# From Machine Learning to Physics, and back again...

Romuald A. Janik

Jagiellonian University
Kraków

RJ 1909.10831
RJ, P. Witaszczyk 2006.04791

## Outline

**What is Machine Learning?**

# Outline

## Outline

# Outline

**What is Machine Learning?**

**From Machine Learning to Physics**
- The problem of computing Entropy
- Entropy from Machine Learning
- 2D Ising model entropy/free energy from Monte Carlo @T

**Interlude: Deep Convolutional Neural Networks**

**From Physics back to Machine Learning**
- Deep Neural Networks and Physics – analogies
- Complexity and effective dimension for neural networks
- Numerical experiments

**Summary**

## Outline

**What is Machine Learning?**

**From Machine Learning to Physics**
- The problem of computing Entropy
- Entropy from Machine Learning
- 2D Ising model entropy/free energy from Monte Carlo @T

**Interlude: Deep Convolutional Neural Networks**

**From Physics back to Machine Learning**
- Deep Neural Networks and Physics – analogies
- Complexity and effective dimension for neural networks
- Numerical experiments

**Summary**

## What is Machine Learning?

- **Machine Learning** comprises a set of algorithms which e.g.
  1. given some set of example data points belonging to various classes,
  2. **learns** based on these examples,
  3. so that it can assign new unseen data points to these classes...
- The renaissance of Machine Learning in recent years came from the success of Deep Neural Networks on image datasets..
- **MNIST**: The *Hello World* of Machine Learning...
- **CIFAR-10** and **CIFAR-100**

## What is Machine Learning?

- **Machine Learning** comprises a set of algorithms which e.g.
  1. given some set of example data points belonging to various classes,
  2. **learns** based on these examples,
  3. so that it can assign new unseen data points to these classes...
- The renaissance of Machine Learning in recent years came from the success of Deep Neural Networks on image datasets..
- **MNIST**: The *Hello World* of Machine Learning...
- **CIFAR-10** and **CIFAR-100**

## What is Machine Learning?

- **Machine Learning** comprises a set of algorithms which e.g.
  1. given some set of example data points belonging to various classes,
  2. **learns** based on these examples,
  3. so that it can assign new unseen data points to these classes...
- The renaissance of Machine Learning in recent years came from the success of Deep Neural Networks on image datasets..
- **MNIST**: The *Hello World* of Machine Learning...
- **CIFAR-10** and **CIFAR-100**

## What is Machine Learning?

- **Machine Learning** comprises a set of algorithms which e.g.
  1. given some set of example data points belonging to various classes,
  2. **learns** based on these examples,
  3. so that it can assign new unseen data points to these classes...
- The renaissance of Machine Learning in recent years came from the success of Deep Neural Networks on image datasets..
- **MNIST**: The *Hello World* of Machine Learning...
- **CIFAR-10** and **CIFAR-100**

## What is Machine Learning?

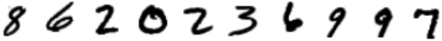
- **Machine Learning** comprises a set of algorithms which e.g.
  1. given some set of example data points belonging to various classes,
  2. **learns** based on these examples,
  3. so that it can assign new unseen data points to these classes...
- The renaissance of Machine Learning in recent years came from the success of Deep Neural Networks on image datasets..
- **MNIST**: The *Hello World* of Machine Learning...
- **CIFAR-10** and **CIFAR-100**

## What is Machine Learning?

- **Machine Learning** comprises a set of algorithms which e.g.
  1. given some set of example data points belonging to various classes,
  2. **learns** based on these examples,
  3. so that it can assign new unseen data points to these classes...
- The renaissance of Machine Learning in recent years came from the success of Deep Neural Networks on image datasets..
- **MNIST**: The *Hello World* of Machine Learning...
- **CIFAR-10** and **CIFAR-100**

## What is Machine Learning?

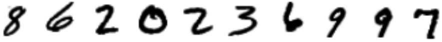
- **Machine Learning** comprises a set of algorithms which e.g.
  1. given some set of example data points belonging to various classes,
  2. **learns** based on these examples,
  3. so that it can assign new unseen data points to these classes...
- The renaissance of Machine Learning in recent years came from the success of Deep Neural Networks on image datasets..
- **MNIST**: The *Hello World* of Machine Learning...



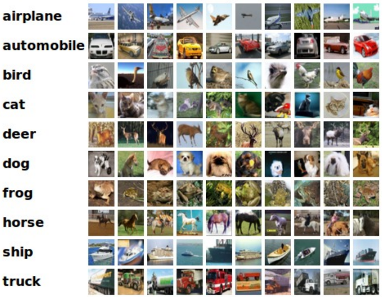
- **CIFAR-10** and **CIFAR-100**

## What is Machine Learning?

- **Machine Learning** comprises a set of algorithms which e.g.
  1. given some set of example data points belonging to various classes,
  2. **learns** based on these examples,
  3. so that it can assign new unseen data points to these classes...
- The renaissance of Machine Learning in recent years came from the success of Deep Neural Networks on image datasets..
- **MNIST**: The *Hello World* of Machine Learning...

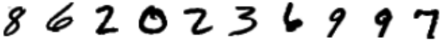


- **CIFAR-10** and **CIFAR-100**

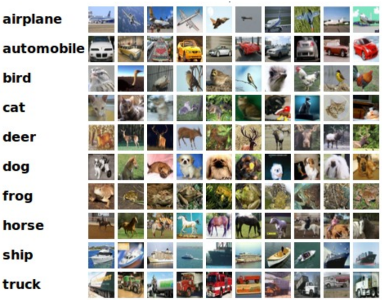


## What is Machine Learning?

- **Machine Learning** comprises a set of algorithms which e.g.
  1. given some set of example data points belonging to various classes,
  2. **learns** based on these examples,
  3. so that it can assign new unseen data points to these classes...
- The renaissance of Machine Learning in recent years came from the success of Deep Neural Networks on image datasets..
- **MNIST**: The *Hello World* of Machine Learning...



- **CIFAR-10** and **CIFAR-100**



## What is Machine Learning?

- **ImageNet**: state of the art...
  1.2 million images (256x256+) in 1000 classes

- There is now of course much more to ML (Reinforcement Learning (e.g. AlphaGo), GAN, language models etc.)...

## What is Machine Learning?

- **ImageNet**: state of the art...
  1.2 million images (256x256+) in 1000 classes

- There is now of course much more to ML (Reinforcement Learning (e.g. AlphaGo), GAN, language models etc.)...

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1 | \underbrace{x_{11} \ldots x_{ij} \ldots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1|\underbrace{x_{11} \dots x_{ij} \dots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1 | \underbrace{x_{11} \dots x_{ij} \dots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1 | \underbrace{x_{11} \ldots x_{ij} \ldots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1 | \underbrace{x_{11} \ldots x_{ij} \ldots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1 | \underbrace{x_{11} \dots x_{ij} \dots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1 | \underbrace{x_{11} \ldots x_{ij} \ldots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1 | \underbrace{x_{11} \ldots x_{ij} \ldots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1 | \underbrace{x_{11} \ldots x_{ij} \ldots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1 | \underbrace{x_{11} \dots x_{ij} \dots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

$$p(y = 1 | \underbrace{x_{11} \ldots x_{ij} \ldots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

## What is Machine Learning?

- There is a wide variety of Machine Learning classification algorithms:
  - Deep Neural Networks
  - Random Forests
  - Gradient Boosted Trees
  - Nearest Neighbours
  - Logistic Regression
- Consider now a two class problem:
  e.g. images showing cats or dogs, $y = 1$ (cat), $y = 0$ (dog)
- Typically these ML classification algorithms do not provide only the *class* $y$ of a new data point (e.g. image) but rather give the **probability** that the class is $y = 1$
- This is really the conditional probability

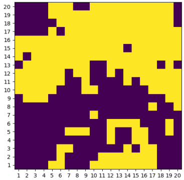
$$p(y = 1 | \underbrace{x_{11} \dots x_{ij} \dots x_{HW}}_{\text{pixels of the image}})$$

- Machine Learning classification algorithms are designed to model (and learn from data!) **very complex conditional probability distributions...**

# Why computing entropy is interesting?

# Why computing entropy is interesting?

**Monte-Carlo configurations**

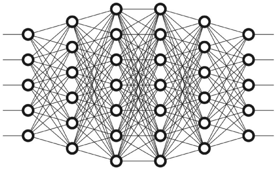



What is the free energy/entropy?

**Spiking neurons**

What is the information content?

**Complexity for Neural Networks**



**Use entropy to define this!**

**Nontrivial in high dimension!**

$\longrightarrow$ **New method**

## Shannon entropy

Suppose that we have an $N$-dimensional binary signal $\{x_i\}_{i=1..N}$ coming from a probability distribution $p(x_1, x_2, \ldots, x_N)$.

The Shannon entropy is defined by

**Key difficulty:** Estimate $p(x_1, x_2, \ldots, x_N)$ from a set of $n$ samples...

- Count the number of occurrencies $k$ of a configuration $\{x_i\}_{i=1..N}$
- Estimate the probability of this configuration as

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

- There are various refinements taking into account finite $n$ effects: Miller-Madow, Chao-Shen, Grassberger, James-Stein, Nemenman-Shafee-Bialek (NSB)

## Shannon entropy

Suppose that we have an $N$-dimensional binary signal $\{x_i\}_{i=1..N}$ coming from a probability distribution $p(x_1, x_2, \ldots, x_N)$.

The Shannon entropy is defined by

**Key difficulty:** Estimate $p(x_1, x_2, \ldots, x_N)$ from a set of $n$ samples...

- Count the number of occurrencies $k$ of a configuration $\{x_i\}_{i=1..N}$
- Estimate the probability of this configuration as

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

- There are various refinements taking into account finite $n$ effects: Miller-Madow, Chao-Shen, Grassberger, James-Stein, Nemenman-Shafee-Bialek (NSB)

## Shannon entropy

Suppose that we have an $N$-dimensional binary signal $\{x_i\}_{i=1..N}$ coming from a probability distribution $p(x_1, x_2, \ldots, x_N)$.

The Shannon entropy is defined by

$$S = -\sum_{\{x_i\}_{i=1..N}} p(x_1, x_2, \ldots, x_N) \log_2 p(x_1, x_2, \ldots, x_N)$$

**Key difficulty:** Estimate $p(x_1, x_2, \ldots, x_N)$ from a set of $n$ samples...

- Count the number of occurrencies $k$ of a configuration $\{x_i\}_{i=1..N}$
- Estimate the probability of this configuration as

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

- There are various refinements taking into account finite $n$ effects: Miller-Madow, Chao-Shen, Grassberger, James-Stein, Nemenman-Shafee-Bialek (NSB)

## Shannon entropy

Suppose that we have an $N$-dimensional binary signal $\{x_i\}_{i=1..N}$ coming from a probability distribution $p(x_1, x_2, \ldots, x_N)$.

The Shannon entropy is defined by

$$S = -\sum_{\{x_i\}_{i=1..N}} p(x_1, x_2, \ldots, x_N) \log_2 p(x_1, x_2, \ldots, x_N)$$

**Key difficulty:** Estimate $p(x_1, x_2, \ldots, x_N)$ from a set of $n$ samples...

- Count the number of occurrencies $k$ of a configuration $\{x_i\}_{i=1..N}$
- Estimate the probability of this configuration as

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

- There are various refinements taking into account finite $n$ effects: Miller-Madow, Chao-Shen, Grassberger, James-Stein, Nemenman-Shafee-Bialek (NSB)

## Shannon entropy

Suppose that we have an $N$-dimensional binary signal $\{x_i\}_{i=1..N}$ coming from a probability distribution $p(x_1, x_2, \ldots, x_N)$.

The Shannon entropy is defined by

$$S = -\langle \log_2 p(x_1, x_2, \ldots, x_N) \rangle \quad \equiv -\frac{1}{n}\sum_{k=1}^{n} \log_2 p(x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)})$$

**Key difficulty:** Estimate $p(x_1, x_2, \ldots, x_N)$ from a set of $n$ samples...

- Count the number of occurrencies $k$ of a configuration $\{x_i\}_{i=1..N}$
- Estimate the probability of this configuration as

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

- There are various refinements taking into account finite $n$ effects: Miller-Madow, Chao-Shen, Grassberger, James-Stein, Nemenman-Shafee-Bialek (NSB)

## Shannon entropy

Suppose that we have an $N$-dimensional binary signal $\{x_i\}_{i=1..N}$ coming from a probability distribution $p(x_1, x_2, \ldots, x_N)$.

The Shannon entropy is defined by

$$S = -\langle \log_2 p(x_1, x_2, \ldots, x_N) \rangle \quad \equiv - \tfrac{1}{n} \textstyle\sum_{k=1}^{n} \log_2 p(x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)})$$

**Key difficulty:** Estimate $p(x_1, x_2, \ldots, x_N)$ from a set of $n$ samples...

- Count the number of occurrencies $k$ of a configuration $\{x_i\}_{i=1..N}$
- Estimate the probability of this configuration as

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

- There are various refinements taking into account finite $n$ effects: Miller-Madow, Chao-Shen, Grassberger, James-Stein, Nemenman-Shafee-Bialek (NSB)

## Shannon entropy

Suppose that we have an $N$-dimensional binary signal $\{x_i\}_{i=1..N}$ coming from a probability distribution $p(x_1, x_2, \ldots, x_N)$.

The Shannon entropy is defined by

$$S = -\langle \log_2 p(x_1, x_2, \ldots, x_N) \rangle \quad \equiv -\frac{1}{n}\sum_{k=1}^{n} \log_2 p(x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)})$$

**Key difficulty:** Estimate $p(x_1, x_2, \ldots, x_N)$ from a set of $n$ samples...

- Count the number of occurrencies $k$ of a configuration $\{x_i\}_{i=1..N}$
- Estimate the probability of this configuration as

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

- There are various refinements taking into account finite $n$ effects: Miller-Madow, Chao-Shen, Grassberger, James-Stein, Nemenman-Shafee-Bialek (NSB)

## Shannon entropy

Suppose that we have an $N$-dimensional binary signal $\{x_i\}_{i=1..N}$ coming from a probability distribution $p(x_1, x_2, \ldots, x_N)$.

The Shannon entropy is defined by

$$S = -\langle \log_2 p(x_1, x_2, \ldots, x_N) \rangle \quad \equiv -\frac{1}{n}\sum_{k=1}^{n} \log_2 p(x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)})$$

**Key difficulty:** Estimate $p(x_1, x_2, \ldots, x_N)$ from a set of $n$ samples...

- Count the number of occurrencies $k$ of a configuration $\{x_i\}_{i=1..N}$
- Estimate the probability of this configuration as

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

- There are various refinements taking into account finite $n$ effects: Miller-Madow, Chao-Shen, Grassberger, James-Stein, Nemenman-Shafee-Bialek (NSB)

## Shannon entropy

Suppose that we have an $N$-dimensional binary signal $\{x_i\}_{i=1..N}$ coming from a probability distribution $p(x_1, x_2, \ldots, x_N)$.

The Shannon entropy is defined by

$$S = -\langle \log_2 p(x_1, x_2, \ldots, x_N) \rangle \quad \equiv -\frac{1}{n}\sum_{k=1}^{n} \log_2 p(x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)})$$

**Key difficulty:** Estimate $p(x_1, x_2, \ldots, x_N)$ from a set of $n$ samples...

- Count the number of occurrencies $k$ of a configuration $\{x_i\}_{i=1..N}$
- Estimate the probability of this configuration as

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

- There are various refinements taking into account finite $n$ effects: Miller-Madow, Chao-Shen, Grassberger, James-Stein, Nemenman-Shafee-Bialek (NSB)

## Shannon entropy

Suppose that we have an $N$-dimensional binary signal $\{x_i\}_{i=1..N}$ coming from a probability distribution $p(x_1, x_2, \ldots, x_N)$.

The Shannon entropy is defined by

$$S = -\langle \log_2 p(x_1, x_2, \ldots, x_N) \rangle \quad \equiv -\tfrac{1}{n} \textstyle\sum_{k=1}^{n} \log_2 p(x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)})$$

**Key difficulty:** Estimate $p(x_1, x_2, \ldots, x_N)$ from a set of $n$ samples...

- Count the number of occurrencies $k$ of a configuration $\{x_i\}_{i=1..N}$
- Estimate the probability of this configuration as

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

- There are various refinements taking into account finite $n$ effects: Miller-Madow, Chao-Shen, Grassberger, James-Stein, Nemenman-Shafee-Bialek (NSB)

**Problems:**

- As we increase the dimensionality $N$, the number of possible configurations grows as $2^N$...
- Then it is quite probable that each configuration occurs at most once in the given data sample...
- This is the generic situation in Physics simulations
  e.g. a $20 \times 20$ 2D Ising model has $2^{400} \sim 10^{120}$ configurations
- All configurations are indeed distinct within the 20000 MC configurations of the above Ising model at $T \geq 2.7$

**We cannot use the above ways of evaluating entropy based on occurrence counts...**

**Problems:**

- As we increase the dimensionality $N$, the number of possible configurations grows as $2^N$...
- Then it is quite probable that each configuration occurs at most once in the given data sample...
- This is the generic situation in Physics simulations e.g. a $20 \times 20$ 2D Ising model has $2^{400} \sim 10^{120}$ configurations
- All configurations are indeed distinct within the 20000 MC configurations of the above Ising model at $T \geq 2.7$

**We cannot use the above ways of evaluating entropy based on occurrence counts...**

**Problems:**

- As we increase the dimensionality $N$, the number of possible configurations grows as $2^N$...
- Then it is quite probable that each configuration occurs at most once in the given data sample...
- This is the generic situation in Physics simulations e.g. a $20 \times 20$ 2D Ising model has $2^{400} \sim 10^{120}$ configurations
- All configurations are indeed distinct within the 20000 MC configurations of the above Ising model at $T \geq 2.7$

**We cannot use the above ways of evaluating entropy based on occurrence counts...**

**Problems:**

- As we increase the dimensionality $N$, the number of possible configurations grows as $2^N$...
- Then it is quite probable that each configuration occurs at most once in the given data sample...
- This is the generic situation in Physics simulations
  e.g. a $20 \times 20$ 2D Ising model has $2^{400} \sim 10^{120}$ configurations
- All configurations are indeed distinct within the 20000 MC configurations of the above Ising model at $T \geq 2.7$

**We cannot use the above ways of evaluating entropy based on occurrence counts...**

**Problems:**

- As we increase the dimensionality $N$, the number of possible configurations grows as $2^N$...
- Then it is quite probable that each configuration occurs at most once in the given data sample...
- This is the generic situation in Physics simulations
e.g. a $20 \times 20$ 2D Ising model has $2^{400} \sim 10^{120}$ configurations
- All configurations are indeed distinct within the 20000 MC configurations of the above Ising model at $T \geq 2.7$

**We cannot use the above ways of evaluating entropy based on occurrence counts...**

**Problems:**

- As we increase the dimensionality $N$, the number of possible configurations grows as $2^N$...
- Then it is quite probable that each configuration occurs at most once in the given data sample...
- This is the generic situation in Physics simulations
  e.g. a $20 \times 20$ 2D Ising model has $2^{400} \sim 10^{120}$ configurations
- All configurations are indeed distinct within the 20000 MC configurations of the above Ising model at $T \geq 2.7$

**We cannot use the above ways of evaluating entropy based on occurrence counts...**

## Entropy in Physics

- In Physics we have the additional structure of a Boltzmann distribution:

$$p_{Boltzmann}(x_1, x_2, \ldots, x_N) = \frac{1}{Z} e^{-\frac{1}{T} E(x_1, x_2, \ldots, x_N)}$$

- $E(x_1, x_2, \ldots, x_N)$ is typically easy to evaluate...
- ... but in $\log p(x_1, x_2, \ldots, x_N)$ we have $\log Z$ which involves a summation over all configurations...
- very difficult (if not impossible) to evaluate
- Indeed entropy is equivalent to the free energy through

$$F = -T \log Z = \langle E \rangle - TS$$

**Use indirect methods to compute the entropy...**

## Entropy in Physics

- In Physics we have the additional structure of a Boltzmann distribution:

$$p_{Boltzmann}(x_1, x_2, \ldots, x_N) = \frac{1}{Z} e^{-\frac{1}{T} E(x_1, x_2, \ldots, x_N)}$$

- $E(x_1, x_2, \ldots, x_N)$ is typically easy to evaluate...
- ... but in $\log p(x_1, x_2, \ldots, x_N)$ we have $\log Z$ which involves a summation over all configurations...
- very difficult (if not impossible) to evaluate
- Indeed entropy is equivalent to the free energy through

$$F = -T \log Z = \langle E \rangle - TS$$

**Use indirect methods to compute the entropy...**

# Entropy in Physics

- In Physics we have the additional structure of a Boltzmann distribution:

$$p_{Boltzmann}(x_1, x_2, \ldots, x_N) = \frac{1}{Z} e^{-\frac{1}{T} E(x_1, x_2, \ldots, x_N)}$$

- $E(x_1, x_2, \ldots, x_N)$ is typically easy to evaluate...
- ... but in $\log p(x_1, x_2, \ldots, x_N)$ we have $\log Z$ which involves a summation over all configurations...
- very difficult (if not impossible) to evaluate
- Indeed entropy is equivalent to the free energy through

$$F = -T \log Z = \langle E \rangle - TS$$

**Use indirect methods to compute the entropy...**

## Entropy in Physics

- In Physics we have the additional structure of a Boltzmann distribution:

$$p_{Boltzmann}(x_1, x_2, \ldots, x_N) = \frac{1}{Z} e^{-\frac{1}{T} E(x_1, x_2, \ldots, x_N)}$$

- $E(x_1, x_2, \ldots, x_N)$ is typically easy to evaluate...
- ... but in $\log p(x_1, x_2, \ldots, x_N)$ we have $\log Z$ which involves a summation over all configurations...
- very difficult (if not impossible) to evaluate
- Indeed entropy is equivalent to the free energy through

$$F = -T \log Z = \langle E \rangle - TS$$

**Use indirect methods to compute the entropy...**

## Entropy in Physics

- In Physics we have the additional structure of a Boltzmann distribution:

$$p_{Boltzmann}(x_1, x_2, \ldots, x_N) = \frac{1}{Z} e^{-\frac{1}{T} E(x_1, x_2, \ldots, x_N)}$$

- $E(x_1, x_2, \ldots, x_N)$ is typically easy to evaluate...
- ... but in $\log p(x_1, x_2, \ldots, x_N)$ we have $\log Z$ which involves a summation over all configurations...
- very difficult (if not impossible) to evaluate
- Indeed entropy is equivalent to the free energy through

$$F = -T \log Z = \langle E \rangle - TS$$

**Use indirect methods to compute the entropy...**

## Entropy in Physics

- In Physics we have the additional structure of a Boltzmann distribution:

$$p_{Boltzmann}(x_1, x_2, \ldots, x_N) = \frac{1}{Z} e^{-\frac{1}{T} E(x_1, x_2, \ldots, x_N)}$$

- $E(x_1, x_2, \ldots, x_N)$ is typically easy to evaluate...
- ... but in $\log p(x_1, x_2, \ldots, x_N)$ we have $\log Z$ which involves a summation over all configurations...
- very difficult (if not impossible) to evaluate
- Indeed entropy is equivalent to the free energy through

$$F = -T \log Z = \langle E \rangle - TS$$

**Use indirect methods to compute the entropy...**

## Entropy in Physics

- In Physics we have the additional structure of a Boltzmann distribution:

$$p_{Boltzmann}(x_1, x_2, \ldots, x_N) = \frac{1}{Z} e^{-\frac{1}{T} E(x_1, x_2, \ldots, x_N)}$$

- $E(x_1, x_2, \ldots, x_N)$ is typically easy to evaluate...
- ... but in $\log p(x_1, x_2, \ldots, x_N)$ we have $\log Z$ which involves a summation over all configurations...
- very difficult (if not impossible) to evaluate
- Indeed entropy is equivalent to the free energy through

$$F = -T \log Z = \langle E \rangle - TS$$

**Use indirect methods to compute the entropy...**

## Temperature integration

- Evaluate the heat capacity $C(T)$ from the variance of the energy $\sigma_E^2(T)$

$$C(T) = \frac{\partial \langle E \rangle}{\partial T} = \frac{\sigma_E^2(T)}{T^2}$$

- Obtain the entropy by a numerical integration over $T$:

$$S(T_0) = \int_0^{T_0} C(T) \frac{dT}{T}$$

- Need to perform separate Monte Carlo simulations for the whole range of temperatures from $T = 0$ to $T = T_0$

## Wang-Landau sampling

Wang, Landau '01

- Does not use the original Monte Carlo configurations at all..
- Need to perform conceptually different Monte Carlo sampling to evaluate the density of states...

## Temperature integration

- Evaluate the heat capacity $C(T)$ from the variance of the energy $\sigma_E^2(T)$

$$C(T) = \frac{\partial \langle E \rangle}{\partial T} = \frac{\sigma_E^2(T)}{T^2}$$

- Obtain the entropy by a numerical integration over $T$:

$$S(T_0) = \int_0^{T_0} C(T) \frac{dT}{T}$$

- Need to perform separate Monte Carlo simulations for the whole range of temperatures from $T = 0$ to $T = T_0$

## Wang-Landau sampling

Wang, Landau '01

- Does not use the original Monte Carlo configurations at all..
- Need to perform conceptually different Monte Carlo sampling to evaluate the density of states...

## Temperature integration

- Evaluate the heat capacity $C(T)$ from the variance of the energy $\sigma_E^2(T)$

$$C(T) = \frac{\partial \langle E \rangle}{\partial T} = \frac{\sigma_E^2(T)}{T^2}$$

- Obtain the entropy by a numerical integration over $T$:

$$S(T_0) = \int_0^{T_0} C(T) \frac{dT}{T}$$

- Need to perform separate Monte Carlo simulations for the whole range of temperatures from $T = 0$ to $T = T_0$

## Wang-Landau sampling

Wang, Landau '01

- Does not use the original Monte Carlo configurations at all..
- Need to perform conceptually different Monte Carlo sampling to evaluate the density of states...

## Temperature integration

- Evaluate the heat capacity $C(T)$ from the variance of the energy $\sigma_E^2(T)$

$$C(T) = \frac{\partial \langle E \rangle}{\partial T} = \frac{\sigma_E^2(T)}{T^2}$$

- Obtain the entropy by a numerical integration over $T$:

$$S(T_0) = \int_0^{T_0} C(T) \frac{dT}{T}$$

- Need to perform separate Monte Carlo simulations for the whole range of temperatures from $T = 0$ to $T = T_0$

## Wang-Landau sampling

Wang, Landau '01

- Does not use the original Monte Carlo configurations at all..
- Need to perform conceptually different Monte Carlo sampling to evaluate the density of states...

## Temperature integration

- Evaluate the heat capacity $C(T)$ from the variance of the energy $\sigma_E^2(T)$

$$C(T) = \frac{\partial \langle E \rangle}{\partial T} = \frac{\sigma_E^2(T)}{T^2}$$

- Obtain the entropy by a numerical integration over $T$:

$$S(T_0) = \int_0^{T_0} C(T) \frac{dT}{T}$$

- Need to perform separate Monte Carlo simulations for the whole range of temperatures from $T = 0$ to $T = T_0$

## Wang-Landau sampling

Wang, Landau '01

- Does not use the original Monte Carlo configurations at all..
- Need to perform conceptually different Monte Carlo sampling to evaluate the density of states...

## Temperature integration

- Evaluate the heat capacity $C(T)$ from the variance of the energy $\sigma_E^2(T)$

$$C(T) = \frac{\partial \langle E \rangle}{\partial T} = \frac{\sigma_E^2(T)}{T^2}$$

- Obtain the entropy by a numerical integration over $T$:

$$S(T_0) = \int_0^{T_0} C(T) \frac{dT}{T}$$

- Need to perform separate Monte Carlo simulations for the whole range of temperatures from $T = 0$ to $T = T_0$

## Wang-Landau sampling

Wang, Landau '01

- Does not use the original Monte Carlo configurations at all..
- Need to perform conceptually different Monte Carlo sampling to evaluate the density of states...

## Temperature integration

- Evaluate the heat capacity $C(T)$ from the variance of the energy $\sigma_E^2(T)$

$$C(T) = \frac{\partial \langle E \rangle}{\partial T} = \frac{\sigma_E^2(T)}{T^2}$$

- Obtain the entropy by a numerical integration over $T$:

$$S(T_0) = \int_0^{T_0} C(T) \frac{dT}{T}$$

- Need to perform separate Monte Carlo simulations for the whole range of temperatures from $T = 0$ to $T = T_0$

## Wang-Landau sampling

Wang, Landau '01

- Does not use the original Monte Carlo configurations at all..
- Need to perform conceptually different Monte Carlo sampling to evaluate the density of states...

## Temperature integration

- Evaluate the heat capacity $C(T)$ from the variance of the energy $\sigma_E^2(T)$

$$C(T) = \frac{\partial \langle E \rangle}{\partial T} = \frac{\sigma_E^2(T)}{T^2}$$

- Obtain the entropy by a numerical integration over $T$:

$$S(T_0) = \int_0^{T_0} C(T) \frac{dT}{T}$$

- Need to perform separate Monte Carlo simulations for the whole range of temperatures from $T = 0$ to $T = T_0$

## Wang-Landau sampling

Wang, Landau '01

- Does not use the original Monte Carlo configurations at all..
- Need to perform conceptually different Monte Carlo sampling to evaluate the density of states...

**Goal:**

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...

## Goal:

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...

## Goal:

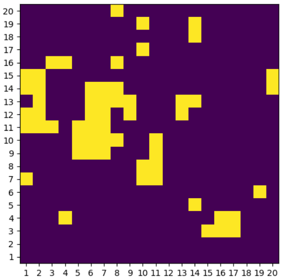
1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...

**Goal:**

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...

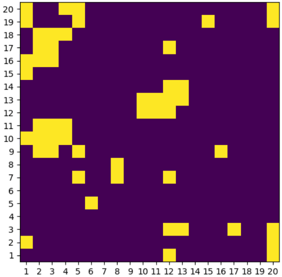
**Goal:**

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...



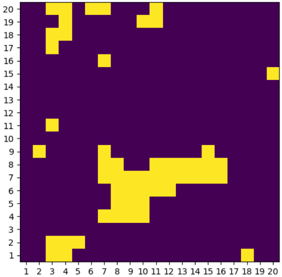
**Goal:**

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...



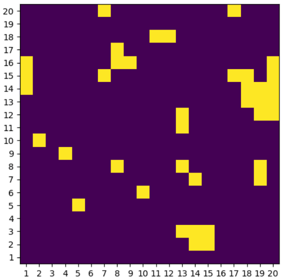
**Goal:**

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...



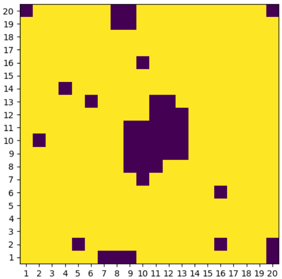
**Goal:**

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...



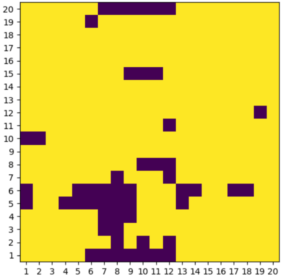
**Goal:**

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...



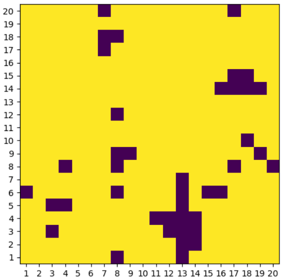
**Goal:**

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...



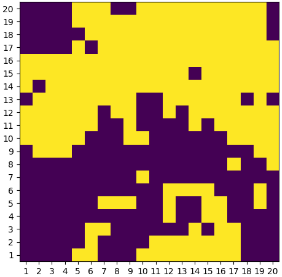
## Goal:

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...



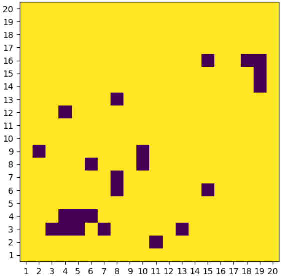
**Goal:**

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...



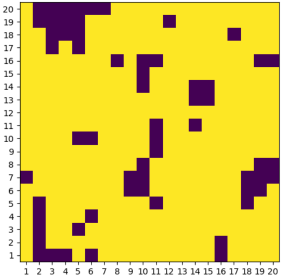
## Goal:

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...



**Goal:**

1. **Find a method for computing entropy from binary configurations which would work in the regime when each configuration appears at most only once**
2. Use it to compute the entropy and free energy directly from Monte Carlo configurations at a given temperature...




$\longrightarrow$ Entropy

**Idea:**

The formula using occurrence counts

$$S = -\sum_{\{x_i\}_{i=1..N}} p(x_1, x_2, \dots, x_N) \log_2 p(x_1, x_2, \dots, x_N)$$

with

$$p(x_1, x_2, \dots, x_N) = \frac{k}{n}$$

treats each configuration as a **structureless "atomic" object**.

**This approach:** Analyze instead **the internal structure** of the configurations...

**use Machine Learning methods...**

**Idea:**

The formula using occurrence counts

$$S = -\sum_{\{x_i\}_{i=1..N}} p(x_1, x_2, \ldots, x_N) \log_2 p(x_1, x_2, \ldots, x_N)$$

with

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

treats each configuration as a **structureless "atomic" object**.

**This approach:** Analyze instead **the internal structure** of the configurations...

**use Machine Learning methods...**

**Idea:**

The formula using occurrence counts

$$S = -\sum_{\{x_i\}_{i=1..N}} p(x_1, x_2, \ldots, x_N) \log_2 p(x_1, x_2, \ldots, x_N)$$

with

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

treats each configuration as a **structureless "atomic" object**.

**This approach:** Analyze instead **the internal structure** of the configurations...

use **Machine Learning** methods...

**Idea:**

The formula using occurrence counts

$$S = -\sum_{\{x_i\}_{i=1..N}} p(x_1, x_2, \ldots, x_N) \log_2 p(x_1, x_2, \ldots, x_N)$$

with

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

treats each configuration as a **structureless "atomic" object**.

**This approach:** Analyze instead **the internal structure** of the configurations...

use **Machine Learning** methods...

**Idea:**

The formula using occurrence counts

$$S = -\sum_{\{x_i\}_{i=1..N}} p(x_1, x_2, \ldots, x_N) \log_2 p(x_1, x_2, \ldots, x_N)$$

with

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

treats each configuration as a **structureless "atomic" object**.

**This approach:** Analyze instead **the internal structure** of the configurations...

use **Machine Learning** methods...

**Idea:**

The formula using occurrence counts

$$S = -\sum_{\{x_i\}_{i=1..N}} p(x_1, x_2, \ldots, x_N) \log_2 p(x_1, x_2, \ldots, x_N)$$

with

$$p(x_1, x_2, \ldots, x_N) = \frac{k}{n}$$

treats each configuration as a **structureless "atomic" object**.

**This approach:** Analyze instead **the internal structure** of the configurations...

**use Machine Learning methods...**

# Entropy from Machine Learning

- Start from the **exact** rewriting

$$p(x_1, x_2, \ldots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdot \ldots$$

- Shannon's entropy $\langle -log_2 p \rangle$ decomposes into a sum of $N$ terms

$$S = S_1 + \Delta S_2 + \Delta S_3 + \ldots \Delta S_N$$

- The first term is just the entropy of the first neuron/spin

$$S_1 = -\langle \log_2 p(x_1) \rangle \equiv -p_1 \log_2 p_1 - (1 - p_1) \log_2 (1 - p_1)$$

  where $p_1 \equiv p(x_1 = 1)$.
- The second term is more interesting...

$$\Delta S_2 = -\langle \log_2 p(x_2|x_1) \rangle$$

## Entropy from Machine Learning

- Start from the **exact** rewriting

$$p(x_1, x_2, \dots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdot \dots$$

- Shannon's entropy $\langle -log_2 p \rangle$ decomposes into a sum of $N$ terms

$$S = S_1 + \Delta S_2 + \Delta S_3 + \dots \Delta S_N$$

- The first term is just the entropy of the first neuron/spin

$$S_1 = -\langle \log_2 p(x_1) \rangle \equiv -p_1 \log_2 p_1 - (1 - p_1) \log_2 (1 - p_1)$$

where $p_1 \equiv p(x_1 = 1)$.

- The second term is more interesting...

$$\Delta S_2 = -\langle \log_2 p(x_2|x_1) \rangle$$

## Entropy from Machine Learning

- Start from the **exact** rewriting

$$p(x_1, x_2, \ldots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdot \ldots$$

- Shannon's entropy $\langle -log_2 p \rangle$ decomposes into a sum of $N$ terms

$$S = S_1 + \Delta S_2 + \Delta S_3 + \ldots \Delta S_N$$

- The first term is just the entropy of the first neuron/spin

$$S_1 = -\langle \log_2 p(x_1) \rangle \equiv -p_1 \log_2 p_1 - (1 - p_1) \log_2(1 - p_1)$$

where $p_1 \equiv p(x_1 = 1)$.

- The second term is more interesting...

$$\Delta S_2 = -\langle \log_2 p(x_2|x_1) \rangle$$

## Entropy from Machine Learning

- Start from the **exact** rewriting

$$p(x_1, x_2, \ldots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\cdot\ldots$$

- Shannon's entropy $\langle -log_2 p\rangle$ decomposes into a sum of $N$ terms

$$S = S_1 + \Delta S_2 + \Delta S_3 + \ldots \Delta S_N$$

- The first term is just the entropy of the first neuron/spin

$$S_1 = -\langle \log_2 p(x_1)\rangle \equiv -p_1 \log_2 p_1 - (1-p_1)\log_2(1-p_1)$$

where $p_1 \equiv p(x_1 = 1)$.

- The second term is more interesting...

$$\Delta S_2 = -\langle \log_2 p(x_2|x_1)\rangle$$

## Entropy from Machine Learning

- Start from the **exact** rewriting

$$p(x_1, x_2, \dots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdot \dots$$

- Shannon's entropy $\langle -log_2 p \rangle$ decomposes into a sum of $N$ terms

$$S = S_1 + \Delta S_2 + \Delta S_3 + \dots \Delta S_N$$

- The first term is just the entropy of the first neuron/spin

$$S_1 = -\langle \log_2 p(x_1) \rangle \equiv -p_1 \log_2 p_1 - (1 - p_1) \log_2 (1 - p_1)$$

where $p_1 \equiv p(x_1 = 1)$.

- The second term is more interesting...

$$\Delta S_2 = -\langle \log_2 p(x_2|x_1) \rangle$$

## Entropy from Machine Learning

- Start from the **exact** rewriting

$$p(x_1, x_2, \ldots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\cdot\ldots$$

- Shannon's entropy $\langle -log_2 p\rangle$ decomposes into a sum of $N$ terms

$$S = S_1 + \Delta S_2 + \Delta S_3 + \ldots \Delta S_N$$

- The first term is just the entropy of the first neuron/spin

$$S_1 = -\langle \log_2 p(x_1)\rangle \equiv -p_1 \log_2 p_1 - (1-p_1)\log_2(1-p_1)$$

where $p_1 \equiv p(x_1 = 1)$.

- The second term is more interesting...

$$\Delta S_2 = -\langle \log_2 p(x_2|x_1)\rangle$$

## Entropy from Machine Learning

$$\Delta S_2 = -\left\langle \log_2 p(x_2|x_1) \right\rangle$$

- $p(x_2|x_1)$ (and $p(x_3|x_1, x_2)$ etc.) is exactly a conditional probability distribution of the kind which is well computed by machine learning classifiers!
- It corresponds to predicting the spin $x_2$ based on the value of the spin $x_1$ interpreted as **classification problems**
- Moreover

  $$\Delta S_2 = -\left\langle \log_2 p(x_2|x_1) \right\rangle$$

  is the **cross-entropy loss** of that machine learning classification problem
- similarly for $p(x_3|x_1, x_2)$ we predict the class (spin) $x_3$ based on the data $x_1, x_2$

# Entropy from Machine Learning

$$\Delta S_2 = -\left\langle \log_2 p(x_2|x_1) \right\rangle$$

- $p(x_2|x_1)$ (and $p(x_3|x_1, x_2)$ etc.) is exactly a conditional probability distribution of the kind which is well computed by machine learning classifiers!
- It corresponds to predicting the spin $x_2$ based on the value of the spin $x_1$ interpreted as **classification problems**
- Moreover

$$\Delta S_2 = -\left\langle \log_2 p(x_2|x_1) \right\rangle$$

  is the **cross-entropy loss** of that machine learning classification problem
- similarly for $p(x_3|x_1, x_2)$ we predict the class (spin) $x_3$ based on the data $x_1, x_2$

## Entropy from Machine Learning

$$\Delta S_2 = -\langle \log_2 p(x_2|x_1) \rangle$$

- $p(x_2|x_1)$ (and $p(x_3|x_1, x_2)$ etc.) is exactly a conditional probability distribution of the kind which is well computed by machine learning classifiers!
- It corresponds to predicting the spin $x_2$ based on the value of the spin $x_1$ interpreted as **classification problems**
- Moreover

$$\Delta S_2 = -\langle \log_2 p(x_2|x_1) \rangle$$

  is the **cross-entropy loss** of that machine learning classification problem
- similarly for $p(x_3|x_1, x_2)$ we predict the class (spin) $x_3$ based on the data $x_1, x_2$

## Entropy from Machine Learning

$$\Delta S_2 = -\langle \log_2 p(x_2|x_1) \rangle$$

- $p(x_2|x_1)$ (and $p(x_3|x_1, x_2)$ etc.) is exactly a conditional probability distribution of the kind which is well computed by machine learning classifiers!
- It corresponds to predicting the spin $x_2$ based on the value of the spin $x_1$ interpreted as **classification problems**
- Moreover
$$\Delta S_2 = -\langle \log_2 p(x_2|x_1) \rangle$$
is the **cross-entropy loss** of that machine learning classification problem
- similarly for $p(x_3|x_1, x_2)$ we predict the class (spin) $x_3$ based on the data $x_1, x_2$

## Entropy from Machine Learning

$$\Delta S_2 = -\langle \log_2 p(x_2|x_1) \rangle$$

- $p(x_2|x_1)$ (and $p(x_3|x_1, x_2)$ etc.) is exactly a conditional probability distribution of the kind which is well computed by machine learning classifiers!
- It corresponds to predicting the spin $x_2$ based on the value of the spin $x_1$ interpreted as **classification problems**
- Moreover

  $$\Delta S_2 = -\langle \log_2 p(x_2|x_1) \rangle$$

  is the **cross-entropy loss** of that machine learning classification problem
- similarly for $p(x_3|x_1, x_2)$ we predict the class (spin) $x_3$ based on the data $x_1$, $x_2$

# Entropy from Machine Learning

## The resulting prescription

- An estimate of the entropy is given by

$$S = S_1 + \Delta S_2 + \Delta S_3 + \ldots \Delta S_N$$

- This is a sum of cross-entropy losses of a sequence of supervised classification problems where we predict the probability of $x_j = 1$ given the values of the previous spins $x_1, x_2, \ldots, x_{j-1}$.
- One can use **any** machine learning classification algorithm to compute the entropy

## The resulting prescription

- An estimate of the entropy is given by

$$S = S_1 + \Delta S_2 + \Delta S_3 + \ldots \Delta S_N$$

- This is a sum of cross-entropy losses of a sequence of supervised classification problems where we predict the probability of $x_j = 1$ given the values of the previous spins $x_1, x_2, \ldots, x_{j-1}$.
- One can use **any** machine learning classification algorithm to compute the entropy

## The resulting prescription

- An estimate of the entropy is given by

$$S = S_1 + \Delta S_2 + \Delta S_3 + \ldots \Delta S_N$$

- This is a sum of cross-entropy losses of a sequence of supervised classification problems where we predict the probability of $x_j = 1$ given the values of the previous spins $x_1, x_2, \ldots, x_{j-1}$.
- One can use **any** machine learning classification algorithm to compute the entropy

## The resulting prescription

- An estimate of the entropy is given by

$$S = S_1 + \Delta S_2 + \Delta S_3 + \ldots \Delta S_N$$

- This is a sum of cross-entropy losses of a sequence of supervised classification problems where we predict the probability of $x_j = 1$ given the values of the previous spins $x_1, x_2, \ldots, x_{j-1}$.
- One can use **any** machine learning classification algorithm to compute the entropy

## The resulting prescription

- An estimate of the entropy is given by

$$S = S_1 + \Delta S_2 + \Delta S_3 + \ldots \Delta S_N$$

- This is a sum of cross-entropy losses of a sequence of supervised classification problems where we predict the probability of $x_j = 1$ given the values of the previous spins $x_1, x_2, \ldots, x_{j-1}$.
- One can use **any** machine learning classification algorithm to compute the entropy

## Variational interpretation

- Given enough data, it is easy to see that the **true** entropy is bounded from above by the **Machine Learning** estimate

$$S_{true} = -\langle \log_2 p_{true} \rangle_{true} \leq -\langle \log_2 \underbrace{p_{ML}}_{p_{trial}} \rangle_{true}$$

- One can therefore consider this as a **variational bound** for entropy...
- ... but with **very nontrivial** trial functions coming from Machine Learning classifiers...
- One can try various ML classifiers and pick the best...
- Comparison of estimates from linear (*logistic regression*) and nonlinear ML classifiers gives novel kind of information on the complexity of data...

## Variational interpretation

- Given enough data, it is easy to see that the **true** entropy is bounded from above by the **Machine Learning** estimate

$$S_{true} = -\langle \log_2 p_{true} \rangle_{true} \leq -\langle \log_2 \underbrace{p_{ML}}_{p_{trial}} \rangle_{true}$$

- One can therefore consider this as a **variational bound** for entropy...
- ... but with **very nontrivial** trial functions coming from Machine Learning classifiers...
- One can try various ML classifiers and pick the best...
- Comparison of estimates from linear (*logistic regression*) and nonlinear ML classifiers gives novel kind of information on the complexity of data...

## Variational interpretation

- Given enough data, it is easy to see that the **true** entropy is bounded from above by the **Machine Learning** estimate

$$S_{true} = -\left\langle \log_2 p_{true} \right\rangle_{true} \leq -\langle \log_2 \underbrace{p_{ML}}_{p_{trial}} \rangle_{true}$$

- One can therefore consider this as a **variational bound** for entropy...
- ... but with **very nontrivial** trial functions coming from Machine Learning classifiers...
- One can try various ML classifiers and pick the best...
- Comparison of estimates from linear (*logistic regression*) and nonlinear ML classifiers gives novel kind of information on the complexity of data...

## Variational interpretation

- Given enough data, it is easy to see that the **true** entropy is bounded from above by the **Machine Learning** estimate

$$S_{true} = -\left\langle \log_2 p_{true} \right\rangle_{true} \leq -\langle \log_2 \underbrace{p_{ML}}_{p_{trial}} \rangle_{true}$$

- One can therefore consider this as a **variational bound** for entropy...
- ... but with **very nontrivial** trial functions coming from Machine Learning classifiers...
- One can try various ML classifiers and pick the best...
- Comparison of estimates from linear (*logistic regression*) and nonlinear ML classifiers gives novel kind of information on the complexity of data...

## Variational interpretation

- Given enough data, it is easy to see that the **true** entropy is bounded from above by the **Machine Learning** estimate

$$S_{true} = -\langle \log_2 p_{true} \rangle_{true} \leq -\langle \log_2 \underbrace{p_{ML}}_{p_{trial}} \rangle_{true}$$

- One can therefore consider this as a **variational bound** for entropy...
- ... but with **very nontrivial** trial functions coming from Machine Learning classifiers...
- One can try various ML classifiers and pick the best...
- Comparison of estimates from linear (*logistic regression*) and nonlinear ML classifiers gives novel kind of information on the complexity of data...

## Variational interpretation

- Given enough data, it is easy to see that the **true** entropy is bounded from above by the **Machine Learning** estimate

$$S_{true} = -\langle \log_2 p_{true} \rangle_{true} \leq -\langle \log_2 \underbrace{p_{ML}}_{p_{trial}} \rangle_{true}$$

- One can therefore consider this as a **variational bound** for entropy...
- ... but with **very nontrivial** trial functions coming from Machine Learning classifiers...
- One can try various ML classifiers and pick the best...
- Comparison of estimates from linear (*logistic regression*) and nonlinear ML classifiers gives novel kind of information on the complexity of data...

## Further remarks

- The specific classification problems depend on the ordering of the spins $x_1, x_2, \ldots, x_N$
- However, the final answer (sum of the cross-entropies) should be independent of the ordering
- This is a nontrivial cross-check of the applicability of a particular machine learning classification algorithm

- One should never evaluate ML models (to get probabilities) on the data which were used for training
- A standard way to sidestep this issue is to use $k$-fold cross-validation

from scikit-learn docs

- ... and use only the predictions on the test folds...
- refinement: shuffle the data differently for each classification problem

## Further remarks

- The specific classification problems depend on the ordering of the spins $x_1, x_2, \ldots, x_N$
- However, the final answer (sum of the cross-entropies) should be independent of the ordering
- This is a nontrivial cross-check of the applicability of a particular machine learning classification algorithm

- One should never evaluate ML models (to get probabilities) on the data which were used for training
- A standard way to sidestep this issue is to use $k$-fold cross-validation

from scikit-learn docs

- ... and use only the predictions on the test folds...
- refinement: shuffle the data differently for each classification problem

## Further remarks

- The specific classification problems depend on the ordering of the spins $x_1, x_2, \ldots, x_N$
- However, the final answer (sum of the cross-entropies) should be independent of the ordering
- This is a nontrivial cross-check of the applicability of a particular machine learning classification algorithm

- One should never evaluate ML models (to get probabilities) on the data which were used for training
- A standard way to sidestep this issue is to use $k$-fold cross-validation

from scikit-learn docs

- ... and use only the predictions on the test folds...
- refinement: shuffle the data differently for each classification problem

## Further remarks

- The specific classification problems depend on the ordering of the spins $x_1, x_2, \ldots, x_N$
- However, the final answer (sum of the cross-entropies) should be independent of the ordering
- This is a nontrivial cross-check of the applicability of a particular machine learning classification algorithm

- One should never evaluate ML models (to get probabilities) on the data which were used for training
- A standard way to sidestep this issue is to use $k$-fold cross-validation

from scikit-learn docs

- ... and use only the predictions on the test folds...
- refinement: shuffle the data differently for each classification problem

## Further remarks

- The specific classification problems depend on the ordering of the spins $x_1, x_2, \ldots, x_N$
- However, the final answer (sum of the cross-entropies) should be independent of the ordering
- This is a nontrivial cross-check of the applicability of a particular machine learning classification algorithm

- One should never evaluate ML models (to get probabilities) on the data which were used for training
- A standard way to sidestep this issue is to use $k$-fold cross-validation

from scikit-learn docs

- ... and use only the predictions on the test folds...
- refinement: shuffle the data differently for each classification problem

## Further remarks

- The specific classification problems depend on the ordering of the spins $x_1, x_2, \ldots, x_N$
- However, the final answer (sum of the cross-entropies) should be independent of the ordering
- This is a nontrivial cross-check of the applicability of a particular machine learning classification algorithm

- One should never evaluate ML models (to get probabilities) on the data which were used for training
- A standard way to sidestep this issue is to use $k$-fold cross-validation

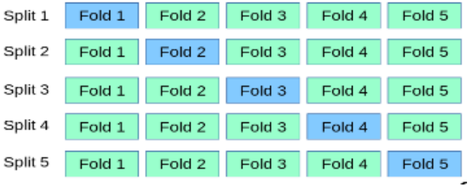
from scikit-learn docs

- ... and use only the predictions on the test folds...
- refinement: shuffle the data differently for each classification problem

## Further remarks

- The specific classification problems depend on the ordering of the spins $x_1, x_2, \ldots, x_N$
- However, the final answer (sum of the cross-entropies) should be independent of the ordering
- This is a nontrivial cross-check of the applicability of a particular machine learning classification algorithm

- One should never evaluate ML models (to get probabilities) on the data which were used for training
- A standard way to sidestep this issue is to use $k$-fold cross-validation



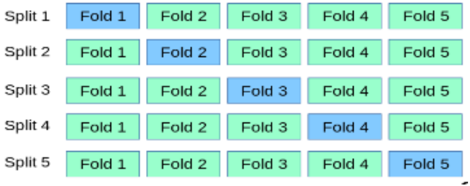

from scikit-learn docs

- ... and use only the predictions on the test folds...
- refinement: shuffle the data differently for each classification problem

## Further remarks

- The specific classification problems depend on the ordering of the spins $x_1, x_2, \ldots, x_N$
- However, the final answer (sum of the cross-entropies) should be independent of the ordering
- This is a nontrivial cross-check of the applicability of a particular machine learning classification algorithm

- One should never evaluate ML models (to get probabilities) on the data which were used for training
- A standard way to sidestep this issue is to use $k$-fold cross-validation



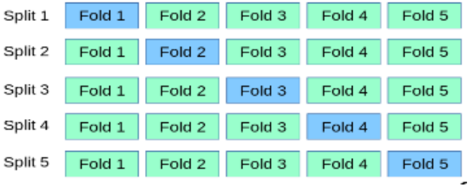

from scikit-learn docs

- ... and use only the predictions on the test folds...
- refinement: shuffle the data differently for each classification problem

## Further remarks

- The specific classification problems depend on the ordering of the spins $x_1, x_2, \ldots, x_N$
- However, the final answer (sum of the cross-entropies) should be independent of the ordering
- This is a nontrivial cross-check of the applicability of a particular machine learning classification algorithm

- One should never evaluate ML models (to get probabilities) on the data which were used for training
- A standard way to sidestep this issue is to use $k$-fold cross-validation




from scikit-learn docs

- ... and use only the predictions on the test folds...
- refinement: shuffle the data differently for each classification problem

## Example

### 2D Ising model:

Evaluate the entropy and free energy from Monte Carlo configurations of the 2D Ising model

- Has two phases and a critical point in between...
- There is a generalization of the Onsager solution to an exact solution on a periodic $L \times L$ lattice due to Kaufman
- At a given temperature, we generate 20000 configurations of the 2D Ising model on a $20 \times 20$ lattice
- We consider a range of temperatures $T = 1.0$ to $T = 4.0$ ($T_c \sim 2.269...$) covering both phases
- There are $\sim 10^{120}$ possible configurations, for $T \geq 2.7$ the Monte Carlo samples involve only distinct configurations...
- We choose a **random** ordering of the lattice sites for the classification problems

# Example

## 2D Ising model:

Evaluate the entropy and free energy from Monte Carlo configurations of the 2D Ising model

- Has two phases and a critical point in between...
- There is a generalization of the Onsager solution to an exact solution on a periodic $L \times L$ lattice due to Kaufman
- At a given temperature, we generate 20000 configurations of the 2D Ising model on a $20 \times 20$ lattice
- We consider a range of temperatures $T = 1.0$ to $T = 4.0$ ($T_c \sim 2.269...$) covering both phases
- There are $\sim 10^{120}$ possible configurations, for $T \geq 2.7$ the Monte Carlo samples involve only distinct configurations...
- We choose a **random** ordering of the lattice sites for the classification problems

## Example

### 2D Ising model:

Evaluate the entropy and free energy from Monte Carlo configurations of the 2D Ising model

- Has two phases and a critical point in between...
- There is a generalization of the Onsager solution to an exact solution on a periodic $L \times L$ lattice due to Kaufman
- At a given temperature, we generate $20000$ configurations of the 2D Ising model on a $20 \times 20$ lattice
- We consider a range of temperatures $T = 1.0$ to $T = 4.0$ ($T_c \sim 2.269...$) covering both phases
- There are $\sim 10^{120}$ possible configurations, for $T \geq 2.7$ the Monte Carlo samples involve only distinct configurations...
- We choose a **random** ordering of the lattice sites for the classification problems

## Example

### 2D Ising model:

Evaluate the entropy and free energy from Monte Carlo configurations of the 2D Ising model

- Has two phases and a critical point in between...
- There is a generalization of the Onsager solution to an exact solution on a periodic $L \times L$ lattice due to Kaufman
- At a given temperature, we generate 20000 configurations of the 2D Ising model on a $20 \times 20$ lattice
- We consider a range of temperatures $T = 1.0$ to $T = 4.0$ ($T_c \sim 2.269...$) covering both phases
- There are $\sim 10^{120}$ possible configurations, for $T \geq 2.7$ the Monte Carlo samples involve only distinct configurations...
- We choose a **random** ordering of the lattice sites for the classification problems

## Example

### 2D Ising model:

Evaluate the entropy and free energy from Monte Carlo configurations of the 2D Ising model

- Has two phases and a critical point in between...
- There is a generalization of the Onsager solution to an exact solution on a periodic $L \times L$ lattice due to Kaufman
- At a given temperature, we generate 20000 configurations of the 2D Ising model on a $20 \times 20$ lattice
- We consider a range of temperatures $T = 1.0$ to $T = 4.0$ ($T_c \sim 2.269...$) covering both phases
- There are $\sim 10^{120}$ possible configurations, for $T \geq 2.7$ the Monte Carlo samples involve only distinct configurations...
- We choose a **random** ordering of the lattice sites for the classification problems

## Example

### 2D Ising model:

Evaluate the entropy and free energy from Monte Carlo configurations of the 2D Ising model

- Has two phases and a critical point in between...
- There is a generalization of the Onsager solution to an exact solution on a periodic $L \times L$ lattice due to Kaufman
- At a given temperature, we generate 20000 configurations of the 2D Ising model on a $20 \times 20$ lattice
- We consider a range of temperatures $T = 1.0$ to $T = 4.0$ ($T_c \sim 2.269...$) covering both phases
- There are $\sim 10^{120}$ possible configurations, for $T \geq 2.7$ the Monte Carlo samples involve only distinct configurations...
- We choose a **random** ordering of the lattice sites for the classification problems

## Example

### 2D Ising model:

Evaluate the entropy and free energy from Monte Carlo configurations of the 2D Ising model

- Has two phases and a critical point in between...
- There is a generalization of the Onsager solution to an exact solution on a periodic $L \times L$ lattice due to Kaufman
- At a given temperature, we generate 20000 configurations of the 2D Ising model on a $20 \times 20$ lattice
- We consider a range of temperatures $T = 1.0$ to $T = 4.0$ ($T_c \sim 2.269...$) covering both phases
- There are $\sim 10^{120}$ possible configurations, for $T \geq 2.7$ the Monte Carlo samples involve only distinct configurations...
- We choose a **random** ordering of the lattice sites for the classification problems

# Example

## 2D Ising model:

Evaluate the entropy and free energy from Monte Carlo configurations of the 2D Ising model

- Has two phases and a critical point in between...
- There is a generalization of the Onsager solution to an exact solution on a periodic $L \times L$ lattice due to Kaufman
- At a given temperature, we generate 20000 configurations of the 2D Ising model on a $20 \times 20$ lattice
- We consider a range of temperatures $T = 1.0$ to $T = 4.0$ ($T_c \sim 2.269...$) covering both phases
- There are $\sim 10^{120}$ possible configurations, for $T \geq 2.7$ the Monte Carlo samples involve only distinct configurations...
- We choose a **random** ordering of the lattice sites for the classification problems

## Example

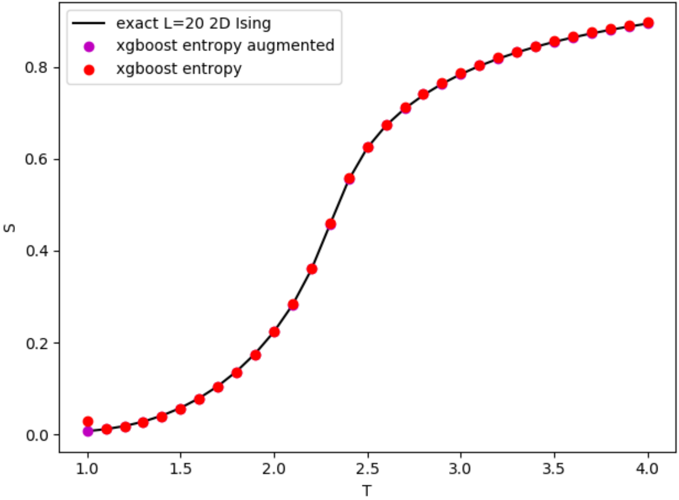
### 2D Ising model:

Evaluate the entropy and free energy from Monte Carlo configurations of the 2D Ising model

- Has two phases and a critical point in between...
- There is a generalization of the Onsager solution to an exact solution on a periodic $L \times L$ lattice due to Kaufman
- At a given temperature, we generate 20000 configurations of the 2D Ising model on a $20 \times 20$ lattice
- We consider a range of temperatures $T = 1.0$ to $T = 4.0$ ($T_c \sim 2.269\ldots$) covering both phases
- There are $\sim 10^{120}$ possible configurations, for $T \geq 2.7$ the Monte Carlo samples involve only distinct configurations...
- We choose a **random** ordering of the lattice sites for the classification problems

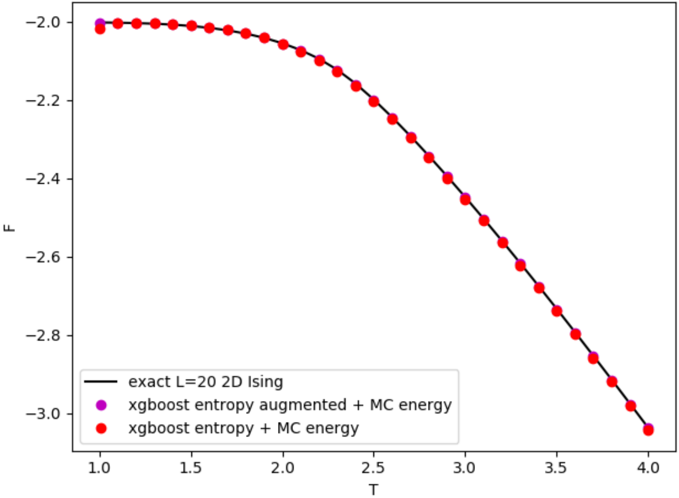
# 2D Ising model

## Entropy (per spin)



## 2D Ising model

**Free energy (per spin)** — from $F = \langle E \rangle - TS$



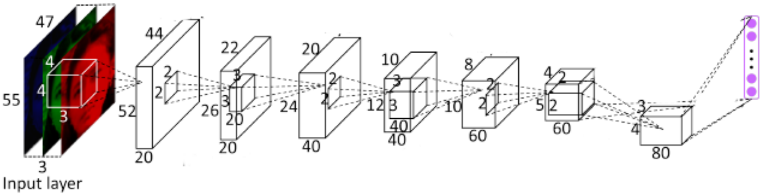
## From Physics back to Machine Learning...

- Contemporary DCNN are much deeper $> 50$ layers...
- Going further (deeper) into the network integrates features from wider range of scales...
- The outputs of interior layers are some (nonlocal) feature representations...
- ... which become more and more global until we reach the semantic classification
- One can view a DCNN as a very nontrivial nonlinear goal-directed analog of a Fourier or wavelet transform of the original image...

**Interlude:** Deep Convolutional Neural Networks (DCNN)

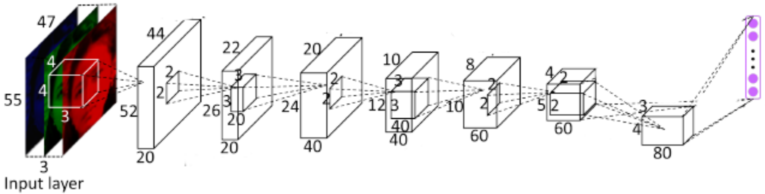
- Contemporary DCNN are much deeper $> 50$ layers...
- Going further (deeper) into the network integrates features from wider range of scales...
- The outputs of interior layers are some (nonlocal) feature representations...
- ... which become more and more global until we reach the semantic classification
- One can view a DCNN as a very nontrivial nonlinear goal-directed analog of a Fourier or wavelet transform of the original image...

## Interlude: Deep Convolutional Neural Networks (DCNN)



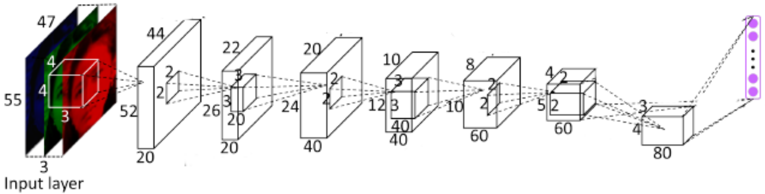

- Contemporary DCNN are much deeper $> 50$ layers...
- Going further (deeper) into the network integrates features from wider range of scales...
- The outputs of interior layers are some (nonlocal) feature representations...
- ... which become more and more global until we reach the semantic classification
- One can view a DCNN as a very nontrivial nonlinear goal-directed analog of a Fourier or wavelet transform of the original image...

## **Interlude:** Deep Convolutional Neural Networks (DCNN)



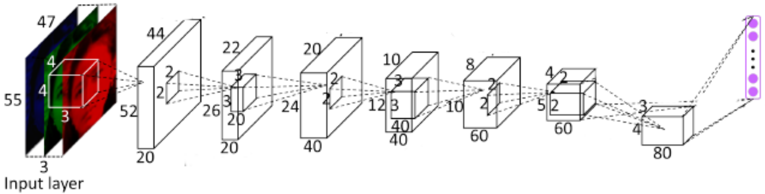

- Contemporary DCNN are much deeper $> 50$ layers...
- Going further (deeper) into the network integrates features from wider range of scales...
- The outputs of interior layers are some (nonlocal) feature representations...
- ... which become more and more global until we reach the semantic classification
- One can view a DCNN as a very nontrivial nonlinear goal-directed analog of a Fourier or wavelet transform of the original image...

## **Interlude:** Deep Convolutional Neural Networks (DCNN)



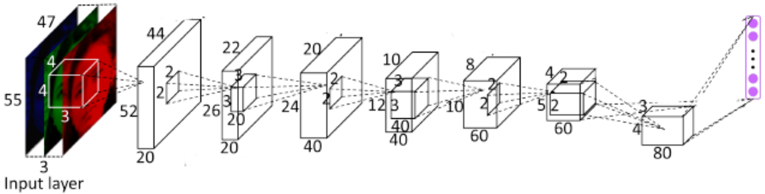

- Contemporary DCNN are much deeper $> 50$ layers...
- Going further (deeper) into the network integrates features from wider range of scales...
- The outputs of interior layers are some (nonlocal) feature representations...
- ... which become more and more global until we reach the semantic classification
- One can view a DCNN as a very nontrivial nonlinear goal-directed analog of a Fourier or wavelet transform of the original image...

## **Interlude:** Deep Convolutional Neural Networks (DCNN)



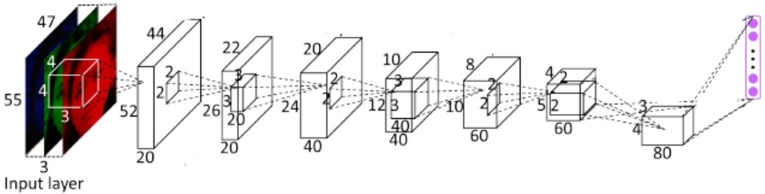

- Contemporary DCNN are much deeper $> 50$ layers...
- Going further (deeper) into the network integrates features from wider range of scales...
- The outputs of interior layers are some (nonlocal) feature representations...
- ... which become more and more global until we reach the semantic classification
- One can view a DCNN as a very nontrivial nonlinear goal-directed analog of a Fourier or wavelet transform of the original image...

## **Interlude:** Deep Convolutional Neural Networks (DCNN)




- Contemporary DCNN are much deeper $> 50$ layers...
- Going further (deeper) into the network integrates features from wider range of scales...
- The outputs of interior layers are some (nonlocal) feature representations...
- ... which become more and more global until we reach the semantic classification
- One can view a DCNN as a very nontrivial nonlinear goal-directed analog of a Fourier or wavelet transform of the original image...

## **Interlude:** Deep Convolutional Neural Networks (DCNN)




- Contemporary DCNN are much deeper $> 50$ layers...
- Going further (deeper) into the network integrates features from wider range of scales...
- The outputs of interior layers are some (nonlocal) feature representations...
- ... which become more and more global until we reach the semantic classification
- One can view a DCNN as a very nontrivial nonlinear goal-directed analog of a Fourier or wavelet transform of the original image...

## Physics analogs

**MERA**

**AdS/CFT**

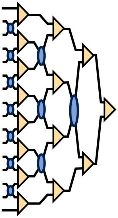
**DCNN**

cf. Hashimoto

represents probability distribution of natural images...

## Physics analogs

**MERA**



**AdS/CFT**

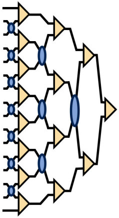
represents a quantum wavefunction

**DCNN**

cf. Hashimoto

represents probability distribution of natural images...

**MERA**



represents a quantum wavefunction

**AdS/CFT**

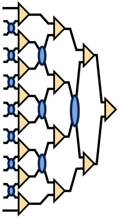
**DCNN**

cf. Hashimoto

represents probability distribution of natural images...

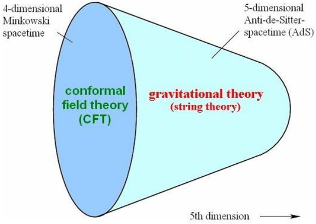
## Physics analogs

**MERA**



represents a quantum wavefunction

**AdS/CFT**




represents a QFT

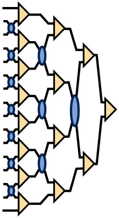
**DCNN**

cf. Hashimoto

represents probability distribution of natural images...

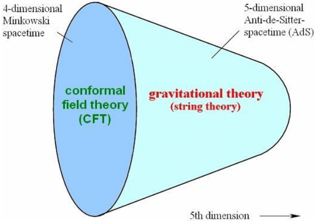
## Physics analogs

**MERA**



represents a quantum wavefunction

**AdS/CFT**




represents a QFT

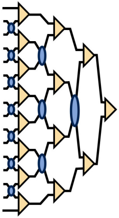
**DCNN**

cf. Hashimoto

represents probability distribution of natural images...

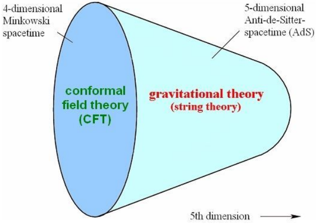
## Physics analogs

### MERA



represents a quantum wavefunction

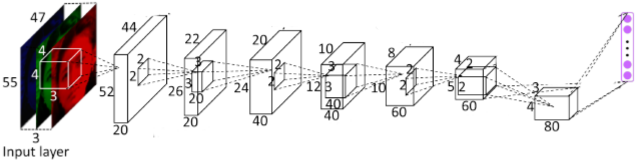
### AdS/CFT




represents a QFT

### DCNN

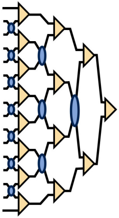
cf. Hashimoto




represents probability distribution of natural images...

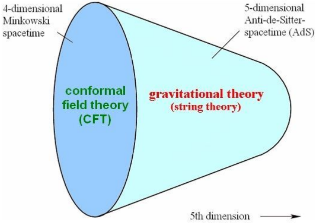
## Physics analogs

### MERA



represents a quantum wavefunction

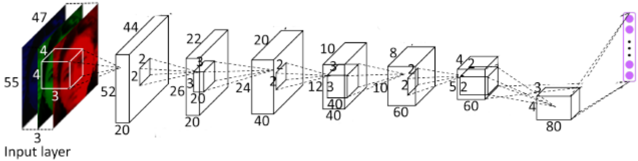
### AdS/CFT




represents a QFT

### DCNN

cf. Hashimoto




represents probability distribution of natural images...

## General questions:

- Look for interesting observables which characterize the behaviour of the network layers as a function of depth (scale)
- Try to interpret it as a characteristic of the dataset (various kinds of images/learning tasks) – here the deep neural network is used as a tool...
- ... and the image dataset is thought as an analog of a "physical system"

**Deep Neural Network** $\sim$ **"holography" for complex probability distributions**

**look for an analog of complexity...**

**General questions:**

- Look for interesting observables which characterize the behaviour of the network layers as a function of depth (scale)
- Try to interpret it as a characteristic of the dataset (various kinds of images/learning tasks) – here the deep neural network is used as a tool...
- ... and the image dataset is thought as an analog of a "physical system"

**Deep Neural Network** $\sim$ **"holography" for complex probability distributions**

**look for an analog of complexity...**

## General questions:

- Look for interesting observables which characterize the behaviour of the network layers as a function of depth (scale)
- Try to interpret it as a characteristic of the dataset (various kinds of images/learning tasks) – here the deep neural network is used as a tool...
- ... and the image dataset is thought as an analog of a "physical system"

**Deep Neural Network** $\sim$ **"holography" for complex probability distributions**

look for an analog of **complexity**...

**General questions:**

- Look for interesting observables which characterize the behaviour of the network layers as a function of depth (scale)
- Try to interpret it as a characteristic of the dataset (various kinds of images/learning tasks) – here the deep neural network is used as a tool...
- ... and the image dataset is thought as an analog of a "physical system"

**Deep Neural Network** $\sim$ **"holography" for complex probability distributions**

**look for an analog of complexity...**

**General questions:**

- Look for interesting observables which characterize the behaviour of the network layers as a function of depth (scale)
- Try to interpret it as a characteristic of the dataset (various kinds of images/learning tasks) – here the deep neural network is used as a tool...
- ... and the image dataset is thought as an analog of a "physical system"

**Deep Neural Network** $\sim$ **"holography" for complex probability distributions**

**look for an analog of complexity...**

**General questions:**

- Look for interesting observables which characterize the behaviour of the network layers as a function of depth (scale)
- Try to interpret it as a characteristic of the dataset (various kinds of images/learning tasks) – here the deep neural network is used as a tool...
- ... and the image dataset is thought as an analog of a "physical system"

**Deep Neural Network** $\sim$ **"holography" for complex probability distributions**

look for an analog of **complexity...**

**General questions:**

- Look for interesting observables which characterize the behaviour of the network layers as a function of depth (scale)
- Try to interpret it as a characteristic of the dataset (various kinds of images/learning tasks) – here the deep neural network is used as a tool...
- ... and the image dataset is thought as an analog of a "physical system"

**Deep Neural Network** $\sim$ **"holography" for complex probability distributions**

**look for an analog of complexity...**

## Complexity in ∼tensor networks and holography

### Circuit complexity

- Pick some family of elementary unitary operators ( *"gates"*)

$$|\Psi\rangle = U_1 \circ U_2 \circ U_3 \circ \ldots \circ U_n \, |REF\rangle$$

- Define complexity as

$$complexity \propto (\log?) \; min \sum_i cost(U_i)$$

### Holography

- Complexity proportional to spatial volume — Susskind

$$complexity \propto V_\perp \int dz \sqrt{g}$$

- "additive" over scales $\longrightarrow$ look at something additive over layers...

**use these only as intuitions...**

## Complexity in ∼tensor networks and holography

### Circuit complexity

- Pick some family of elementary unitary operators (*"gates"*)

$$|\Psi\rangle = U_1 \circ U_2 \circ U_3 \circ \ldots \circ U_n \, |REF\rangle$$

- Define complexity as

$$complexity \propto (\log?) \; min \sum_i cost(U_i)$$

### Holography

- Complexity proportional to spatial volume Susskind

$$complexity \propto V_\perp \int dz \sqrt{g}$$

- "additive" over scales $\longrightarrow$ look at something additive over layers...

**use these only as intuitions...**

## Complexity in $\sim$tensor networks and holography

### Circuit complexity

- Pick some family of elementary unitary operators ( *"gates"*)

$$|\Psi\rangle = U_1 \circ U_2 \circ U_3 \circ \ldots \circ U_n \, |REF\rangle$$

- Define complexity as

$$complexity \propto (\log ?) \; min \sum_i cost(U_i)$$

### Holography

- Complexity proportional to spatial volume — Susskind

$$complexity \propto V_\perp \int dz \sqrt{g}$$

- "additive" over scales $\longrightarrow$ look at something additive over layers...

**use these only as intuitions...**

## Complexity in ∼tensor networks and holography

### Circuit complexity

- Pick some family of elementary unitary operators ( *"gates"*)

$$|\Psi\rangle = U_1 \circ U_2 \circ U_3 \circ \ldots \circ U_n \, |REF\rangle$$

- Define complexity as

$$complexity \propto (\log?) \; min \sum_i cost(U_i)$$

### Holography

- Complexity proportional to spatial volume Susskind

$$complexity \propto V_\perp \int dz \sqrt{g}$$

- "additive" over scales $\longrightarrow$ look at something additive over layers...

**use these only as intuitions...**

## Complexity in ∼tensor networks and holography

### Circuit complexity

- Pick some family of elementary unitary operators ( *"gates"*)

$$|\Psi\rangle = U_1 \circ U_2 \circ U_3 \circ \ldots \circ U_n \, |REF\rangle$$

- Define complexity as

$$complexity \propto (\log?) \; min \sum_i cost(U_i)$$

### Holography

- Complexity proportional to spatial volume Susskind

$$complexity \propto V_\perp \int dz \sqrt{g}$$

- "additive" over scales $\longrightarrow$ look at something additive over layers...

**use these only as intuitions...**

## Complexity in ∼tensor networks and holography

### Circuit complexity

- Pick some family of elementary unitary operators ( *"gates"*)

$$|\Psi\rangle = U_1 \circ U_2 \circ U_3 \circ \ldots \circ U_n \, |REF\rangle$$

- Define complexity as

$$complexity \propto (\log?)\; min \sum_i cost(U_i)$$

### Holography

- Complexity proportional to spatial volume — Susskind

$$complexity \propto V_\perp \int dz \sqrt{g}$$

- "additive" over scales $\longrightarrow$ look at something additive over layers...

**use these only as intuitions...**

## Complexity in ∼tensor networks and holography

### Circuit complexity

- Pick some family of elementary unitary operators ( *"gates"*)

$$|\Psi\rangle = U_1 \circ U_2 \circ U_3 \circ \ldots \circ U_n \, |REF\rangle$$

- Define complexity as

$$complexity \propto (\log?) \; min \sum_i cost(U_i)$$

### Holography

- Complexity proportional to spatial volume Susskind

$$complexity \propto V_\perp \int dz \sqrt{g}$$

- "additive" over scales $\longrightarrow$ look at something additive over layers...

**use these only as intuitions...**

# Complexity in ∼tensor networks and holography

## Circuit complexity

- Pick some family of elementary unitary operators ( *"gates"*)

$$|\Psi\rangle = U_1 \circ U_2 \circ U_3 \circ \ldots \circ U_n \, |REF\rangle$$

- Define complexity as

$$complexity \propto (\log?) \; min \sum_i cost(U_i)$$

## Holography

- Complexity proportional to spatial volume Susskind

$$complexity \propto V_\perp \int dz \sqrt{g}$$

- "additive" over scales $\longrightarrow$ look at something additive over layers...

**use these only as intuitions...**

# Complexity in ∼tensor networks and holography

## Circuit complexity

- Pick some family of elementary unitary operators ( *"gates"*)

$$|\Psi\rangle = U_1 \circ U_2 \circ U_3 \circ \ldots \circ U_n \, |REF\rangle$$

- Define complexity as

$$complexity \propto (\log?) \; min \sum_i cost(U_i)$$

## Holography

- Complexity proportional to spatial volume Susskind

$$complexity \propto V_\perp \int dz \sqrt{g}$$

- "additive" over scales $\longrightarrow$ look at something additive over layers...

use these only as intuitions...

## Complexity in ∼tensor networks and holography

### Circuit complexity

- Pick some family of elementary unitary operators ( *"gates"*)

$$|\Psi\rangle = U_1 \circ U_2 \circ U_3 \circ \ldots \circ U_n \, |REF\rangle$$

- Define complexity as

$$complexity \propto (\log?) \; min \sum_i cost(U_i)$$

### Holography

- Complexity proportional to spatial volume — Susskind

$$complexity \propto V_\perp \int dz \sqrt{g}$$

- "additive" over scales $\longrightarrow$ look at something additive over layers...

**use these only as intuitions...**

## Complexity for neural networks – heuristics

**Do not count parameters** but estimate the complexity of a **particular computation** – **the neural network architecture is kept fixed!**

- In modern neural networks we have typically ReLU neurons

$$y_k = ReLU(W_{ki}x_i + b_k)$$

  which act either as a linear function or as zero
- It does not make sense to count gates which collapse under composition

$$A_1 \circ A_2 \circ A_3 \circ A_4 = A$$

- Both the linear function and zero collapse under composition!
- Nontriviality arises with changing the mode of operation (switching on/off)
- **Idea: Define complexity in terms of the switching behaviour (nonlinearity!).**

## Complexity for neural networks – heuristics

**Do not count parameters** but estimate the complexity of a **particular computation** – **the neural network architecture is kept fixed!**

- In modern neural networks we have typically ReLU neurons

$$y_k = ReLU(W_{ki}x_i + b_k)$$

  which act either as a linear function or as zero
- It does not make sense to count gates which collapse under composition

$$A_1 \circ A_2 \circ A_3 \circ A_4 = A$$

- Both the linear function and zero collapse under composition!
- Nontriviality arises with changing the mode of operation (switching on/off)
- **Idea: Define complexity in terms of the switching behaviour (nonlinearity!).**

# Complexity for neural networks – heuristics

**Do not count parameters** but estimate the complexity of a **particular computation** – **the neural network architecture is kept fixed!**

- In modern neural networks we have typically ReLU neurons

$$y_k = ReLU(W_{ki} x_i + b_k)$$

  which act either as a linear function or as zero
- It does not make sense to count gates which collapse under composition

$$A_1 \circ A_2 \circ A_3 \circ A_4 = A$$

- Both the linear function and zero collapse under composition!
- Nontriviality arises with changing the mode of operation (switching on/off)
- **Idea: Define complexity in terms of the switching behaviour (nonlinearity!).**

## Complexity for neural networks – heuristics

**Do not count parameters** but estimate the complexity of a **particular computation** – **the neural network architecture is kept fixed!**

- In modern neural networks we have typically ReLU neurons

$$y_k = ReLU(W_{ki}x_i + b_k)$$

  which act either as a linear function or as zero
- It does not make sense to count gates which collapse under composition

$$A_1 \circ A_2 \circ A_3 \circ A_4 = A$$

- Both the linear function and zero collapse under composition!
- Nontriviality arises with changing the mode of operation (switching on/off)
- **Idea: Define complexity in terms of the switching behaviour (nonlinearity!).**

## Complexity for neural networks – heuristics

**Do not count parameters** but estimate the complexity of a **particular computation** – **the neural network architecture is kept fixed!**

- In modern neural networks we have typically ReLU neurons

$$y_k = ReLU(W_{ki} x_i + b_k)$$

  which act either as a linear function or as zero
- It does not make sense to count gates which collapse under composition

$$A_1 \circ A_2 \circ A_3 \circ A_4 = A$$

- Both the linear function and zero collapse under composition!
- Nontriviality arises with changing the mode of operation (switching on/off)
- **Idea:** **Define complexity in terms of the switching behaviour (nonlinearity!).**

## Complexity for neural networks – heuristics

**Do not count parameters** but estimate the complexity of a **particular computation** – **the neural network architecture is kept fixed!**

- In modern neural networks we have typically ReLU neurons

$$y_k = ReLU(W_{ki}x_i + b_k)$$

  which act either as a linear function or as zero

- It does not make sense to count gates which collapse under composition

$$A_1 \circ A_2 \circ A_3 \circ A_4 = A$$

- Both the linear function and zero collapse under composition!
- Nontriviality arises with changing the mode of operation (switching on/off)
- **Idea:** **Define complexity in terms of the switching behaviour (nonlinearity!).**

## Complexity for neural networks – heuristics

**Do not count parameters** but estimate the complexity of a **particular computation** – **the neural network architecture is kept fixed!**

- In modern neural networks we have typically ReLU neurons

$$y_k = ReLU(W_{ki} x_i + b_k)$$

  which act either as a linear function or as zero
- It does not make sense to count gates which collapse under composition

$$A_1 \circ A_2 \circ A_3 \circ A_4 = A$$

- Both the linear function and zero collapse under composition!
- Nontriviality arises with changing the mode of operation (switching on/off)
- **Idea:** **Define complexity in terms of the switching behaviour (nonlinearity!).**

## Complexity for neural networks – heuristics

**Do not count parameters** but estimate the complexity of a **particular computation** – **the neural network architecture is kept fixed!**

- In modern neural networks we have typically ReLU neurons

$$y_k = ReLU(W_{ki} x_i + b_k)$$

  which act either as a linear function or as zero
- It does not make sense to count gates which collapse under composition

$$A_1 \circ A_2 \circ A_3 \circ A_4 = A$$

- Both the linear function and zero collapse under composition!
- Nontriviality arises with changing the mode of operation (switching on/off)
- **Idea: Define complexity in terms of the switching behaviour (nonlinearity!).**

## Complexity for neural networks

- Define a binary **nonlinearity** variable ($z_k = 1$ on, $z_k = 0$ off)

$$z_k = \theta(W_{ki} x_i + b_k)$$

- Define the complexity of a layer as the Shannon entropy of the nonlinearities $z_k$:

**complexity** $\equiv$ **entropy of nonlinearity**

- We can study the nonlinearity of the neural network's operation as a function of depth...
- Using the algorithm from the first part of the talk, we can effectively compute the complexity for each layer...

... and add them up if we want

## Complexity for neural networks

- Define a binary **nonlinearity** variable ($z_k = 1$ on, $z_k = 0$ off)

$$z_k = \theta(W_{ki} x_i + b_k)$$

- Define the complexity of a layer as the Shannon entropy of the nonlinearities $z_k$:

$$\textbf{complexity} \equiv \textbf{entropy of nonlinearity}$$

- We can study the nonlinearity of the neural network's operation as a function of depth...
- Using the algorithm from the first part of the talk, we can effectively compute the complexity for each layer...

... and add them up if we want

## Complexity for neural networks

- Define a binary **nonlinearity** variable ($z_k = 1$ on, $z_k = 0$ off)

$$z_k = \theta(W_{ki} x_i + b_k)$$

- Define the complexity of a layer as the Shannon entropy of the nonlinearities $z_k$:

**complexity** $\equiv$ **entropy of nonlinearity**

- We can study the nonlinearity of the neural network's operation as a function of depth...
- Using the algorithm from the first part of the talk, we can effectively compute the complexity for each layer...

... and add them up if we want

## Complexity for neural networks

- Define a binary **nonlinearity** variable ($z_k = 1$ on, $z_k = 0$ off)

$$z_k = \theta(W_{ki} x_i + b_k)$$

- Define the complexity of a layer as the Shannon entropy of the nonlinearities $z_k$:

**complexity** $\equiv$ **entropy of nonlinearity**

- We can study the nonlinearity of the neural network's operation as a function of depth...
- Using the algorithm from the first part of the talk, we can effectively compute the complexity for each layer...

... and add them up if we want

## Complexity for neural networks

- Define a binary **nonlinearity** variable ($z_k = 1$ on, $z_k = 0$ off)

$$z_k = \theta(W_{ki} x_i + b_k)$$

- Define the complexity of a layer as the Shannon entropy of the nonlinearities $z_k$:

**complexity $\equiv$ entropy of nonlinearity**

- We can study the nonlinearity of the neural network's operation as a function of depth...
- Using the algorithm from the first part of the talk, we can effectively compute the complexity for each layer...

... and add them up if we want

## Complexity for neural networks

- Define a binary **nonlinearity** variable ($z_k = 1$ on, $z_k = 0$ off)

$$z_k = \theta(W_{ki} x_i + b_k)$$

- Define the complexity of a layer as the Shannon entropy of the nonlinearities $z_k$:

**complexity** $\equiv$ **entropy of nonlinearity**

- We can study the nonlinearity of the neural network's operation as a function of depth...
- Using the algorithm from the first part of the talk, we can effectively compute the complexity for each layer...

... and add them up if we want

## Effective dimension of feature representations

- **Complexity** by design ignored details of linear operation of the network...
- To capture information on the linear regime, introduce a complementary observable **effective dimension** of the feature representation of the given layer...

**Algorithm:**

1. Compute the PCA of the layer output
2. Get the `variance ratio explained` $r_i$ ($\sum r_i = 1$)
3. Compute Shannon entropy of the $r_i$'s
4. Define

$$\textit{effective dimension} \equiv \exp(S[\{r_i\}])$$

## Effective dimension of feature representations

- **Complexity** by design ignored details of linear operation of the network...
- To capture information on the linear regime, introduce a complementary observable **effective dimension** of the feature representation of the given layer...

**Algorithm:**

1. Compute the PCA of the layer output
2. Get the `variance ratio explained` $r_i$ ($\sum r_i = 1$)
3. Compute Shannon entropy of the $r_i$'s
4. Define

$$\textit{effective dimension} \equiv \exp(S[\{r_i\}])$$

## Effective dimension of feature representations

- **Complexity** by design ignored details of linear operation of the network...
- To capture information on the linear regime, introduce a complementary observable **effective dimension** of the feature representation of the given layer...

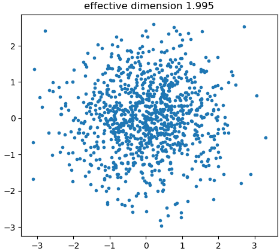
**Algorithm:**

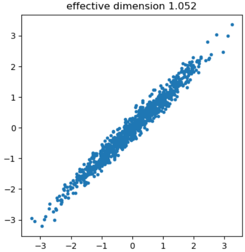
1. Compute the PCA of the layer output
2. Get the `variance ratio explained` $r_i$ ($\sum r_i = 1$)
3. Compute Shannon entropy of the $r_i$'s
4. Define

$$\textit{effective dimension} \equiv \exp(S[\{r_i\}])$$

## Effective dimension of feature representations

- **Complexity** by design ignored details of linear operation of the network...
- To capture information on the linear regime, introduce a complementary observable **effective dimension** of the feature representation of the given layer...



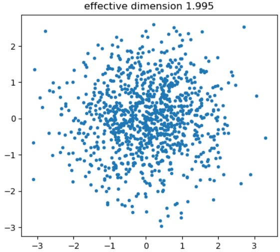

**Algorithm:**

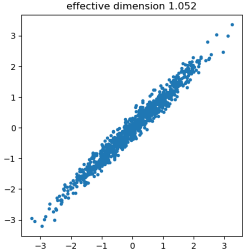
1. Compute the PCA of the layer output
2. Get the `variance ratio explained` $r_i$ ($\sum r_i = 1$)
3. Compute Shannon entropy of the $r_i$'s
4. Define

$$\textit{effective dimension} \equiv \exp(S[\{r_i\}])$$

## Effective dimension of feature representations

- **Complexity** by design ignored details of linear operation of the network...
- To capture information on the linear regime, introduce a complementary observable **effective dimension** of the feature representation of the given layer...



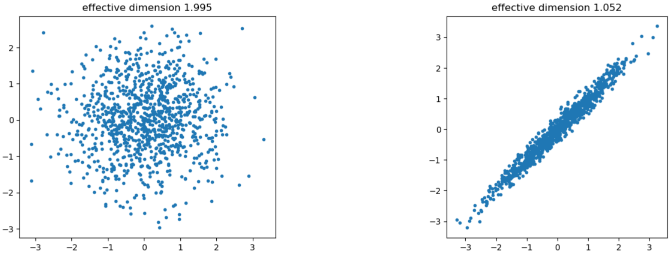

**Algorithm:**

1. Compute the PCA of the layer output
2. Get the `variance ratio explained` $r_i$ ($\sum r_i = 1$)
3. Compute Shannon entropy of the $r_i$'s
4. Define

$$\textit{effective dimension} \equiv \exp(S[\{r_i\}])$$

## Effective dimension of feature representations

- **Complexity** by design ignored details of linear operation of the network...
- To capture information on the linear regime, introduce a complementary observable **effective dimension** of the feature representation of the given layer...



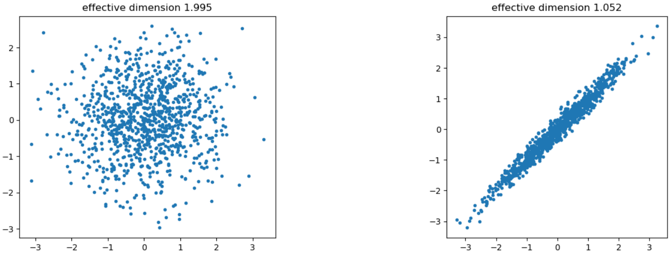

**Algorithm:**

1. Compute the PCA of the layer output
2. Get the `variance ratio explained` $r_i$ ($\sum r_i = 1$)
3. Compute Shannon entropy of the $r_i$'s
4. Define

$$\textit{effective dimension} \equiv \exp(S[\{r_i\}])$$

## Effective dimension of feature representations

- **Complexity** by design ignored details of linear operation of the network...
- To capture information on the linear regime, introduce a complementary observable **effective dimension** of the feature representation of the given layer...



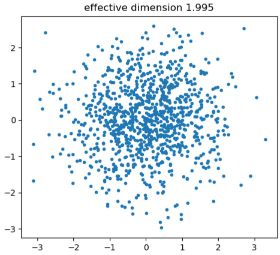

**Algorithm:**

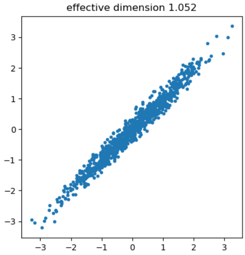
1. Compute the PCA of the layer output
2. Get the `variance ratio explained` $r_i$ ($\sum r_i = 1$)
3. Compute Shannon entropy of the $r_i$'s
4. Define

$$\textit{effective dimension} \equiv \exp(S[\{r_i\}])$$

## Effective dimension of feature representations

- **Complexity** by design ignored details of linear operation of the network...
- To capture information on the linear regime, introduce a complementary observable **effective dimension** of the feature representation of the given layer...



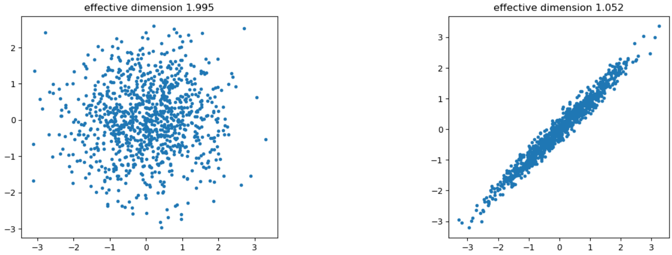

**Algorithm:**

1. Compute the PCA of the layer output
2. Get the `variance ratio explained` $r_i$ ($\sum r_i = 1$)
3. Compute Shannon entropy of the $r_i$'s
4. Define

$$\textit{effective dimension} \equiv \exp(S[\{r_i\}])$$

## Effective dimension of feature representations

- **Complexity** by design ignored details of linear operation of the network...
- To capture information on the linear regime, introduce a complementary observable **effective dimension** of the feature representation of the given layer...




**Algorithm:**

1. Compute the PCA of the layer output
2. Get the `variance ratio explained` $r_i$ ($\sum r_i = 1$)
3. Compute Shannon entropy of the $r_i$'s
4. Define

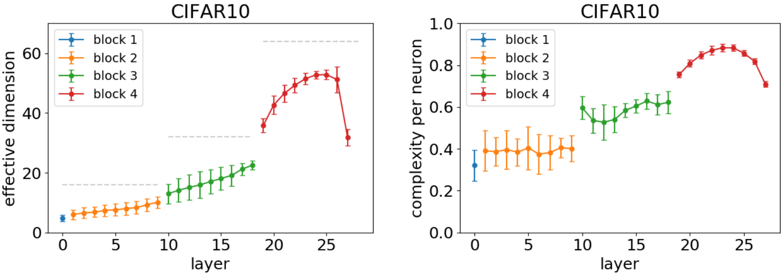
$$\textit{effective dimension} \equiv \exp(S[\{r_i\}])$$

## Numerical experiments

## “Holographic” plots

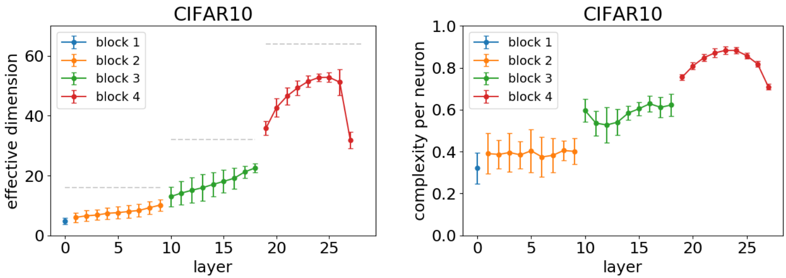
- The different colors represent different stages of a `resnet-56` network (version for CIFAR-10 dataset)
- The network produces richer representations as we go deeper into the network..
- ...more nonlinearity and higher effective dimension...
- We observe increase when going to more difficult datasets – especially at the higher levels of the network

## “Holographic” plots



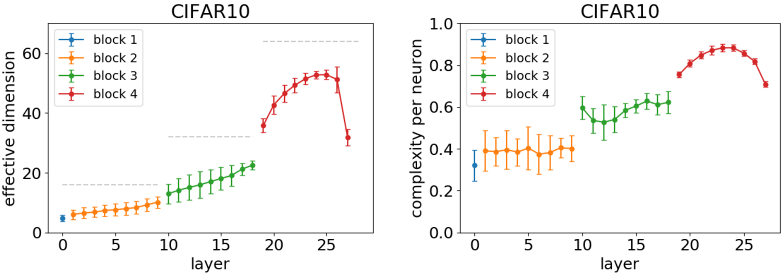

- The different colors represent different stages of a `resnet-56` network (version for CIFAR-10 dataset)
- The network produces richer representations as we go deeper into the network..
- ...more nonlinearity and higher effective dimension...
- We observe increase when going to more difficult datasets – especially at the higher levels of the network

## “Holographic” plots



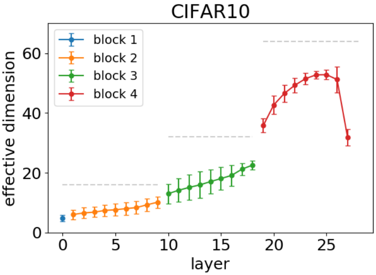

- The different colors represent different stages of a `resnet-56` network (version for CIFAR-10 dataset)
- The network produces richer representations as we go deeper into the network..
- ...more nonlinearity and higher effective dimension...
- We observe increase when going to more difficult datasets – especially at the higher levels of the network

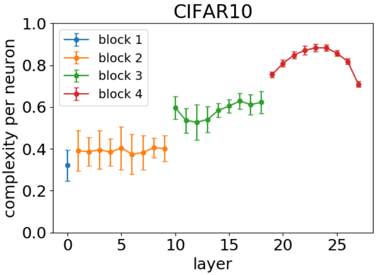
# “Holographic” plots




- The different colors represent different stages of a `resnet-56` network (version for CIFAR-10 dataset)
- The network produces richer representations as we go deeper into the network..
- ...more nonlinearity and higher effective dimension...
- We observe increase when going to more difficult datasets – especially at the higher levels of the network

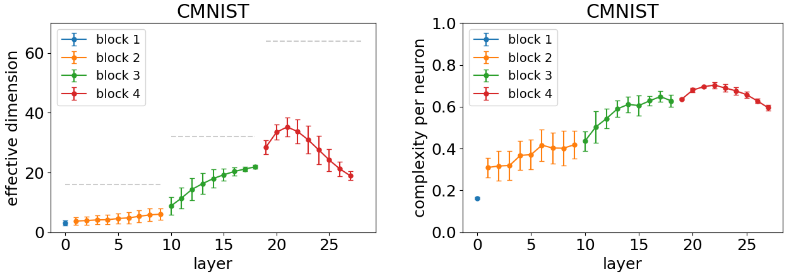
## "Holographic" plots



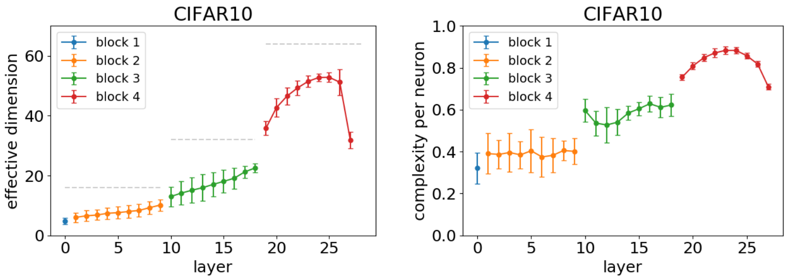

- The different colors represent different stages of a `resnet-56` network (version for CIFAR-10 dataset)
- The network produces richer representations as we go deeper into the network..
- ...more nonlinearity and higher effective dimension...
- We observe increase when going to more difficult datasets – especially at the higher levels of the network

## "Holographic" plots




- The different colors represent different stages of a `resnet-56` network (version for CIFAR-10 dataset)
- The network produces richer representations as we go deeper into the network..
- ...more nonlinearity and higher effective dimension...
- We observe increase when going to more difficult datasets – especially at the higher levels of the network

## “Holographic” plots




- The different colors represent different stages of a `resnet-56` network (version for CIFAR-10 dataset)
- The network produces richer representations as we go deeper into the network..
- ...more nonlinearity and higher effective dimension...
- We observe increase when going to more difficult datasets – especially at the higher levels of the network

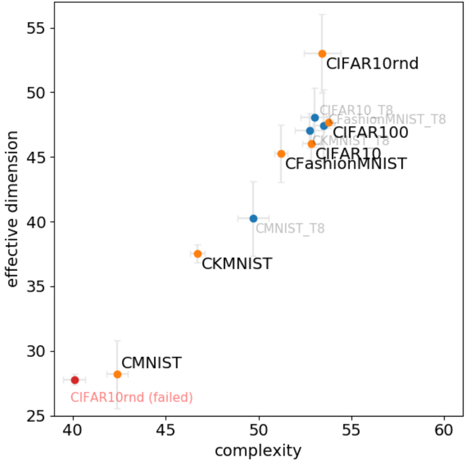
## Dataset difficulty

- We evaluated complexity and effective dimension averaged over the topmost layers of the `resnet-56` network for a variety of datasets

- We find a clear correlation with the intuitive dataset difficulty

## Dataset difficulty

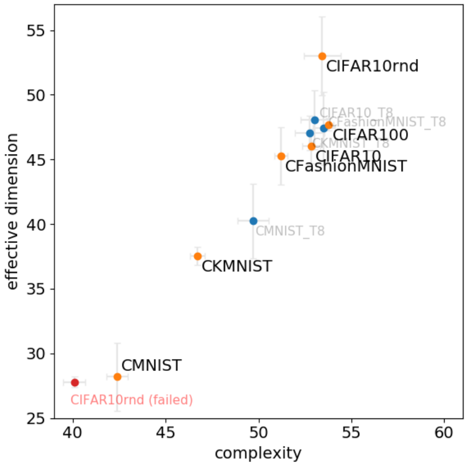
- We evaluated complexity and effective dimension averaged over the topmost layers of the `resnet-56` network for a variety of datasets




- We find a clear correlation with the intuitive dataset difficulty

## Dataset difficulty

- We evaluated complexity and effective dimension averaged over the topmost layers of the `resnet-56` network for a variety of datasets




- We find a clear correlation with the intuitive dataset difficulty

**What is the origin of the rise in complexity?**

- Is the switching behaviour of individual neurons becoming closer to optimal ($p = 0.5$)?
- Is the switching increasingly **independent**?

**What is the origin of the rise in complexity?**

- Is the switching behaviour of individual neurons becoming closer to optimal ($p = 0.5$)?
- Is the switching increasingly **independent**?

**What is the origin of the rise in complexity?**

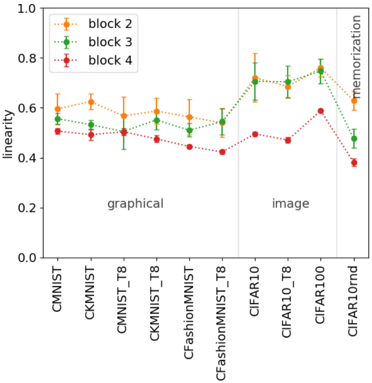
- Is the switching behaviour of individual neurons becoming closer to optimal ($p = 0.5$)?
- Is the switching increasingly **independent**?

## Total correlation

**Question:** Does the increase in complexity occur due to switching behaviour being more close to optimal ($p = 0.5$) or due to increased **independence** of the nonlinearities?

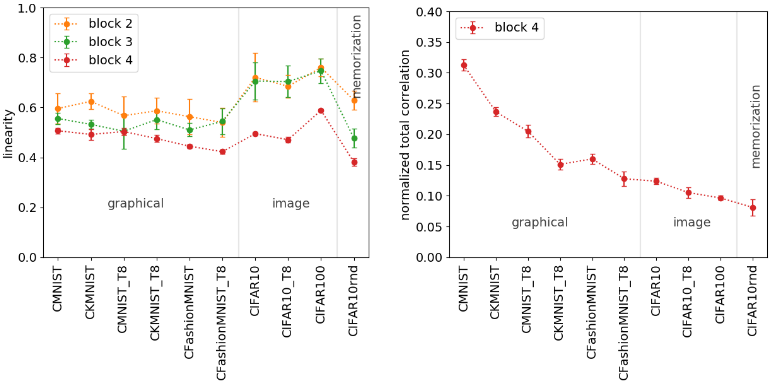
## Total correlation

**Question:** Does the increase in complexity occur due to switching behaviour being more close to optimal ($p = 0.5$) or due to increased **independence** of the nonlinearities?



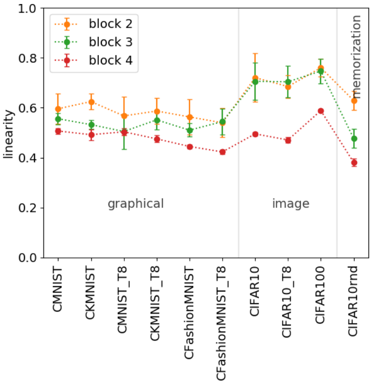
## Total correlation

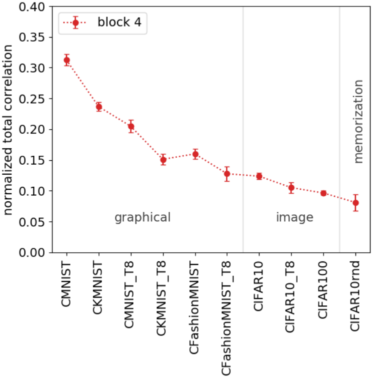
**Question:** Does the increase in complexity occur due to switching behaviour being more close to optimal ($p = 0.5$) or due to increased **independence** of the nonlinearities?



## Total correlation

**Question:** Does the increase in complexity occur due to switching behaviour being more close to optimal ($p = 0.5$) or due to increased **independence** of the nonlinearities?

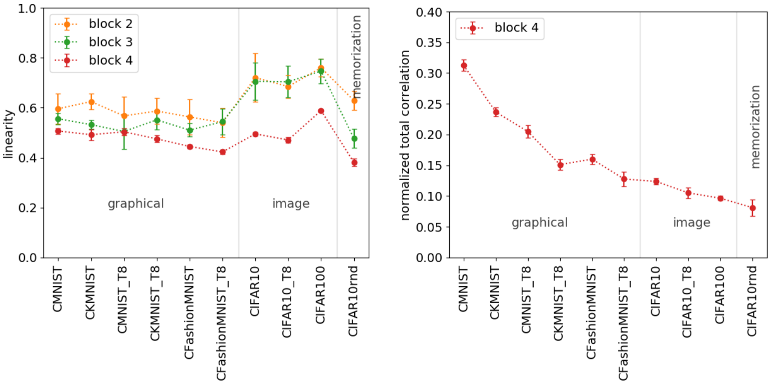



$$\textit{normalized total correlation} = \frac{H(Z_1) + \ldots + H(Z_C) - H(Z_1, \ldots Z_C)}{H(Z_1) + \ldots + H(Z_C)}$$

## Total correlation

**Question:** Does the increase in complexity occur due to switching behaviour being more close to optimal ($p = 0.5$) or due to increased **independence** of the nonlinearities?




For more complicated datasets, the higher level neurons fire in a more independent way – they extract more independent features

## Total correlation during training

- Since independent switching was the deciding factor for complexity differences between various tasks, it is interesting to study its buildup during training...

**Clear power-law scaling**!

- The scaling exponent is not universal – it seems to depend on the dataset and learning rate
- It is a challenge to understand this behaviour!

## Total correlation during training

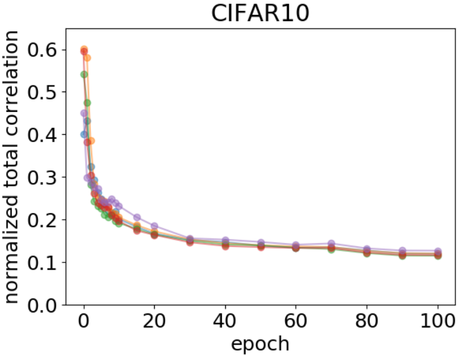
- Since independent switching was the deciding factor for complexity differences between various tasks, it is interesting to study its buildup during training...

**Clear power-law scaling**!

- The scaling exponent is not universal – it seems to depend on the dataset and learning rate
- It is a challenge to understand this behaviour!

## Total correlation during training

- Since independent switching was the deciding factor for complexity differences between various tasks, it is interesting to study its buildup during training...

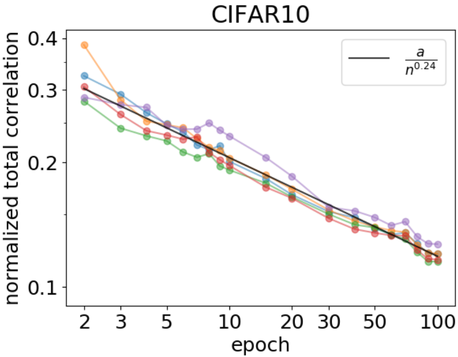



**Clear power-law scaling!**

- The scaling exponent is not universal – it seems to depend on the dataset and learning rate
- It is a challenge to understand this behaviour!

## Total correlation during training

- Since independent switching was the deciding factor for complexity differences between various tasks, it is interesting to study its buildup during training...

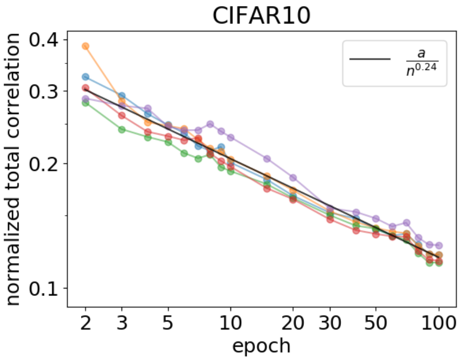



**Clear power-law scaling**!

- The scaling exponent is not universal – it seems to depend on the dataset and learning rate
- It is a challenge to understand this behaviour!

## Total correlation during training

- Since independent switching was the deciding factor for complexity differences between various tasks, it is interesting to study its buildup during training...

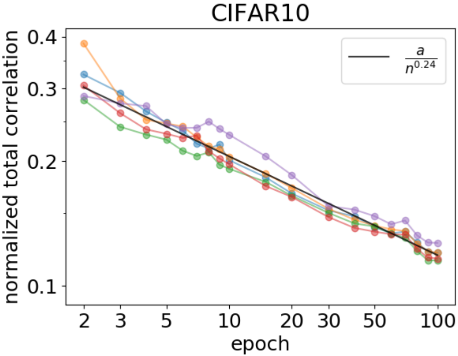



**Clear power-law scaling**!

- The scaling exponent is not universal – it seems to depend on the dataset and learning rate
- It is a challenge to understand this behaviour!

## Total correlation during training

- Since independent switching was the deciding factor for complexity differences between various tasks, it is interesting to study its buildup during training...




**Clear power-law scaling**!

- The scaling exponent is not universal – it seems to depend on the dataset and learning rate
- It is a challenge to understand this behaviour!

## Summary

- We developed a flexible method for leveraging Machine Learning to estimate entropy from Monte Carlo configurations
- ... by translating the problem into a sequence of classification tasks
- One can use *any* ML classification algorithm for that...
- We defined a notion of *complexity* for neural networks which measures nonlinearity of information processing
- ... and a complementary measure of *effective dimension* of feature representations
- Clear correlation with the intuitive difficulty of datasets

  DNN as "holography" for probability distributions...
- Insights into training...
- There is room for fruitful exchanges between Machine Learning and Physics

## Summary

- We developed a flexible method for leveraging Machine Learning to estimate entropy from Monte Carlo configurations
- ... by translating the problem into a sequence of classification tasks
- One can use *any* ML classification algorithm for that...
- We defined a notion of *complexity* for neural networks which measures nonlinearity of information processing
- ... and a complementary measure of *effective dimension* of feature representations
- Clear correlation with the intuitive difficulty of datasets

  DNN as "holography" for probability distributions...
- Insights into training...
- There is room for fruitful exchanges between Machine Learning and Physics

## Summary

- We developed a flexible method for leveraging Machine Learning to estimate entropy from Monte Carlo configurations
- ... by translating the problem into a sequence of classification tasks
- One can use *any* ML classification algorithm for that...
- We defined a notion of *complexity* for neural networks which measures nonlinearity of information processing
- ... and a complementary measure of *effective dimension* of feature representations
- Clear correlation with the intuitive difficulty of datasets

  DNN as "holography" for probability distributions...
- Insights into training...
- There is room for fruitful exchanges between Machine Learning and Physics

## Summary

- We developed a flexible method for leveraging Machine Learning to estimate entropy from Monte Carlo configurations
- ... by translating the problem into a sequence of classification tasks
- One can use *any* ML classification algorithm for that...
- We defined a notion of *complexity* for neural networks which measures nonlinearity of information processing
- ... and a complementary measure of *effective dimension* of feature representations
- Clear correlation with the intuitive difficulty of datasets

  DNN as "holography" for probability distributions...
- Insights into training...
- There is room for fruitful exchanges between Machine Learning and Physics

## Summary

- We developed a flexible method for leveraging Machine Learning to estimate entropy from Monte Carlo configurations
- ... by translating the problem into a sequence of classification tasks
- One can use *any* ML classification algorithm for that...
- We defined a notion of *complexity* for neural networks which measures nonlinearity of information processing
- ... and a complementary measure of *effective dimension* of feature representations
- Clear correlation with the intuitive difficulty of datasets

  DNN as "holography" for probability distributions...
- Insights into training...
- There is room for fruitful exchanges between Machine Learning and Physics

## Summary

- We developed a flexible method for leveraging Machine Learning to estimate entropy from Monte Carlo configurations
- ... by translating the problem into a sequence of classification tasks
- One can use *any* ML classification algorithm for that...
- We defined a notion of *complexity* for neural networks which measures nonlinearity of information processing
- ... and a complementary measure of *effective dimension* of feature representations
- Clear correlation with the intuitive difficulty of datasets

  DNN as "holography" for probability distributions...
- Insights into training...
- There is room for fruitful exchanges between Machine Learning and Physics

## Summary

- We developed a flexible method for leveraging Machine Learning to estimate entropy from Monte Carlo configurations
- ... by translating the problem into a sequence of classification tasks
- One can use *any* ML classification algorithm for that...
- We defined a notion of *complexity* for neural networks which measures nonlinearity of information processing
- ... and a complementary measure of *effective dimension* of feature representations
- Clear correlation with the intuitive difficulty of datasets

DNN as "holography" for probability distributions...

- Insights into training...
- There is room for fruitful exchanges between Machine Learning and Physics

## Summary

- We developed a flexible method for leveraging Machine Learning to estimate entropy from Monte Carlo configurations
- ... by translating the problem into a sequence of classification tasks
- One can use *any* ML classification algorithm for that...
- We defined a notion of *complexity* for neural networks which measures nonlinearity of information processing
- ... and a complementary measure of *effective dimension* of feature representations
- Clear correlation with the intuitive difficulty of datasets

DNN as "holography" for probability distributions...

- Insights into training...
- There is room for fruitful exchanges between Machine Learning and Physics

## Summary

- We developed a flexible method for leveraging Machine Learning to estimate entropy from Monte Carlo configurations
- ... by translating the problem into a sequence of classification tasks
- One can use *any* ML classification algorithm for that...
- We defined a notion of *complexity* for neural networks which measures nonlinearity of information processing
- ... and a complementary measure of *effective dimension* of feature representations
- Clear correlation with the intuitive difficulty of datasets

  DNN as "holography" for probability distributions...
- Insights into training...
- There is room for fruitful exchanges between Machine Learning and Physics

## Summary

- We developed a flexible method for leveraging Machine Learning to estimate entropy from Monte Carlo configurations
- ... by translating the problem into a sequence of classification tasks
- One can use *any* ML classification algorithm for that...
- We defined a notion of *complexity* for neural networks which measures nonlinearity of information processing
- ... and a complementary measure of *effective dimension* of feature representations
- Clear correlation with the intuitive difficulty of datasets

  DNN as "holography" for probability distributions...
- Insights into training...
- There is room for fruitful exchanges between Machine Learning and Physics