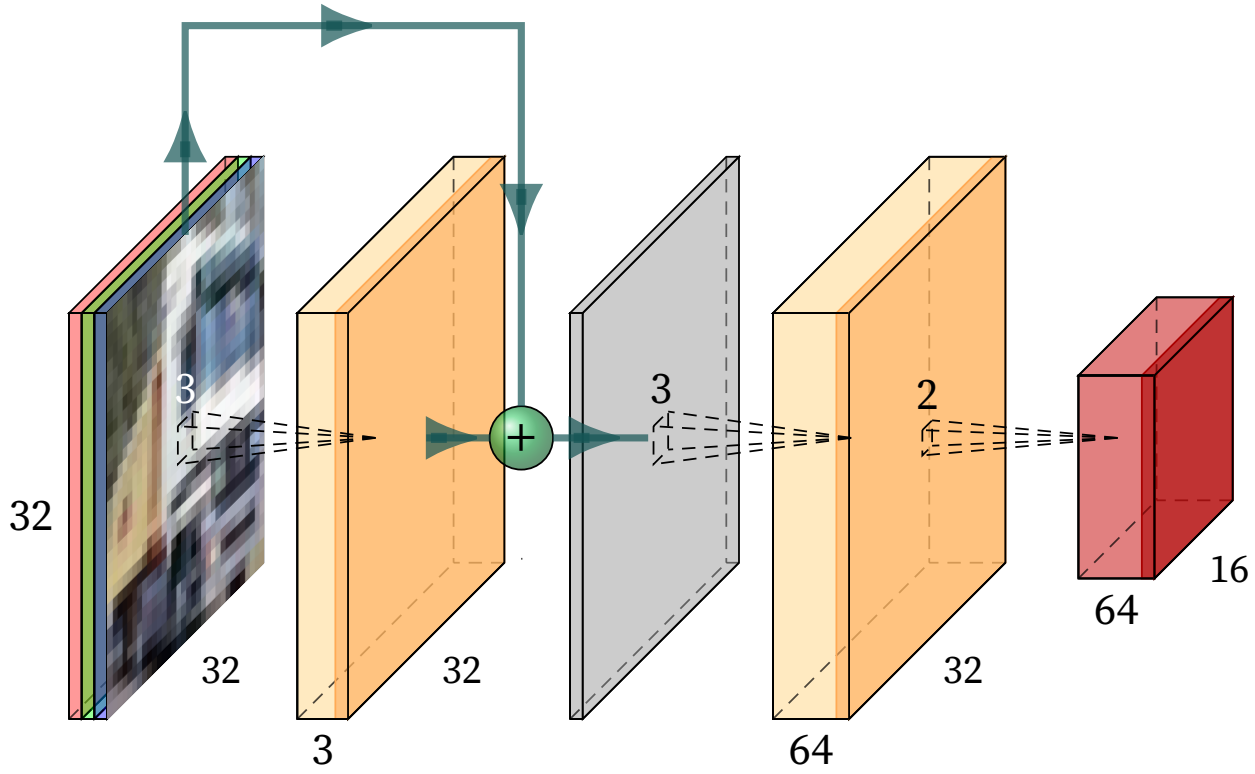
CHALMERS
UNIVERSITY OF TECHNOLOGY



Explaining Deep Neural Networks using Information Theory and Geometry

Master's thesis in Data science and AI

ZIYONG CHEONG

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025

Master's thesis 2025

Explaining Deep Neural Networks using Information Theory and Geometry

ZIYONG CHEONG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

Explaining Deep Neural Networks using Information Theory and Geometry
ZIYONG CHEONG

Master's thesis 2025
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31-772 1000

Supervisor: Giuseppe Durisi, Department of Electrical Engineering
Examiner: Giuseppe Durisi, Department of Electrical Engineering

Acknowledgements, dedications, and similar personal statements in this thesis, reflect the author's own views.

Cover: Illustration of a convolutional neural network used to classify images. Specifically, f_+^1 as described in section 2.3.

Typeset in LaTeX
Gothenburg, Sweden 2025

Explaining Deep Neural Networks using Information Theory and Geometry
ZIYONG CHEONG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Deep neural networks (DNNs) have achieved remarkable success in various machine learning tasks, yet a satisfactory explanation for their generalisation performance remains elusive. In this thesis, we stand on the shoulders of giants and explore two extensions to the information bottleneck (IB) principle, which formalises the idea that DNNs are ultimately compression machines which turn input data into minimal sufficient representations.

The first extension is based on geometry and replaces the information-theoretic IB with neural collapse (NC), a phenomenon where the penultimate layer representations of the training data converge to a regular simplex, a configuration of points that maximises pairwise distance. The second extension enforces more realistic constraints on the computation of mutual information used in IB, by taking into account the functional family used to decode said information. This leads to the natural definition of $\mathcal{V}$-information.

We build on these ideas by extending them to all layers of a DNN. Motivated by real-world computational constraints on computing NC and $\mathcal{V}$-information, we develop a (to our knowledge) novel algorithm for measuring NC, and (re)discover theoretical motivations behind many current practices in DNN training. Experiments indicate that compression, both geometric and information-theoretic, is not necessary for generalisation, yet DNNs *do* compress in a predictable manner, with compression increasing as one moves deeper into the network.

Keywords: neural collapse, information theory, decodable information bottleneck, compression, generalisation

Acknowledgements

I would like to express my deepest gratitude to Janek for being extremely supportive throughout this thesis project and helping me carry some of the emotional (and computational) load. I also want to extend my thanks to my supervisor/examiner Giuseppe for his fantastic feedback and guidance, as well as to my other supervisor, Bernhard, for the invaluable brainstorming sessions the three of us had together.

I am grateful to my family for their support, especially my mum, who made me the person I am today, and my sister, who has been a fountain of empathy for the hardships involved with being a foreign student.

I also want to thank my friends for being a great support network and helping me unwind and vent after particularly stressful days. Finally, I want to thank those at C3SE for providing me with ample computational resources to run my experiments.

Ziyong Cheong, Gothenburg, 2025

Contents

1. The Question of Generalisation

Deep neural networks (DNNs) currently dominate the field of supervised learning and machine learning in general. This is in large part due to their remarkable ability to *generalise*, i.e., perform well on data unseen during training. However, *why* they generalise remains an open question. In fact, their generalisation performance is in some sense paradoxical, as highlighted by Zhang et al. (2021), who showed that DNNs can easily fit randomly labelled data, a result reproduced in figure 1.1.

1.1. The Information Bottleneck

One attempt to explain this is the information bottleneck (IB) principle (Tishby and Zaslavsky 2015; Shwartz-Ziv and Tishby 2017), which posits that learning happens due to compression of the input data into a representation that is *minimal sufficient*. Formally, they adopt the framework of statistical learning theory (Vapnik 1999), which assumes that the training and test datasets consist of independent and identically distributed (i.i.d.) samples drawn from the unknown joint distribution $p_{\mathbf{x},\mathsf{y}}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathbf{x}$ is the multivariate random variable (m.r.v.)—i.e., the random vector—of the input, y is the random variable (r.v.) of the label, and $\mathcal{X}$, $\mathcal{Y}$ their supports. For a full overview of the notation used in this thesis, see chapter A.

A representation is *sufficient* when its m.r.v. $\mathbf{h}$ contains all information necessary to predict the corresponding label. In other words, $\mathrm{I}(\mathbf{h};\mathsf{y}) = \mathrm{I}(\mathbf{x};\mathsf{y})$ (Cover and Thomas 2005, p. 36), where

$$\mathrm{I}(\mathsf{a};\mathsf{b}) = \mathbb{E}_{\mathsf{a},\mathsf{b}}\left[\log\left(\frac{p_{\mathsf{a},\mathsf{b}}(\mathsf{a},\mathsf{b})}{p_{\mathsf{a}}(\mathsf{a})p_{\mathsf{b}}(\mathsf{b})}\right)\right] \tag{1.1}$$

is the *mutual information* (MI) between (m.)r.v.s a and b. The representation is then *minimal sufficient* when it is also as efficient as possible, i.e., it can be expressed as a

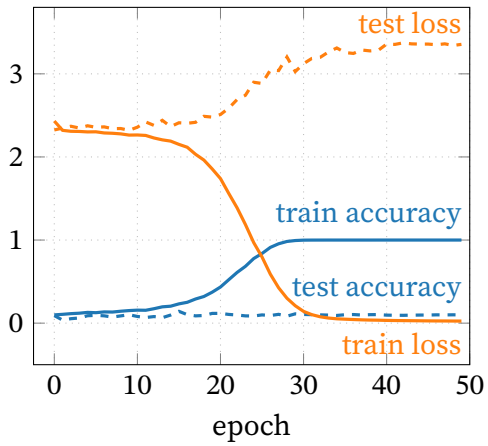


Figure 1.1.: The accuracy and loss curves of a DNN perfectly fitting randomly labelled data. Despite clearly being overparametrised, the same DNN achieves a test accuracy of 91% on nonrandom labels (c.f. figure 3.1b with more details in sections 2.3 and 3.1).

function of any other sufficient representation. This can be formulated in terms of the Markov chain $y \to \mathbf{x} \to \mathbf{g} \to \mathbf{h}$ (Cover and Thomas 2005, p. 37), where $\mathbf{g}$ is the m.r.v. of some sufficient representation and $a \to b$ means b only depends on a.

As an immediate consequence of the data processing inequality (Cover and Thomas 2005, pp. 34–35), which informally states that information cannot be gained through computation alone, we have $I(\mathbf{g}; y) = I(\mathbf{h}; y) = I(\mathbf{x}; y) \geq I(\mathbf{g}; \mathbf{x}) \geq I(\mathbf{h}; \mathbf{x})$ and thus

$$\mathbf{h} = \operatorname*{arg\,min}_{\mathbf{g} : I(\mathbf{g};y)=I(\mathbf{x};y)} I(\mathbf{g}; \mathbf{x}).$$

The Lagrangian relaxation of the constraint $I(\mathbf{g}; y) = I(\mathbf{x}; y)$, which may not be attainable depending on the architecture at hand, results in the IB objective

$$\operatorname*{arg\,min}_{\mathbf{h}} I(\mathbf{h}; \mathbf{x}) + \beta(I(\mathbf{x}; y) - I(\mathbf{h}; y)) = \operatorname*{arg\,min}_{\mathbf{h}} I(\mathbf{h}; \mathbf{x}) - \beta\, I(\mathbf{h}; y), \tag{1.2}$$

where the Lagrangian multiplier β controls the trade-off between retaining information relevant to the labels (by minimising $-I(\mathbf{h}; y)$) and discarding irrelevant information (by minimising $I(\mathbf{h}; \mathbf{x})$).

Issues with the IB Objective

The story goes that since training a DNN corresponds to computing (1.2), the DNN learns to compress the input data into a minimal sufficient representation, naturally resulting in both good training performance and generalisation. To substantiate this claim, Shwartz-Ziv and Tishby (2017) ran empirical tests on a multilayer perceptron (MLP) trained on an artificial dataset and observed that a DNN learns a minimal sufficient representation in two distinct phases. In the first phase, both $I(\mathbf{h}; \mathbf{x})$ and $I(\mathbf{h}; y)$ increase, as the DNN learns to extract information from the input data. In the second phase, $I(\mathbf{h}; \mathbf{x})$ decreases while $I(\mathbf{h}; y)$ continues to increase (or slightly decreases, indicating overfitting), due to the DNN "forgetting" irrelevant information in the input data.

The problem with this story is that their results may not be inherent to all DNNs. Saxe et al. (2019) tested Shwartz-Ziv and Tishby's claims and found that their results are largely an artifact of the way MI was estimated and the nonlinearity function used in the MLP. Since $p_{\mathbf{x},y}$ is unknown and must thus be estimated from the training data, the distribution $p_{\mathbf{h},y}$ necessary (and sufficient) to compute $I(\mathbf{h}; y)$ must also be estimated. Shwartz-Ziv and Tishby did this by discretising $\mathbf{h}$—measured just after the tanh nonlinear activation functions of each layer. The exact binning used for each entry of $\mathbf{h}$ is visualised in figure 1.2. Since $\tanh(\phi)$ is approximately linear for ϕ with $|\phi| \approx 0$, in order for the DNN to learn nonlinear functions, ϕ must saturate the nonlinearity, i.e., $|\phi| \gg 0$.[1] This means most entries of $\mathbf{h}$ fall in either the top or bottom bin, effectively reducing the information content of $\mathbf{h}$ to a single bit per entry.

[1] This is part of why weight decay has a regularising effect, as it prevents the preactivation values from becoming too large in magnitude.

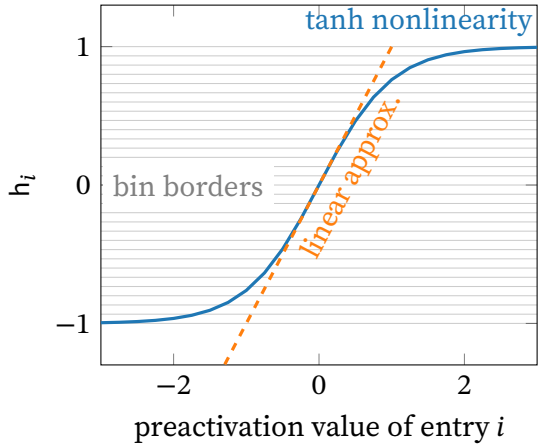


Figure 1.2.: The bin borders and possible values of the ith entry h_i of $\mathbf{h}$. The interval $[-1, 1]$ was divided into 30 equal-width bins. The dashed line shows how tanh is well approximated by a linear function for small values.

There is also a fundamental issue, formalised in the following theorem, concerning the computation of $\mathrm{I}(\mathbf{h}; \mathbf{x})$.

Theorem 1. *For nondegenerate continuous r.v.s* a *and* $\mathsf{b} = f(\mathsf{a})$ *where* f *is deterministic,* $\mathrm{I}(\mathsf{a}; \mathsf{b}) = \infty$.

Proof. Since f is deterministic, the conditional distribution $p_{\mathsf{b}|\mathsf{a}=a}$ is a Dirac delta distribution, i.e., $p_{\mathsf{b}|\mathsf{a}=a}(b) = \delta(b - f(a))$ can be thought of as the distribution that is zero everywhere except at $b - f(a)$. Formally, $\delta(u) = \lim_{w \to 0} \delta_w(u)$, where

$$\delta_w(u) = \begin{cases} \frac{1}{w}, & u \in \left[-\frac{w}{2}, \frac{w}{2}\right] \\ 0, & \text{otherwise} \end{cases}$$

is the uniform distribution over the interval $[-w/2, w/2]$. To ease notation, we drop the subscripts from (1.1) when clear from context and obtain

$$\begin{aligned} \mathrm{I}(\mathsf{a}; \mathsf{b}) &= \mathbb{E}_{\mathsf{a},\mathsf{b}}\left[\log\left(\frac{p(\mathsf{a}, \mathsf{b})}{p(\mathsf{a})p(\mathsf{b})}\right)\right] \\ &= \mathbb{E}_{\mathsf{a},\mathsf{b}}\left[\log\left(\frac{p(\mathsf{a}, \mathsf{b})}{p(\mathsf{a})}\right) - \log p(\mathsf{b})\right] \\ &= \mathbb{E}_{\mathsf{a},\mathsf{b}}[\log p(\mathsf{b}|\mathsf{a}) - \log p(\mathsf{b})] \\ &= \mathbb{E}_{\mathsf{a},\mathsf{b}}[\log p(\mathsf{b}|\mathsf{a})] + \mathbb{E}_{\mathsf{b}}[-\log p(\mathsf{b})]. \end{aligned}$$

The first term can be rewritten as

$$\begin{aligned} \mathbb{E}_{\mathsf{a},\mathsf{b}}[\log p(\mathsf{b}|\mathsf{a})] &= \mathbb{E}_{\mathsf{a}}[\mathbb{E}_{\mathsf{b}|\mathsf{a}=a}[\log p_{\mathsf{b}|\mathsf{a}=a}(\mathsf{b})]] \\ &= \int_{-\infty}^{\infty} p(a) \int_{-\infty}^{\infty} p_{\mathsf{b}|\mathsf{a}=a}(b) \log p_{\mathsf{b}|\mathsf{a}=a}(b)\, db\, da \\ &= \int_{-\infty}^{\infty} p(a) \int_{-\infty}^{\infty} \delta(b - f(a)) \log \delta(b - f(a))\, db\, da. \end{aligned} \tag{1.3}$$

The variable substitution $u = b - f(a)$ yields

$$
\begin{aligned}
&= \int_{-\infty}^{\infty} p(a) \lim_{w \to 0} \left(\int_{-w/2}^{w/2} \delta_w(u) \log \delta_w(u) \, du \right) da \\
&= \mathbb{E}_a \Big[\lim_{w \to 0} (-\log w) \Big],
\end{aligned}
$$

which is infinite. The theorem then follows from the finiteness of $\mathbb{E}_b[-\log p(b)]$ for nondegenerate b. □

Theorem 1 has the immediate consequence that, for any deterministic DNN with continuous inputs and representations, any meaningful observation of $I(\mathbf{h}; \mathbf{x})$ must have been a result of the estimator used. In fact, an even stronger result holds (see theorem 2) whose proof is omitted here for brevity. The motivated reader is referred to Geiger (2021) for an overview of the conflicting literature surrounding the IB principle.

Theorem 2 (Amjad and Geiger 2019). *For almost every deterministic DNN,* $I(\mathbf{h}; \mathbf{x})$ *is infinite if* $\mathbf{x}$ *is continuous and piecewise constant if* $\mathbf{x}$ *is discrete.* □

1.2. Neural Collapse

In the process of reviewing existing literature about the IB principle, Geiger (2021) noticed that in almost all cases, any information-theoretic compression, i.e., decrease in $I(\mathbf{h}; \mathbf{x})$—which as we now know, must have been a result of the estimator used—could have been explained by *geometric* compression instead. This geometric compression takes two forms: An *absolute* compression, where the diameter of $\mathcal{H}$ shrinks, and a *relative* compression in the form of clustering, where the distances between representations within a class decrease compared to their distances between different classes. Empirical studies (Goldfeld et al. 2019; Basirat, Geiger, and Roth 2021) qualitatively confirm this.

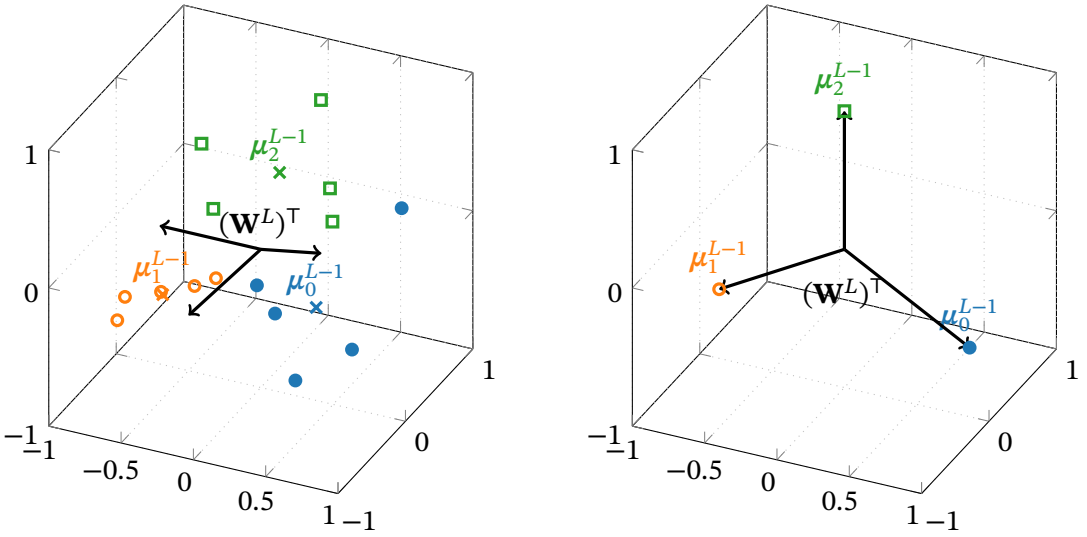
To quantify this geometric compression, we turn to a related phenomenon known as *neural collapse* (NC) (Papyan, Han, and Donoho 2020). To describe NC, we first formally define a DNN. A DNN with L layers is defined as the function composition

$$f^{L:1} = f^L \circ f^{L-1} \circ \cdots \circ f^1$$

where each layer $f^l = \sigma^l \circ T^l$ consists of a parametric transformation T^l with parameters $\mathbf{W}^l$ and a nonlinear activation function σ^l. We sometimes write $f_\theta^{L:1}$ to highlight the parameters $\theta = \{\mathbf{W}^l\}_{l=1}^L$ of the DNN. *Training* a DNN is then nothing more than finding the parameters $\hat{\theta}$ that minimise some objective function. Usually, $\sigma^1 = \cdots = \sigma^{L-1}$, $\sigma^L = \text{softmax}$ and T^L is a matrix multiplication.

For classification problems, each class $c = 1, \ldots, C$ of the training dataset has n_c samples $\{\mathbf{x}_{c,i}\}_{i=1}^{n_c}$, with $N = \sum_{c=1}^C n_c$ samples in total. Let $\mathbf{h}_{c,i}^l = f^{l:1}(\mathbf{x}_{c,i})$ be the (flattened) representation vector of the ith sample of class c after layer l. Their within-class mean is then $\boldsymbol{\mu}_c^l = {}^1/_{n_c} \sum_{i=1}^{n_c} \mathbf{h}_{c,i}^l$, and their (empirical) within-class covariance matrix is

$$\boldsymbol{\Sigma}_W^l = \frac{1}{N} \sum_{c=1}^{C} \sum_{i=1}^{n_c} (\mathbf{h}_{c,i}^l - \boldsymbol{\mu}_c^l)(\mathbf{h}_{c,i}^l - \boldsymbol{\mu}_c^l)^\top.$$



(a) No clear structure is visible apart from some clustering of the representations.

(b) The representations of each class have collapsed to their respective class means (NC1), which are themselves equidistant from each other (NC2) and aligned with the rows of the weight matrix (NC3).

Figure 1.3.: The (fictitious) configuration of the penultimate layer early in training (a) and after training beyond zero training error (b). For the sake of visualisation, $C = 3$ with centred three-dimensional representations, i.e., $\bar{\boldsymbol{\mu}}^{L-1} = \mathbf{0}$.

The global mean at layer l is $\bar{\boldsymbol{\mu}}^l = {}^1\!/_N \sum_{c=1}^{C} \sum_{i=1}^{n_c} \mathbf{h}_{c,i}^l$ and so the corresponding (empirical) between-class covariance matrix is

$$\boldsymbol{\Sigma}_B^l = \frac{1}{N} \sum_{c=1}^{C} n_c (\boldsymbol{\mu}_c^l - \bar{\boldsymbol{\mu}}^l)(\boldsymbol{\mu}_c^l - \bar{\boldsymbol{\mu}}^l)^\top.$$

Finally, the centred mean of class c at layer l is given by $\tilde{\boldsymbol{\mu}}_c^l = \boldsymbol{\mu}_c^l - \bar{\boldsymbol{\mu}}^l$. For more details on notation, see chapter A. Those familiar with linear discriminant analysis (Fisher 1936) may recognise $\boldsymbol{\Sigma}_W^l$ and $\boldsymbol{\Sigma}_B^l$ as the (normalised) within-class and between-class scatter matrices, respectively.

NC can now be defined as the phenomenon characterised by four properties (NC1–4), visualised in figure 1.3, of the penultimate layer activations $\mathbf{h}_{c,i}^{L-1}$ that occur when training beyond zero training error:

NC1: The (within-class) sample variance of $\{\mathbf{h}_{c,i}^{L-1}\}_{i=1}^{n_c}$ vanishes as all values collapse to $\boldsymbol{\mu}_c^{L-1}$.

NC2: The centred class means $\{\tilde{\boldsymbol{\mu}}_c^{L-1}\}_{c=1}^{C}$ converge to a regular $(C-1)$-simplex (up to isometry), i.e., $\|\tilde{\boldsymbol{\mu}}_c^{L-1}\| \to \rho$ for all c, and all distinct pairs $\tilde{\boldsymbol{\mu}}_c^{L-1}$, $\tilde{\boldsymbol{\mu}}_{c'}^{L-1}$ with $c \neq c'$ become equiangular with inner product $-\rho^2/(C-1)$.

NC3: The matrix $\mathbf{M}^{L-1} = \left[\tilde{\boldsymbol{\mu}}_1^{L-1}, \ldots, \tilde{\boldsymbol{\mu}}_C^{L-1}\right]$ converges to $(\mathbf{W}^L)^\top$ up to scaling.

NC4: As a result of NC1–3, f^L converges to simple nearest centre classification, i.e., $f^L(\mathbf{z}) \to \arg\min_c \|\mathbf{z} - \boldsymbol{\mu}_c^{L-1}\|$.

As the following discussion applies to any layer l, we drop the layer superscript to simplify notation. To measure geometric compression, one could naively use a generalised version of NC1, $\boldsymbol{\Sigma}_W \to \mathbf{0}$, as it directly measures the geometric compression of the representations. To avoid storing the entire matrix, $\operatorname{tr}(\boldsymbol{\Sigma}_W) = {}^1\!/_N \sum_{c=1}^C \sum_{i=1}^{n_c} \|\mathbf{h}_{c,i} - \boldsymbol{\mu}_c\|^2$ could be used as is done in (Masarczyk et al. 2023; Zangrando et al. 2024). One caveat of this approach is that a decrease in within-class variance could simply be a byproduct of a decrease in the total covariance

$$\boldsymbol{\Sigma}_T = \frac{1}{N} \sum_{c=1}^{C} \sum_{i=1}^{n_c} (\mathbf{h}_{c,i} - \bar{\boldsymbol{\mu}})(\mathbf{h}_{c,i} - \bar{\boldsymbol{\mu}})^\top = \boldsymbol{\Sigma}_W + \boldsymbol{\Sigma}_B.$$

To account for this, Papyan, Han, and Donoho (2020) and He and Su (2023) instead used $\operatorname{tr}(\boldsymbol{\Sigma}_W \boldsymbol{\Sigma}_B^+)$, a form of inverse signal ($\boldsymbol{\Sigma}_B$) to noise ($\boldsymbol{\Sigma}_W$) ratio, where $(\cdot)^+$ denotes the Moore-Penrose pseudoinverse. We shall do the same. To see why the pseudoinverse is necessary, recall that, by definition, $\sum_{c=1}^C \tilde{\boldsymbol{\mu}}_c = \sum_{c=1}^C (\boldsymbol{\mu}_c - \bar{\boldsymbol{\mu}}) = \mathbf{0}$ and so $\operatorname{rank} \mathbf{M} \leq C - 1$. Now let $\mathbf{N} = \operatorname{diag}(n_1, \ldots, n_C) \in \mathbb{N}^{C \times C}$ with $\operatorname{rank} \mathbf{N} = C$. Then,

$$\operatorname{rank} \boldsymbol{\Sigma}_B = \operatorname{rank}\left({}^1\!/_N \mathbf{M}\mathbf{N}\mathbf{M}^\top\right) \leq \min\{\operatorname{rank} \mathbf{M}, \operatorname{rank} \mathbf{N}\} \leq C - 1, \tag{1.4}$$

and so $\boldsymbol{\Sigma}_B$ is only invertible in the rare case that the representation is at most $(C-1)$-dimensional.

1.3. $\mathcal{V}$-Information and the Decodable Information Bottleneck

In a concurrent attempt to fix the issues with the IB principle, Dubois et al. (2020) introduced the *decodable information bottleneck* (DIB) based on the concept of *variational information* ($\mathcal{V}$-information) defined by Xu et al. (2019). The core motivation behind $\mathcal{V}$-information was to take into account computational limitations when computing MI.

A well-known result in information theory is that MI is invariant under invertible transformations (Kraskov, Stögbauer, and Grassberger 2004; Amjad and Geiger 2019). This has immediate undesirable side effects: Consider, e.g., $\mathrm{I}(\mathbf{h}^{L-1}, \mathrm{y})$ where f^L is a linear classifier and $\mathbf{h}^{L-1}$ is (easily) linearly separable. Now consider the transformation which flips the sign of a fixed set of entries of $\mathbf{h}^{L-1}$. Clearly, this transformation is invertible and therefore does not change the MI. However, applying such a transformation would likely result in the representation no longer being linearly separable, drastically reducing the DNN's performance. Another intuitive example is the case of cryptography, where an invertible transformation (encryption) is applied to some input text; practically speaking, no information on the original text can be extracted from its encrypted form.

To see where one might introduce computational limitations, recall the MI decomposition in the proof of theorem 1. Let $\mathcal{U} = \{\mathcal{H} \to \mathbb{P}(\mathcal{Y})\}$ be the family of functions that

map representations to distributions over the labels. Ergo, a function $f \in \mathcal{U}$ takes a representation as input and outputs a probability mass function over $\mathcal{Y}$. To emphasise the difference between the input and output of f, we write $f[\cdot](\cdot)$ instead of $f(\cdot)(\cdot)$. Then the negative of the term in (1.3) can alternatively be written in variational form as

$$\begin{aligned}
-\mathbb{E}_{\mathsf{a},\mathsf{b}}[\log p(\mathsf{b}|\mathsf{a})] &= \mathbb{E}_{\mathsf{a}}\Bigg[\mathbb{E}_{\mathsf{b}|\mathsf{a}=a}[-\log p_{\mathsf{b}|\mathsf{a}=a}(\mathsf{b})] + \overbrace{\inf_{f\in\mathcal{U}} \mathbb{E}_{\mathsf{b}|\mathsf{a}=a}\left[-\log \frac{f[a](\mathsf{b})}{p_{\mathsf{b}|\mathsf{a}=a}(\mathsf{b})}\right]}^{KL}\Bigg] \\
&= \inf_{f\in\mathcal{U}} \mathbb{E}_{\mathsf{a}}\left[\mathbb{E}_{\mathsf{b}|\mathsf{a}=a}\left[\log \frac{1}{p_{\mathsf{b}|\mathsf{a}=a}(\mathsf{b})} + \log \frac{p_{\mathsf{b}|\mathsf{a}=a}(\mathsf{b})}{f[a](\mathsf{b})}\right]\right] \\
&= \inf_{f\in\mathcal{U}} \mathbb{E}_{\mathsf{a}}[\mathbb{E}_{\mathsf{b}|\mathsf{a}=a}[-\log f[a](\mathsf{b})]] \\
&= \inf_{f\in\mathcal{U}} \mathbb{E}_{\mathsf{a},\mathsf{b}}[-\log f[\mathsf{a}](\mathsf{b})],
\end{aligned}$$

since $KL = 0$ for $f[a] = p_{\mathsf{b}|\mathsf{a}=a}$. Ergo, $\mathbb{E}_{\mathsf{a},\mathsf{b}}[-\log p(\mathsf{a}|\mathsf{b})]$ corresponds to the expected cross-entropy loss of predicting b with an infinitely powerful DNN f (with softmax output) given a as input, suggesting a natural way to introduce computational constraints: Restrict the function family to $\mathcal{V} \subset \mathcal{U}$.

The resulting $\mathcal{V}$-information, for the case $\mathsf{a} = \mathbf{h}$ and $\mathsf{b} = \mathsf{y}$, is called the *sufficiency* term and is defined as

$$\mathrm{I}_{\mathcal{V}}(\mathbf{h} \to \mathsf{y}) = \mathbb{E}_{\mathsf{y}}[-\log p(\mathsf{y})] - \inf_{f\in\mathcal{V}} \mathbb{E}_{\mathbf{h},\mathsf{y}}[-\log f[\mathbf{h}](\mathsf{y})].$$

In the case of an L-layered DNN, a natural choice for each layer l is to choose the family of functions $\mathcal{V}^l \triangleq \{f_\theta^{L:l+1}\}_\theta$ corresponding to all possible parametrisations of the decoder (architecture) $f^{L:l+1}$. The sufficiency term $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathsf{y})$ then captures the amount of relevant information in $\mathbf{h}^l$ that is extractable by the decoder $f_{\hat\theta}^{L:l+1}$ with optimal parameters $\hat\theta$.[2] In this case, minimising $-\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathsf{y})$ with respect to the encoder $f^{l:1}$ is equivalent to minimising the expected error of the *entire DNN* $f^{L:1}$, rejustifying the second term of (1.2).

To rejustify the first term, Dubois et al. (2020) argued that instead of considering the full MI $\mathrm{I}(\mathbf{h};\mathbf{x})$, one should instead only consider the information in $\mathbf{h}$ that is decodable by functions in $\mathcal{V}$. This has the issue that $\mathcal{X} \neq \mathcal{Y}$ in general, and so $\mathrm{I}_{\mathcal{V}}(\mathbf{h} \to \mathbf{x})$ is undefined. To circumvent this, they instead considered the set of r.v.s corresponding to all possible (re)labellings of $\mathbf{x}|\mathsf{y} = y$,

$$\mathrm{Dec}(\mathbf{x}, y) = \{\mathsf{n}_y \mid \exists t' : \mathcal{X} \to \mathcal{Y} \text{ s.t. } t'(\mathbf{x}|\mathsf{y} = y) = \mathsf{n}_y\},$$

dubbed the y decomposition of $\mathbf{x}$. Since n_y takes values in $\mathcal{Y}$, the $\mathcal{V}$-information from $\mathbf{h}|\mathsf{y} = y$ to it is well-defined, and the average over all labels $y \in \mathcal{Y}$ and their corresponding decompositions,

$$\mathrm{I}_{\mathcal{V}}(\mathbf{h} \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y})) = \frac{1}{|\mathcal{Y}|}\sum_{y\in\mathcal{Y}} \frac{1}{|\mathrm{Dec}(\mathbf{x}, y)|} \sum_{\mathsf{n}_y \in \mathrm{Dec}(\mathbf{x},y)} \mathrm{I}_{\mathcal{V}}(\mathbf{h}|\mathsf{y} = y \to \mathsf{n}_y),$$

[2] We assume $\mathcal{V}^l$ is finite and so $f_{\hat\theta}^{L:l+1}$ exists. This is (unsatisfyingly) justified by the finite numerical accuracy of real-world DNNs.

called the *minimality* term, replaces $\mathrm{I}(\mathbf{h}; \mathbf{x})$ in (1.2). Intuitively, for a given layer l, the term $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ measures the average capacity of a decoder $f^{L:l+1} \in \mathcal{V}^l$ to predict arbitrary relabellings of samples with true label y given the m.r.v. $\mathbf{h}^l|\mathsf{y} = y$ of their representations. Therefore, if $f^{l:1}$ minimises $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$, then *every* $f^{L:l+1}$ struggles to extract information from $\mathbf{h}^l$ pertaining to *any* label $y' \in \mathcal{Y}$ (on average). In other words, very little information remains in $\mathbf{h}^l$.

The DIB objective is then obtained by plugging the sufficiency and minimality terms into (1.2):

$$\arg\min_{\mathbf{h}^l} \mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y})) - \beta\, \mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathsf{y}) \tag{1.5}$$

Since $1/|\mathcal{Y}| \sum_{y \in \mathcal{Y}} \mathrm{I}_{\mathcal{V}^l}(\mathbf{h}|\mathsf{y} = y \to \mathsf{n}_y) \approx \mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathsf{y})$ for the case $\mathsf{n}_y = \mathsf{y}$,[3] an encoder $f^{l:1}$ which computes (1.5) forces *every* decoder $f^{L:l+1} \in \mathcal{V}^l$ to assign the same (correct) prediction to samples which have the same label; ergo, $f^{L:l+1}$ cannot distinguish between train and test samples and must generalise. This is formalised in the following theorem, whose proof is omitted here for brevity.

Theorem 3 (Dubois et al. 2020). *Assume a deterministic labelling function t of the input, i.e., $\mathsf{y} = t(\mathbf{x})$. Under some mild assumptions on $\mathcal{V}^l$, if $f^{l:1}$ maximises $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathsf{y})$ and minimises $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$, then any decoder $f^{L:l+1} \in \mathcal{V}^l$ that minimises the training error will also minimise the test error.* □

1.4. Our Contributions

Our main contributions are threefold:

- We describe a two-pass algorithm for computing $\mathrm{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$ which has time and space complexity linear in the dimensionality of the representations of each layer.
- We empirically analyse the evolution of $\mathrm{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$ for every layer l during four training scenarios (two models × two datasets) on both the training and test dataset. Correlation between compression and generalisation was only seen in one scenario.
- We also empirically analyse $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathsf{y})$ and $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ in the same way as $\mathrm{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$. In addition, we plot their information planes and find no compression phase.

[3] In fact, equality holds if we assume a uniform distribution over $\mathcal{Y}$, corresponding to the case of a balanced dataset. This assumption will be discussed further in section 2.2.

2. Measuring Compression Throughout a DNN

With the theory now established, we can proceed to the practical aspects. Extending the work done in (Shwartz-Ziv and Tishby 2017), the goal is to measure both geometric and infomation-theoretic compression for every layer of a DNN as it trains. In addition, to probe the effectiveness of NC and the DIB as generalisation metrics, their values should be computed on both the training and test data. This, however, raises several challenges.

2.1. Computing $\operatorname{tr}\left(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+\right)$

The main concern when computing the geometric compression metric $\operatorname{tr}\left(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+\right)$ is computational complexity; even storing $\{\{\{\mathbf{h}_{c,i}^l\}_{i=1}^{n_c}\}_{c=1}^{C}\}_{l=1}^{L}$, i.e., every representation of every sample for every layer, is undesirable if not infeasible. Consider, e.g., the DNN picured on the cover page, whose architecture is described in section 2.3. The representation obtained by feeding it a 32×32 RGB image is then $16 \times 16 \times 64 = D$ dimensional; hence, just storing the representations of, say, $N = 50\,000$ samples at 32-bit precision would require $4ND$ bytes, or around 3 GiB. On top of that, (naïvely) computing $\boldsymbol{\Sigma}_W^l$ would require adding N matrices of size $D \times D$ per layer l. While computing $\boldsymbol{\Sigma}_B^l$ only involves C additions, directly computing its pseudoinverse using singular value decomposition (SVD) is in $\mathcal{O}(D^3)$, just like the matrix multiplication $\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+$.

In Papyan, Han, and Donoho (2020), since the dimensionality of $\mathbf{h}^{L-1}$ is rather small, the authors can afford to simply direct compute $\operatorname{tr}\left(\boldsymbol{\Sigma}_W^{L-1}(\boldsymbol{\Sigma}_B^{L-1})^+\right)$. Other literature (see section 2.4) either compute an approximation of $\operatorname{tr}\left(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+\right)$ or only compute it after certain epochs. We instead opted to use a (to our knowledge) novel algorithm to compute $\operatorname{tr}\left(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+\right)$ in a distributed manner using *two* passes over the dataset. The algorithm for a given layer l is described in algorithm 1 where the layer superscript is omitted for clarity. The algorithm is described for a balanced dataset, a decision which will be justified in section 2.2. It can, however, be easily extended to imbalanced datasets. The following theorem states the correctness, runtime and space complexity of the algorithm.

Theorem 4. *Algorithm 1 correctly computes* $\operatorname{tr}\left(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+\right)$ *for a given layer* l *in* $\mathcal{O}(N(E + CD))$ *time and* $\mathcal{O}(CD)$ *space, where* N *is the number of samples,* E *is the cost of evaluating the encoder* $f^{l:1}$ *on a single sample* $\mathbf{x}$*,* D *is the dimensionality of* $f^{l:1}(\mathbf{x})$*, and* C *is the number of classes.*

Proof. See section B.1. □

Algorithm 1 thus trades off the time and space complexity of storing the representations and covariance matrices with the time complexity E of evaluating the encoder $f^{l:1}$ on every sample an extra time. However, this trade-off worsens as l increases; not only does E increase as the encoder becomes deeper, but—as mentioned earlier—D also tends to shrink in most modern DNN architectures. In this case, one can still avoid storing

Algorithm 1 An algorithm for computing $\operatorname{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$ for a given layer l. The algorithm assumes a balanced dataset and performs two passes over it. The layer superscript is suppressed for ease of reading.

Input: samples with corresponding labels $\{\mathbf{x}_n, y_n\}_{n=1}^N$, the number of classes C, the encoder $f^{l:1}$

for $c = 1, \ldots, C$ **do**
  $\mathbf{s}_c \leftarrow \mathbf{0}$
for $n = 1, \ldots, N$ **do** ▷ *first pass*
  if $y_n = c$ **then**
    $\mathbf{s}_c \leftarrow \mathbf{s}_c + f^{l:1}(\mathbf{x}_n)$ ▷ *class totals*
$\bar{\boldsymbol{\mu}} \leftarrow \frac{1}{N}\sum_{c=1}^{C} \mathbf{s}_c$ ▷ *global mean*
for $c = 1, \ldots, C$ **do**
  $\boldsymbol{\mu}_c \leftarrow \frac{C}{N}\mathbf{s}_c$ ▷ *class means*
$\mathbf{M} \leftarrow [\boldsymbol{\mu}_1 - \bar{\boldsymbol{\mu}}, \ldots, \boldsymbol{\mu}_C - \bar{\boldsymbol{\mu}}]$
$\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top \leftarrow \mathbf{M}^\top\mathbf{M}$ ▷ *eigendecomposition*
$\operatorname{diag}(\lambda_1, \ldots, \lambda_C) \leftarrow \boldsymbol{\Lambda}$
$\mathcal{I}_{\neq 0} \leftarrow \{i \in \{1, \ldots, C\} \mid \lambda_i \neq 0\}$ ▷ *indices of nonzero eigenvalues*
$(\lambda_1, \ldots, \lambda_R) \leftarrow (\lambda_j)_{j \in \mathcal{I}_{\neq 0}}$ ▷ $R = \operatorname{rank}\mathbf{M}$
$\mathbf{V} \leftarrow (Q_{ij})_{i \in \{1,\ldots,C\}, j \in \mathcal{I}_{\neq 0}}$
$s_{\mathrm{tr}} \leftarrow 0$
for $n = 1, \ldots, N$ **do** ▷ *second pass*
  if $y_n = c$ **then**
    $\boldsymbol{\kappa}_{c,i} \leftarrow \mathbf{V}^\top\mathbf{M}^\top(f^{l:1}(\mathbf{x}_n) - \boldsymbol{\mu}_c)$
    $s_{\mathrm{tr}} \leftarrow s_{\mathrm{tr}} + \|\boldsymbol{\kappa}_{c,i} \oslash (\lambda_1, \ldots, \lambda_R)^\top\|^2$
return $\frac{C}{N} s_{\mathrm{tr}}$

the representations by also computing $\mathbf{G}^l = \sum_{c=1}^{C} \sum_{i=1}^{n_c} \mathbf{h}_{c,i}^l (\mathbf{h}_{c,i}^l)^\top$ in the first pass and exploiting

$$\mathbf{\Sigma}_W^l = \frac{\mathbf{G}^l}{N} - \bar{\boldsymbol{\mu}}^l (\bar{\boldsymbol{\mu}}^l)^\top - \mathbf{\Sigma}_B^l \tag{2.1}$$

to avoid the second pass. For the sake of simplicity, we only use algorithm 1 in our experiments, and amortise E by integrating the first pass into the main training loop. The proof of (2.1) is delegated to section B.2.

2.2. Computing $\mathcal{V}^l$-Information

Much like $\operatorname{tr}(\mathbf{\Sigma}_W^l (\mathbf{\Sigma}_B^l)^+)$, naïvely computing the $\mathcal{V}^l$-information metrics $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$ and $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ would also incur a tremendous computational cost. In the case of the sufficiency term $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$, while the entropy term $\mathbb{E}_{\mathrm{y}}[-\log p(\mathrm{y})]$ is a constant, the cross-entropy term $\ell_{CE} = \inf_{f^{L:l+1} \in \mathcal{V}^l} \mathbb{E}_{\mathbf{h}^l, \mathrm{y}}\big[-\log f^{L:l+1}[\mathbf{h}^l](\mathrm{y})\big]$ is more problematic. In order to compute ℓ_{CE}, one has to essentially train the decoder $f^{L:l+1}$ from scratch to convergence. Since the goal is to track the evolution of $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$ throughout training, this has to be done periodically (e.g., at every epoch).

The real challenge, however, is computing the minimality term $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$. Recall that computing $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ involves computing a nested average: First, the average $\mathcal{V}^l$-information between the representation m.r.v. $\mathbf{h}^l|\mathrm{y} = y$ and all possible relabellings of the underlying sample m.r.v. $\mathbf{x}|\mathrm{y} = y$ is computed for a fixed label y. Then, the average of these values is taken over all possible $y \in \mathcal{Y}$. This is clearly infeasible even for small datasets, as the total number of possible relabellings $\sum_{c=1}^{C} C^{n_c}$ is exponential.

To resolve this, one could approximate the average using Monte Carlo (MC) sampling. This, however, introduces randomness into the computation, and raises the question of how many MC relabellings are needed to obtain a good approximation. Surprisingly, Dubois et al. (2020) noticed in their tests that even a small number of MC relabellings suffice.[1] This, they argue, is because many of the possible relabellings result in the same $\mathcal{V}$-information value. This makes sense, as $\mathcal{V}$ is, e.g., invariant to permutations of $\mathcal{Y}$.

It is thus reasonable to instead choose relabellings that adequately "cover" the space of all possible relabellings, i.e., relabellings that are as mutually independent as possible. (Dubois et al. 2020, Algorithm 1) describes an algorithm that generates such relabellings. This algorithm has been reformulated to use our notation as algorithm 2. Intuitively, the algorithm generates relabellings by flipping the relationship between class and index; each sample $\mathbf{x}_{c,i}$ is now grouped by its class index i instead of its class c. More concretely, if the new labels $r(\mathbf{x}_{c,i}) = y_0, \ldots, y_{N_D-1}$ of $\mathbf{x}_{c,i}$ are treated as a single vector label $\mathbf{y}_i = (y_0, \ldots, y_{N_D-1})$, then since i fully determines $\mathbf{y}_i$, the ith samples $\{\mathbf{x}_{c,i}\}_{c=1}^{C}$ of each class all get assigned the new (vector) label $\mathbf{y}_i$. Figure 2.1 provides a visualisation of r and figure 2.2 shows its associated DNN.

Recall that a DNN should generalise if it minimises $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ while maximising $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$, i.e., if it struggles to predict $r(\mathbf{x}_{c,i})$ given $\mathbf{x}_{c,i}$. The relabellings defined

[1] In their case, four.

Algorithm 2 (Dubois et al. 2020, Algorithm 1) reformulated using our notation.

Input: samples $\{\{\mathbf{x}_{c,i}\}_{i=1}^{n_c}\}_{c=1}^{C}$ grouped by class and ordered
Output: the new labels $r(\mathbf{x}_{c,i})$ for each sample $\mathbf{x}_{c,i}$

$N_D \leftarrow \lceil \log_C(\max_{c=1,\dots,C} n_c) \rceil$
for $c = 1, \dots, C$ **do**
  for $i = 1, \dots, n_c$ **do**
    $y_{N_D-1} \cdots y_0 \leftarrow i$ in base C with leading zeros ▷ $i = \sum_{n_d=0}^{N_D-1} y_{n_d} C^{n_d}$
    $r(\mathbf{x}_{c,i}) \leftarrow y_0, \dots, y_{N_D-1}$

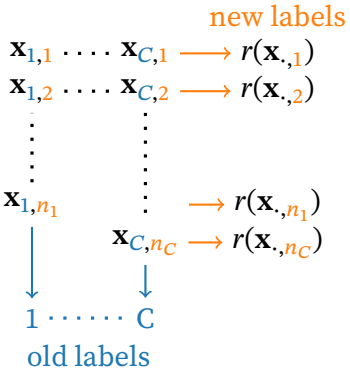


Figure 2.1.: The relabelling function r defined in algorithm 2. Each sample $\mathbf{x}_{c,i}$ is assigned the new labels $r(\mathbf{x}_{c,i})$ which are the digits of its class index i in base C.

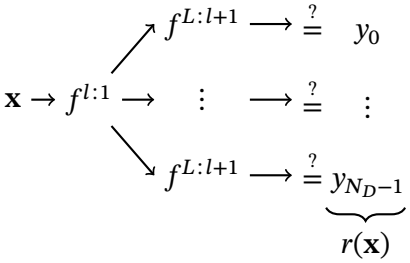


Figure 2.2.: The DNN that computes $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ for relabellings generated using algorithm 2. Since only the decoders should be trained, the encoder is frozen.

in algorithm 2 are especially difficult to predict under two conditions: If the dataset is balanced, i.e., $n_1 = \cdots = n_C$, then the set of samples which share new labels is as diverse as possible, containing exactly one sample from each class. If the dataset is large, then more samples from the same class get assigned different new (vector) labels, further increasing the difficulty. These conditions reflect the folklore that DNNs trained on larger balanced datasets generalise better.

In addition, a balanced dataset also results in each relabelling being balanced, maximising the entropy terms of $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathsf{y})$ and $\mathrm{I}_{\mathcal{V}}(\mathbf{h}|\mathsf{y} = y \to \mathsf{n}_y)$ to the constant value

$$\mathbb{E}_{\mathsf{y}}[-\log p(\mathsf{y})] = \mathbb{E}_{\mathsf{n}_y}[-\log p(\mathsf{n}_y)] = -\sum_{c=1}^{C} \frac{n_c}{N} \log \frac{n_c}{N} = -\sum_{c=1}^{C} \frac{1}{C} \log \frac{1}{C} = \log C.$$

2.3. Experiment Protocol

We considered four scenarios consisting of all possible combinations of two datasets and two DNN architectures, namely the more difficult CIFAR-10 (Krizhevsky 2009) dataset and the easier Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017) dataset, as well as one powerful and one weaker DNN. Both compression metrics were measured on both the training and test data, where care was taken to ensure that the DNNs behave the same during training and testing; no dropout or data augmentation was used, and any batch normalisation was computed *without* storing running statistics. Also recall from the last section that dataset size plays a role in the value of $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$, and so both training sets were downsampled to match the size of their respective test sets.

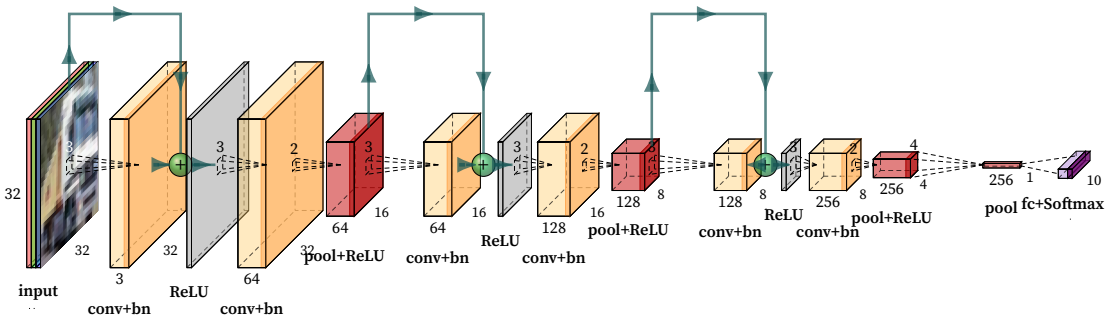
The more complex convolutional neural network (CNN) $f_+^{4:1}$ and its simpler derivative $f_-^{4:1}$ are pictured in figure 2.3. The CNN $f_+^{4:1}$ is based on the one described in (Jordan 2024), with the whitening layer removed, the layers within each block reordered and skip connections added. The CNN $f_-^{4:1}$ is $f_+^{4:1}$ with layers removed from each block, larger 5×5 convolutional kernels, larger 3×3 pooling kernels and shallower convolutional layers. The larger pooling kernels result in the final pooling layer being redundant, and so it was also removed.

Formally, for both CNNs, $\sigma^1(\cdot) = \cdots = \sigma^3(\cdot) = \max\{0, \cdot\}$ are rectified linear units (ReLU) and $\sigma^4 = \mathrm{softmax}$. We also have

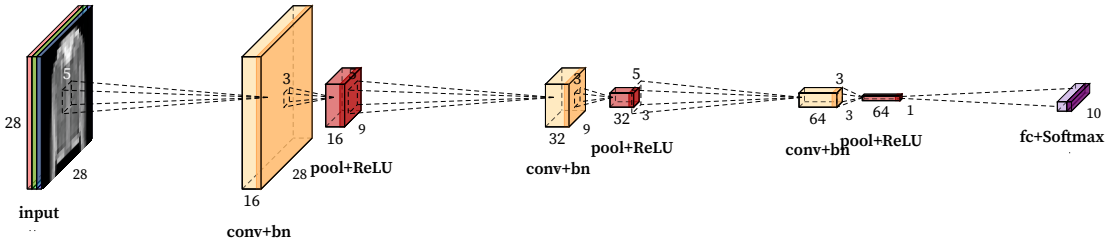
$$\begin{aligned} T_+^l(\cdot) &= (\mathrm{pool}(2 \times 2) \circ \mathrm{bn} \circ \mathrm{conv}(3 \times 3, d_l))(\max\{0, \cdot + (\mathrm{bn} \circ \mathrm{conv}(3 \times 3, d_{l-1}))(\cdot)\}) \\ T_-^l &= \mathrm{pool}(3 \times 3) \circ \mathrm{bn} \circ \mathrm{conv}(5 \times 5, {}^{d_l}/_4) \end{aligned}$$

for $l = 1, 2, 3$, where $(d_0, d_1, d_2, d_3) = (3, 64, 128, 256)$, $\mathrm{pool}(k \times k)$ is a max-pooling layer with a $k \times k$ kernel and stride k, $\mathrm{conv}(k \times k, d)$ is a convolutional layer with a $k \times k$ kernel of depth d padded such that the output has the same height and width as the input. Since each convolutional layer is followed by a batch normalisation (Ioffe and Szegedy 2015) layer

$$\mathrm{bn}(\cdot) = \frac{\cdot - \hat{\mu}}{\hat{\sigma}} \gamma + \beta$$



(a) The CNN $f_{+}^{4:1}$ based on the architecture described in (Jordan 2024). All convolutions use a 3×3 kernel and all pooling layers use a 2×2 kernel.



(b) The CNN $f_{-}^{4:1}$ which is a simplified version of $f_{+}^{4:1}$. All convolutions use a 5×5 kernel and all pooling layers use a 3×3 kernel.

Figure 2.3.: The architectures of the CNNs used. **conv+bn** denotes a convolutional layer followed by batch normalisation, **pool** denotes a max-pooling layer, and **fc** denotes a fully-connected layer. All convolutions are padded such that the output has the same height and width as the input. The tensor dimensions pictured here are correct but are in general dependent on the input size.

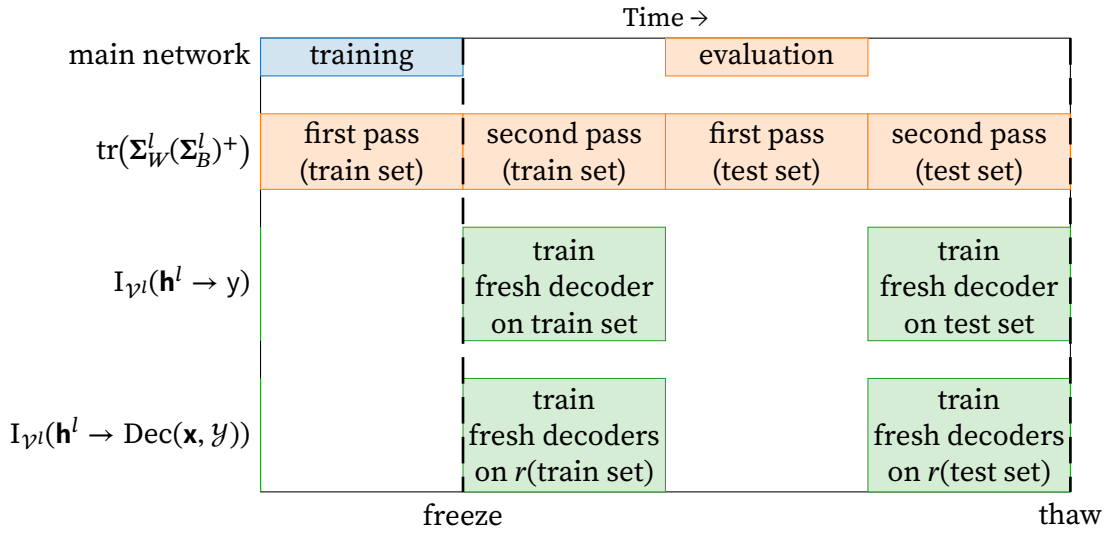


Figure 2.4.: Timeline of one epoch of the main network. Blue bars indicate that gradients were computed and backpropagated, orange bars indicate no values were used for backpropagation, and green bars indicate that a separate $\mathcal{V}$-information network was trained.

with batch mean $\hat{\mu}$, batch standard deviation $\hat{\sigma}$, and learnable parameters γ, β, the convolutional layers do *not* have a learnable bias. Finally,

$$T_+^4 = \mathrm{fc}(d_3, 10) \circ \mathrm{pool}(\sqrt{D_3/d_3} \times \sqrt{D_3/d_3})$$
$$T_-^4 = \mathrm{fc}(d_3/4, 10)$$

where $\mathrm{fc}(i, o)$ is a fully-connected layer with i inputs and o outputs, and D_3 is the dimensionality of $\mathbf{h}^3$.

Both models were trained the same way on cross-entropy loss using the AdamW optimiser (Loshchilov and Hutter 2018) and a constant learning rate of $2 \cdot 10^{-3}$. Any hyperparameters not mentioned here were set to the default values used by PyTorch 2.6 (Ansel et al. 2024). Each experiment was run until the training loss reached 0.02, with a hard limit of 50 epochs. At the end of each training epoch, $\mathrm{tr}(\Sigma_W^l(\Sigma_B^l)^+)$, $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$ and $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ were computed for the layers $l = 1, 2, 3$ on both the training and test data, with the sufficiency and minimality networks being trained the same way as the main network. In addition, $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$ and $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ were also computed for the case $l = 0$ corresponding to an empty encoder. The training procedure for one epoch is represented by the Gantt chart in figure 2.4 and the full source code is available at `https://github.com/ziyong-was-taken/Compression-In-DNNs`.

2.4. Related Work

Intermediate Neural Collapse While other literature also measures $\mathrm{tr}\big(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+\big)$ for every layer throughout training, they either only measure an approximation $\frac{\mathrm{tr}(\boldsymbol{\Sigma}_W^l)}{\mathrm{tr}(\boldsymbol{\Sigma}_B^l)}$, only measure $\mathrm{tr}\big(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+\big)$ at the end of training (or for a few set epoch milestones), or do not measure $\mathrm{tr}\big(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+\big)$ on the test data (Parker et al. 2023; Rangamani et al. 2023; Wang et al. 2024). One notable exception is (He and Su 2023, figure S8), who only considered an 8-layer fully-connected network on Fashion-MNIST.

Decodable Information Bottleneck In their original paper, Dubois et al. (2020) only measure $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$ and $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ as a result of minimising (1.5) directly, with a focus on the effect of β. Kleinman et al. (2021) only measure the test $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$ during training for synthetic training and test datasets. Lyu, Aminian, and Rodrigues (2023) also use DNNs to measure $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$, but they use a fixed $\mathcal{V}$ instead of the one corresponding to the decoder. In addition, instead of using $\mathrm{Dec}(\mathbf{x}, y)$ to fix the definitional problem with $\mathrm{I}_{\mathcal{V}}(\mathbf{h} \to \mathbf{x})$, they simply use a different functional family $\{\mathcal{H} \to \mathcal{X}\}$ and replace the cross-entropy term with a squared error term.

3. Experimental Results

To ease notation, when the analysis is for the entire CNN, the layer superscripts of each CNN are dropped.

3.1. Accuracy and Loss Curves

We begin by analysing the accuracy and loss curves of the four scenarios shown in figure 3.1. Both networks achieve 100% training accuracy for both datasets, with f_- on CIFAR-10 being the only scenario which did not achieve the 0.02 training loss threshold. The final test accuracies and losses, rounded to three significant figures, are as follows:

Scenario	Test Accuracy	Test Loss
f_+ on CIFAR-10	75.3%	0.757
f_- on CIFAR-10	62.6%	1.260
f_+ on Fashion-MNIST	91.0%	0.323
f_- on Fashion-MNIST	88.4%	0.387

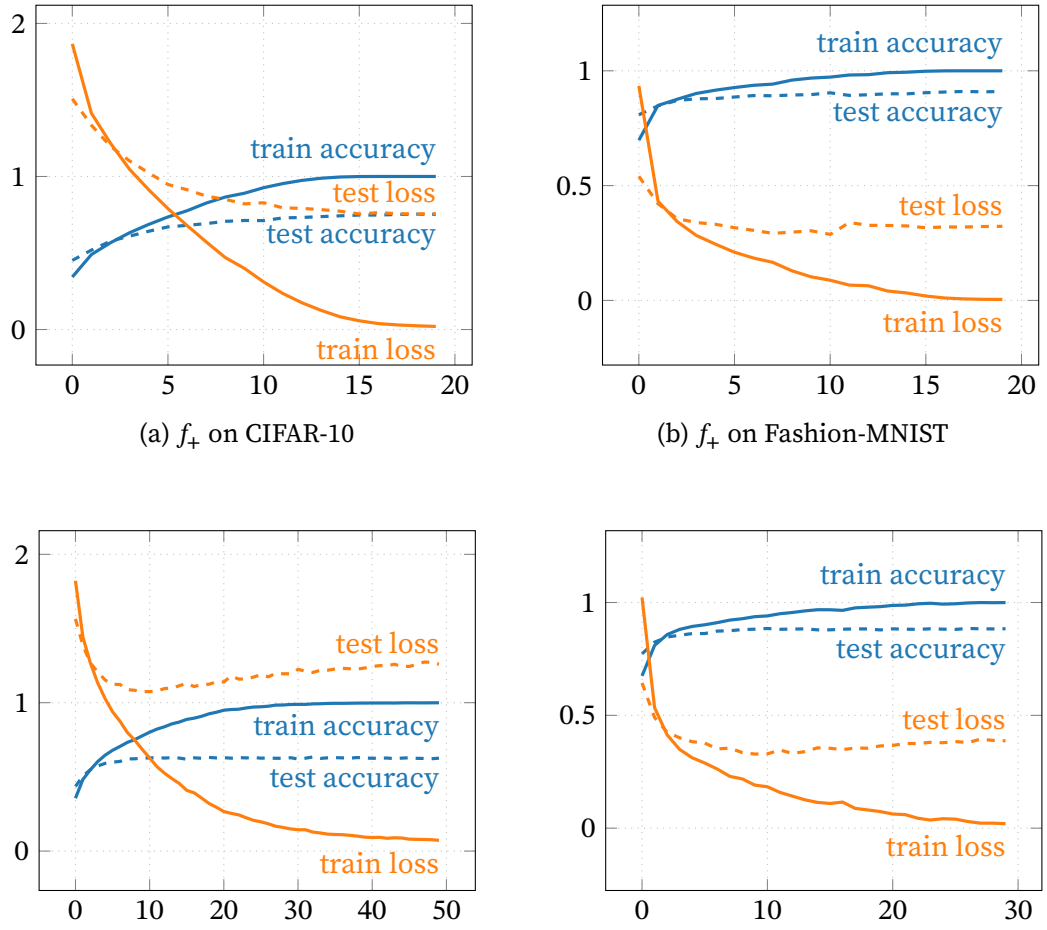
Unsurprisingly, f_- performs worse than f_+ on both datasets, with a smaller difference on the easier Fashion-MNIST dataset. On both datasets, and in particular on CIFAR-10, f_- shows signs of overfitting, as the test loss starts to increase after the 10th epoch.

3.2. Geometric Compression

The results are more interesting when we look at the NC metric $\operatorname{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$ of f_+ and f_- on both datasets, shown in figure 3.2. Here, two distinct phenomena can be seen.

First, the training $\operatorname{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$ is consistently lower than the test $\operatorname{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$, indicating that both CNNs compress the training data more than the test data. However, the difference in compression seems to only be mildly correlated with the generalisation gap, if at all, as evidenced by figure 3.3. One notable exception is the case of f_- on CIFAR-10, where the compression gap for layers $l = 2, 3$ positively correlates with the generalisation gap. This is perhaps due to the overfitting of f_- on CIFAR-10 being more pronounced than in the other scenarios.

Second, the values of $\operatorname{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$ are roughly the same for all layers during the beginning of training, with the layers quickly diverging such that deeper layers compress more. Interestingly, the NC values of each layer align well with the equi-separation law detailed in (He and Su 2023) which states that the ratio $r = \operatorname{tr}\big(\boldsymbol{\Sigma}_W^{l+1}(\boldsymbol{\Sigma}_B^{l+1})^+\big)/\operatorname{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$ is constant for all $l = 1, \ldots, L-1$, i.e., the compression is exponential in l. To visualise this, we instead plot the Pearson correlation coefficient, a measure of linear dependency, between $\log(\operatorname{tr}(\boldsymbol{\Sigma}_W^{\mathsf{l}}(\boldsymbol{\Sigma}_B^{\mathsf{l}})^+))$ and l, where l is the discrete random variable uniformly distributed over $\{1, \ldots, L\}$. This can be seen in figure 3.4, where it is clear that $\log(\operatorname{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+))$ is per-



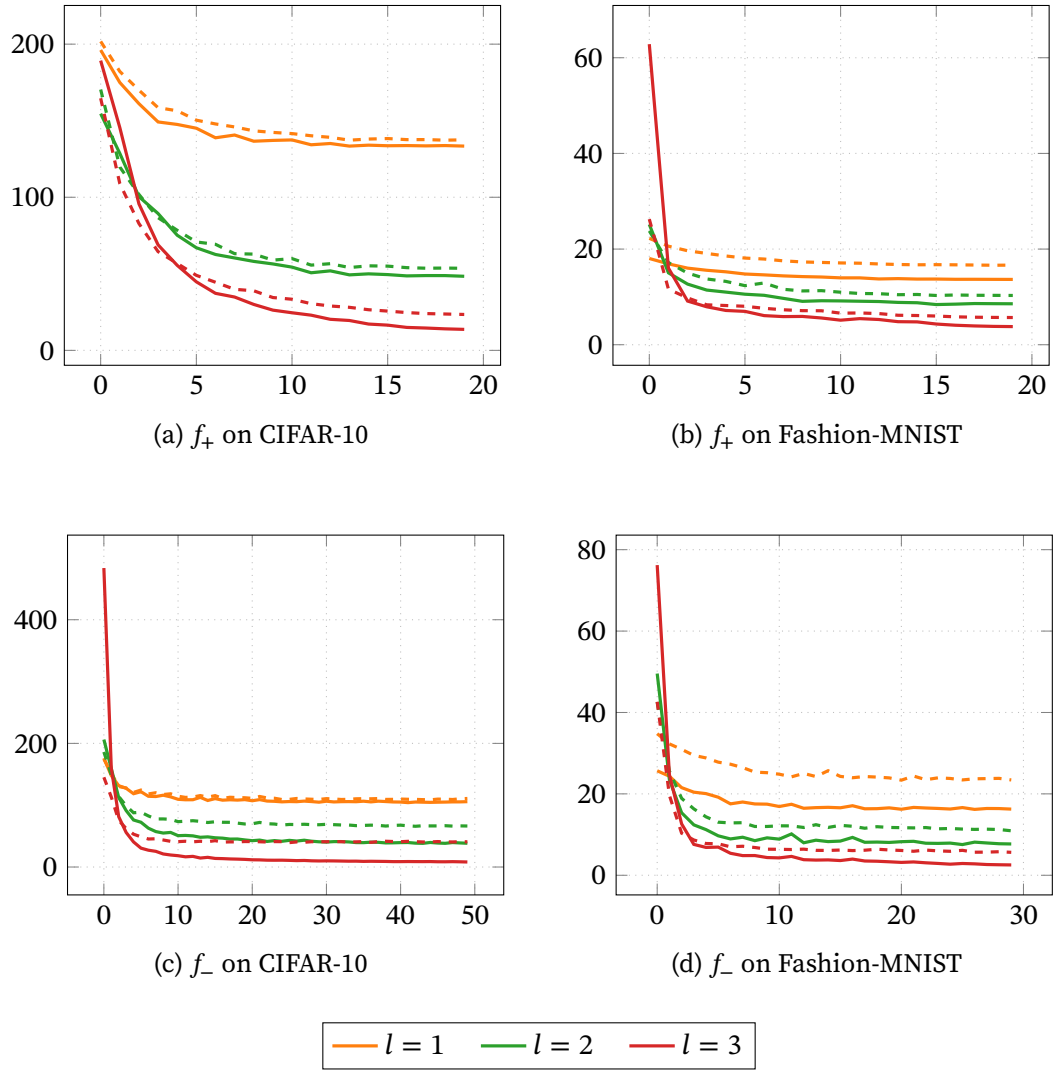
(a) f_+ on CIFAR-10

(b) f_+ on Fashion-MNIST

(c) f_- on CIFAR-10

(d) f_- on Fashion-MNIST

Figure 3.1.: The accuracy and loss curves for the four scenarios. The x-axis of each plot denotes the epoch. Solid lines denote training metrics, while dashed lines denote test metrics.



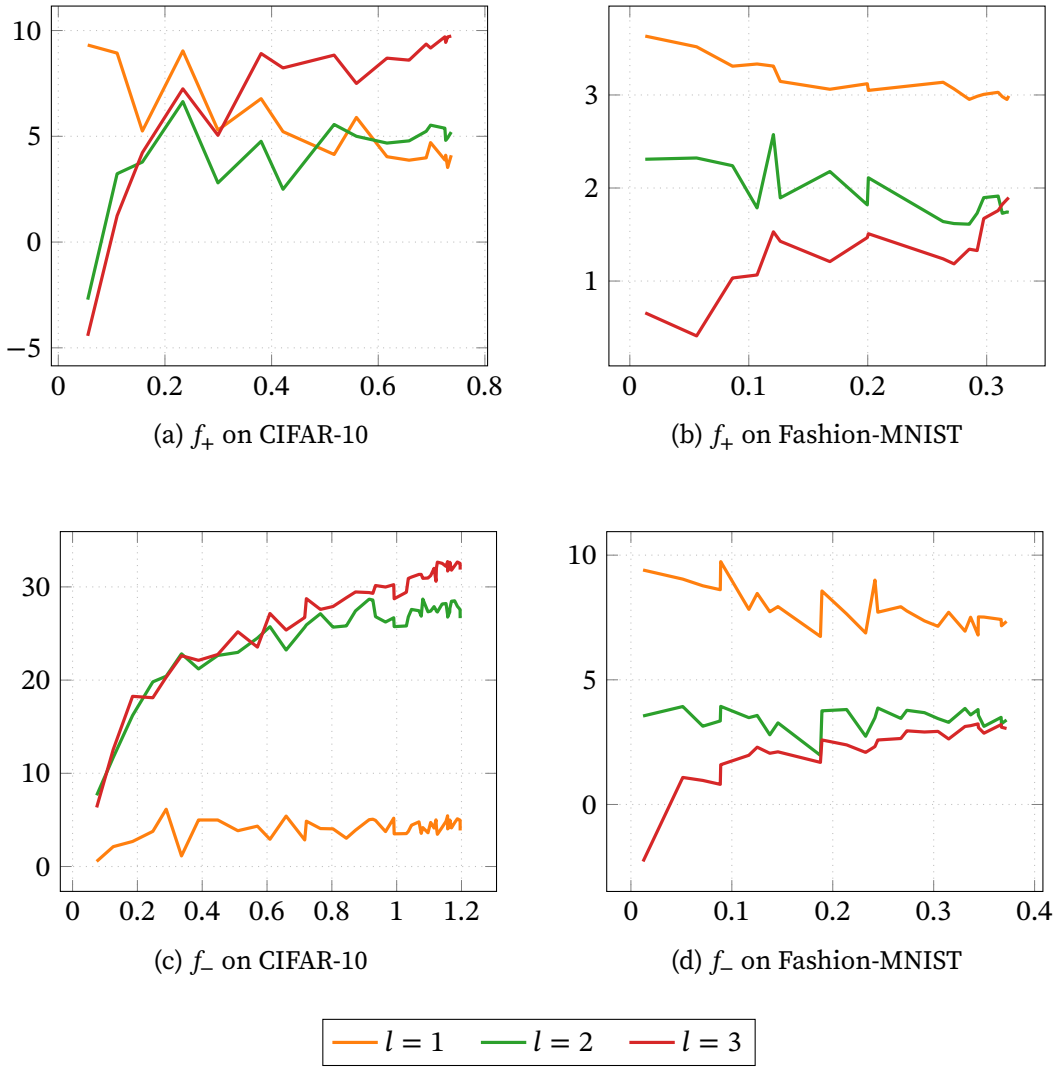
(a) f_+ on CIFAR-10

(b) f_+ on Fashion-MNIST

(c) f_- on CIFAR-10

(d) f_- on Fashion-MNIST

Figure 3.2.: The NC metric $\mathrm{tr}\left(\Sigma_W^l(\Sigma_B^l)^+\right)$ of f_+ and f_- computed on both the training and test datasets after each epoch. The x-axis of each plot denotes the epoch, while the y-axis denotes $\mathrm{tr}\left(\Sigma_W^l(\Sigma_B^l)^+\right)$. Solid lines denote training metrics, and dashed lines denote test metrics.



(a) f_+ on CIFAR-10

(b) f_+ on Fashion-MNIST

(c) f_- on CIFAR-10

(d) f_- on Fashion-MNIST

Figure 3.3.: Plots comparing the generalisation gap (the test loss minus the training loss) on the x-axis with the compression gap (the test value of $\mathrm{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$ minus its training value) on the y-axis. Only the points with positive generalisation gaps were considered.

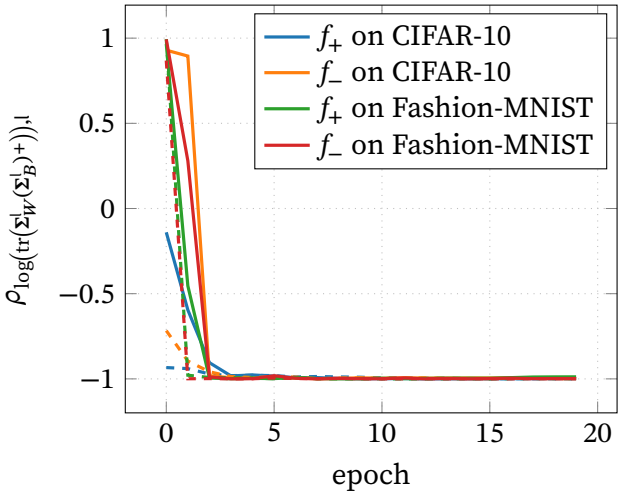


Figure 3.4.: Plots showing how "constant" r is for each scenario on both the training and test datasets. As usual, solid lines denote the training set, while dashed lines denote the test set.

fectly linear in l. This is perhaps unsurprising, given that a version of the equi-separation law for the simplified NC metric $\mathrm{tr}(\boldsymbol{\Sigma}_W^l)/\mathrm{tr}(\boldsymbol{\Sigma}_B^l)$ was proven in (Wang et al. 2024). It can also be seen that the equi-separation law comes into effect much sooner than NC itself.

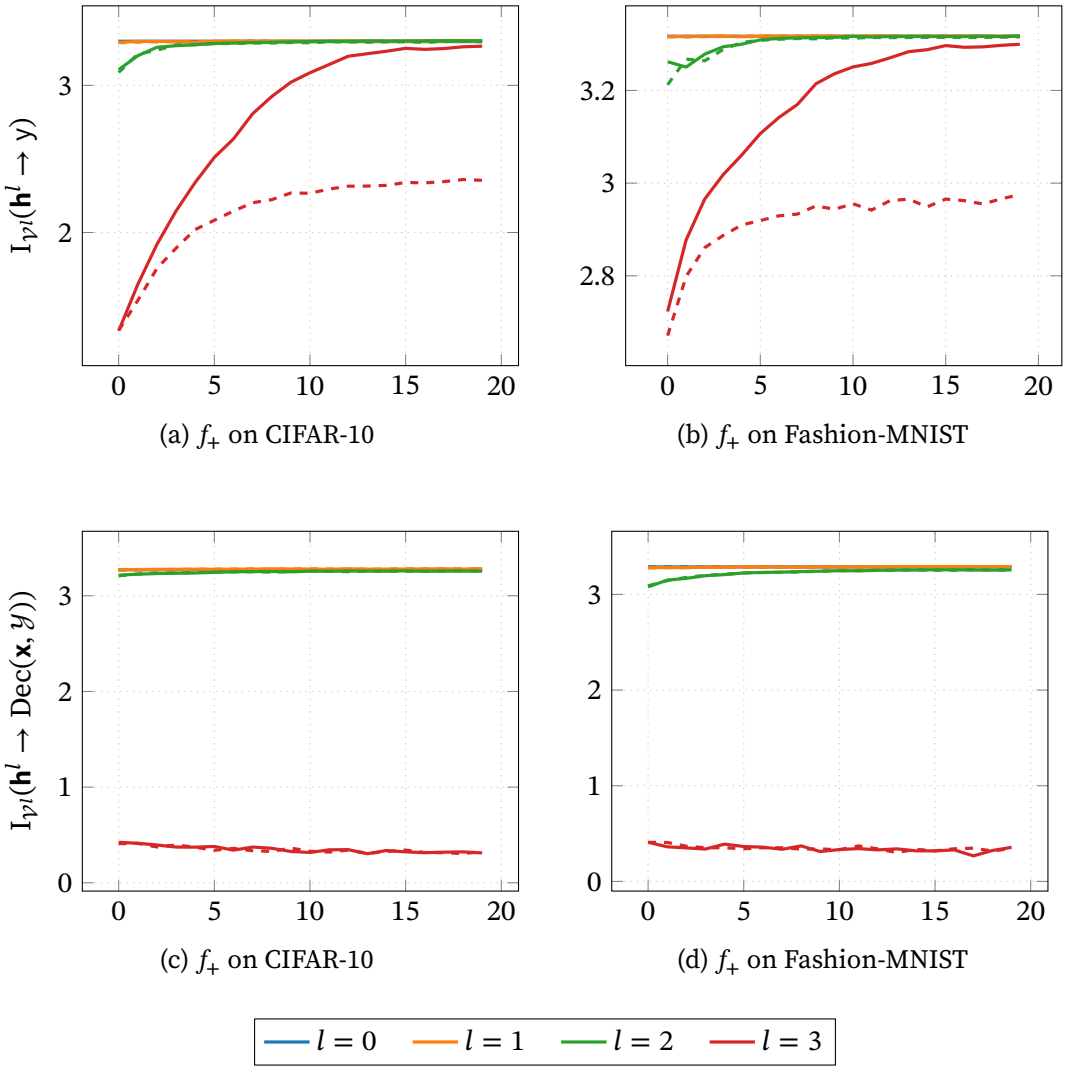
3.3. Information-Theoretic Compression

We now look at the information-theoretic side of things and begin with the results of the more powerful f_+ network.

The f_+ Network

The sufficiency term $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$ and the minimality term $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ of f_+ on both datasets are shown in figure 3.5. The general trajectory of $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$ and $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ is similar between the two datasets, although the sufficiency on Fashion-MNIST has a much higher magnitude than that on CIFAR-10. This tracks with f_+ achieving a higher train and test accuracy on Fashion-MNIST than on CIFAR-10. The clear divergence between the train and test $\mathrm{I}_{\mathcal{V}^3}(\mathbf{h}^3 \to \mathrm{y})$, with the test value plateauing at a much lower value than the train value, also seems to coincide with the generalisation gap seen in figures 3.1a and 3.1b, suggesting that f_+^4 is the only layer that is actually learning something meaningful about the data. In addition, the fact that the representations $\mathbf{h}^0$, $\mathbf{h}^1$, and to a lesser extent $\mathbf{h}^2$, very quickly saturate $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$ to its maximum value of $\log_2 10 \approx 3.32$ bits suggests that even $f_+^{4:3}$ suffices to decode the initially random representations $\mathbf{h}^2$.

It should then be no surprise that the minimality term for $l = 0, 1, 2$ also quickly saturates, as predicting the new labels $r(\mathbf{x})$ given an unaltered representation $\mathbf{h} = f^{l:1}(\mathbf{x})$ is easier than predicting the true label from a jumbled representation. Interestingly, the train and test $\mathrm{I}_{\mathcal{V}^3}(\mathbf{h}^3 \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ do *not* diverge from each other over the course of training, which suggests that f_+ equally compresses the train and test $\mathbf{h}^3$ despite the



(a) f_+ on CIFAR-10

(b) f_+ on Fashion-MNIST

(c) f_+ on CIFAR-10

(d) f_+ on Fashion-MNIST

Figure 3.5.: The first row is the sufficiency term $I_{\mathcal{V}^l}(\mathbf{h}^l \to y)$ and the second row is the minimality term $I_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ of f_+ for both datasets. The x-axis of each plot denotes the epoch of the main network's training. Solid lines denote training metrics, while dashed lines denote test metrics. While it may look like the blue $l = 0$ curves are missing, they are simply hidden behind the orange $l = 1$ curves.

large generalisation gap. Furthermore, the train and test $I_{\mathcal{V}^3}(\mathbf{h}^3 \to \text{Dec}(\mathbf{x}, \mathcal{Y}))$ remaining low throughout the training of the main network implies that f_+^4 is simply too weak to extract any extraneous information from $\mathbf{h}^3$. This is to be expected, given that f_+^4 is merely an affine linear transformation with $256 \times 10 + 10 = 2570$ learnable weights. The information planes pictured in figure 3.6 are thus rather uneventful.

The f_- Network

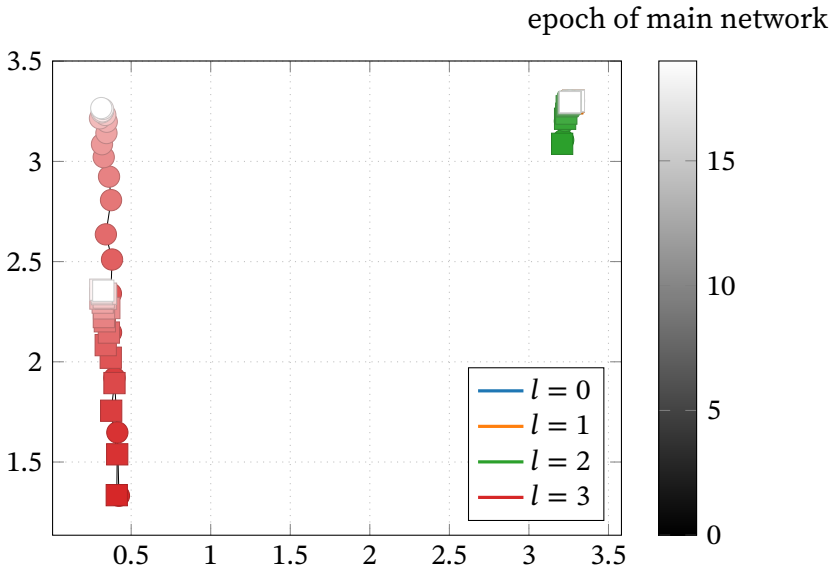
The results for f_- are more intriguing and are shown in figure 3.7. It can be seen that $l = 2$ plays a more active role in the sufficiency term for both datasets, with a small but noticeable train-test gap forming during training. Compared to f_+, the larger gap for $l = 3$ seems to reflect f_-'s larger generalisation gap. Unlike its stronger counterpart, the minimality term of f_- *does* change during training, with it increasing for $l = 2$ on CIFAR-10 and for $l = 1$ on Fashion-MNIST. Like in the case of f_+, we see that compression increases with depth, something we also saw when looking at the NC metric.

The cut-off layer for Fashion-MNIST, i.e., the layer l where $I_{\mathcal{V}^l}(\mathbf{h}^l \to \text{Dec}(\mathbf{x}, \mathcal{Y}))$ sharply drops, happens at $l = 1$ instead of at $l = 2$ as was the case for every other scenario. This implies that f_- is compressing the representations faster (in terms of depth) than f_+. In fact, $I_{\mathcal{V}^1}(\mathbf{h}^1 \to \text{Dec}(\mathbf{x}, \mathcal{Y}))$ and $I_{\mathcal{V}^2}(\mathbf{h}^2 \to \text{Dec}(\mathbf{x}, \mathcal{Y}))$ do not saturate, contradicting our reasoning for their saturation in f_+, and implying that f_- is compressing more than f_+ overall. One could argue that if we simply compare the dimensionality of the representations of each layer:

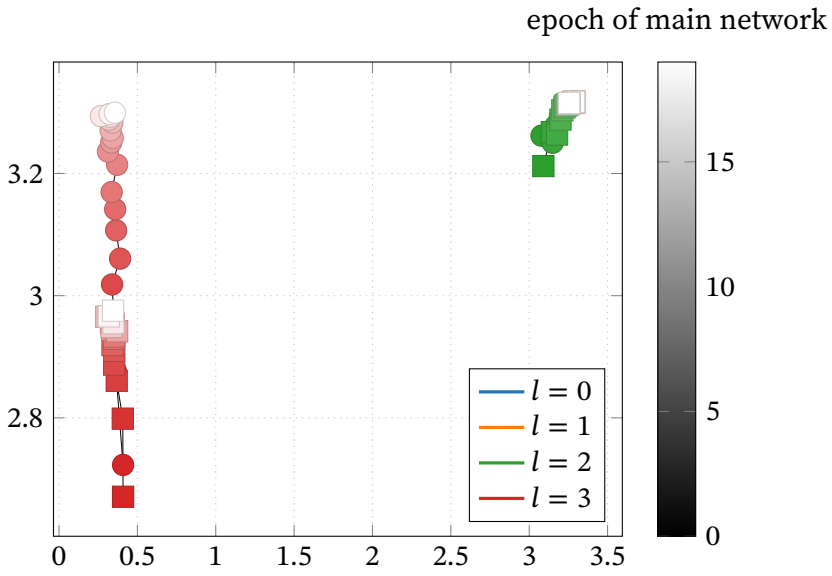
	Dimensionality of $\mathbf{h}^l$		
Scenario	$l = 1$	$l = 2$	$l = 3$
f_+ on CIFAR-10	$16 \times 16 \times 64$	$8 \times 8 \times 128$	$4 \times 4 \times 256$
f_- on CIFAR-10	$10 \times 10 \times 16$	$3 \times 3 \times 32$	$1 \times 1 \times 64$
f_+ on Fashion-MNIST	$14 \times 14 \times 64$	$7 \times 7 \times 128$	$3 \times 3 \times 256$
f_- on Fashion-MNIST	$9 \times 9 \times 16$	$3 \times 3 \times 32$	$1 \times 1 \times 64$

Then the representations of f_- are much smaller than those of f_+, necessarily resulting in compression. However, this does not explain the difference in compression for the case $l = 0$. Furthermore, although $\mathbf{h}^2$ is the same size for f_- on both datasets, $I_{\mathcal{V}^2}(\mathbf{h}^2 \to \text{Dec}(\mathbf{x}, \mathcal{Y}))$ behaves very differently.

More strangeness appears in the information planes of f_- shown in figure 3.8. Here, the curves of the fitting phases of $l = 2$ for CIFAR-10 and $l = 1$ for Fashion-MNIST are almost horizontal, meaning that the network seemingly sacrifices compression of $\mathbf{h}^l$ for no tangible gain in its predictive power. In addition, as in the case of f_+, we see no signs of a compression phase.

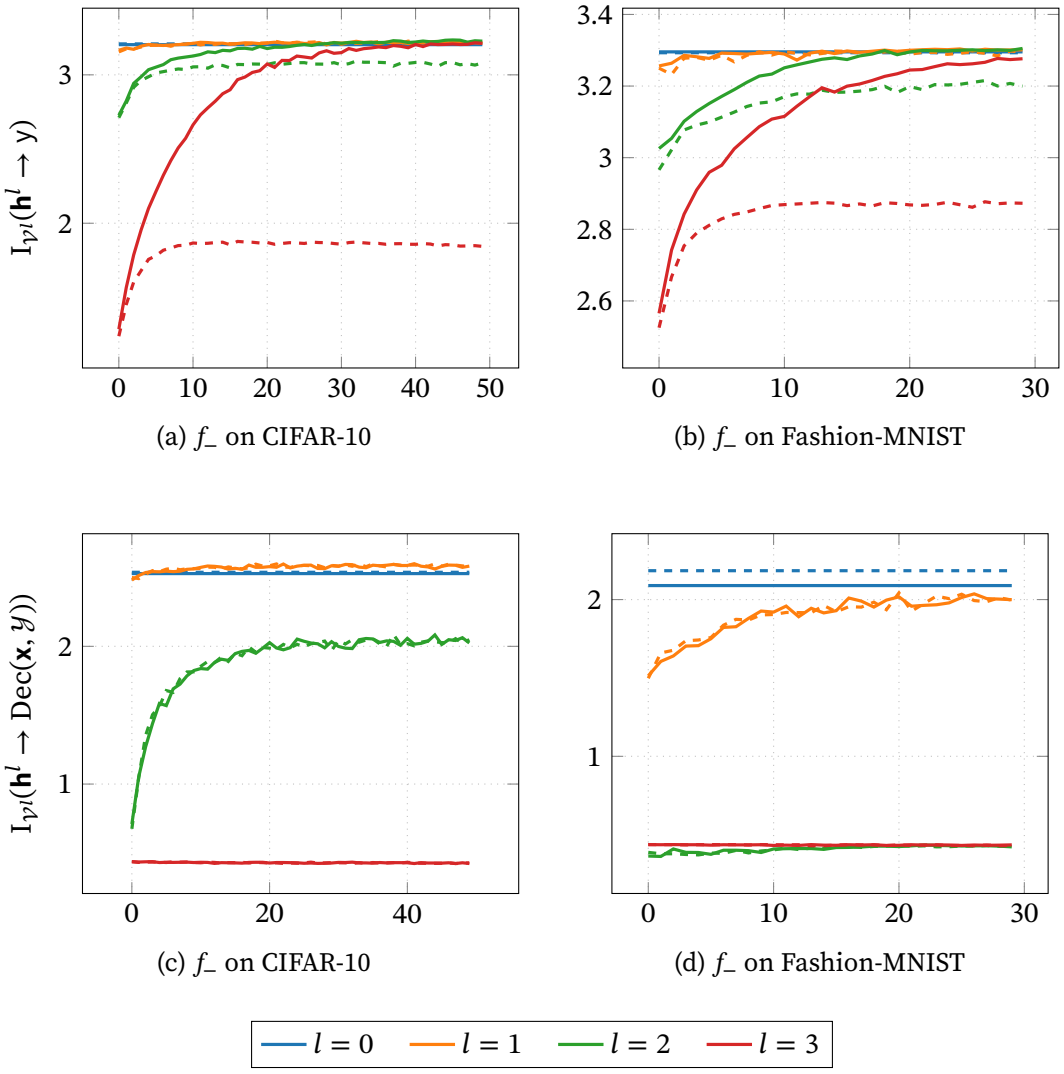


(a) f_+ on CIFAR-10



(b) f_+ on Fashion-MNIST

Figure 3.6.: The information planes of f_+ on CIFAR-10 and Fashion-MNIST. As is standard, the x-axis denotes the minimality term $I_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ and the y-axis denotes the sufficiency term $I_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$. Square markers denote the test metrics, while circular markers denote the training metrics. While it may look like the blue $l = 0$ and orange $l = 1$ curves are missing, they are simply at $(\log_2 10, \log_2 10)$ and thus hidden behind the green $l = 2$ curves.



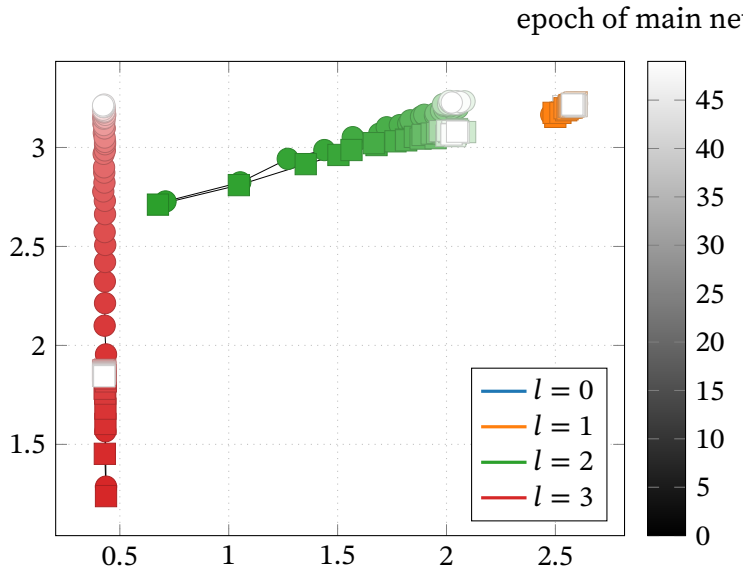
(a) f_- on CIFAR-10

(b) f_- on Fashion-MNIST

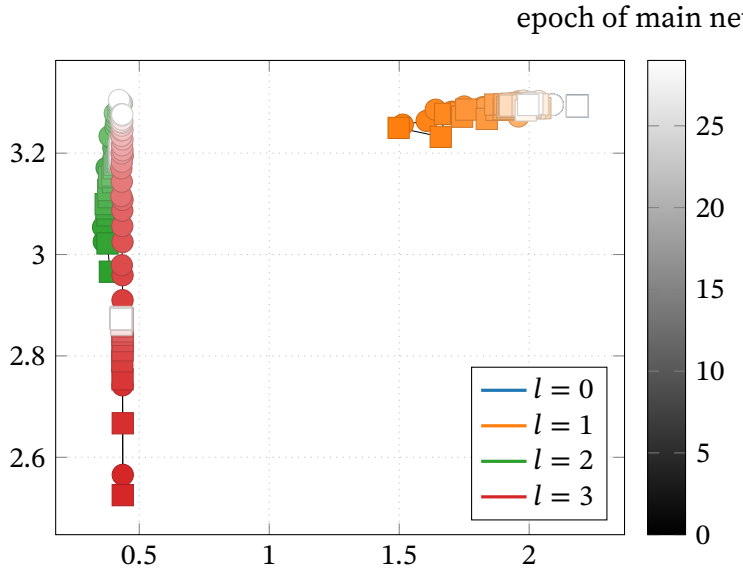
(c) f_- on CIFAR-10

(d) f_- on Fashion-MNIST

Figure 3.7.: The first row is the sufficiency term $I_{\mathcal{V}l}(\mathbf{h}^l \to y)$ and the second row is the minimality term $I_{\mathcal{V}l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, y))$ of f_- for both datasets. The x-axis of each plot denotes the epoch of the main network's training. Solid lines denote training metrics, while dashed lines denote test metrics.



(a) $f_{\_}$ on CIFAR-10



(b) $f_{\_}$ on Fashion-MNIST

Figure 3.8.: The information planes of $f_{\_}$ on CIFAR-10 and Fashion-MNIST. As is standard, the x-axis denotes the minimality term $I_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ and the y-axis denotes the sufficiency term $I_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$. Square markers denote the test metrics, while circular markers denote the training metrics. While it may look like the blue $l = 0$ curves are missing, they are simply hidden behind the orange $l = 1$ curves.

4. Discussion and Future Work

While the results may seem disheartening at first glance, they still raise many important questions about compression and generalisation in DNNs. Why does compression not correlate with generalisation despite intuition suggesting otherwise? Why does geometric compression increase exponentially with depth, while information-theoretic compression happens so much more suddenly? Perhaps the sudden drop in $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ is the result of a compounding effect: Increasing l not only results in a stronger encoder, but also a *weaker* decoder, while $\mathrm{tr}(\boldsymbol{\Sigma}_W^l(\boldsymbol{\Sigma}_B^l)^+)$ only depends on the encoder. What could have been the reason for the strange "reverse compression phase" seen in figure 3.8?

One possible explanation is that our experimental setup is flawed. Indeed, if one defines $p_{\mathbf{x},\mathrm{y}}$ *not* as a discrete uniform distribution over the training and test set, as we have implicitly done in our experiments, but rather as a more general continuous distribution over the space of *all* possible inputs and labels, then the nonentropy expectations in $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$ and $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ correspond *not* to the average cross-entropy train (or test) loss, but rather to the *expected* cross-entropy loss over said distribution.

Such a loss function is then nontrivial to optimise, as one has to avoid overfitting by either utilising some held-out validation set which, as discussed in section 2.2, could negatively affect $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{Dec}(\mathbf{x}, \mathcal{Y}))$ as a side effect of reducing the dataset size; or by using some form of regularisation which would by definition affect $\mathrm{I}_{\mathcal{V}^l}(\mathbf{h}^l \to \mathrm{y})$.

There is also the question of what effect our specific choice of DNN architecture had on the results. Was the compression observed simply an artifact of the natural funnel shape of f_+ and f_-? As we argued in section 3.3, only considering the size of the representations cannot suffice, yet the fact that both monotonically decrease with depth may suggest the existence of some form of scaling law, akin to the equi-separation law of NC. What role did the translational invariance of CNNs play in the results concerning generalisation? What if we soften our assumption of a balanced dataset and introduce some form of class imbalance? Clearly, more research is needed.

Bibliography

Amjad, Rana Ali and Bernhard C. Geiger (2019-04-02). "Learning Representations for Neural Network-Based Classification Using the Information Bottleneck Principle". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.9, pp. 2225–2239. DOI: 10.1109/TPAMI.2019.2909031.

Ansel, Jason et al. (2024-04-27). "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation". In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. Vol. 2. ASPLOS '24. La Jolla, CA, USA: Association for Computing Machinery, pp. 929–947. ISBN: 9798400703850. DOI: 10.1145/3620665.3640366.

Basirat, Mina, Bernhard C. Geiger, and Peter M. Roth (2021-06-03). "A Geometric Perspective on Information Plane Analysis". In: *Entropy* 23.6, pp. 1–16. ISSN: 1099-4300. DOI: 10.3390/e23060711. URL: https://www.mdpi.com/1099-4300/23/6/711.

Cover, Thomas M. and Joy A. Thomas (2005-04-07). *Elements of Information Theory*. Wiley. ISBN: 9780471748823. DOI: 10.1002/047174882x.

Dubois, Yann, Douwe Kiela, David J. Schwab, and Ramakrishna Vedantam (2020-12-06). "Learning Optimal Representations with the Decodable Information Bottleneck". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 18674–18690. URL: https://proceedings.neurips.cc/paper_files/paper/2020/hash/d8ea5f53c1b1eb087ac2e356253395d8-Abstract.html.

Fisher, R. A. (1936-09). "The Use of Multiple Measurements in Taxonomic Problems". In: *Annals of Eugenics* 7.2, pp. 179–188. ISSN: 2050-1439. DOI: 10.1111/j.1469-1809.1936.tb02137.x.

Geiger, Bernhard C. (2021-06-30). "On Information Plane Analyses of Neural Network Classifiers—A Review". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.12, pp. 7039–7051. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2021.3089037.

Goldfeld, Ziv, Ewout van den Berg, Kristjan Greenewald, Igor Melnyk, Nam Nguyen, Brian Kingsbury, and Yury Polyanskiy (2019-06-09). "Estimating Information Flow in Deep Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 2299–2308. URL: https://proceedings.mlr.press/v97/goldfeld19a.html.

He, Hangfeng and Weijie J. Su (2023-08-28). "A law of data separation in deep learning". In: *Proceedings of the National Academy of Sciences* 120.36, pp. 1–8. ISSN: 1091-6490. DOI: 10.1073/pnas.2221704120.

Ioffe, Sergey and Christian Szegedy (2015-06-07). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". en. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 448–

456. URL: https://proceedings.mlr.press/v37/ioffe15.html (visited on 06/10/2025).

Jordan, Keller (2024). *94% on CIFAR-10 in 3.29 Seconds on a Single GPU*. arXiv: 2404.00498 [cs.LG].

Kleinman, Michael, Alessandro Achille, Daksh Idnani, and Jonathan Kao (2021-01-12). "Usable Information and Evolution of Optimal Representations During Training". en. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=p8agn6bmTbr (visited on 02/05/2025).

Kraskov, Alexander, Harald Stögbauer, and Peter Grassberger (2004-06-23). "Estimating mutual information". In: *Physical Review E* 69.6, p. 066138. DOI: 10.1103/PhysRevE.69.066138. (Visited on 06/03/2025).

Krizhevsky, Alex (2009-04-08). *Learning multiple layers of features from tiny images*. Tech. rep. Department of Computer Science, University of Toronto. URL: https://www.cs.toronto.edu/~kriz/cifar.html.

Loshchilov, Ilya and Frank Hutter (2018-12-21). "Decoupled Weight Decay Regularization". en. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

Lyu, Zhaoyan, Gholamali Aminian, and Miguel R. D. Rodrigues (2023-07-14). "On Neural Networks Fitting, Compression, and Generalization Behavior via Information-Bottleneck-like Approaches". In: *Entropy* 25.7, pp. 1–28. ISSN: 1099-4300. DOI: 10.3390/e25071063. URL: https://www.mdpi.com/1099-4300/25/7/1063.

Masarczyk, Wojciech, Mateusz Ostaszewski, Ehsan Imani, Razvan Pascanu, Piotr Mił oś, and Tomasz Trzcinski (2023-12-10). "The Tunnel Effect: Building Data Representations in Deep Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., pp. 76772–76805. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/f249db9ab5975586f36df46f8958c008-Abstract-Conference.html.

Papyan, Vardan, X. Y. Han, and David L. Donoho (2020-09-21). "Prevalence of neural collapse during the terminal phase of deep learning training". In: *Proceedings of the National Academy of Sciences* 117.40, pp. 24652–24663. ISSN: 1091-6490. DOI: 10.1073/pnas.2015509117.

Parker, Liam, Emre Onal, Anton Stengel, and Jake Intrater (2023-07-28). "Neural Collapse in the Intermediate Hidden Layers of Classification Neural Networks". In: *Workshop on High-dimensional Learning Dynamics, ICML 2023*. URL: https://drive.google.com/file/d/1Ql51dYcYjZdTipep1XYKe3iTGTZzMBku/view.

Rangamani, Akshay, Marius Lindegaard, Tomer Galanti, and Tomaso A. Poggio (2023-07-23). "Feature learning in deep classifiers through Intermediate Neural Collapse". en. In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 28729–28745. URL: https://proceedings.mlr.press/v202/rangamani23a.html.

Saxe, Andrew M., Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D. Tracey, and David D. Cox (2019-12-20). "On the information bottleneck theory of deep learning*". en. In: *Journal of Statistical Mechanics: Theory and Experiment* 2019.12, pp. 124020–124054. ISSN: 1742-5468. DOI: 10.1088/1742-5468/ab3985. (Visited on 02/06/2025).

Shwartz-Ziv, Ravid and Naftali Tishby (2017). "Opening the Black Box of Deep Neural Networks via Information". arXiv: 1703.00810 [cs.LG].

Tishby, Naftali and Noga Zaslavsky (2015-06-25). "Deep learning and the information bottleneck principle". In: *2015 IEEE Information Theory Workshop (ITW)*. IEEE, pp. 1–5. DOI: 10.1109/ITW.2015.7133169.

Vapnik, Vladimir N. (1999-11-19). *The Nature of Statistical Learning Theory*. Springer New York. ISBN: 9781475732641. DOI: 10.1007/978-1-4757-3264-1.

Wang, Peng, Xiao Li, Can Yaras, Zhihui Zhu, Laura Balzano, Wei Hu, and Qing Qu (2024). "Understanding Deep Representation Learning via Layerwise Feature Compression and Discrimination". arXiv: 2311.02960 [cs.LG].

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017-08-28). *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. DOI: 10.48550/arXiv.1708.07747. arXiv: 1708.07747 [cs.LG].

Xu, Yilun, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon (2019-12-20). "A Theory of Usable Information under Computational Constraints". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=r1eBeyHFDH.

Zangrando, Emanuele, Piero Deidda, Simone Brugiapaglia, Nicola Guglielmi, and Francesco Tudisco (2024). "Neural Rank Collapse: Weight Decay and Small Within-Class Variability Yield Low-Rank Bias". arXiv: 2402.03991 [cs.LG].

Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals (2021-02-22). "Understanding deep learning (still) requires rethinking generalization". In: *Commun. ACM* 64.3, pp. 107–115. ISSN: 0001-0782. DOI: 10.1145/3446776.

A. Notation

Letters written in sans-serif font denote random variables of their serif counterparts, e.g., y and y. Tangential to this, **bold** letters denote vectors—or matrices in the case of **BOLD UPPERCASE** letters—e.g., $\mathbf{x}$ and $\mathbf{M}$. Logically, **bold sans-serif** letters denote random vectors, e.g., $\mathbf{h}$. Sets are denoted by calligraphic uppercase letters, e.g., $\mathcal{X}$. A set consisting of enumerated elements is denoted $\{x_n\}_{n=1}^N$, where we try to use a lowercase letter n for the arbitrary index, and the corresponding uppercase letter N for the total number of elements. When unimportant, the total number of elements is omitted, which gives $\{x_n\}_n$. A *sequence* of enumerated elements is treated as a vector and denoted by replacing the braces with parentheses, e.g., $(x_n)_{n=1}^N$; a sequence of vectors is treated as a matrix, e.g., $((M_{ij})_{i=1}^I)_{j=1}^J$.

The expectation $\mathbb{E}_{\mathsf{a}}[f(\mathsf{a})] = \mathbb{E}_{\mathsf{a}\sim p_{\mathsf{a}}}[f(\mathsf{a})]$ of $f(\mathsf{a})$ is taken over the distribution p_{a} with respect to the random variable a. Unless otherwise specified, $p(a) = p_{\mathsf{a}}(a)$ always means the probability density (or mass) function of the random variable a evaluated at a. The conditional random variable $\mathsf{b}|\mathsf{a} = a$ is, as the notation suggests, the random variable b conditioned on the event $\mathsf{a} = a$. The Euclidean norm of a vector $\mathbf{x}$ is denoted by $\|\mathbf{x}\| = \sqrt{\sum_i x_i^2}$. The elementwise vector division is denoted $\mathbf{x} \oslash \mathbf{y} = (x_i/y_i)_i$. We adopt the dot notation for partial application so $f(x, \cdot)$ is a function that takes an input y and returns $f(x, y)$. The function f can also be curried as $f[x](y) = f(x, y)$ where $f[x] = f(x, \cdot)$.

B. Proofs

B.1. Proof of Theorem 4

As in algorithm 1, the layer superscript is omitted for simplicity. We begin with the correctness proof. It is clear that algorithm 1 correctly computes the global mean $\bar{\boldsymbol{\mu}}$. In addition, recall that for a balanced dataset, $n_1 = \cdots = n_C = N/C$, and so algorithm 1 correctly computes the class means $\{\boldsymbol{\mu}_c\}_{c=1}^C$. We also have that the between-class covariance matrix simplifies to

$$\begin{aligned}\boldsymbol{\Sigma}_B &= \frac{1}{N}\sum_{c=1}^{C} n_c(\boldsymbol{\mu}_c - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}_c - \bar{\boldsymbol{\mu}})^\top \\ &= \frac{1}{C}\sum_{c=1}^{C}(\boldsymbol{\mu}_c - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}_c - \bar{\boldsymbol{\mu}})^\top \\ &= \frac{1}{C}\mathbf{M}\mathbf{M}^\top,\end{aligned} \tag{B.1}$$

where $\mathbf{M} = [\boldsymbol{\mu}_1 - \bar{\boldsymbol{\mu}}, \ldots, \boldsymbol{\mu}_C - \bar{\boldsymbol{\mu}}] \in \mathbb{R}^{D\times C}$ is the matrix of centred means.

Now consider the compact SVD $\mathbf{U}\mathbf{S}\mathbf{V}^\top$ of $\mathbf{M}$ with semi-orthogonal matrices $\mathbf{U} \in \mathbb{R}^{D\times R}$ and $\mathbf{V} \in \mathbb{R}^{C\times R}$—i.e., $\mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}_R$—and diagonal matrix $\mathbf{S} = \operatorname{diag}(\sigma_1, \ldots, \sigma_R) \in \mathbb{R}^{R\times R}$ with $\operatorname{rank}\mathbf{S} = R = \operatorname{rank}\mathbf{M} = \operatorname{rank}\boldsymbol{\Sigma}_B \overset{(1.4)}{\leq} C - 1$. We then have

$$\mathbf{U} = \mathbf{M}\mathbf{V}\mathbf{S}^{-1}, \tag{B.2}$$

$$\mathbf{M}\mathbf{M}^\top = \mathbf{U}\mathbf{S}\mathbf{V}^\top\mathbf{V}\mathbf{S}\mathbf{U}^\top = \mathbf{U}\mathbf{S}^2\mathbf{U}^\top, \tag{B.3}$$

$$\mathbf{M}^\top\mathbf{M} = \mathbf{V}\mathbf{S}\mathbf{U}^\top\mathbf{U}\mathbf{S}\mathbf{V}^\top = \mathbf{V}\mathbf{S}^2\mathbf{V}^\top. \tag{B.4}$$

The substitution of (B.3) into (B.1) yields $\boldsymbol{\Sigma}_B = \mathbf{U}(\mathbf{S}^2/C)\mathbf{U}^\top$ and so

$$\begin{aligned}\boldsymbol{\Sigma}_B^+ &= \mathbf{U}\left(\frac{\mathbf{S}^2}{C}\right)^{-1}\mathbf{U}^\top && \text{(tedious but straightforward to verify[1])} \\ &= C\mathbf{U}\mathbf{S}^{-2}\mathbf{U}^\top \\ &= C\mathbf{M}\mathbf{V}\mathbf{S}^{-1}\mathbf{S}^{-2}\mathbf{S}^{-1}\mathbf{V}^\top\mathbf{M}^\top && \text{(substitution of (B.2))} \\ &= C\mathbf{M}\mathbf{V}\mathbf{S}^{-4}\mathbf{V}^\top\mathbf{M}^\top. && \text{(B.5)}\end{aligned}$$

Furthermore, (B.3) and (B.4) can be converted to eigendecompositions of $\mathbf{M}\mathbf{M}^\top$ and $\mathbf{M}^\top\mathbf{M}$ respectively by zero-padding $\mathbf{S}$ and extending $\mathbf{U}$ and $\mathbf{V}$ to orthonormal matrices. This means that $\mathbf{M}\mathbf{M}^\top$ and $\mathbf{M}^\top\mathbf{M}$ share the nonzero eigenvalues $\{\lambda_r\}_{r=1}^R$ computed in

[1] One checks the definition of the Moore-Penrose pseudoinverse: $\boldsymbol{\Sigma}_B\boldsymbol{\Sigma}_B^+\boldsymbol{\Sigma}_B = \boldsymbol{\Sigma}_B$, $\boldsymbol{\Sigma}_B^+\boldsymbol{\Sigma}_B\boldsymbol{\Sigma}_B^+ = \boldsymbol{\Sigma}_B^+$, $(\boldsymbol{\Sigma}_B\boldsymbol{\Sigma}_B^+)^\top = \boldsymbol{\Sigma}_B\boldsymbol{\Sigma}_B^+$, and $(\boldsymbol{\Sigma}_B^+\boldsymbol{\Sigma}_B)^\top = \boldsymbol{\Sigma}_B^+\boldsymbol{\Sigma}_B$.

algorithm 1. In other words, $\mathbf{S}^2 = \operatorname{diag}(\lambda_1, ..., \lambda_R)$, which means

$$\begin{aligned}
\operatorname{tr}(\boldsymbol{\Sigma}_W \boldsymbol{\Sigma}_B^+) &= \frac{1}{N}\sum_{c=1}^{C}\sum_{i=1}^{n_c} \operatorname{tr}((\mathbf{h}_{c,i} - \boldsymbol{\mu}_c)(\mathbf{h}_{c,i} - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_B^+) \\
&= \frac{1}{N}\sum_{c=1}^{C}\sum_{i=1}^{n_c} (\mathbf{h}_{c,i} - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_B^+ (\mathbf{h}_{c,i} - \boldsymbol{\mu}_c) && (\operatorname{tr}(\mathbf{a}\mathbf{b}^\top) = \mathbf{b}^\top\mathbf{a}) \\
&= \frac{C}{N}\sum_{c=1}^{C}\sum_{i=1}^{n_c} (\mathbf{h}_{c,i} - \boldsymbol{\mu}_c)^\top \mathbf{M}\mathbf{V}\mathbf{S}^{-4} \underbrace{\mathbf{V}^\top\mathbf{M}^\top(\mathbf{h}_{c,i} - \boldsymbol{\mu}_c)}_{\boldsymbol{\kappa}_{c,i} \in \mathbb{R}^R} && (\text{substitution of (B.5)}) \\
&= \frac{C}{N}\sum_{c=1}^{C}\sum_{i=1}^{n_c} \|\boldsymbol{\kappa}_{c,i}^\top \mathbf{S}^{-2}\|^2,
\end{aligned}$$

which is exactly the value returned by algorithm 1.

The space complexity is straightforward; $\mathbf{M}$ is clearly the largest variable explicitly stored and storing it requires $\mathcal{O}(CD)$ space. The runtime complexity is also not too difficult to prove. During the first pass, $f^{l:1}$ is evaluated N times and N additions of D-dimensional vectors are performed. Between the first and second passes, under the assumption that $C \ll D$, every calculation is at most linear in D. During the second pass, $f^{l:1}$ is evaluated N times again, and the two matrix multiplications $\mathbf{M}^\top(f^{l:1}(\mathbf{x}_n) - \boldsymbol{\mu}_c) \triangleq \mathbf{z}$, $\mathbf{V}^\top\mathbf{z}$, which cost $\mathcal{O}(CD)$ and $\mathcal{O}(RC) \subseteq \mathcal{O}(CD)$ respectively, are also performed N times. Therefore, the total runtime is $\mathcal{O}(N(E + CD))$.

B.2. Proof of (2.1)

The proof of (2.1) is rather straightforward. As is usual, the layer superscript is omitted for simplicity. Recall that $\boldsymbol{\Sigma}_T = \boldsymbol{\Sigma}_W + \boldsymbol{\Sigma}_B$ and so it suffices to show that $\boldsymbol{\Sigma}_T = \mathrm{G}^l/N - \bar{\boldsymbol{\mu}}\bar{\boldsymbol{\mu}}^\top$. This is easily shown from the definition of $\boldsymbol{\Sigma}_T$:

$$\begin{aligned}
\boldsymbol{\Sigma}_T &= \frac{1}{N}\sum_{c=1}^{C}\sum_{i=1}^{n_c} (\mathbf{h}_{c,i} - \bar{\boldsymbol{\mu}})(\mathbf{h}_{c,i} - \bar{\boldsymbol{\mu}})^\top \\
&= \frac{1}{N}\sum_{c=1}^{C}\sum_{i=1}^{n_c} (\mathbf{h}_{c,i}\mathbf{h}_{c,i}^\top - \bar{\boldsymbol{\mu}}\mathbf{h}_{c,i}^\top - \mathbf{h}_{c,i}\bar{\boldsymbol{\mu}}^\top + \bar{\boldsymbol{\mu}}\bar{\boldsymbol{\mu}}^\top) \\
&= \mathbf{G}^l - \bar{\boldsymbol{\mu}}\left(\frac{1}{N}\sum_{c=1}^{C}\sum_{i=1}^{n_c} \mathbf{h}_{c,i}^\top\right) - \left(\frac{1}{N}\sum_{c=1}^{C}\sum_{i=1}^{n_c} \mathbf{h}_{c,i}\right)\bar{\boldsymbol{\mu}}^\top + \bar{\boldsymbol{\mu}}\bar{\boldsymbol{\mu}}^\top \\
&= \mathbf{G}^l - \bar{\boldsymbol{\mu}}\bar{\boldsymbol{\mu}}^\top - \bar{\boldsymbol{\mu}}\bar{\boldsymbol{\mu}}^\top + \bar{\boldsymbol{\mu}}\bar{\boldsymbol{\mu}}^\top \\
&= \mathbf{G}^l - \bar{\boldsymbol{\mu}}\bar{\boldsymbol{\mu}}^\top.
\end{aligned}$$