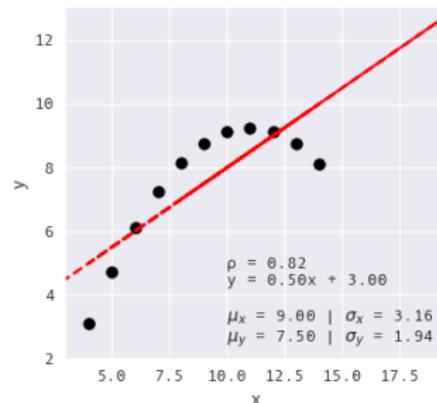
X Mean: 30.3685136  
Y Mean: 53.0126900  
X SD : 13.4415721  
Y SD : 15.5344370  
Corr. : +0.0396375

# Anscombe's Quartet

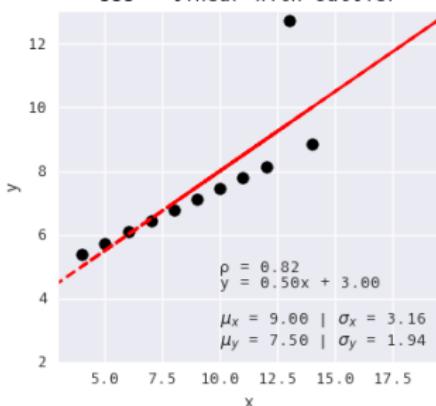
I - linear



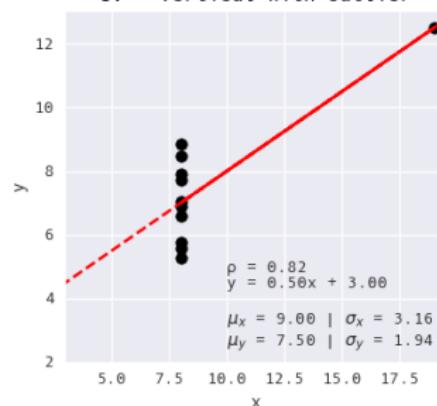
II - non-linear

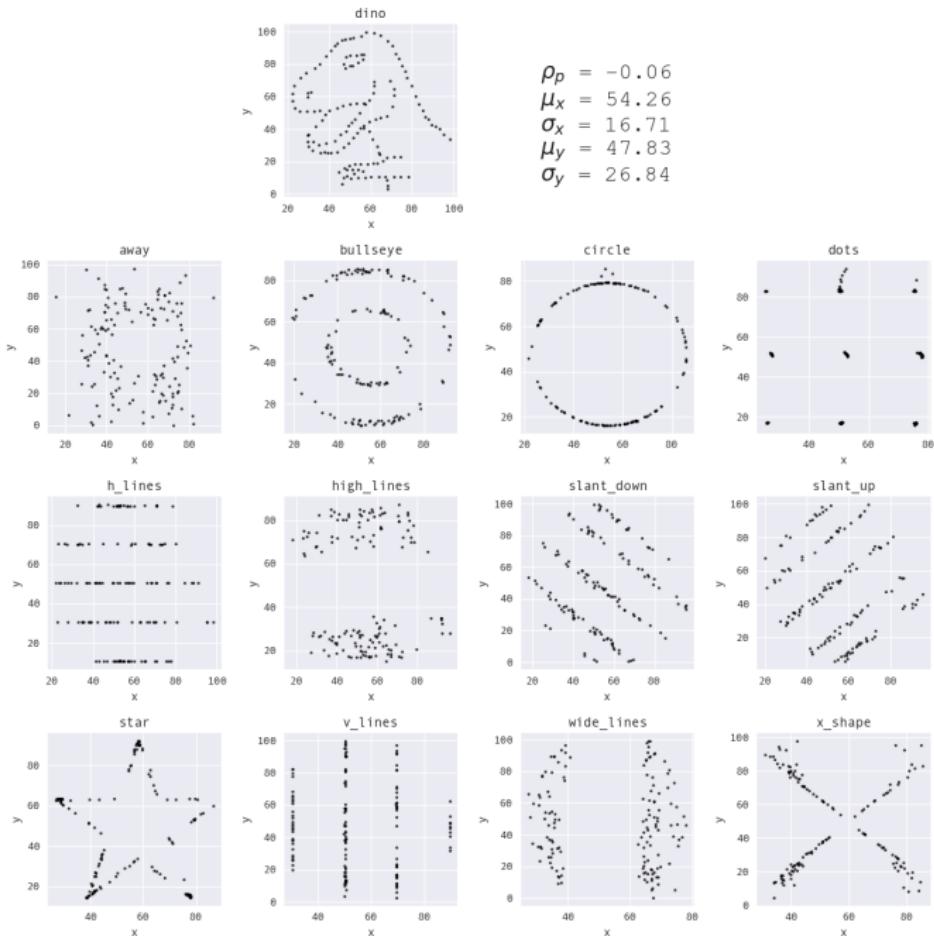


III - linear with outlier



IV - vertical with outlier





# Data Morph

## How to Fit Any Dataset with a Single Parameter

# The Core Result

Boué, L., 2019. Real Numbers, Data Science and Chaos: How to Fit any Dataset with a Single Parameter. arXiv, Code.

Main theorem: Any dataset can be fit using

$$f_\alpha(x) = \sin^2(2^{x\tau} \arcsin \sqrt{\alpha})$$

where:

- $\alpha \in \mathbb{R}$  is a single learned parameter
- $x \in [0, \dots, n]$  takes integer values
- $\tau \in \mathbb{N}$  controls accuracy

Properties:

- Continuous and differentiable
- Arbitrary precision fit
- Single real-valued parameter

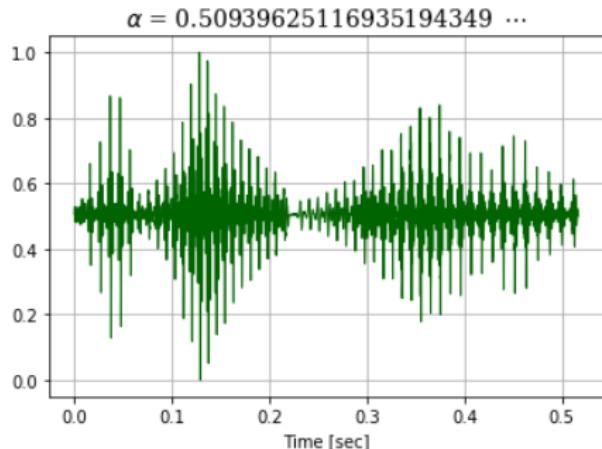
# Demonstration: Animal Shapes



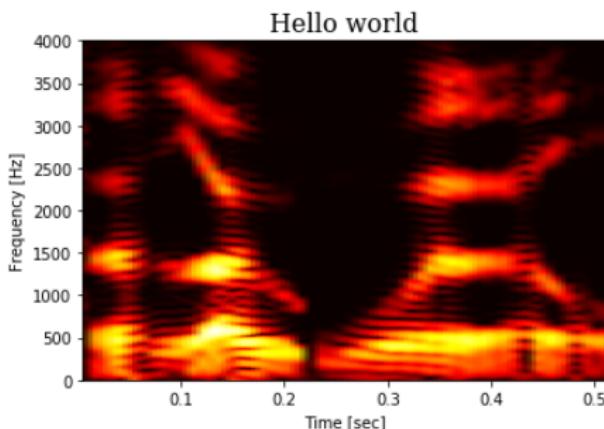
Figure: Generated using different values of  $\alpha$

- Each shape is a scatter plot  $(x, y)$
- $x \in \mathbb{N}$  are integer values
- $y = f_\alpha(x)$  gives y-coordinates
- Different  $\alpha$  values = different shapes

# Audio Signal Example



(a) Waveform



(b) Spectrogram

Figure: "Hello world" audio signal

Processing:

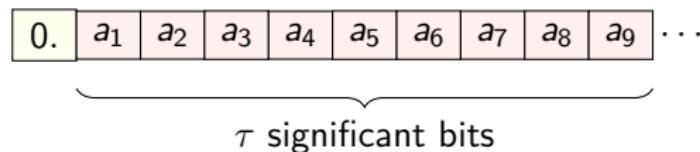
- Sample at 11kHz
- Values determined by  $f_\alpha$
- Complex waveform from single  $\alpha$

# Fixed-point Binary Representation

For  $\alpha \in [0, 1]$ :

$$\alpha = \sum_{n=1}^{+\infty} \frac{a_n}{2^n}$$

where  $a_n \in \{0, 1\}$



In practice:

- Truncate to  $\tau$  bits
- Error bound:  $|\alpha - \alpha_{\text{approx}}| \leq \frac{1}{2^\tau}$

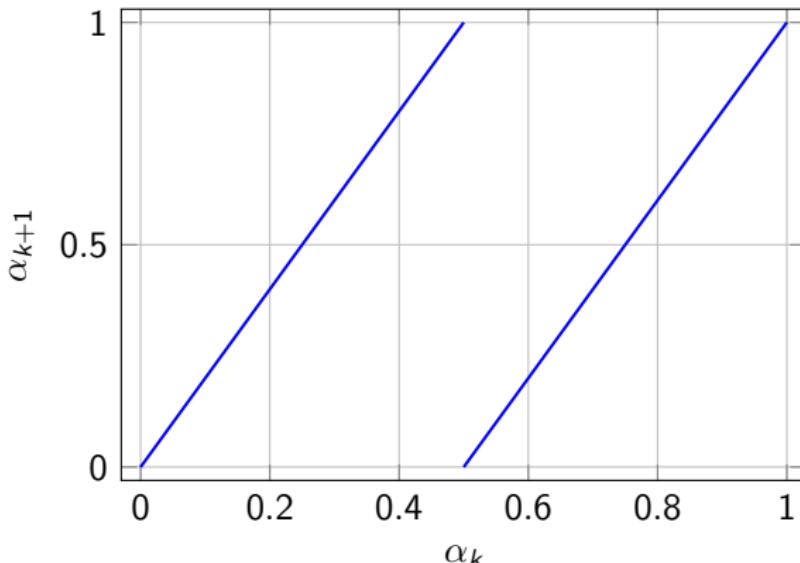
# The Dyadic Transformation

Definition:

$$\mathcal{D}(\alpha_k) = 2\alpha_k \bmod 1$$

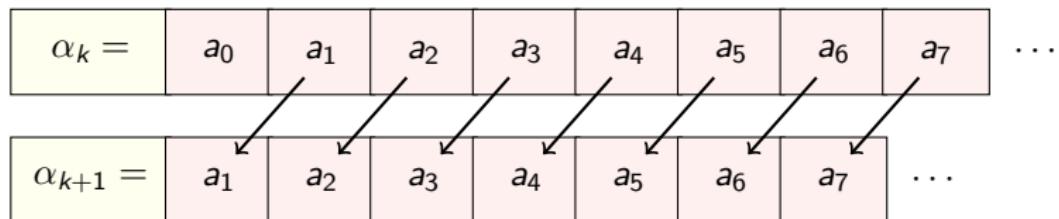
Properties:

- Maps  $[0,1]$  to itself
- Piecewise linear
- Exhibits chaos



# Bit-Shift Property

In binary,  $\mathcal{D}$  is a left shift:



Key properties:

- Each iteration loses 1 bit
- After  $\tau$  iterations, significant bits lost
- Shows sensitive dependence on initial conditions

# Initial Implementation

Convert decimal to binary:

```
def decimalToBinary(decimalInitial):
    return reduce(lambda acc, _:
        [dyadicMap(acc[0]), acc[1] + ('0' if acc[0] < 0.5 else
            '1')],
        range(tau), [decimalInitial, ''])[1]
```

The dyadic map:

```
dyadicMap = lambda x: (2 * x) % 1
```

# Encoding Strategy

Converting  $\mathcal{X} = [x_0, \dots, x_n]$  to  $\alpha_0$ :

- ① Convert each  $x_i$  to  $\tau$ -bit binary
- ② Concatenate all strings
- ③ Convert to decimal  $\alpha_0$

0.	$x_{\text{bin}}^0$	$x_{\text{bin}}^1$	$\dots$														
----	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	---------

First  $\tau$  bits encode  $x_0$ , next  $\tau$  bits encode  $x_1$ , etc.

# Historical Background

Origins:

- Population demographics model
- Studied by Robert May (1976)
- Canonical example of chaos

Definition:

$$z_{k+1} = \mathcal{L}(z_k) = rz_k(1 - z_k)$$

We focus on  $r = 4$  case where:

- System is fully chaotic
- Maps  $[0,1]$  to itself
- No stable fixed points

# Properties of the Logistic Map

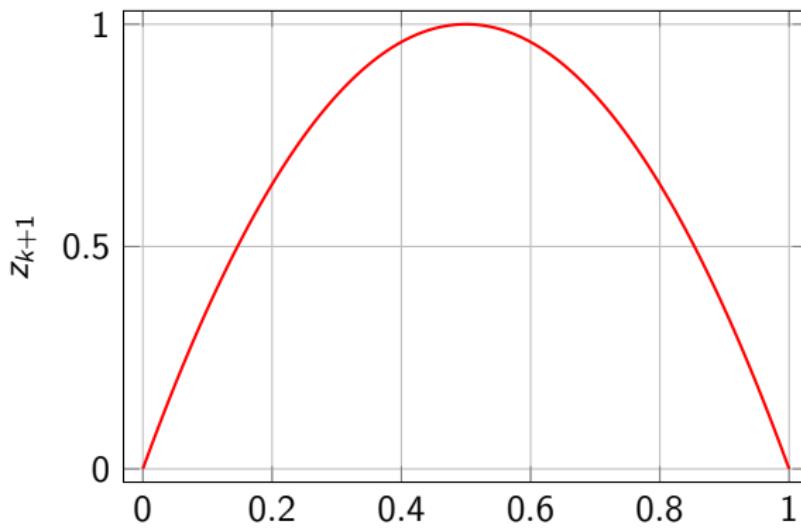
Mathematical structure:

$$\mathcal{L}(z_k) = 4z_k(1 - z_k)$$

Key features:

- Continuous and differentiable
- Maximum at  $z = 1/2$
- Quadratic nonlinearity

Logistic Map for  $r = 4$



# Contrast with Dyadic Map

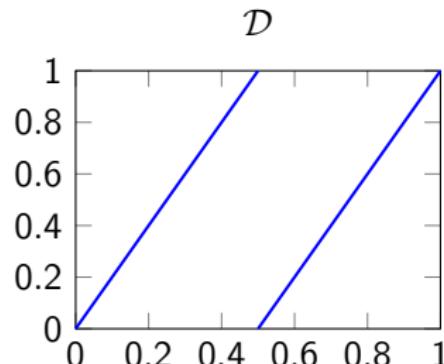
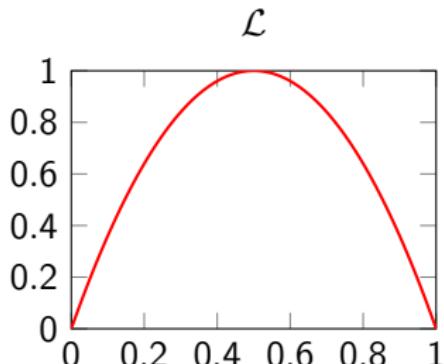
Comparison:

Logistic Map  $\mathcal{L}$ :

- Smooth
- Quadratic
- Continuous

Dyadic Map  $\mathcal{D}$ :

- Piecewise linear
- Uses modulo
- Discontinuous



# The Bridge Function $\phi$

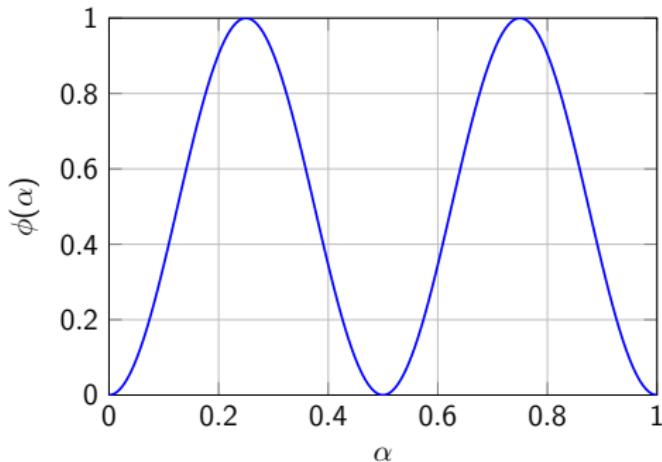
Definition:

$$\phi(\alpha) = \sin^2(2\pi\alpha)$$

Properties:

- Continuous and differentiable
- Maps  $[0,1]$  to  $[0,1]$
- Periodic with period 1
- Has continuous inverse

The Bridge Function



# The Inverse Bridge

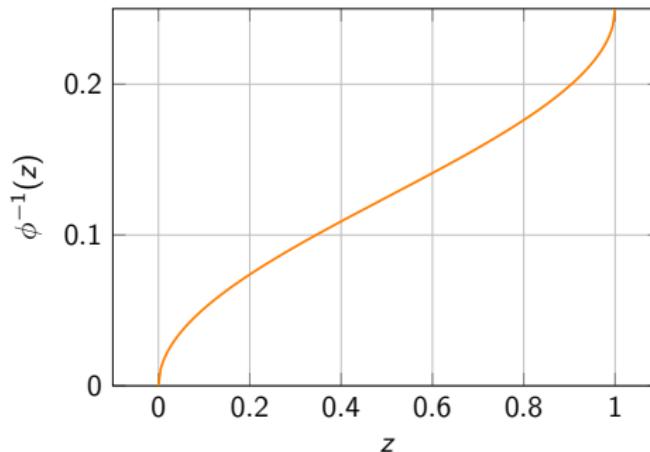
Definition:

$$\phi^{-1}(z) = \frac{\arcsin \sqrt{z}}{2\pi}$$

Properties:

- Also continuous
- Maps  $[0,1]$  to  $[0,1/4]$
- Composition yields identity

The Inverse Bridge



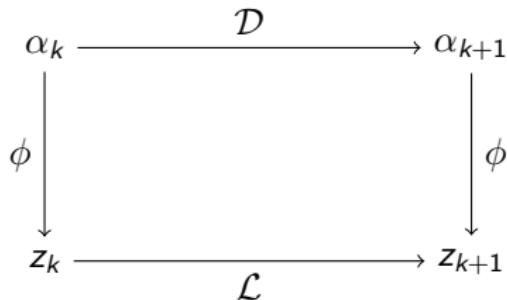
# The Conjugacy Relation

Note that

$$z_{k+1} = \mathcal{L}(z_k) = 4\phi(\alpha_k)(1 - \phi(\alpha_k)) = 4\sin^2(2\pi\alpha_k)\cos^2(2\pi\alpha_k) = \sin^2(2\pi \cdot 2\alpha_k)$$

Key equation:

$$\mathcal{L} \circ \phi = \phi \circ \mathcal{D}$$



Implications:

- Same dynamics in both spaces
- Can work in either representation
- Smooth version available

# Deriving the Final Formula

Starting with:

$$z_k = \phi(\alpha_k) = \sin^2(2\pi\alpha_k)$$

From dyadic map:

$$\alpha_k = 2^{k\tau} \alpha_0 \bmod 1$$

Combining gives:

$$f_\alpha(x) = \sin^2(2^{x\tau} \arcsin \sqrt{\alpha})$$

This is our elegant final result!

# Setting Up

Required imports and precision:

```
from mpmath import mp, pi, sin, asin, sqrt
import numpy as np
from functools import reduce

# Set precision
mp.dps = 1000 # decimal digits
tau = 8         # bits per sample
```

Basic helper functions:

```
# Dyadic map
dyadicMap = lambda x: (2 * x) % 1

# Bridge function
phi = lambda alpha: sin(2 * pi * alpha)**2
phiInv = lambda z: asin(sqrt(z)) / (2 * pi)
```

# Binary Conversion

Converting between representations:

```
def decimalToBinary(decimalInitial):
    return reduce(lambda acc, _:
        [dyadicMap(acc[0]), acc[1] + ('0' if acc[0] < 0.5 else
            '1')], 
        range(tau), [decimalInitial, ''])

def binaryToDecimal(binaryInitial):
    return reduce(lambda acc, val:
        acc + int(val[1]) / 2**val[0] + 1), enumerate(
            binaryInitial),
        mp.mpf(0.0))
```

# Dataset Processing

Encoding the dataset:

```
# Convert dataset to binary
binaryInitial = ''.join(map(decimalToBinary, xs))
decimalInitial = binary.ToDecimal(binaryInitial)

print('Binary initial:', binaryInitial[:50], '...')
print('Decimal initial:', float(decimalInitial))
```

The decoder function:

```
def logisticDecoder(k):
    return sin(2**((k*tau) * asin(sqrt(decimalInitial))))**2

# Recover all samples
decodedValues = [float(logisticDecoder(_)) for _ in range(len(xs))]
```

# Error Checking

Verify theoretical bounds:

```
# Maximum allowed error
maxError = pi / 2**(tau - 1)

# Check all errors
normalizedErrors = [abs(decoded - true) / maxError
    for decoded, true in zip(decodedValues, xs)]
]

# Verify bounds
assert all(e <= 1.0 for e in normalizedErrors)
print('Maximum normalized error:', max(normalizedErrors))
```

Example output:

```
Maximum normalized error: 0.8732
All errors within theoretical bound
```

# Animal Shape Example

Complete process:

- ① Generate x-coordinates:  $x \in [0, \dots, n]$
- ② Choose appropriate  $\alpha$  value
- ③ Compute  $y = f_\alpha(x)$  for each  $x$
- ④ Plot resulting  $(x, y)$  pairs



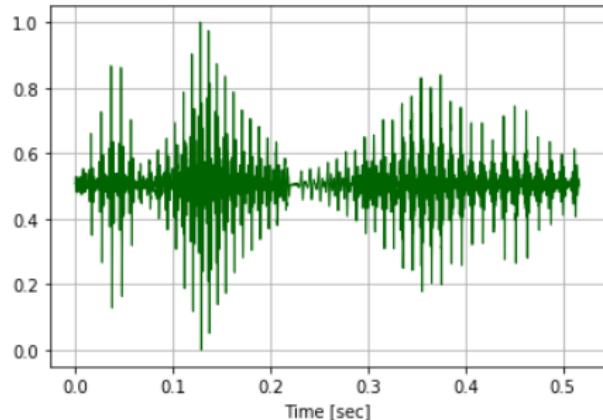
Figure: Different  $\alpha$  values generate different shapes

# Audio Signal Generation

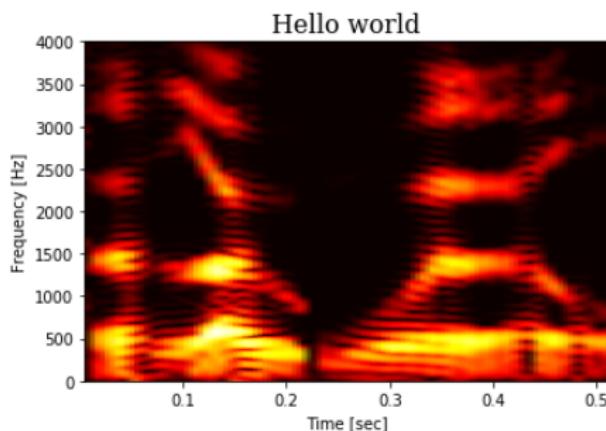
Process:

- ① Choose sampling rate (11kHz)
- ② Generate time points  $t_i$
- ③ Compute  $f_\alpha(i)$  for each  $t_i$
- ④ Scale to audio range [-1,1]

$$\alpha = 0.50939625116935194349 \dots$$



(a) Time domain



(b) Frequency domain

Figure: "Hello world" audio encoding

# Image Generation

CIFAR-10 process:

- ① Generate 3072 values ( $32 \times 32 \times 3$ )
- ② Reshape into RGB channels
- ③ Scale to [0,255] range
- ④ Stack into final image

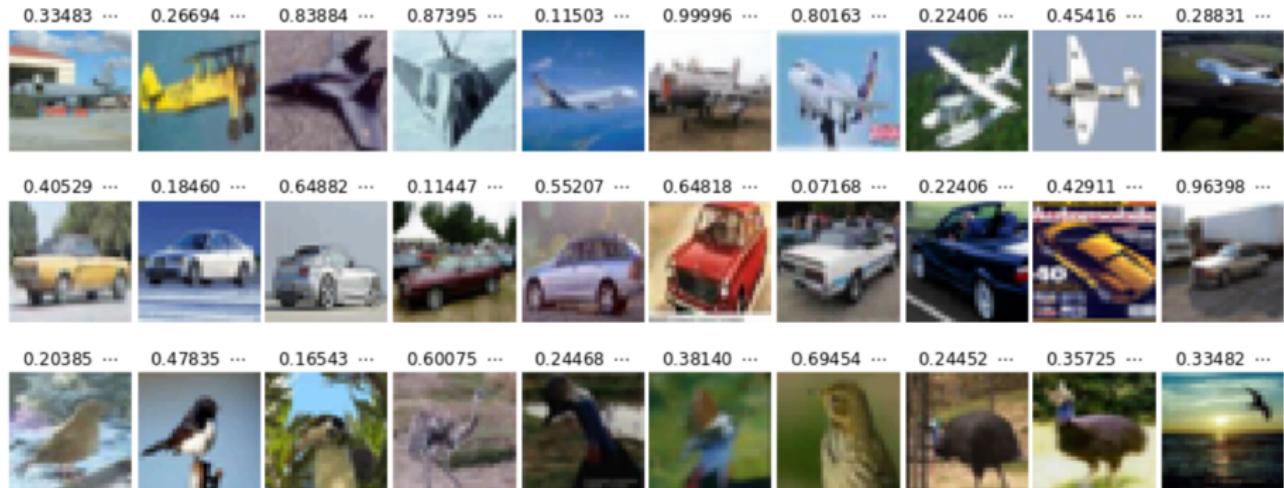


Figure: Generated CIFAR-10 style images

# Generalization Analysis

Time series example:

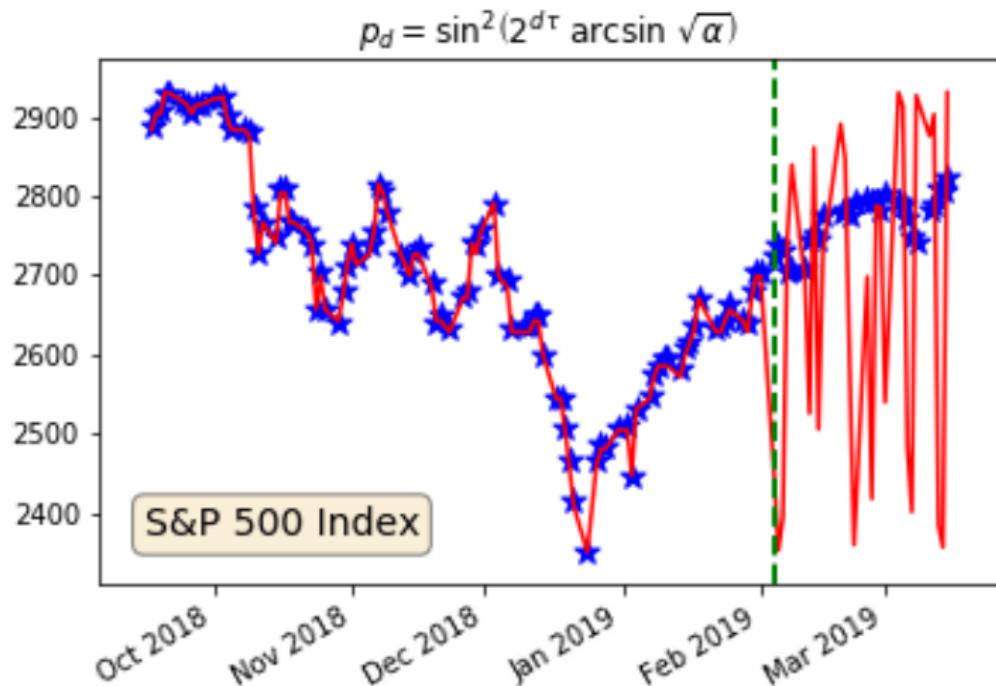


Figure: S&P 500 predictions showing no generalization