Stockholms universitet

Grokking through the lens of information theory

Ashkan Safaee

Kandidatuppsats i matematisk statistik
Bachelor Thesis in Mathematical Statistics

Kandidatuppsats 2026:3
Matematisk statistik
Maj 2026

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Matematiska institutionen

Stockholm University

Mathematical Statistics
Stockholm University
Bachelor Thesis **2026:3**
http://www.math.su.se

Grokking through the lens of information theory

Ashkan Safaee*

May 2026

Abstract

Grokking, the phenomenon where neural networks undergo a sudden transition from memorization to generalization long after the training error has disappeared, remains poorly understood, with implications for AI safety, alignment, and interpretability. Two information theoretic frameworks have been proposed to study representation learning in deep neural networks: Tishby's information bottleneck framework, later adopted by Tishby and Zaslavsky and Shwartz–Ziv and Tishby to deep neural networks, which described an initial fitting phase followed by a representation compression on the information plane, and Clauw et al.'s multivariate framework which identifies feature learning, emergence, and decoupling phases through synergy and redundancy between neurons. This thesis synthesizes the two to build an analytical bridge between them and study the phenomenon of grokking. We propose that the decoupling phase proposed by Clauw et al. corresponds to the representation compression phase predicted in the information bottleneck framework, with weight decay as an enabling factor to drive the network from pure memorization towards a generalizing, compressed representation. Establishing this connection rigorously is left to future work.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: ashkansafaee@gmail.com. Supervisor: Ola Hössjer, Leo Levenius.

Sammanfattning

Grokking, fenomenet där neurala nätverk genomgår en plötslig övergång från memorering till generalisering långt efter att träningsfel har minskat, är ännu inte helt klarlagt. Detta har implikationer för AI-säkerhet, AI-anpassning och tolkningsbarhet. Två informationsteoretiska ramverk har föreslagits för att studera representationsinlärning inom neurala nätverk: för det första Tishbys information bottleneck-ramverk, som senare har vidareutvecklats av Tishby och Zaslavsky, samt Shwartz–Ziv och Tishby till neurala nätverk, vilket beskrev en initial anpassningsfas följt av representationskompression på informationsplanet. För det andra har Clauw et al. givit ett multivariat ramverk som identifierar attributinlärning, emergens och frikopplingsfaser genom synergi och redundans mellan neuroner i ett neuralt nätverk. Denna uppsats syntetiserar dessa två ramverk och bygger en analytisk brygga mellan dem för att studera fenomenet grokking. Vi föreslår att frikopplingsfasen som Clauw et al. beskriver korresponderar mot representationskompressionsfasen som information bottleneck-ramverket predicerar. Vidare betonar vi viktdämpning (weight decay) som en möjliggörande faktor som driver nätverket från ren memorering mot en generaliserande, komprimerad representation. Att etablera denna koppling rigoröst lämnas till framtida studier.

Acknowledgements

My heartfelt gratitude to my family for their patience, support, and encouragement over the last couple of years, which have culminated in this thesis.

I would also like to thank my supervisors Ola Hössjer and Leo Levenius for their guidance throughout this work. Your knowledge and involvement brought clarity to a dense subject. Thanks also to Chun-Biu Li, whose teaching in Statistical Information Theory and Deep Learning first sparked my interest in this subject.

Use of AI tools: AI-based tools were used during the preparation of this thesis for generating illustrative figures, identifying relevant literature, providing feedback on mathematical notation and consistency, and improving academic writing conventions and language. All intellectual content, theoretical claims, and conclusions are the author's own.

Contents

1 Introduction

1.1 Background and motivation

In the context of AI safety and alignment — the research field dedicated to ensuring that artificial intelligence acts in accordance with human goals, principles, and ethics — it is critical to understand what a network is actually doing, even regarding 'hard to explain' phenomena. One such phenomenon is *grokking*. Grokking occurs in certain types of neural network settings and can informally be described as 'generalization way after overfitting.' That is, a network may struggle to perform well on a task initially, only to exhibit near-perfect generalization after a prolonged period of training [10]. Effectively, the model moves from merely memorizing answers to learning the inherent methods required to solve the task. In one striking example, Nanda et al. [9] reverse-engineered a neural network trained on modular addition and discovered that the network discovered a generalizing algorithm based on the discrete Fourier transform and trigonometric identities. While exciting for the field, this phenomenon also raises concerns for AI safety and alignment [1]; if models can undergo sudden, delayed shifts in capability, it implies that unpredictable behaviors could remain hidden long after a model appears to have converged.

1.2 Problem statement

In this thesis, we use statistical information theory as a lens to explore the following question:

> *How does the phenomenon of grokking relate to the representation compression phase predicted by the information bottleneck framework?*

1.3 Method overview

This thesis synthesized the information bottleneck principle, originally developed by Tishby, Pereira & Bialek in 1999, as applied to deep neural networks by Tishby and Zaslavsky and Shwartz–Ziv and Tishby with the multivariate information theoretic framework by Clauw et al. The aim is to build an analytical bridge between the two frameworks to study the phenomenon of grokking. The scope is theoretical: no simulations or empirical experiments have been conducted.

1.4 Structure of the thesis

Section 2 lays the foundations of statistical information theory as a preliminary framework. In Section 3, we introduce deep learning and describe a modern 'vanilla' neural network architecture, representing the most fundamental structure in the current evolution of the field. This section also introduces the phenomenon of *grokking*, characterized as delayed generalization. Section 4 is dedicated to studying deep neural networks through the lens of statistical information theory, with a particular focus on the information plane. Finally, we describe the information dynamics under grokking and conclude with several hypotheses regarding the underlying mechanics of this phenomenon, studied through the framework of information theory.

2 Statistical information theory

Neural networks can be viewed as a tradeoff between compression and complexity [15]. As we will show in later sections, by increasing dimension in neural networks (adding more layers and neurons) we impose more complexity, and when making a prediction we compress from higher dimensions onto for example a binary outcome of two. This tradeoff, between compression and complexity is beneficial to observe through the lens of information theory and its key idea of entropy.

2.1 Basic concepts of information theory

Definition 2.1 (Entropy)**.** Let $p(x)$ be the probability mass function of a random variable $X \in \mathcal{X}$. The *entropy* of X is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x).$$

The use of logarithm of base 2 gives us the entropy measured in bits. This measure captures "the average uncertainty in the random variable", or equivalently, "the uncertainty of a single random variable" [2]. Compared to variance, which is the squared distance from the mean (squared spread), entropy measures the information content of a probability distribution. Intuitively, we can see this as the higher the entropy, the more uncertain we are about a system we are observing, and vice versa. Zero entropy is the deterministic, and rather unrealistic, case where we have full knowledge of a system. The conditional entropy $H(X|Y)$ measures the remaining uncertainty in X given knowledge of Y. In order to define this concept (see Equation (1) below), we will first introduce two other concepts: relative entropy and mutual information.

Definition 2.2 (Joint entropy)**.** The *joint entropy* of a pair of random variables (X, Y) with joint probability mass function $p(x, y)$ is defined as

$$H(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y).$$

2.1.1 Relative entropy and mutual information

Definition 2.3 (Relative entropy)**.** *Relative entropy* is defined as the Kullback–Leibler (KL) divergence between two distributions P and Q with probability mass

functions $p(x)$ and $q(x)$ as

$$D(p||q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}.$$

Intuitively, we can see this as how different an approximating probability distribution Q is from the true distribution P. The KL-divergence is not symmetric and does not satisfy the triangle inequality, so it is not a distance in the true mathematical sense. It is still "often useful to think of relative entropy as a 'distance' between distributions" [2]. Further, Cover and Thomas note that in statistics, it is the expected logarithm of the likelihood ratio between P and Q when P is the true distribution.

Definition 2.4 (Mutual information)**.** Consider two random variables with a joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$. *Mutual information* $I(X;Y)$ is the relative entropy between the joint distribution and the product distribution $p(x)p(y)$:

$$\begin{aligned} I(X;Y) &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= D(p(x,y) \| p(x)p(y)) \\ &= \mathbb{E}_{p(x,y)} \log \frac{p(X,Y)}{p(X)p(Y)}. \end{aligned}$$

Cover and Thomas generalize this for continuous and a mixture of discrete and continuous variables in section 8 of [2]. Notice how the relationship between entropy and mutual information is connected via

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned} \tag{1}$$

where the mutual information is the reduction in uncertainty of X due to the knowledge of Y. The relationship between different types of entropy is expressed in a Venn diagram as in Figure 1, where we can distinguish how the mutual information is the intersection of the entropy of X and the entropy of Y.

Theorem 2.5. *Mutual information is invariant to invertible transformations, [13]. That is, for invertible functions ψ and ϕ*

$$I(X;Y) = I(\psi(X); \phi(Y)).$$

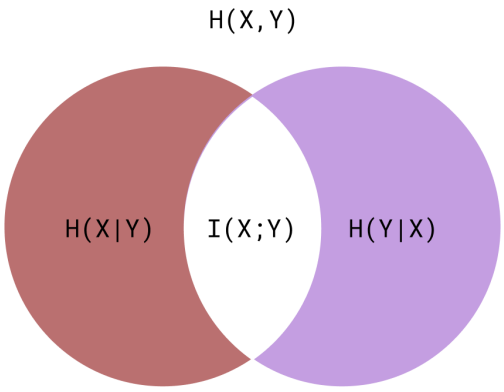


Figure 1: Relationship between entropy and mutual information. Adapted from Cover & Thomas [2].

2.1.2 Multi mutual information

Definition 2.6 (Interaction information)**.** For three variables V_1, V_2, V_3, the *interaction information* (or *co-information*) is defined as

$$I(V_1; V_2; V_3) = \sum_{i=1}^{3} H(V_i) - \sum_{i<j} H(V_i, V_j) + H(V_1, V_2, V_3).$$

Unlike bivariate mutual information, interaction information can be negative, allowing us to distinguish between two distinct informational states: synergy and redundancy.

Definition 2.7 (Synergy)**.** A negative value of interaction information indicates *synergy*, where the variables taken together are more informative than they are when taken separately.

Definition 2.8 (Redundancy)**.** A positive value of interaction information indicates *redundancy*, where terms are redundant in the information that they provide about the remaining variable.

Definitions 2.6, 2.7, and 2.8 can be found in Schneidman et al. [12]. Interaction information compares the information that two variables jointly provide about a third variable than what these two variables provide separately. However, for systems with $n \geq 4$ variables, interaction information is known to violate the interpretation that positive values indicate pure synergy and negative values indicate pure redundancy. This mathematical limitation motivates the introduction of O-information,

which captures only interactions that go beyond pairwise relationships, providing a more robust measure for high-dimensional systems [11].

Definition 2.9 (O-information)**.** The *O-information* of the system described by the random vector $\mathbf{X}^n$ is defined as

$$\Omega(\mathbf{X}^n) = (n-2)H(\mathbf{X}^n) + \sum_{j=1}^{n}[H(X_j) - H(\mathbf{X}^n_{-j})], \tag{2}$$

where $\mathbf{X}^n_{-j} = (X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_n)$. Beautifully, for the specific case of $n = 3$, the O-information simplifies and is mathematically equivalent to the interaction information

$$\Omega(X_1, X_2, X_3) = I(X_1; X_2; X_3) = I(X_i; X_j) - I(X_i; X_j|X_k),$$

for any permutation $\{i, j, k\} = \{1, 2, 3\}$. This expression explicitly measures the difference between redundancy and synergy. When $\Omega(X_1, X_2, X_3) > 0$, redundancy dominates. In contrast, when $\Omega(X_1, X_2, X_3) < 0$, synergy dominates [6].

2.2 The information bottleneck principle

The objective of supervised learning is "to capture and efficiently represent the relevant information in the input variable about the output" [15]. This is ultimately what we try to achieve with the information bottleneck principle [14].

2.2.1 Sufficient statistics

Before unpacking what the information bottleneck principle is, we introduce two well-known concepts for statisticians in information theoretic terms, namely *sufficient statistics* and *minimal sufficient statistics* which both rely on the *data-processing inequality* [2].

Theorem 2.10 (Data-processing inequality)**.** *If $Y \to X \to Z$ forms a Markov chain, then $I(X;Y) \geq I(Z;Y)$.*

That is, X is better at predicting Y than Z. We can now define the ideal representation $T(X)$ of our data X with respect to a target Y:

Definition 2.11 (Sufficient statistics)**.** A statistic $T(X)$ is called *sufficient* for Y if $I(T(X);Y) = I(X;Y)$.

In other words, a sufficient statistic retains all predictive power regarding Y and is called sufficient for Y if it contains all the information in X about Y.

Definition 2.12 (Minimal sufficient statistic)**.** A statistic $T(X)$ is a *minimal sufficient statistic* if it is a function of every other sufficient statistic.

Thus, the minimal sufficient statistic is the 'simplest' or most compressed mapping that captures all relevant information.

2.2.2 Rate distortion theory

To understand the information bottleneck principle, we first consider rate distortion theory, originally developed for lossy data compression [2]. A practical analogy is the compression of a high-resolution image into a small JPEG file. The rate represents the number of bits used to store the information (compression), while distortion measures the resulting loss of visual quality. In representation learning, Tishby et al. [14] define 'relevant information' as "the information that [a signal] x provides about another signal y." Intuitively, this implies that the only features of the input variable X worth retaining are those that assist in predicting the target Y; all other information should be treated as noise and compressed away. Let X be the input variable with a fixed probability mass function $p(x)$, and let Z denote its compressed representation. In the information bottleneck framework, we assume the Markov chain $Y \to X \to Z$. To satisfy the information bottleneck principle, Z must balance the trade-off between two competing measures:

- **Rate:** The amount of information the representation Z retains about the input X, measured by the mutual information

$$R = I(Z;X) = \mathbb{E}_{p(Z,X)}\left[\log \frac{p(X,Z)}{p(X)p(Z)}\right].$$

- **Distortion:** The expected distortion between X and its compressed representation Z is defined as

$$D = \mathbb{E}_{p(X,Z)}[d(X,Z)],$$

  where $d : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}^+$ is a distortion function, with $\mathcal{X}$ the input space and $\mathcal{Z}$ the reconstruction space, measuring the cost of representing x by z. One counterintuitive part of this theory is that instead of compressing each random variable separately (since they are independent), it turns out that joint descriptions are more efficient than individual ones — describing X_1 and X_2 requires fewer bits for the same distortion than compressing them separately.

Intuitively, this process implies that "we squeeze the information that X provides about Y through a 'bottleneck' " [14]. The objective of the network is to minimize the functional

$$\mathcal{L}(p(z|x)) = I(Z;X) - \beta I(Z;Y). \tag{3}$$

The positive Lagrange multiplier β acts as a 'resource budget'. When β is small, the model is forced to be extremely concise, potentially losing too much information. As β increases, the model is allowed to capture more complex features to improve accuracy, moving along the so-called information bottleneck frontier.

2.2.3 Application to deep learning

Consider an image classification problem with high-dimensional data such as a 1000×500 pixel image and a binary target $Y \in \{0, 1\}$, e.g., classifying whether an image contains a dog or not. This vast difference suggest that "most of the entropy of X is not very informative about Y" [15]. In practice, this optimization acts as a filter for the 500,000 pixels in X. By minimizing $I(Z;X)$ (see Equation (1)) under a constraint on $I(Z;Y)$, the network is forced to discard the vast majority of the image data that is irrelevant to the binary classification. The data processing inequality ensures that we are merely distilling existing information rather than creating it, resulting in the most compact representation of the image that still preserves its predictive properties.

3 Deep learning

In supervised learning, we aim to either solve a classification or regression problem where deep learning has proven to be highly successful in a range of problems. "The quintessential example of a deep learning model is the feedforward deep network" [4]. In such a network, information flows in a single direction, from the input $\mathbf{x}$ through a series of hidden layers to the output Y, as illustrated in Figure 2. Each hidden layer i can be viewed as a latent representation, which we denote Z_i to stay consistent with the information bottleneck framework.

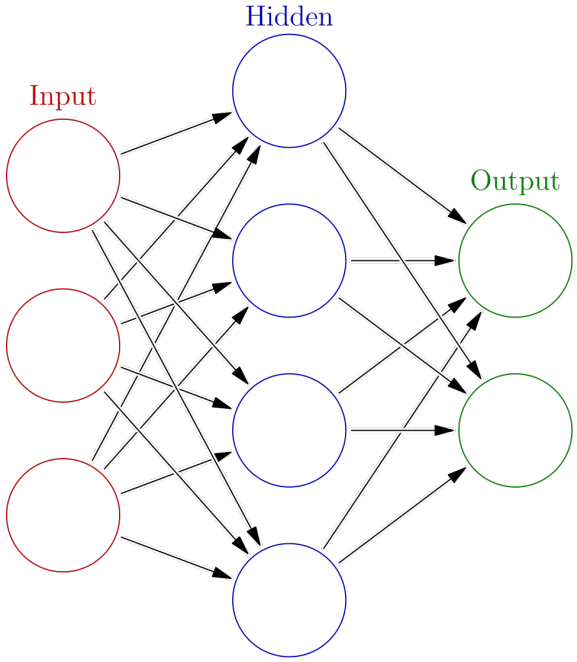


Figure 2: A colored 'vanilla' neural network diagram [3]. The network has three inputs, two outputs and a hidden layer consisting of four hidden units.

In this thesis, we treat the hidden layers as the 'bottleneck' variables (Z) through which the input X is compressed and distilled into relevant feature for predicting Y.

3.1 Training a neural network

In supervised neural networks, a feedforward network takes an input, applies a series of transformations to it, and produces an output. Training the model involves a process where a loss function is minimized by adjusting the networks parameters θ, such as weights and biases. "Feedforward neural networks are called networks because they are typically represented by composing together many different functions" [4]. For example, multiple functions $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ form the composition $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. In this framework, the depth of the network is defined by the number of composed layers. Crucially, since the composition of any number of affine (linear) transformations is itself an affine (linear) transformation, the model would collapse into a linear output without intervention. It is the addition of the **activation function** between these layers that provides the architecture to learn from complex, non-linear mappings required for high-dimensional data.

3.1.1 The architecture: From linear to nonlinear

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a sample of n examples where each example has d dimensions. Let $\mathbf{Z} \in \mathbb{R}^{n \times h}$ denote the hidden representation (the bottleneck variable). Since the hidden and output layers are fully connected, we introduce the hidden layer weights $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times h}$ and biases $\mathbf{b}^{(1)} \in \mathbb{R}^{1 \times h}$; and output layer weights $\mathbf{W}^{(2)} \in \mathbb{R}^{h \times q}$ and biases $\mathbf{b}^{(2)} \in \mathbb{R}^{1 \times q}$ [16]. For a one-hidden-layer feedforward network (also called a multilayer perceptron), the representation $\mathbf{Z}$ and the final output $\mathbf{Y} \in \mathbb{R}^{n \times q}$ are calculated as

$$\mathbf{Z} = \sigma(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}),$$
$$\mathbf{Y} = \mathbf{Z}\mathbf{W}^{(2)} + \mathbf{b}^{(2)},$$

where σ is a nonlinear activation function. Mathematically, each layer applies an affine transformation $\mathbf{W}^\top \mathbf{x} + \mathbf{b}$ to each of the n examples, followed by the activation function σ. The primary purpose of the activation function is to introduce non-linearity, enabling the network to learn complex mappings that would be impossible with purely linear transformations [7]. It is worth noting that in most neural networks, there are multiple hidden layers to increase the network's capacity to learn complex representations. Here, we have restricted ourselves to one hidden layer for simplicity's sake.

3.1.2 Activation functions

A *neuron* is the connected unit or node in the artificial network. Activation functions "decide whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it" [16]. Some of the most popular choices of activation functions are ReLU or sigmoid. A deeper dive into these functions is outside the scope of this thesis but we will give a brief description to build intuition around their purpose. We give brief definitions of two common activation functions [16]:

ReLU Given an element x, the function is defined as the maximum of that element and 0 as

$$\mathrm{ReLU}(x) = \max(x, 0).$$

ReLU is a piecewise linear function, that is differentiable at all values except for 0. It retains only positive values and discards negatives by setting the corresponding activations to 0. In the non-differentiable case, we default to set the value to 0, when the input is 0.

Sigmoid The sigmoid function is defined over the domain $\mathbb{R}$, with the outputs in the interval $(0, 1)$, as the function

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}.$$

We can think of the sigmoid as a special case of softmax. Sigmoids are especially valuable in binary classification problems when we want to interpret the outputs as probabilities.

3.1.3 Parameter estimation: Maximum likelihood estimation

Notation: We use ℓ for the log likelihood (and loss function) throughout this thesis, reserving $\mathcal{L}$ for the information bottleneck Lagrangian (Equation (3)).

When training a neural network, a systematic approach is required to select the parameters $\boldsymbol{\theta}$ that best describe the underlying data. Rather than an ad-hoc choice of an estimator, we employ the well-established statistical principle of *maximum likelihood estimation* (MLE). The objective is to determine a parameter vector $\boldsymbol{\theta}$ that maximizes the log likelihood $\ell(\boldsymbol{\theta}; \mathbf{X})$ of observations of m independently distributed random variables $\mathbf{X} = (\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(m)})$ under a chosen model distribution p_{model} [4]:

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\text{model}}(\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \mathbf{X}).$$

As the sample size $m \to \infty$, we can apply the law of large numbers. This principle states that the empirical average of a function converges with probability 1 to its expectation under the true data-generating distribution

$$\frac{1}{m} \sum_{i=1}^{m} f(\mathbf{X}^{(i)}) \to \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[f(\mathbf{X})].$$

Applying this to the log likelihood, the asymptotic limit $\boldsymbol{\theta}_{ML}$ of the maximum likelihood estimator can be expressed as an expectation over the true distribution

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})\right]. \tag{4}$$

For a finite m in Equation (4) "any loss consisting of a negative log likelihood is a cross-entropy between the empirical distribution defined by the training set and the probability distribution defined by the model" [4].

3.1.4 Optimization: Stochastic gradient descent

Nearly all of deep learning is "powered by one very important algorithm: *stochastic gradient descent* (SGD)" [4]. Before unpacking the stochastic nature of the algorithm, we first establish the framework of *gradient descent* by recalling foundational results from multivariable calculus.

Gradient descent: For a continuous and differentiable function $f(\boldsymbol{\theta})$, a necessary condition for a local minimum is that the gradient $\nabla f(\boldsymbol{\theta})$ must be zero. Gradient descent utilizes partial derivatives to identify a path that reduces $f(\boldsymbol{\theta})$ iteratively. The steepest descent direction is given by the negative gradient, $-\nabla f(\boldsymbol{\theta})$, allowing us to construct the update rule

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha_k \nabla f(\boldsymbol{\theta}_k),$$

where $\alpha_k > 0$ is the **step size**, commonly referred to as the *learning rate* in the deep learning literature.

Theorem 3.1. *Assume $\nabla f(\boldsymbol{\theta}) \neq 0$. The direction $\mathbf{d} = -\nabla f(\boldsymbol{\theta})$ is a descent direction.*

In practice, the algorithm begins at an initial parameter estimate $\boldsymbol{\theta}_0$ and iterates until a convergence criterion is met. Recalling the maximum likelihood estimation from Equation (4), computing the expectation exactly "is very expensive because it requires evaluating the model on every example in the entire dataset" [16]. Modern machine learning algorithms overcome this by randomly sampling smaller subsets of the data.

Stochastic gradient descent: The transition from gradient descent to stochastic gradient descent (SGD) is motivated by computational efficiency. Instead of calculating the gradient over the entire dataset, we compute it over a smaller subset, or *mini-batch*, where it is "crucial that the minibatches be selected randomly" [4].
By ensuring that samples are drawn *i.i.d.* (*independent and identically distributed*) from the training set, the mini-batch gradient serves as an *unbiased estimator* of the true population gradient [16], since

$$\mathbb{E}_i \nabla f_i(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\boldsymbol{\theta}) = \nabla f(\boldsymbol{\theta}).$$

While individual steps may be noisy due to this sampling variance, the algorithm, on average, follows the correct descent direction. Multiple variations of SGD exist,

such as those incorporating momentum or adaptive learning rates, to ensure faster convergence, though a definitions of these remain outside the scope of this thesis.

3.1.5 Backpropagation

In a feedforward network, the training process begins with a *forward pass*: an input $\mathbf{X}$ is propagated through the hidden layers to produce a prediction $\hat{\mathbf{Y}}$. This prediction is evaluated against the true target $\mathbf{Y}$ using a scalar cost function $\ell(\boldsymbol{\theta})$, which quantifies the prediction error. Once this cost is computed, the backpropagation algorithm "allows the information from the cost to then flow backward through the network in order to compute the gradient" [4]. In other words, we compute $\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})$ to determine how much each parameter contributed to the error, allowing us to update the weights via stochastic gradient descent. While our ultimate goal is to reach a local minimum where the gradient is zero ($\nabla\ell = 0$), solving this analytically in a deep neural network is computationally difficult. Instead, backpropagation achieves this systematically by computing the gradients of the weights and biases for each hidden layer through the multivariable chain rule [4]. Recalling our conceptualization of the network as a composition of functions $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$, the gradient with respect to the input is calculated as

$$\frac{\partial f}{\partial \mathbf{x}} = \frac{\partial f^{(3)}}{\partial f^{(2)}} \frac{\partial f^{(2)}}{\partial f^{(1)}} \frac{\partial f^{(1)}}{\partial \mathbf{x}}.$$

Back-propagation "sequentially calculates and stores the gradients of intermediate variables and parameters within the neural network in the reversed order" [16]. The forward pass carries the information, the back-propagation the cost. Without these two passes, the network cannot learn.

3.1.6 Regularization (weight decay)

Regularization, also called *weight decay*, is standard practice in many supervised learning settings to improve numerical stability and avoid *overfitting*, effectively reducing test error. The regularizing objective function $\tilde{\ell}$ is given by [4]

$$\tilde{\ell}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = \ell(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \alpha\mathcal{R}(\boldsymbol{\theta}),$$

where $\alpha \in [0, \infty)$ is a hyperparameter that weights the relative contribution of the norm penalty term, $\mathcal{R}$, relative to the standard objective function ℓ. For deep neural networks, the norm penalty $\mathcal{R}$ typically only penalizes the *weights* of the affine transformation at each layer "and leaves the biases unregularized" [4]. Taking

the gradient and applying stochastic gradient descent with learning rate η, and noting that $\mathcal{R}$ only penalizes the weights, i.e., $\boldsymbol{\theta} = \mathbf{w}$, the L2 choice $\mathcal{R}(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top\mathbf{w}$ gives

$$\tilde{\ell}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \frac{\alpha}{2}\mathbf{w}^\top\mathbf{w} + \ell(\mathbf{w}; \mathbf{X}, \mathbf{y}).$$

Straightforward but tedious derivations show that the gradient is

$$\nabla_{\mathbf{w}}\tilde{\ell}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha\mathbf{w} + \nabla_{\mathbf{w}}\ell(\mathbf{w}; \mathbf{X}, \mathbf{y}),$$

with weight update

$$\mathbf{w} \leftarrow (1 - \eta\alpha)\mathbf{w} - \eta\nabla_{\mathbf{w}}\ell(\mathbf{w}; \mathbf{X}, \mathbf{y}).$$

Often $\mathcal{R}$ is a quadratic function of the weights of the affine transformations of the neural network. Alternatively, the L_1 norm (LASSO) can be implemented for more sparse solutions. In the context of representation learning and grokking, weight decay is often a critical hyperparameter, as it implicitly biases the network towards finding simpler, more generalizing sub-networks once the training data has been fitted.

3.1.7 Generalization and learning curves

At the core of statistical learning, we must determine whether what we have observed in our data is justified only for a single observation, or if it generalizes to unseen samples from the same distribution. To quantify this, we differentiate between the *training error* and the *generalization error* [5]. The former represents the average loss over the finite training set, while the latter represents the expected loss over the entire data-generating distribution p_{data}. In order to define the training error we assume that training data $(\mathbf{X}, \mathbf{y}) = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^m$ is available, and that it is used to fit a nonlinear regression model $\mathbf{Y}_i = f(\mathbf{x}_i, \boldsymbol{\theta}) + \varepsilon_i$ for $i = 1, \ldots, m$.

Definition 3.2 (Training error)**.** The *training error* is defined as

$$\mathcal{E}_{\text{train}}(\boldsymbol{\theta}) = \frac{\ell(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y})}{m}.$$

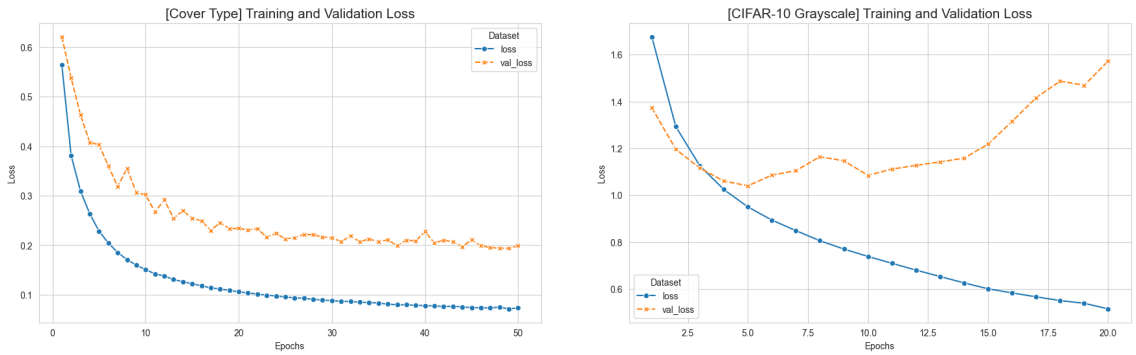
Definition 3.3 (Generalization error)**.** The *generalization error* is defined as

$$\mathcal{E}_{\text{test}}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{X},\mathbf{Y})\sim p_{\text{data}}}\left[\ell\left(\mathbf{Y}, f(\mathbf{X}; \boldsymbol{\theta})\right)\right].$$

The difference between these two quantities, $\mathcal{E}_{\text{test}}(\boldsymbol{\theta}) - \mathcal{E}_{\text{train}}(\boldsymbol{\theta})$, is known as the *generalization gap*. To visualize the optimization process and monitor the generalization gap, it is standard practice to plot the training and test errors iteratively

over time (measured in epochs). An *epoch* is a complete pass of the entire training dataset through the algorithm to update its internal parameters. These resulting plots are known as *learning curves*.

Figure 3 illustrates two distinct training dynamics from prior coursework in Statistical Deep Learning (MT7042) from Stockholm University. In Figure 3a, a neural network was trained on approximately 580,000 observations for a 7-class cover type classification problem with 54 features. Due to severe class imbalance, oversampling from minority classes was applied to adjust the class distribution. The resulting learning curve shows a steady decrease in both training and generalization error over 50 epochs (blue and orange curves respectively), with a stable gap between the two — indicating a consistent generalization gap without overfitting. In contrast, Figure 3b illustrates a neural network trained on an image classification problem with 60,000 images uniformly distributed across 10 classes. Here, we observed severe overfitting after just 5 epochs. This widening generalization gap indicates that the model has begun to memorize the statistical noise of the specific training sample rather than learning the underlying distribution p_{data}. (In the original project, multiple regularization techniques were subsequently implemented to stabilize the network during training)



(a) A well-behaved learning curve showing convergence and good generalization.

(b) A learning curve exhibiting severe overfitting after epoch 5.

Figure 3: Empirical learning curves demonstrating different generalization behaviors. Figures from previous coursework at Stockholm University.

3.1.8 Grokking: A delayed generalization

In the sci-fi novel *Stranger in a Strange Land* (1961), the word 'grok' means to understand something so deeply that it becomes part of you. In the context of machine learning, Power et al. [10] demonstrated that neural networks can exhibit unusual behavior compared to traditional learning processes. They observed that for specific algorithmic problems, networks learn through a process of 'grokking' a

pattern, which improves generalization performance from random chance to perfection long after the training error has vanished. When training a network on a modular addition table, it has been noted that models "abruptly transition to a generalizing solution after a large number of training steps, despite initially overfitting" [9, 10]. While some suggest that this might be due to stochastic gradient descent performing a random walk on the optimal manifold, the authors conclude that their results are consistent with Barak et al. (2022, as cited in [10]), showing that the network instead makes continuous progress toward the generalizing algorithm. In the experiments by Nanda et al. [9], networks trained on a modular addition task consistently exhibited grokking. The authors demonstrate that the network initially achieves 100% training accuracy with a training loss that quickly declines, while test accuracy remains low and test loss remains high, as can be seen in Figure 4. However, after approximately 10,000 epochs, the network begins to generalize, with test accuracy reaching near 100%. Informally, this transition suggests that the network moves from memorizing specific answers to learning the underlying algorithmic method required to solve the problem. While Nanda et al. rely on specific progress measures to analyze the network's internal state, such measures are argued to be prone to subjective bias and lack scalability to larger models [1]. Furthermore, Clauw et al. contend that these measures are task-specific, proposing instead the use of information theory as a "task-independent tool to identify emergent sub-networks in neural networks for mechanistic interpretability [1]".

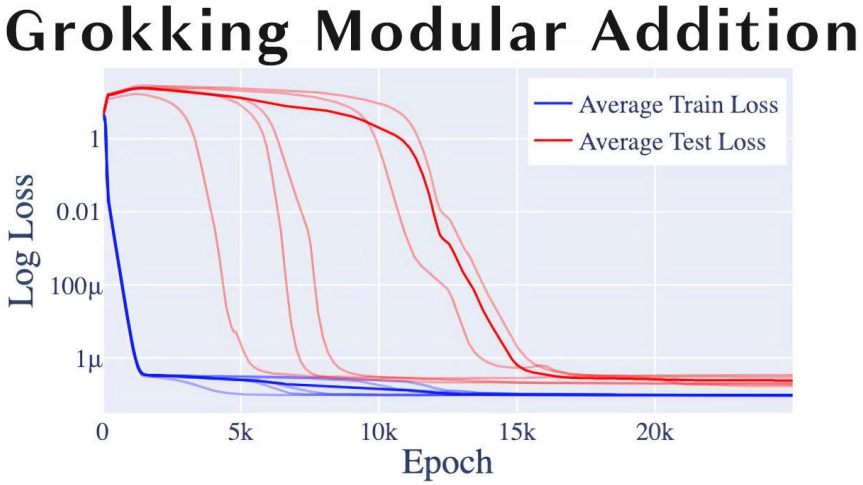


Figure 4: Training and test loss curves showing the sudden emergence of generalization (grokking) in a neural network learning modular addition. Image by Neel Nanda, used under CC BY-SA 4.0. [8].

4 Information theory and deep learning

As discussed previously, mutual information can be used to quantify the relationship between input variables, the network's hidden layers, and its output [13]. Here, we discuss the findings of Clauw et al. [1] regarding compression-driven phase transitions.

4.1 The information plane

Recall that the primary objective of the information bottleneck method (Equation (3)) is to "find a maximally compressed mapping of the input variable that preserves as much as possible the information on the output variable" [15]. By plotting the mutual information $I(Z;X)$ against $I(Z;Y)$, as can be seen in Figure 5, we can visualize the network's evolution through the lens of different values for β. This visualization is commonly referred to as the *information plane*, the *information curve*, or the *IB distortion curve*. Within this plane, there exists a theoretical information limit (called IB-limit in Figure 5) representing the optimal tradeoff between compression and prediction. Furthermore, the information plane may have *bifurcation points* (points where the optimal solutions splits into multiple sub-optimal curves) "at critical values of β" [15] (called β-curves in Figure 5). Optimally, deep neural networks (DNNs) should learn to extract the most efficient informative features — approximating minimal sufficient statistics — with the most compact architecture possible. We go on to study the information paths of deep neural networks in the information plane.

4.1.1 Visualizing DNNs in the information plane

Two properties of mutual information are critical in the context of DNNs: its invariance to invertible transformations (see Theorem 2.5) and the data processing inequality (see Theorem 2.10) [13].

Invariance to invertible transformation: Since invertible transformations retain all information, mutual information cannot distinguish between different representations that carry the same information. This is precisely why mutual information is well-suited to studying DNNs: invertible transformations of the representations "generate equivalent representations even if the individual neurons encode entirely different features of the input". However, this invariance comes at a cost — mutual information is "insensitive to the complexity of the function $f(x)$ or the class

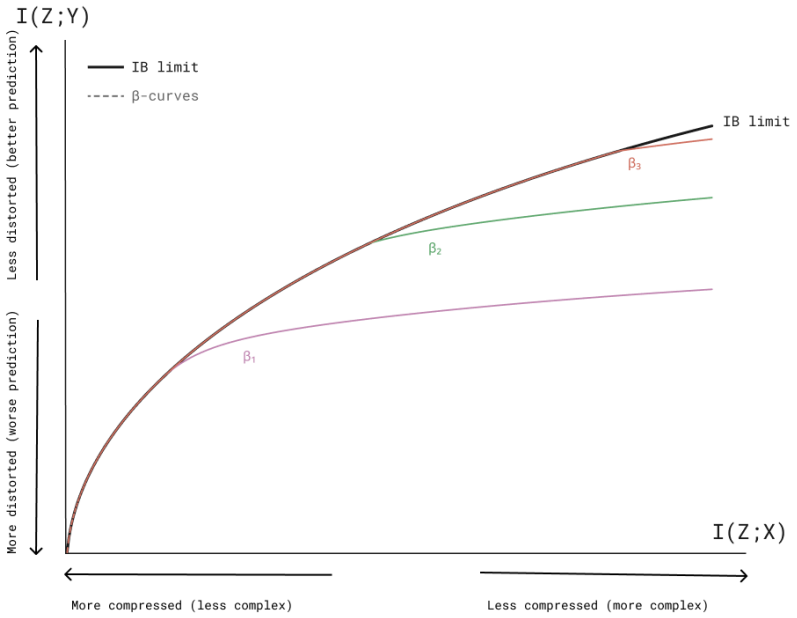


Figure 5: The information plane illustrating the IB limit and sub-optimal β-curves. Adapted from [6].

it comes from" [13], meaning we can mistake a low-complexity classifier for a high-complexity one using mutual information alone. The cure is adding noise, which we leave outside the scope of this thesis.

The data processing inequality: The data processing inequality also plays a crucial role here: it guarantees that no transformation in the network can generate information that was not already present, and as an immediate consequence "information about Y that is lost in one layer cannot be recovered in higher layers" [13].

These fundamental properties make DNNs particularly powerful to study through the information-theoretic lens, and motivate the claim that "optimized DNN layers should approach the information bottleneck (IB) bound" [13]. When plotting $I(Z;Y)$ against $I(Z;X)$, "two optimization phases are clearly visible in all cases" [13]. The initial phase, which is fast and relatively short, is termed *empirical error minimization* (ERM). During this phase, the network rapidly increases the mutual information $I(Z;Y)$ between Z and the labels Y, effectively "fitting" the data. Following this *fitting phase*, the network enters a significantly longer training phase where the layers Z reduce their mutual information $I(Z;X)$ regarding the input X. This process effectively compresses the internal representations to retain only the most relevant features and is referred to as the *representation compression phase*.

The full trajectory of a DNN through these two phases is illustrated in Figure 6. While Shwartz–Ziv and Tishby attribute the onset of the compression phase to a decrease in the signal-to-noise ratio (SNR) of the gradients, this thesis focuses on the informational dynamics of the representations themselves rather than the underlying stochastic gradient mechanics.

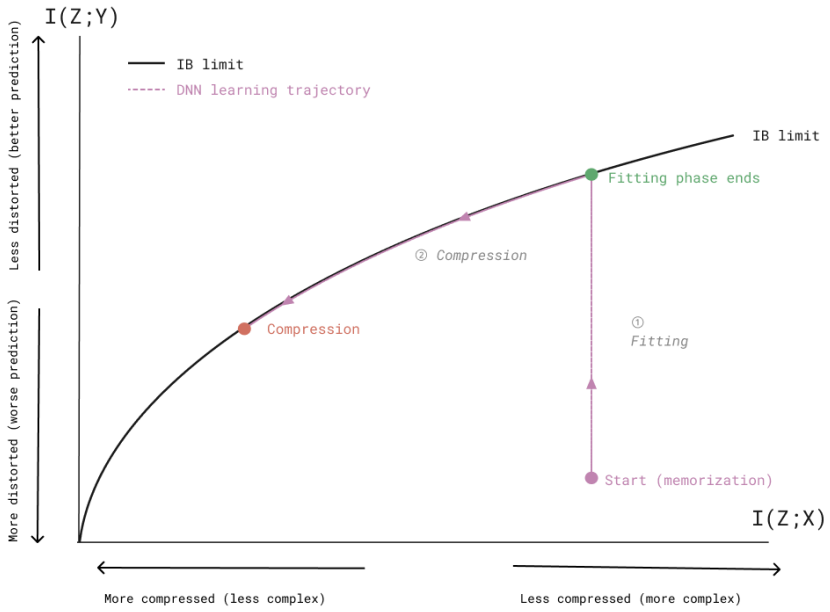


Figure 6: Learning trajectory of a DNN in the information plane, showing the fitting phase followed by representation compression.

4.2 Information dynamics under grokking

A note on measurement: in Section 4.1, the hidden layer Z is treated as a single high-dimensional unit, and information is measured between Z and input/output via the bivariate mutual information $I(Z;Y)$ and $I(Z;X)$. Clauw et al. instead partition the hidden layer Z into a number of clusters (10 in their experiments) and compute multivariate measures — synergy and redundancy — from the feature vectors of these clusters. Unless otherwise stated, the following is based on [1]. Clauw et al. investigate higher-order interactions between neurons to understand the mechanics of grokking. While pairwise mutual information has been shown to identify important features, it can only measure the dependency between two variables. Multivariate mutual information instead "provides a fine-grained analysis of the statistical interactions between multiple variables by decomposing the dependencies into synergy and redundancy". We therefore shift our analytical lens away from the layer-level measures of the information bottleneck framework — $I(Z;X)$ and

$I(Z;Y)$ — toward the neuron-level interactions within a single layer. To quantify these interactions within a network, the authors search for combinations of neurons (multiplets) that maximize either synergy (see Definition 2.7) (yielding the lowest, or most negative, O-information) or redundancy (see Definition 2.8) (yielding the highest, or most positive, O-information) according to Definition 2.9. That is, synergy refers to the cooperation between variables as a whole, and redundancy is the shared information between variables. In the empirical analysis, these normalized metrics are tracked along the loss. Monitoring this informational evolution over time "reflects the training dynamics, to measure the model's progress during training". This analysis identifies three distinct phases during training, namely:

- **Feature Learning:** In the first phase, a low synergy is observed while redundancy is initially high. "The lack of coupling between features suggests the network is independently learning features, while the high redundancy indicates it is extracting basic patterns with similar properties."

- **Emergence:** During this phase, the model trades off redundancy for synergy. "This suggests a critical phase transition from memorization where the model attempts to combine features to emerge a generalizing synergistic sub-network that is growing as it explores." In the low weight decay setting, this phase does not directly lead to a generalizing solution; instead, the network enters a *divergent* phase before re-entering a *delayed emergence* sub-phase where "the model escaped the sub-optimal solution and is now able to combine features to form a generalizing solution."

- **Decoupling:** In this phase, synergy decreases, redundancy increases, and test accuracy increases. "This suggests the model has found a generalizing sub-network but not all features are relevant so it removes coupled interactions for compression."

Furthermore, weight decay (see Section 3.1.6) is demonstrated to play a critical role in the onset of grokking. Low weight decay (that is, a small amount of regularization) severely delays the *emergence phase*. Increasing weight decay can accelerate this emergence, while excessively high weight decay may prevent the model from achieving grokking altogether. As illustrated in Figure 7, "synergy peaks early during training could predict grokking" [1].

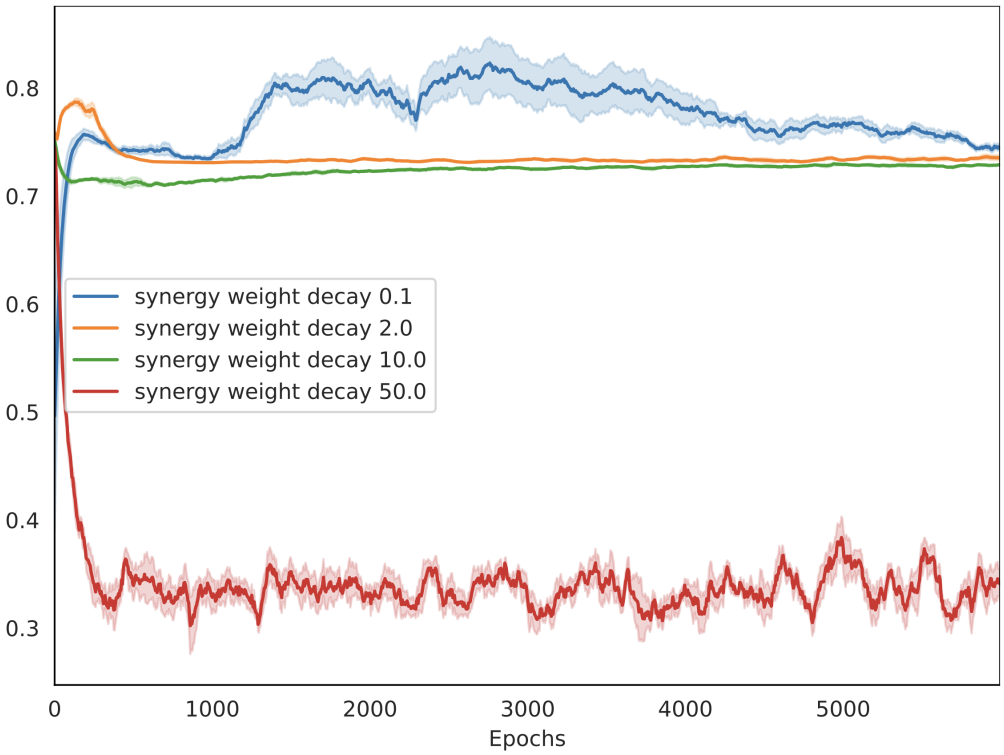


Figure 7: Synergy across training epochs for models trained with weight decay values of 0.1, 2.0, 10, and 50. Models with low to moderate weight decay (0.1, 2.0) exhibit an early synergy peak prior to grokking, while higher weight decay values (10, 50) do not. Reproduced from [1] under CC BY 4.0.

The interplay between synergy dynamics and information compression suggests a possible link between Clauw et al.'s findings and the broader information bottleneck framework. We return to this connection in the discussion.

5 Conclusions

5.1 Summary

This thesis has reviewed two complementary frameworks for understanding representation learning in deep neural networks: the information bottleneck principle by Tishby and Zaslavsky [15] with the corresponding information plane, and the extension to multivariate mutual information by Clauw et al. [1] using synergy and redundancy to quantify interactions between neurons. Both frameworks were used to interpret the phenomenon of grokking — the delayed generalization first reported by Power et al. [10] where neural networks abruptly transition from memorization to generalization long after the training error has vanished. The bottleneck framework formulates representation learning as an optimization problem between rate and distortion governed by the Lagrange multiplier β (see Equation 3). This framework quantifies the relationship between input variables, the network's hidden layers, and its output. Deep neural networks pass through two distinct phases in the information plane: an initial empirical error minimization phase (fitting phase) followed by a longer representation compression phase, as illustrated in Figure 6. Clauw et al.'s framework instead operated at the neuron level - utilizing O-information to decompose multi mutual information into synergy and redundancy. Their analysis identifies three distinct phases — feature learning, emergence, and decoupling — with weight decay playing a critical role in driving the network through the emergence phase toward generalization. Across both frameworks, the goal of supervised deep learning can be expressed in information-theoretic terms as a tradeoff between compression and prediction: the network must first capture relevant information about the target before discarding what is not. Grokking, viewed through this lens, becomes a candidate example for delayed emergence — a subject we will return to in the interpretation that follows.

5.2 Interpretation

What does grokking look like through the lens of the information bottleneck framework? We propose that the three phases Clauw et al. identify (feature learning, emergence, and decoupling) map onto distinct regions of the information plane, with decoupling corresponding to compression, as illustrated in Table 1. While the two are measured at different scales — pairwise interactions between neurons versus mutual information between a layer and the input — they share a common signature: the network reducing redundant or coupled information after finding a generalizing

solution. As Clauw et al. describe it, grokking itself is "an emergent phase transition where the model collectively combines features to solve arithmetic tasks" [1], which we interpret as the network entering the emergence-then-decoupling phase that corresponds to representation compression in the information bottleneck framework. Whether this correspondence is mechanistic or merely analogous remains an open question.

Table 1: Proposed correspondence between the phases identified by Tishby and Zaslavsky (mutual information framework) and Clauw et al. (redundancy/synergy framework).

Learning phase	**Mutual information**	**Redundancy/Synergy**
1	Empirical Error Minimization	Feature Learning
2	Empirical Error Minimization	Emergence
3	Representation Compression	Decoupling

5.3 Points forward

Measuring different scales: A critical limitation of this interpretation is that the two frameworks use different measures. Although both stem from mutual information, they capture different types of interactions — pairwise mutual information between a layer and the input/output in Tishby's framework, versus higher-order interactions between neurons in Clauw et al.'s framework. A rigorous test of the proposed correspondence would require mapping these scales onto one another, for instance by tracking $I(Z;X)$, $I(Z;Y)$, synergy, and redundancy simultaneously on the same model during training.

Linear separability condition: Tishby and Zaslavsky also propose a connection between IB phase transitions and a linear separability condition involving second-order correlations in the data [15]. Exploring whether this condition aligns with the emergence phase identified by Clauw et al. is a promising direction for future work, but lies beyond the scope of this thesis.

Connections to double descent: Recent research has begun exploring connections between grokking and double descent. This thesis does not address double descent through an information-theoretic lens, leaving a unified account of these related phenomena to future work.

References

[1] Kenzo Clauw, Sebastiano Stramaglia, and Daniele Marinazzo. Information-theoretic progress measures reveal grokking is an emergent phase transition. *arXiv preprint arXiv:2408.08944*, 2024.

[2] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2006.

[3] Glosser.ca. Colored neural network. https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg, 2013. Wikimedia Commons. Licensed under CC BY-SA 3.0. Accessed: 2026-03-19.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[5] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2 edition, 2009.

[6] Chun-Biu Li. Information theory (lecture notes). Department of Mathematics, Stockholm University, 2025. Course MT7037.

[7] Chun-Biu Li. Statistical aspects of deep learning (lecture notes). Department of Mathematics, Stockholm University, 2026. Course MT7042.

[8] Neel Nanda. Training and test loss curves for grokking in modular addition, 2023. Available via Wikimedia Commons under CC BY-SA 4.0. Accessed 28 January 2026.

[9] Neel Nanda, Lawrence Chan, Tom Liberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations (ICLR)*, 2023.

[10] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[11] Fernando E. Rosas, Pedro A. M. Mediano, Michael Gastpar, and Henrik J. Jensen. Quantifying high-order interdependencies via multivariate extensions of the mutual information. *Phys. Rev. E*, 100:032305, Sep 2019.

[12] Elad Schneidman, Susanne Still, Michael J Berry II, and William Bialek. Network information and connected correlations. *Physical Review Letters*, 91(23):238701, 2003.

[13] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[14] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[15] Zaslavsky Tishby. Deep learning and the information bottleneck principle. *arXiv preprint arXiv:1503.02406*, 2015.

[16] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. https://d2l.ai.