MSc Artificial Intelligence

Master Thesis

# Greedy InfoMax for Self-Supervised Representation Learning

by

Sindy Löwe

Student-ID: 11594969

September 12, 2019

36 European Credits
January 2019 - September 2019

*Supervisor:*
Bastiaan S. Veeling
*Co-supervisor:*
Peter O'Connor

*Assessor:*
Prof. Dr. Max Welling

University of Amsterdam

## *Abstract*

We propose a novel deep learning method for local self-supervised representation learning that does not require labels nor end-to-end backpropagation but exploits the natural order in data instead. Inspired by the observation that biological neural networks appear to learn without backpropagating a global error signal, we split a deep neural network into a stack of gradient-isolated modules. Each module is trained to maximize the Mutual Information between its consecutive outputs using the InfoNCE bound from Oord et al. (2018). Despite this greedy training, we demonstrate that each module improves upon the output of its predecessor, and that the representations created by the top module yield highly competitive results on downstream classification tasks in the audio and visual domain. The proposal enables optimizing modules asynchronously, allowing for large-scale distributed training of very deep neural networks on unlabeled datasets.

*Keywords*

Self-supervised Learning · Representation Learning · Mutual Information Optimization · Backpropagation

## *Acknowledgments*

A master's study and the writing of a thesis, in particular, are significant endeavors that are impossible to achieve by yourself. Here, I would like to thank those who helped me on my way.

First of all, I would like to thank my supervisor, Bastiaan Veeling. Throughout the work on my thesis, I genuinely enjoyed our exciting and fruitful discussions that often led me to reflect on new angles of my research.

Additionally, I would like to thank Peter O'Connor for co-supervising me. Thank you for providing me with lots of valuable inputs and for helping me to determine and define the topic of my thesis.

Besides my supervisors, I would like to thank my fellow students for turning this master's program into the best experience possible. Working together with you has taught me many lessons outside of our lectures and enjoying life in Amsterdam with you did not only guarantee for fun but also brought along friendships that I hope will last a lifetime.

In particular, I would like to thank Joop Pascha for his love and support. Thank you for believing in me, for encouraging me to do my best and for always providing a shoulder to lean on.

Outside of Amsterdam, I would like to thank my family and especially my parents for their continuous support, for always being there for me and for encouraging me to follow my goals.

*September 2019*

# Contents

# 1 *Introduction*

Building human-level general Artificial Intelligence (AI) has been a major goal in computer science ever since the creation of this scientific field (Turing, 1950). Back then, expectations that machines would soon reach and surpass human-level intelligence were high. In 1965, Nobel price winner Herbert Simon famously predicted that "machines will be capable, within twenty years, of doing any work a man can do". In hindsight, achieving general AI turned out to be much more challenging.

Recently, the tech world shows new enthusiasm to solve human-level AI. Deep learning enabled machines to surpass human performance in various fields, such as image classification (He et al., 2016a), the ancient Chinese game of Go (Silver et al., 2017) and StarCraft II (Vinyals et al., 2019). However, these successes are very restricted. The high performance of these algorithms depends on raw compute power incomparable to the efficiency of the human brain, both in terms of energy consumption and sample efficiency. At the same time, machines outperform humans only in very narrow, well-defined domains. In most areas essential to human intelligence, such as reasoning, common-sense and basic motor control, algorithms still fail to approach the cognitive abilities of a four-year-old.

Overall, building human-level AI has proven itself to be a daunting task. The search space of possible solutions is vast and, given our failure of finding any solution yet, probably very sparsely populated (Hassabis et al., 2017). So far, the human brain provides the only proof that a solution exists at all. We argue that it can thus provide valuable insights into the aspects necessary for achieving higher-level general intelligence.

Originally, Artificial Neural Networks (ANNs) were inspired by biological neural networks (McCulloch and Pitts, 1943), and neuroscience continues to play a role in the development of new approaches (Cox and Dean, 2014). Nonetheless, most recent advances are guided by insights into the mathematics of efficient optimization. ANNs nowadays are typically optimized using batch-wise stochastic gradient descent and the weights are updated using backpropagation. Additionally, numerous tricks such as better weight initializations (Sutskever et al., 2013), dropout (Srivastava et al., 2014) or batch normalization (Ioffe and Szegedy, 2015) have been developed to improve the stability and efficiency of the training process and the final testing results. There are little apparent connections between these approaches and neuroscience (Marblestone et al., 2016).

In this thesis, we challenge the most commonly applied learning algorithm for ANNs, end-to-end backpropagation of a global, supervised loss function. Although empirically proven to be very successful (Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016a), this approach is considered biologically implausible for several reasons. For one, supervised learning requires enormous labeled datasets to ensure generalization. In contrast, children receive mostly unlabeled inputs, and they can learn to categorize a new image class based on a handful of samples. Additionally, despite some evidence for top-down connections in the brain, there does not appear to be a global objective that is optimized by backpropagating error signals (Crick, 1989; Marblestone et al., 2016). Instead, the biological brain is highly modular and learns predominantly based on local information (Caporale and Dan, 2008).

In addition to this lack of a natural counterpart, the supervised training of neural networks with end-to-end backpropagation suffers from practical disadvantages as well. Supervised learning requires labeled inputs, which are expensive to obtain. As a result, it is not applicable to the majority of available data. On top of that, it suffers from a higher risk of overfitting, as the number of parameters required for a deep model often exceeds the number of labeled data points at hand. End-to-end backpropagation, on the other hand, creates a substantial memory overhead in a naïve implementation, as the entire computational graph, including all parameters, activations and gradients, needs to fit in a processing unit's working memory. In the literature, there exist two approaches to prevent this which either require the recomputation of intermediate outputs (Salimans and Bulatov, 2017) or expensive reversible layers (Jacobsen et al., 2018). Nonetheless, this inhibits the application of deep learning models to high-dimensional input data that surpasses current memory constraints. This problem is perpetuated as end-to-end training does not allow for an exact way of asynchronously optimizing individual layers (Jaderberg et al., 2017). In a globally optimized network, every layer needs to wait for its predecessors to provide its inputs, as well as for its successors to provide gradients. This forward and backward locking of the network caused by the backpropagation algorithm additionally impedes the efficiency of hardware accelerator design due to a lack of locality.

In order to eliminate these problems associated with end-to-end backpropagation of a global, supervised loss function, we take inspiration from biological learning processes. First, we remove end-to-end backpropagation by dividing a deep architecture into consecutive modules that we train greedily using a local loss per module. Second, we train our model on unlabeled data by using a self-supervised loss that exploits the natural order of the data. Overall, this resembles the learning processes found in the brain more closely and prevents the computational issues described above, as we will argue in Chapter 6.

Based on this, we propose a novel learning approach for deep neural networks, Greedy InfoMax (GIM). It encodes unlabeled high-dimensional sequential or spatial data iteratively, module by module. All modules are trained with a self-supervised loss that enforces the individual modules to maximally preserve the information of their inputs. This enables the stacked model to collectively create compact representations that can be used for downstream tasks. We will describe this new algorithm in Chapter 3 and demonstrate its efficacy in Chapter 4. On top of that, we will provide a thorough discussion of the relevant background knowledge in Chapter 2. In Chapter 5, we will embed our method into the current research and compare it to other biologically inspired alternatives of the backpropagation algorithm.

Overall, our contributions are as follows:

- On audio and image classification tasks, the proposed Greedy InfoMax algorithm achieves competitive performance to its end-to-end trained counterpart, the Contrastive Predictive Coding (CPC) model from Oord et al. (2018), despite greedy self-supervised training without a global objective.

- Our proposal enables asynchronous, decoupled training of neural networks, allowing for training arbitrarily deep networks on larger-than-memory input data.

- We show that Mutual Information maximization is especially suitable for layer-by-layer greedy optimization, and argue that this reduces the problem of vanishing gradients.

# 2 Background

In our proposed method, Greedy InfoMax (GIM), we combine two concepts: the learning of representations from data by Mutual Information maximization, and training a neural network without end-to-end backpropagation. In this chapter, we will lay the mathematical foundations for these two concepts and motivate their usage. After that, we will provide a review on how they can be inspired biologically.

## 2.1 Mutual Information Estimation for Representation Learning

Recently, approaches based on Mutual Information (MI) maximization have received increased attention in the machine learning community, especially in the field of representation learning (Oord et al., 2018; Hjelm et al., 2019; Hénaff et al., 2019; Tian et al., 2019; Bachman et al., 2019; Sun et al., 2019). In this section, we will lay the mathematical foundations for the relevant information-theoretic concepts and motivate how this approach can be useful for the creation of representations from data. The method which we develop in this thesis is based on one MI maximization approach in particular, namely Contrastive Predictive Coding (CPC) (Oord et al., 2018), which we describe at the end of this section.

### 2.1.1 Mutual Information

Before exploring the applicability of information-theoretic approaches to representation learning, we will lay the mathematical foundations for them. These involve the definitions for Shannon Entropy, Conditional Entropy and Mutual Information and some of their core properties.[1]

[1] All definitions and propositions are taken from Cramer and Fehr (2011).
Note: We will express all definitions and propositions for discrete random variables. They can be trivially extended to continuous probability distributions by replacing the sums with integrals over the set of possible values.

The Shannon Entropy is a measure for the amount of uncertainty in a random variable. Suppose we are given a probabilistic event $\mathcal{A}$ with probability $P[\mathcal{A}]$ for some probability measure $P$. We can express how surprised we are when event $\mathcal{A}$ occurs using the *surprisal value* $\log \frac{1}{P[\mathcal{A}]}$. This value describes that events with high probabilities are less surprising than events with low probabilities. An event $\mathcal{A}$ with probability $P[\mathcal{A}] = 1$, for example, will result in a surprisal value of zero since its occurrence can be anticipated entirely. The *expected* surprisal value for a random variable $X$ can then indicate the amount of uncertainty in that variable or, interpreted differently, how much information is gained by revealing its outcome. This expected surprisal value of a random variable is more commonly known as the (Shannon) entropy:

**Definition 2.1: Entropy**

*Let $X$ be a random variable with image $\mathcal{X}$. Then the entropy $H(X)$ is defined as*

$$H(X) := \mathop{\mathbb{E}}_{X}\left[\log \frac{1}{P_X(X)}\right] \tag{2.1}$$

$$= \sum_{x\in\mathcal{X}} P_X(x) \log \frac{1}{P_X(x)} . \tag{2.2}$$

A constant distribution lower bounds the entropy. As there is no uncertainty about its outcome, it yields an entropy of zero. A uniform distribution yields the upper bound. Here, all outcomes are equally likely, resulting in the highest possible uncertainty. The following proposition conveys this intuition:

**Proposition 2.2**

*Let $X$ be a random variable with image $\mathcal{X}$. Then it holds that*

$$0 \leq H(X) \leq \log|\mathcal{X}| , \tag{2.3}$$

*where equality on the left side holds iff $\exists x \in \mathcal{X}$: $P_X(x) = 1$. Equality on the right side holds iff $\forall x \in \mathcal{X} : P_X(x) = \frac{1}{|\mathcal{X}|}$.*

For the proof of this proposition, see Appendix A.1. The probability distribution over the random variable $X$ might change given an event $\mathcal{A}$. Correspondingly, the entropy of $X$ can change according to this conditional probability distribution $P_{X|\mathcal{A}}$. This leads us to the definition of the conditional entropy:

**Definition 2.3: Conditional Entropy**

*Let $X, Z$ be random variables with images $\mathcal{X}, \mathcal{Z}$. Then the conditional entropy $H(X|Z)$ of $X$ given $Z$ is defined as*

$$H(X|Z) := \sum_{z\in\mathcal{Z}} P_Z(z) H(X|Z=z) \tag{2.4}$$

$$= \sum_{z\in\mathcal{Z}} P_Z(z) \sum_{x\in\mathcal{X}} P_{X|Z}(x|z) \log \frac{1}{P_{X|Z}(x|z)} \tag{2.5}$$

$$= \sum_{x\in\mathcal{X}, z\in\mathcal{Z}} P_{XZ}(x,z) \log \frac{P_Z(z)}{P_{XZ}(x,z)} . \tag{2.6}$$

Similar to the Shannon entropy, the conditional entropy can be interpreted as the uncertainty in $X$ when $Z$ is given. On average, additional information (i.e. knowing $Z$) can only decrease our uncertainty about $X$, which is expressed in the following proposition:

**Proposition 2.4**

*Let $X, Z$ be random variables with images $\mathcal{X}, \mathcal{Z}$. Then it holds that*

$$0 \leq H(X|Z) \leq H(X) , \tag{2.7}$$

*where equality on the left side holds iff $\forall z \in \mathcal{Z}, \exists x \in \mathcal{X} : P_{X|Z}(x|z) = 1$, i.e. X is determined by Z. Equality on the right side holds iff X and Z are independent.*

For the proof of this proposition, see Appendix A.2. Using the entropy and conditional entropy, we can define the Mutual Information as follows:

**Definition 2.5: Mutual Information**

*Let $X, Z$ be random variables. Then we can express the Mutual Information (MI) as*

$$I(X, Z) := H(X) - H(X|Z) , \tag{2.8}$$

*where $H(X)$ is the entropy of X, and $H(X|Z)$ the conditional entropy of X given Z.*

Following the interpretation of the entropy as a measure of uncertainty, the Mutual Information (MI) can be understood as the decrease in uncertainty in X when given Z. Thus, MI expresses the dependence between random variables. We can rewrite it as follows:

$$\begin{aligned} I(X, Z) = & \sum_{x \in \mathcal{X}} P_X(x) \log \frac{1}{P_X(x)} \\ & - \sum_{x \in \mathcal{X}, z \in \mathcal{Z}} P_{XZ}(x, z) \log \frac{P_Z(z)}{P_{XZ}(x, z)} && (2.9) \\ = & \sum_{x \in \mathcal{X}, z \in \mathcal{Z}} P_{XZ}(x, z) \log \frac{P_{XZ}(x, z)}{P_X(x) P_Z(z)} && (2.10) \\ = & \, KL(P_{XZ} \mid\mid P_X P_Z) , && (2.11) \end{aligned}$$

where $KL(\cdot \mid\mid \cdot)$ denotes the Kullback-Leibler (KL) divergence, a measure of how different one probability distribution is from another. This gives us another interpretation of the MI as the difference between the joint probability distribution of the random variables $X$ and $Z$ and the product of their marginals.

Following propositions 2.2 and 2.4, MI exhibits the ensuing properties:

$$\begin{aligned} & I(X, Z) \geq 0 && \text{(Non-negativity)} \\ & I(X, Z) = 0 \iff X, Z \text{ independent} . && (2.12) \end{aligned}$$

Additionally, from $P_{XZ} = P_{ZX}$ and $P_X P_Z = P_Z P_X$ and thus $KL(P_{XZ} \,||\, P_X P_Z) = KL(P_{ZX} \,||\, P_Z P_X)$, we get that

$$I(X, Z) = I(Z, X)\ . \tag{Symmetry}$$

Kinney and Atwal (2014) have shown that in contrast to correlation, MI captures non-linear statistical dependencies between variables and can thus act as a measure of true dependence. As a result, it provides a practical method for equitably quantifying associations in large datasets.

However, in practice, estimating MI is notoriously difficult. In most settings, we do not have access to the true data density $P_X$. This problem can be circumvented by using a stochastic approximation, assuming we have access to a sufficiently large empirical distribution $\hat{P}_X$. The second arising problem, unfortunately, is much harder to circumvent: computing the marginal $P_Z = \sum_{x \in \mathcal{X}} P_{XZ}(x, z)$ is often intractable, especially in high-dimensional spaces. Therefore, in practice, one usually makes use of tractable lower and upper bounds. These allow for optimizing the MI even when it is intractable. When one tries to maximize the MI, for instance, this can be achieved implicitly by maximizing the tractable lower bound.

### 2.1.2 *Mutual Information Estimation for Representation Learning*

Given that we can maximize the MI between two random variables, how does this help us with the creation of better representations of data? In order to answer this question, we will first describe the general setup of representation learning. Then, we will develop an intuition on how information-theoretic approaches can help us to create the desired features.

Since no clear objective for representation learning has been identified yet, we will only provide an intuitive description of the task setting: given unlabeled input data $X$, learn a function $g$ that maps the input into some, typically lower-dimensional space. The created representations $Z$ are considered to be better when they help to achieve higher performances on supervised downstream tasks or to solve these tasks more efficiently, e.g. by requiring less annotated data. However, the target task might be unknown a priori or might change throughout the employment of the created representations. Thus, in order to create useful features, Bengio et al. (2013) argue that one should disentangle as many explanatory factors as possible while discarding as little information about the input as practical.

This is where information-theoretic approaches lend themselves naturally for representation learning. By maximizing the MI between the input and the output of a model, we can ensure that as little information about the input is discarded as possible. This approach is inspired by the InfoMax principle (Linsker, 1988) and has first been applied to unsupervised representation learning by Bridle et al. (1992).

However, two fundamental problems arise when maximizing the MI between the input and the output of a model. The first problem is that the MI $I(X, Z)$ is at its maximum when $X = Z$ and thus $H(X|Z) = 0$, i.e. when the output $Z$ of an algorithm is equal to its input $X$. Then, all information of the input is present in the representation, but arguably no meaningful transformation has been applied. Consequently, we can not assume that the InfoMax principle inherently pushes for a disentangled representation.

The second fundamental problem of applying information theory to representation learning: when measuring the MI between two variables, no value judgments are made. Redundancy is discounted, but the MI cannot discern between useful and irrelevant information. According to information theory, a maximally unpredictable signal is the most informative one[2]. In contrast, random disturbances in the data are most likely to be useless for representation learning tasks and should usually be discarded as noise.

[2] This follows from the interpretation of entropy as a measure of how informative the outcome of a random experiment is. We gain the most information when the entropy is at its maximum, which is reached by a uniformly distributed random variable (see Proposition 2.2)

Krause et al. (2010) have shown how these theoretical restrictions can indeed lead to degenerate solutions in practice. They train a probabilistic classifier without supervision by maximizing the MI between the input $X$ and the output $Z$, which they interpret as a probability distribution over a discrete random variable (the class label) $Y$. They found that a classifier trained with the InfoMax objective tends to fragment the output space into a large number of categories, essentially classifying each data point into its own category.

Consequentially, some regularization is necessary to create useful representations using information-theoretic approaches. In the work mentioned above (Krause et al., 2010), they suggest penalizing complex decision boundaries in order to yield sensible clustering solutions.

A more generic solution exploits the structure that is present in the input data. Becker (1996) took a first step in this direction proposing to extract features that are coherent across inputs. Here, "coherence" describes that parts of an input can be predicted by other parts. This coherence may arise over different sensory channels, e.g. when we see a cup of coffee, smell and taste its aroma and feel the warm cup in our hands. Alternatively, we may exhibit coherence of a signal over time. These temporal coherences were subsequently coined slow features (Wiskott and Sejnowski, 2002). To illustrate: a patch of a few milliseconds of raw speech utterances shares information with neighboring patches such as the speaker identity, emotion, and phonemes, while it does not share these with random patches drawn from other utterances. Similarly, a small patch from a natural image shares many aspects with neighboring patches such as the depicted object or lighting conditions and thus exhibits spatial coherence.

These coherences can be exploited for extracting useful features from data. Suppose we are given an input $X$ and let $X^{(1)}, X^{(2)}$ be coherent parts of this input. Then we can employ encoders $g_1, g_2$ which possible share parameters and optimize them with the following objective:

$$\max_{g_1 \in \mathcal{G}_1, g_2 \in \mathcal{G}_2} I(g_1(X^{(1)}), g_2(X^{(2)})) \,, \tag{2.13}$$

where $\mathcal{G}_1, \mathcal{G}_2$ describe function classes that can specify structural constraints of the encoders. Thus, our objective for representation learning becomes to maximize the MI between representations of coherent parts of the input.

By applying the data processing inequality[3] multiple times, one can show that (Tschannen et al., 2019):

$$I(g_1(X^{(1)}), g_2(X^{(2)})) \leq I(X; g_1(X^{(1)}), g_2(X^{(2)})) \,. \tag{2.14}$$

[3] **Data Processing Inequality** Let $X \to Y \to Z$ be a Markov Chain (i.e. $P_{Z|XY} = P_{Z|Y}$). Then it holds that $I(X,Y) \geq I(X,Z)$.

Thus, Equation (2.13) maximizes a lower bound of the InfoMax objective ($\max_{g \in \mathcal{G}} I(X, g(X)) = \max I(X,Z)$). Nonetheless, it manages to resolve the two aforementioned problems associated with this objective. For one, $g_1$ and $g_2$ have to extract and disentangle features that are present in both $X^{(1)}$ and $X^{(2)}$, such that their encodings become maximally informative of each other. This new objective also resolves the second problem, since random noise is most likely to get filtered as an uninformative part of the input. An additional advantage of this approach is that the representations $g_1(X^{(1)})$ and $g_2(X^{(2)})$ are typically lower-dimensional than the original input $X$ and as a result, the MI becomes easier to estimate.

As argued before, there are many possibilities for coherence to arise in data and consequentially, there are various approaches based on Equation (2.13) that exploit different modalities and aspects of the data. Becker (1996), for example, propose to extract coherent features from different receptive fields ($X^{(1)}$, $X^{(2)}$) by maximizing the MI between the outputs of different network modules ($g_1, g_2$) and show that this method is capable of extracting higher-order features from data. More recently, Hjelm et al. (2019) demonstrate that exploiting the spatial coherence in natural images by applying $g_1$ on the entire image $X^{(1)}$ to extract global features and $g_2$ on image patches $X^{(2)}$ to extract local features can be highly effective for representation learning.

Recent work (Tschannen et al., 2019) suggests that the success of these MI maximization methods might only be loosely attributed to the properties of MI itself. Instead, they provide an alternative interpretation based on the triplet loss known from deep metric learning. In this setting, one tries to create representations such that the *distance* between $g_1(X^{(1)})$

and $g_2(X^{(2)})$ is smaller than the distance between $g_1(X^{(1)})$ and some representation of a non-coherent part of the input $g(X^{(j)})$. Nonetheless, the goal remains similar, namely to create representations that preserve the coherences present in the input. Since MI maximization provides an intuitive interpretation of this goal, we continue to use the information-theoretic framework.

The method that we develop in this thesis is based on Contrastive Predictive Coding (Oord et al., 2018). This representation learning approach exploits the temporal coherence of sequential data. It extracts useful features by maximizing the MI between the extracted representations of temporally nearby patches. Thus, $X^{(1)}$ corresponds to a patch at time $t$ and $X^{(2)}$ to future patches at $t+k$. Next, we will provide a more in-depth description of this self-supervised end-to-end training approach.

### 2.1.3 Contrastive Predictive Coding and the InfoNCE loss

Contrastive Predictive Coding (CPC) works by combining two submodels. As depicted in Figure 2.1, it first processes the sequential input signal $x$ using a deep encoding model $g_{enc}(x_t) = z_t$. Additionally, it produces a representation $c_t$ that aggregates the information of all patches up to time-step $t$ using an autoregressive model $g_{ar}(z_{0:t}) = c_t$. Then, the MI between the extracted representations $z_{t+k}$ and $c_t$ of temporally nearby patches is maximized by employing a specifically designed global probabilistic loss: Following the principles of Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010), CPC takes a bag $X = \{z_{t+k}, z_{j_1}, z_{j_2}, ...z_{j_{N-1}}\}$ for each delay $k$, with one "positive sample" $z_{t+k}$ which is the encoding of the input that follows $k$ time-steps after $c_t$, and $N-1$ "negative samples" $z_{j_n}$ which are uniformly drawn from all available encoded input sequences.

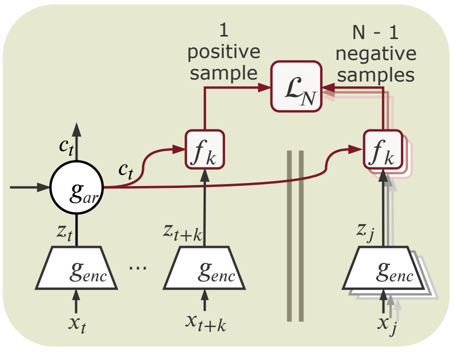


Figure 2.1: Contrastive Predictive Coding (CPC) trains the two submodels $g_{enc}$ and $g_{ar}$, as well as the classifier $f_k$, by contrasting temporally nearby representation pairs (i.e. positive samples $(z_{t+k}, c_t)$) against random pairs (i.e. negative samples $(z_j, c_t)$).

Each pair of encodings $(z_j, c_t)$ is scored using a function $f(\cdot)$ to predict how likely it is that the given $z_j$ is the positive sample $z_{t+k}$. In practice, Oord et al. (2018) use a log-bilinear model

$$f_k(z_j, c_t) = \exp\left(z_j^T W_k c_t\right), \tag{2.15}$$

with a unique weight-matrix $W_k$ for each $k$-steps-ahead prediction. The scores from $f(\cdot)$ are used to predict which sample in the bag $X$ is correct, leading to the InfoNCE loss:

$$\mathcal{L}_N = -\sum_k \mathop{\mathbb{E}}_X \left[\log \frac{f_k(z_{t+k}, c_t)}{\sum_{z_j \in X} f_k(z_j, c_t)}\right]. \tag{2.16}$$

This loss is used to optimize both the encoding model $g_{enc}$ and the auto-regressive model $g_{ar}$ to extract the features that are consistent over neighboring patches but which diverge between random pairs of patches. At the same time, the scoring model $f_k$ learns to use those features to classify the matching pair correctly. In practice, the loss is trained using stochastic gradient descent with mini-batches drawn from a large dataset of sequences, and negative samples drawn uniformly from all sequences in the minibatch.

As demonstrated in the appendix of Oord et al. (2018) and proven by Poole et al. (2018), $-\mathcal{L}_N$ can be reformulated as a lower bound on the MI $I(z_{t+k}, c_t)$. Intuitively, the connection between $-\mathcal{L}_N$ and the MI can be made by the formulation of MI as $I(X, Z) = KL(P_{XZ} \mid\mid P_X P_Z)$ and the following idea: If a classifier can distinguish between samples from the joint $P_{XZ}$ and those from the marginal $P_X P_Z$, than $X$ and $Z$ have a high MI. In the setting of CPC, this implies that if the classifier $f_k$ can distinguish the positive pair of encodings $(z_{t+k}, c_t)$ from the negative pairs $(z_j, c_t)$, than $z_{t+k}$ and $c_t$ have a high MI.

Minimizing the loss $\mathcal{L}_N$ thus optimizes the MI between consecutive patch representations $I(z_{t+k}, c_t)$, which in itself lower bounds the MI $I(x_{t+k}, c_t)$ between the future input $x_{t+k}$ and the current representation $c_t$.[4]

[4] This argumentation is equivalent to the bound $I(g_1(X^{(1)}), g_2(X^{(2)})) \leq I(X; g_1(X^{(1)}), g_2(X^{(2)}))$ proven by Tschannen et al. (2019) mentioned above.

## 2.2 *Backpropagation and its Biological Plausibility*

Recent state of the art results in various areas such as computer vision (Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016a), natural language understanding (Devlin et al., 2019; Yang et al., 2019) and reinforcement learning (Silver et al., 2016; Vinyals et al., 2019) were achieved with the help of deep learning. They all rely on ANNs that are optimized using backpropagation.

Originally, ANNs were biologically inspired, and they have shown to develop representations similar to those found in the brain areas involved in similar tasks, such as Gabor-like filters in the early visual system. Backpropagation's biological plausibility, on the other hand, is much more debatable (Crick, 1989; Whittington and Bogacz, 2019). In this section, we will first provide a short description of ANNs and how to optimize their parameters using the backpropagation algorithm. Then, we will depict some of its computational disadvantages before discussing aspects of this learning algorithm that are biologically implausible. Finally, we will include a short discussion on Backpropagation Through Time. This algorithm is typically used for optimizing the parameters in Recurrent Neural Networks, which are ANNs specialized for sequential input data.

### 2.2.1 *Backpropagation in Artificial Neural Networks*

Artificial Neural Networks (ANNs) are parametric, non-linear representation learning functions. They are typically hierarchical, such that the model can be split into a number of layers $1, \dots, L$. Each layer applies a non-linear operation on its input and propagates its result to the following layer. We can express this more formally, such that given an input $x$, an ANN does the following computation:

$$o^L(x; W^1, \dots, W^L) = h^L(h^{L-1}(\dots h^1(x, W^1) \dots, W^{L-1}), W^L) \ , \tag{2.17}$$

where $W^l$ describes the parameters for layer $l$ and $h^l(o^{l-1}, W^l)$ is a differentiable, non-linear function on the layer-input $o^{l-1}$, which is either the output of the previous layer or the input to the network ($o^0 = x$).

In the following, without loss of generality, we will focus on the most elementary type of a neural network using only fully-connected layers. This allows us to split the calculations of each layer into separate units, commonly described as neurons. Formally, the calculation for a neuron $j$ in layer $l$ can be described as

$$z_j^l = \sum_{k=1}^{n} w_{kj}^l o_k^{l-1} \tag{2.18}$$

$$o_j^l = \varphi^l(z_j^l) \ , \tag{2.19}$$

where $\varphi^l$ is a non-linear, differentiable activation function, $o_k^{l-1}$ the output of neuron $k$ in the previous layer $l-1$ (with neurons $k \in \{1, \dots, n\}$) and $w_{kj}^l$ the weight for the connection from neuron $k$ in layer $l-1$ to neuron $j$ in layer $l$.

When using ANNs in a supervised setting, the goal is to find the optimal parameters $W*$ such that given inputs $X$, the network produces outputs that are close to some target values $Y$. In order to do so, one formalizes the discrepancy between the output of the network and the targets as the loss $\mathcal{L}$, which we try to minimize:

$$W* \leftarrow \arg\min_{W} \sum_{(x,y) \subseteq \{X,Y\}} \mathcal{L}(y, o^L(x; W)) \ , \tag{2.20}$$

where $W$ is the concatenation of all weight matrices $W^1, \dots, W^L$.

Backpropagation (Rumelhart et al., 1985) is the algorithm most commonly used to optimize the parameters in ANNs based on the loss $\mathcal{L}$. In order to effectively enable the network to produce better outputs, all weights $W^1, \dots, W^L$ within the architecture need to be adjusted, as they all influence the output of the final layer. Backpropagation provides a theoretically grounded approach to calculate the necessary weight changes. It assigns credit to the individual weights throughout all layers of the network describing which weights need to be adjusted, in which direction and by how much.

For calculating the weight changes, backpropagation makes use of the gradients of the loss function. Generally, gradients indicate in which direction one would have to change the parameters of a function (here, the weights) in order to achieve the steepest decrease in the function value (here, the loss). Thus, backpropagation makes use of gradient descent, i.e. it takes steps proportional to the negative of the gradients for finding a (local) minimum of the loss. The main insight of backpropagation is then to make use of the chain rule in order to calculate the gradients with respect to the parameters of the inner layers of the network. Going back to the example of a network with fully-connected layers, backpropagation provides the following update rules for all weights $w_{ij}$ of the network (discarding the layer index $l$ here for brevity):

$$\begin{aligned} \Delta w_{ij} &= -\eta \frac{\partial \mathcal{L}}{\partial w_{ij}} && (2.21) \\ &= -\eta o_i \delta_j && (2.22) \end{aligned}$$

where

$$\delta_j = \begin{cases} \frac{\partial \mathcal{L}}{\partial o_j} \frac{\partial \varphi(z_j)}{\partial z_j} & \text{if } j \text{ is an output neuron} \\ \left(\sum_{m \in M} w_{jm} \delta_m\right) \frac{\partial \varphi(z_j)}{\partial z_j} & \text{if } j \text{ is an inner neuron} \end{cases} \tag{2.23}$$

where $M$ contains all the neurons receiving inputs from neuron $j$. See Appendix A.3 for the derivation of these update rules. Thus, the update of the weight $w_{ij}$ depends on three factors (Equation (2.22)): the learning rate $\eta$, the input from the previous neuron $o_i$ and an error term $\delta_j$. This error term needs to be calculated differently depending on whether $j$ is an inner neuron or an output neuron (Equation (2.23)). If it is an output neuron, $\delta_j$ is the product of the derivative of the loss function with respect to the output of the neuron $j$ and the derivate of the activation function $\varphi$ with respect to its input. If $j$ is an inner neuron, its influence on the final loss is more intricate as it may influence the behavior of a number of downstream neurons, including several output neurons. As a result, $\delta_j$ is more involved: it takes a sum of all the error terms $\delta_m$ of the neurons that receive inputs from neuron $j$ weighted by their respective connection strengths $w_{jm}$ and multiplies this sum with the derivate of the activation function $\varphi$.

### 2.2.2 *Computational Disadvantages of Backpropagation*

Although empirically proven to be highly effective, the backpropagation algorithm suffers from several practical disadvantages:

*Memory Overhead* In a naïve implementation, it creates a substantial memory overhead as the entire computational graph, including all parameters, activations, and gradients, needs to fit in a processing unit's working memory. Currently known solutions to address this require recomputation of intermediate outputs (Salimans and Bulatov, 2017) or expensive reversible layers (Jacobsen et al., 2018). This hinders the application of deep learning models to high-dimensional input data that surpasses current memory constraints.

*Synchronous Training* Optimizing a network using backpropagation results in several forms of locking (Jaderberg et al., 2017). On the one hand, it is forward locked, since every layer needs to wait for its predecessors to provide its inputs. On the other hand, it is backward locked, as no layer can update its parameters before its successors have provided it with the necessary gradients. Thus, end-to-end training with backpropagation does not allow for an exact way of asynchronously optimizing individual layers. This inhibits the efficiency of hardware accelerator design due to a lack of locality.

*Vanishing Gradients* When training a neural network with $L$ layers using backpropagation, the weight update of the lower layers is computed using the chain rule, effectively multiplying with the gradients of higher layer's activation functions up to $L$ times. Typically, these gradients range between zero and one, which can thus have the effect of exponentially decreasing the size of the gradients with which to update the lower layers. The resulting vanishing gradients can then fail to update the weights of the lower layers appropriately, impeding overall network performance. There are several possible solutions to this problem, such as layer-wise pretraining (Bengio et al., 2007), the usage of ReLU nonlinearities (Krizhevsky et al., 2012), residual networks (He et al., 2016a) and LSTMs (Hochreiter and Schmidhuber, 1997) for recurrent networks.

### 2.2.3 *Biologically Implausible Aspects of Backpropagation*

In addition to these computational disadvantages, backpropagation has several aspects that reduce its biological plausibility:

*Local Error Representation* Generally, biological synapses are adjusted based on local signals, i.e. based on signals from the neurons that they are immediately connected to (Caporale and Dan, 2008). This is in stark contrast to the weight updates in backpropagation which depend on gradient information originating from a global error calculated in the final layer (see Equation (2.23)). Despite some evidence for top-down connections in the brain, there does not appear to be a comparable global objective that is optimized by backpropagating error signals in such a way (Crick, 1989). Early theories proposed that the brain might nonetheless optimize a global error that is signaled to all neurons using neuromodulators (Mazzoni et al., 1991; Williams, 1992). However, such an approach is inherently slow and does not scale with the size of the network (Werfel et al., 2004). More recent work has shown that backpropagation may be approximated in more biologically plausible ways by computing weight updates based on local information (Whittington and Bogacz, 2017; Scellier and Bengio, 2017). Nevertheless, these approaches still rely on supervised learning and thus require the representation of a global target value. Even if the brain were to receive a target value for each of its inputs, it remains an unclear how this value would be implemented.

*Weight Symmetry* Backpropagation works by first forward-propagating a given input through a network, comparing the produced output to the target value, and updating the weights by backpropagating the gradients of the loss. In this process, the same weights are used twice: once in the forward pass when they are applied to the input and once in the backward pass to scale the backpropagating gradients according to the connection strengths. The weights thus need to be symmetrical. However, such bidirectional connections are only sparingly present in the brain (Song et al., 2005). This leaves the question of how the brain would be able to implement backpropagation if it cannot appropriately propagate the errors back to the neurons that caused them. Lillicrap and Santoro (2019); Xiao et al. (2019) proposed a possible solution by showing that backpropagation can still work when the backward-pass makes use of random weights. Nonetheless, a related problem remains, namely that backpropagation in the brain would require a distinct mechanism for propagating gradients, which allows it to backpropagate its errors without influencing the ongoing neural activity.

*Other Aspects* In addition to the two aforementioned critique points on the biological plausibility of backpropagation, there are several aspects regarding the modeling and optimization process of ANNs themselves that do not correspond to the biological brain. First of all, biological neurons communicate using binary values, better known as spikes, while artificial neurons typically send continuous outputs. Secondly, most ANNs make use of supervised training, i.e. they are optimized on datasets in which every input has a corresponding target value. In addition to that, the number of required input-label pairs is often enormous. In contrast, humans often learn new concepts using only a few samples (Gopnik et al., 1999).

Another critical point to note is that the ANNs considered so far work on non-temporal inputs and targets. The brain, on the other side, receives a constant stream of inputs and thus needs to work on temporal sequences. There are neural networks that take sequential inputs and model temporal dependencies, named Recurrent Neural Networks (RNNs). They are usually trained using Backpropagation Through Time for which there is even less evidence that a similar process may be implemented in the brain. We will provide a short discussion on this algorithm in the next section.

### 2.2.4 *Additional Notes on Backpropagation Through Time*

Backpropagation Through Time (BPTT) works by unfolding a neural network over time while keeping the parameters constant. Then, the weights are updated by applying backpropagation on the unfolded network. As a result, it exhibits the same biologically implausible aspects as backpropagation itself. On top of that, by unfolding the network over time for every sequence to be learned, it adds two major biologically implausible aspects.

In a standard ANN, activation values need to be stored during a single forward/backward pass, i.e. each neuron needs to save its own state until all weights are updated. When using BPTT however, each neuron needs to store its activations from all points in the past over which the network is unfolded, i.e. each neuron needs to store its own history of states (Lillicrap and Santoro, 2019).

The second criticism is that BPTT is not suitable for online learning. Instead, it requires a network to forward-pass the entire input sequence, unfold the resulting network and back-propagate errors over the unfolded network, before weights can be updated and before a new sequence can be processed. While there is some evidence for memory replay in the brain (Wilson and McNaughton, 1994; Foster and Wilson, 2006), it mainly learns online.

## 2.3 *Neuroscientific Background and Inspiration*

Given these biologically implausible aspects of backpropagation, the brain likely adjusts its synaptic connections using different mechanisms. In this section, we will provide a review of known learning processes in the brain, focusing specifically on theories connected to MI and layer-wise learning based on prediction errors.

We start our review at the most fine-grained level of learning: the adjustment of individual synapses. Hebb (1949) proposed an early theory for synaptic learning that is still influential today. He stated that "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." This synaptic learning rule is known as Hebbian Learning and commonly abbreviated as "neurons that fire together, wire together". Many studies provide strong experimental support for this theory, finding synaptic changes reflecting this learning rule in diverse neural circuits, such as in the hippocampus (Bliss and Lømo, 1973) and various neocortical areas (Artola and Singer, 1993; Iriki et al., 1989; Hirsch et al., 1992).

In addition to Hebbian learning which focuses on the correlation between pre- and post-synaptic spikes, several studies pointed out the importance of the temporal order in which these spikes occur (Levy and Steward, 1983; Gustafsson et al., 1987; Debanne et al., 1994). This temporal dependency is captured in the synaptic learning theory of Spike-Time Dependent Plasticity (STDP) (Caporale and Dan, 2008). It postulates that pre-synaptic activity that precedes post-synaptic firing can induce a persistent strengthening of the involved synapse while reversing the temporal order can cause the synapse to weaken (see Figure 2.2). Many experimental studies support this theory and also delineate the critical time window to be within tens of milliseconds (Bi and Poo, 2001; Debanne et al., 1998; Magee and Johnston, 1997).

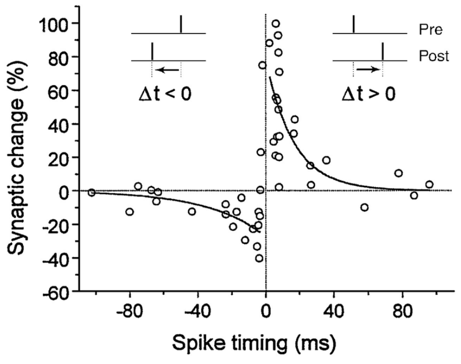


Figure 2.2: Temporal dependency of synaptic weight changes reflecting the theory of Spike-Time Dependent Plasticity (STDP). When the post-synaptic spike occurs after the pre-synaptic one ($\Delta t > 0$), the synaptic connection is strengthened. When the spiking order is reversed ($\Delta t < 0$), connection strengths are weakened. Data points reflect the behavior of hippocampal cells and were taken from Bi and Poo (1998), the figure is taken from Asl (2018).

While these theories governing the synaptic plasticity in the brain are widely accepted, there is a lot more debate about the more abstract learning concepts that they implement. One line of work tries to connect them with information theory. An early step in this direction was taken by Linsker (1988), who developed the InfoMax principle. It theorizes that the brain learns to process its perceptions by maximally preserving the information of the input activities in each layer. According to his theory, a Hebb-type rule may induce this self-organizing behavior. Linsker also showed that his approach can successfully model the emergence of oriented and center-surround receptive fields in the primary visual cortex of mammals. Further studies showed that MI maximization can also model other early stages of perception, such as the gain control in the blowfly's eye (Laughlin, 1987), and properties of the mammalian retina's receptive fields (Atick and Redlich, 1990). More recently, Toyoizumi et al. (2005) developed a learning rule that maximizes the MI between the input and the output of a layer based on the rules of Spike-Time Dependent Plasticity (STDP).

Overall, neurons learn primarily based on local information. Some theories suggest that this local information is generated as the brain tries to predict its future inputs. The resulting prediction error could then be used for learning (Friston, 2010). Rao and Ballard (1999) indicate that this process may happen at each layer within the brain. Their theory named Predictive Coding describes how each level in a hierarchical network attempts to predict the activity of the next lower level. This prediction is subtracted from the bottom-up sensory representation, which results in an error that is sent back to the higher level. There, the error is used to correct the estimate of future bottom-up sensory signals. Rao and Ballard (1999) showed that a three-level hierarchical network constructed according to this theory and trained on several thousand natural image patches could elicit some extra-classical receptive field effects with similar properties to those found in the visual cortex.

Our proposal draws motivation from these theories, resulting in a method that learns to preserve the information of each layer's input by creating representations that are predictive of future inputs.

# 3 Greedy InfoMax

In this thesis, we pose the question if we can effectively optimize the Mutual Information (MI) between representations at each layer of a model in isolation, enjoying the many practical benefits that greedy training (decoupled, isolated training of parts of a model) provides. In doing so, we introduce a novel approach for self-supervised representation learning: Greedy InfoMax (GIM).

As depicted on the left side of Figure 3.1, we take a conventional deep learning architecture and divide it by depth into a stack of $M$ modules. This decoupling can happen at the individual layer level or, for example, at the level of blocks found in residual networks (He et al., 2016b). Rather than training this model end-to-end, we prevent gradients from flowing between modules and employ a local self-supervised loss instead.

As shown on the right side of Figure 3.1, each encoding module $g_{enc}^m$ within our architecture maps the output from the previous module $z_t^{m-1}$ to an encoding

$$z_t^m = g_{enc}^m(\text{GradientBlock}(z_t^{m-1})) \ . \tag{3.1}$$

In order to restrict modules to train locally, gradient flow between modules is inhibited. This is formulated using the gradient blocking operator defined as:

$$\text{GradientBlock}(x) \triangleq x \tag{3.2}$$

$$\nabla \, \text{GradientBlock}(x) \triangleq 0 \ . \tag{3.3}$$

Oord et al. (2018) propose to use the output of an autoregressive model $g_{ar}(z_{0:t}) = c_t$ to contrast against future predictions $z_{t+k}$. However, our preliminary results showed that this did not improve results if applied at every module in the stack. Also, optimizing it requires Backpropagation Through Time (BPTT), which is considered biologically implausible. Therefore, we train each module $g_{enc}^m$ using the following module-local InfoNCE loss:

$$f_k^m(z_{t+k}^m, z_t^m) = \exp\left( {z_{t+k}^m}^T W_k^m z_t^m \right) \tag{3.4}$$

$$\mathcal{L}_N^m = -\sum_k \mathop{\mathbb{E}}_X \left[ \log \frac{f_k^m(z_{t+k}^m, z_t^m)}{\sum_{z_j^m \in X} f_k^m(z_j^m, z_t^m)} \right] \ . \tag{3.5}$$

After convergence of all modules, the scoring functions $f_k^m(\cdot)$ can be discarded, leaving a conventional feed-forward neural network architecture that extracts features $z_t^M$ for downstream tasks:

$$z_t^M = g_{enc}^M \left( g_{enc}^{M-1} \left( \cdots g_{enc}^1 (x_t) \right) \right) \ . \tag{3.6}$$

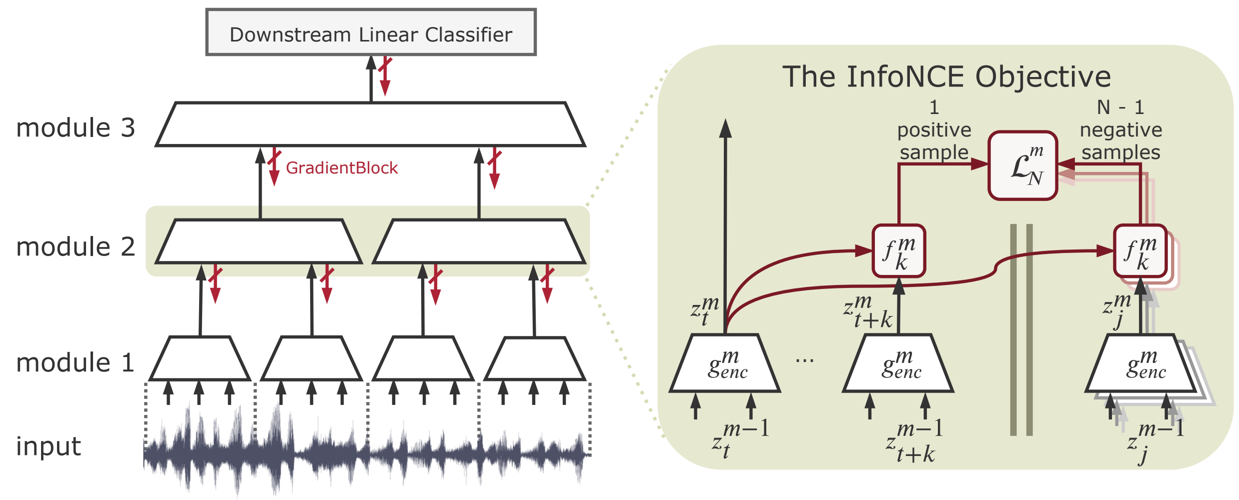


Figure 3.1: The Greedy InfoMax Learning Approach. **(Left)** For the self-supervised learning of representations, we stack a number of modules through which the input is forward-propagated in the usual way, but gradients do not propagate backward. Instead, every module is trained greedily using a local loss. **(Right)** Every encoding module maps its inputs $z_t^{m-1}$ at time-step $t$ to $g_{enc}^m(\text{GradientBlock}(z_t^{m-1})) = z_t^m$, which is used as the input for the following module. The InfoNCE objective is used for its greedy optimization. This loss is calculated by contrasting the predictions of a module for its future representations $z_{t+k}^m$ against negative samples $z_j^m$, which enforces each module to maximally preserve the information of its inputs. We optionally employ an additional autoregressive module $g_{ar}$, which is not depicted here.

For some downstream tasks, a broad context is essential. For example, in speech recognition, the receptive field of $z_t^M$ might not carry the full information required to distinguish phonetic structures. To provide this context, we reintroduce the autoregressive model $g_{ar}$ as an independent module that we optionally append to the stack of encoding modules, resulting in a context-aggregate representation $c_t^M$:

$$c_t^M = g_{ar}^M \left( \text{GradientBlock} \left( z_{0:t}^{M-1} \right) \right) . \tag{3.7}$$

We train this autoregressive module independently using the module-local InfoNCE loss with the following adjusted scoring function:

$$f_k^M(z_{t+k}^{M-1}, c_t^M) = \exp \left( \text{GradientBlock} \left( z_{t+k}^{M-1} \right)^T W_k^M c_t^M \right) . \tag{3.8}$$

## 3.1 *Mutual Information Maximization*

Similarly to the InfoNCE loss in Equation (2.16), our module-local InfoNCE loss in Equation (3.5) maximizes a lower bound on the MI $I(z_{t+k}^m, z_t^m)$ between nearby patch representations, encouraging the extraction of slow features.

Additionally, it follows from Oord et al. (2018), that the module-local InfoNCE loss also maximizes the lower bound of the MI $I(z_{t+k}^{m-1}, z_t^m)$ between the future input to a module and its current representation. This can be seen as a maximization of the MI between the input and the output of a module, subject to the constraint of temporal disparity. Thus, the InfoNCE loss can successfully enforce each module to preserve the information of its inputs, while providing the necessary regularization (Krause et al., 2010; Hu et al., 2017) for circumventing degenerate solutions. These factors contribute to ensuring that the greedily optimized modules provide meaningful inputs to their successors and that the network as a whole provides useful features for downstream tasks without the use of a global error signal.

## 3.2 Practical benefits

Training a neural network greedily with module-local losses and without end-to-end backpropagation gives rise to several practical benefits. Most importantly, when applying GIM to high-dimensional inputs, we can optimize each module in sequence to decrease the memory costs during training. In the most memory-constrained scenario, individual modules can be trained, frozen, and their outputs stored as a dataset for the next module to train on. This effectively removes the depth of the network as a factor of the memory complexity.

Additionally, GIM provides a highly flexible framework for the training of neural networks. It enables the training of individual parts of an architecture at varying update frequencies. When a higher level of abstraction is needed, GIM allows for adding new modules on top at any moment of the optimization process without having to fine-tune previous results.

Another benefit to note is that GIM can circumvent a potential cause of vanishing gradients. In an end-to-end optimized neural network with $L$ layers, the gradients of the lower layers are computed by application of the chain rule, effectively multiplying with the gradients of higher layer's activation functions up to $L$ times. With typical values of these gradients ranging from zero to one, this has the effect of exponentially decreasing the size of the gradients to the lower layers by a factor of $L$. When training the network greedily with GIM, this factor is reduced significantly, as we do not backpropagate through the entire stack of network layers. Instead, gradients are only propagated through the layers that make up one individually trainable subpart (i.e. a module) of the network. As a result, GIM provides a natural way to reduce the risk of vanishing gradients.

Last but not least, GIM allows for training models on larger-than-memory input data with architectures that would otherwise exceed memory limitations. Leveraging the conventional pooling and strided layers found in common network architectures, we can start with small patches of the input, greedily train the first module, extract the now compressed representation spanning larger windows of the input and train the following module using these.

# 4 Experiments

We test the applicability of the proposed Greedy InfoMax (GIM) approach to the visual and audio domain. In both settings, a feature-extraction model is divided by depth into modules and trained without labels using GIM. The representations created by the final (frozen) module are then used as the input for a linear classifier. Its accuracy scores provide us with a proxy for the quality and generalizability of the representations created by the self-supervised model.

## 4.1 Vision

To apply Greedy InfoMax to natural images, we impose a top-down ordering on 2D images. We follow Oord et al. (2018); Hénaff et al. (2019) by extracting a grid of partly-overlapping patches from the image to restrict the receptive fields of the representations (Figure 4.1). First, we encode each patch $x_{i,j}$ in row $i$ and column $j$ of this grid into its corresponding representation $z_{i,j}$. Then, we predict up to K encoded patches $z_{i+K,j}$ in the rows underneath, skipping the first overlapping patch $z_{i+1,j}$. Random contrastive samples are drawn with replacement from all samples available inside a batch, using 16 contrastive samples for each evaluation of the loss. No autoregressive model $g_{ar}$ is used for GIM in this regime.

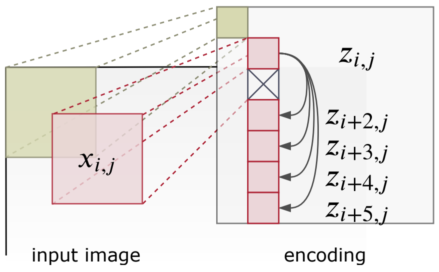


Figure 4.1: We impose a top-down ordering on images by predicting the encodings of patches in subsequent rows of the same column.

### 4.1.1 Experimental Details

We focus on the STL-10 dataset (Coates et al., 2011), an image recognition dataset for developing unsupervised feature learning algorithms. It consists of 100,000 unlabeled images, which we use for the self-supervised training of our GIM model. The remaining 5,000 labeled training images and 800 testing images containing ten classes are employed for examining the usefulness of our learned representations for image classification tasks.

For data augmentation, we take random $64 \times 64$ crops from the $96 \times 96$ images, flip them horizontally with probability 0.5 and convert them to grayscale. We divide each image of $64 \times 64$ pixels into a total of $7 \times 7$ local patches, each of size $16 \times 16$ with 8 pixels overlap. The patches are encoded by a ResNet-50 v2 model (He et al., 2016b) without batch normalization (Ioffe and Szegedy, 2015) which we split into three gradient-isolated modules. In our main experiment, we train these modules in sync and with a coherent learning rate. After convergence, a linear classifier is trained – without finetuning the representations – using a conventional softmax activation and cross-entropy loss. This linear classifier accepts the patch representations $z_{i,j}^M$ from the final module and first average-pools these, resulting in

| **Method** | **Accuracy** |
|---|---|
| Deep InfoMax (Hjelm et al., 2019) | 78.2% |
| Predsim (Nøkland and Eidnes, 2019) | 80.8% |
| Randomly initialized | 27.0% |
| Supervised | 71.4% |
| Greedy Supervised | 65.2% |
| CPC | **82.1**% |
| Greedy InfoMax (GIM) | **82.0**% |

Table 4.1: STL-10 classification results on the test set. The GIM model performs virtually the same as the CPC model, despite a lack of end-to-end backpropagation and without the use of a global objective.

a single vector representation $z^M$. Remaining implementation details are presented in Appendix B.1.

### 4.1.2 *Results*

As shown in Table 4.1, *GIM* performs as well as the end-to-end trained *CPC* counterpart, despite its unsupervised features being optimized greedily without any backpropagation between modules. An equivalent *randomly initialized* feature extraction model exhibits poor performance, showing that GIM extracts useful features. Training the feature extraction model end-to-end and fully *supervised* performs worse, likely due to the small size of the annotated dataset resulting in overfitting. Although regularization techniques (DeVries and Taylor, 2017) may be able to circumvent this, the self-supervised methods do not appear to require regularization as they benefit from the full unlabeled dataset. Using a *greedy supervised* approach for training the feature model impedes performance. Here, we train modules separately as done with the GIM model, but use a supervised loss function instead. This result suggests that MI maximization is unique in its direct applicability to greedy optimization.

In comparison with the recently proposed *Deep InfoMax* model from Hjelm et al. (2019) the InfoNCE-based methods come out favorably. Deep InfoMax uses a slightly different end-to-end MI maximization approach, AlexNet (Krizhevsky et al., 2012) as their feature-extraction model, and an additional hidden layer in the supervised classification model. Finally, we see that we outperform the state-of-the-art biologically inspired *Predsim* model from Nøkland and Eidnes (2019). This approach trains individual layers of a VGG like architecture (Simonyan and Zisserman, 2014) using two supervised loss functions.

### 4.1.3 *Iterative Training*

GIM provides a significant advantage arising from the greedy nature of optimization: it can effectively remove the depth of the network as a factor of the memory complexity. Every module is trained with its own local loss function that depends solely on this particular module's inputs and outputs. This greedy optimization enables us to decouple the training of the individual parts of the network. In the most memory-restricted

| Method | GPU memory |
|---|---|
| Supervised | 6.3 GB |
| Greedy Supervised - 1st module | **2.5** GB |
| CPC | 7.7 GB |
| GIM - 1st module | **2.5** GB |
| GIM - all modules | 7.0 GB |

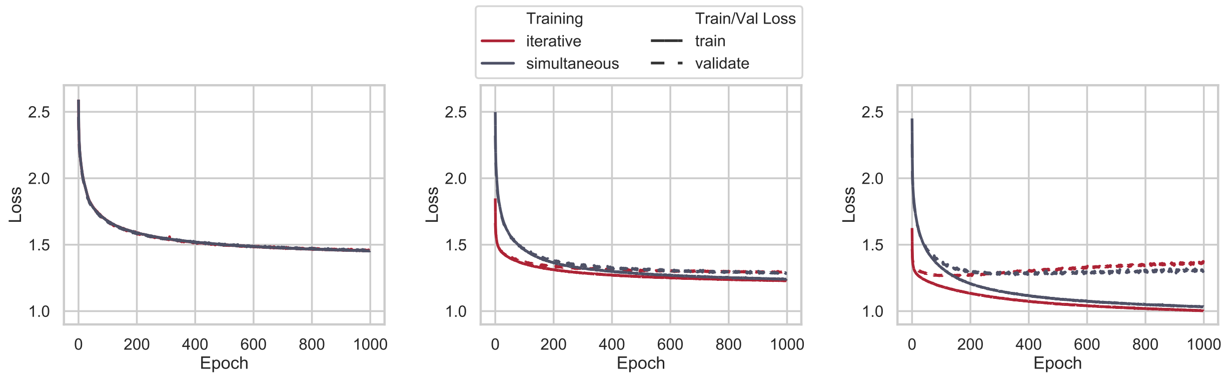
Table 4.2: GPU memory consumption during training. All compared models consist of the ResNet-50 v2 architecture and only differ in their respective training approach. GIM allows for memory-efficient greedy training.

setting, we can train one module after another, saving the previous module's output as a dataset for the next module to train on. As a result, we can train the entire network while only having to fit one individual module with its corresponding inputs, outputs, and gradients into the GPU memory at a time.

Measuring the allocated GPU memory of the previously studied models during training (Table 4.2), indicates that this theoretical benefit holds in practice as well. After splitting the architecture into three separately trainable modules, we can reduce the GPU memory consumption by a factor of 2.8 by training the modules individually (*GIM - 1st module*) compared to training them simultaneously (*GIM - all modules*). On top of this benefit from training modules individually, training *all modules* simultaneously with GIM reduces the memory footprint by approximately 10% in comparison to *CPC* as it does not employ an autoregressive PixelCNN (Van den Oord et al., 2016) model on top of the encoding ResNet architecture.

Now, the question remains whether asynchronous training influences the quality of the representations created by GIM. In order to test this, we train a second GIM model iteratively. In this setting, we train the lowest module until convergence and fix its parameters before training the next module on top of it and repeat this process until all modules have been trained. After training each module for 300 epochs, this *iteratively* trained model achieves an accuracy of 79.8% on the image classification downstream task. Thus, the performance declines slightly in comparison to the *simultaneously* trained model, as previously shown in Table 4.1 with 82.0% accuracy.

In order to examine potential causes for this difference in performance, we plot the training curves of the two models in Figure 4.2. As expected, the learning curves of the first module (Figure 4.2a) reflect that there is no difference in its training in the two models. Modules two and three (Figures 4.2b and 4.2c), however, reveal two differences between the iteratively and the simultaneously trained model. First, the *iteratively* trained modules start with lower losses. This can be explained by them receiving clean inputs from their already converged predecessors from the beginning of their training. Second, the *iteratively* trained modules show a larger divergence between the training and validation loss, indicating stronger overfitting. We can thus conclude that the noisier inputs that the higher



(a) First Module (b) Second Module (c) Third Module

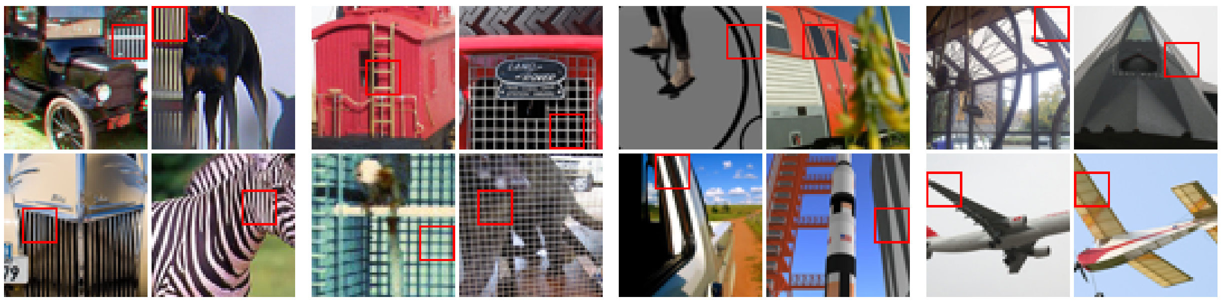
Figure 4.2: Training curves for optimizing all modules *simultaneously* (blue) or *iteratively*, one at a time (red). While there is no difference in the training methods for the first module (**a**), later modules (**b, c**) start out with a lower loss and tend to overfit more when trained iteratively on top of already converged modules.

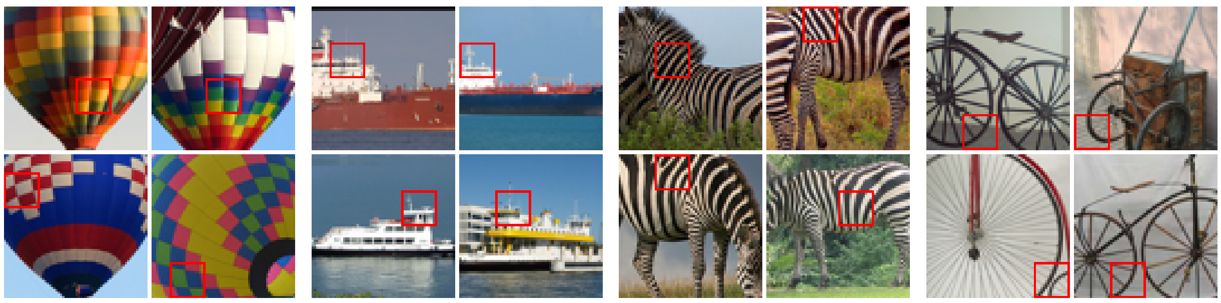
modules in the *simultaneously* trained model receive from their untrained predecessors provide some kind of regularization for their training.

Interestingly, the validation losses do not diverge at epoch 300 yet, even though we found a difference in downstream performance at that point of training (79.8% vs. 82.0% accuracy). Additionally, even though training and validation losses deviate more strongly after 1000 epochs, the difference in performance on the downstream task stays relatively constant with 80.6% accuracy for the *iteratively* trained model and 82.7% accuracy for the *simultaneously* trained model. These results illustrate that the loss achieved during the training with GIM cannot directly indicate the downstream performance of the created representations.

### 4.1.4 *Visualization of Feature Abstraction*

In Figure 4.3, we visualize patches that individual neurons in the three modules of the GIM model are sensitive to. While in the first module neurons show a clear preference for edges of specific orientations, they focus on more complex patterns in the second module. The third module shows the highest level of abstraction. Its neurons are highly selective for class-specific patterns that are rather agnostic of the specific coloring or orientation. These visualizations illustrate that individual modules learn to iteratively increase the level of abstraction in their representations even though the GIM approach optimizes each module greedily without end-to-end backpropagation.


(a) First Module


(b) Second Module

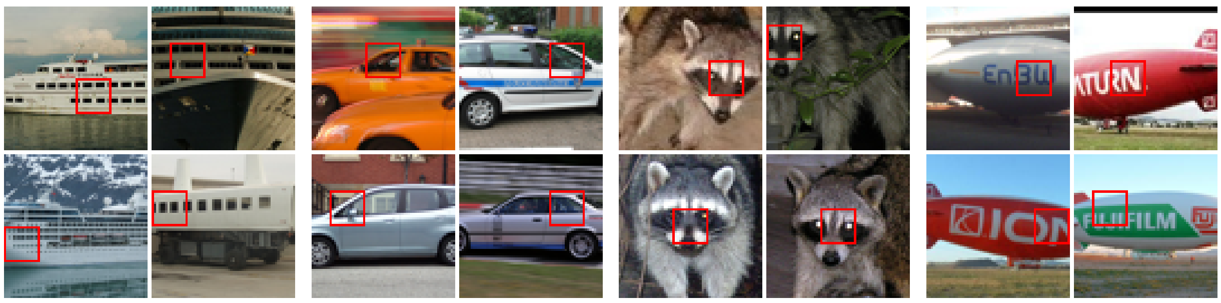

(c) Third Module

Figure 4.3: Groups of four image patches that activate a certain neuron, at three individually trained levels of the module stack. All modules are trained greedily without labels.

| Method | Phone Classification Accuracy | Speaker Classification Accuracy |
|---|---|---|
| Randomly initialized | 27.6% | 1.9% |
| MFCC features | 39.7% | 17.6% |
| Supervised | 77.7% | 98.9% |
| Greedy Supervised | 73.4% | 98.7% |
| CPC (Oord et al., 2018)* | 64.9% | 99.6% |
| Greedy InfoMax (GIM) | 62.5% | 99.4% |

Table 4.3: Results for classifying speaker identity and phone labels in the LibriSpeech dataset. All models use the same audio input sizes and the same architecture. GIM creates representations that are useful for audio classification tasks despite its greedy training and lack of a global objective.

*In the original implementation, Oord et al. (2018) achieved 64.6% for the phone and 97.4% for the speaker classification task.

## 4.2 Audio

We evaluate GIM in the audio domain on the sequence-global task of *speaker* classification and the local task of *phone* classification (distinct phonetic sounds that make up pronunciations of words). These two tasks are interesting for self-supervised representation learning as the former requires representations that discriminate speakers but are invariant to content, while the latter requires the opposite. Strong performance on both tasks thus suggests strong generalization and disentanglement.

### 4.2.1 Experimental Details

We follow the setup of Oord et al. (2018) unless specified otherwise and use a 100-hour subset of the publicly available LibriSpeech dataset (Panayotov et al., 2015). It contains the utterances of 251 different speakers with aligned phone labels divided into 41 classes. These phone labels were provided by Oord et al. (2018) who obtained them by force-aligning phone sequences using the Kaldi toolkit (Povey et al., 2011) and pre-trained models on Librispeech (Panayotov, 2014).

We first train the self-supervised model consisting of five convolutional layers and an autoregressive, single-layer Gated Recurrent Unit (GRU). For the training of GIM, we split this architecture into six separately trained modules, five encoding, and one autoregressive. After convergence, a linear multi-class classifier is trained on top of the context-aggregate representation $c^M$ without fine-tuning the representations. Remaining implementation details are presented in Appendix B.2.

### 4.2.2 Results

Following Table 4.3, we analyze the performance of models on phone and speaker classification accuracy. *Randomly initialized* features perform poorly, demonstrating that both tasks require complex representations. The traditional, hand-engineered *MFCC features* are commonly used in speech recognition systems (Ganchev et al., 2005), and improve over the random features but provide limited linear separability on both tasks. On the speaker classification task, *CPC* and *GIM* outperform the *supervised* baselines despite their feature models having been trained without labels, and GIM without end-to-end back-

| Method | Accuracy |
|---|---|
| **Speaker Classification** | |
| Greedy InfoMax (GIM) | 99.4% |
| GIM without BPTT | 99.2% |
| GIM without $g_{ar}$ | 99.1% |
| **Phone Classification** | |
| Greedy InfoMax (GIM) | 62.5% |
| GIM without BPTT | 55.5% |
| GIM without $g_{ar}$ | 50.8% |

Table 4.4: Ablation studies on the LibriSpeech dataset for improving the biological plausibility of GIM as well as its memory efficiency by restricting the flow of gradients in the autoregressive module or by removing it altogether.

propagation. In this setting, both *GIM* and *Greedy Supervised*, where individual layers are trained greedily similar to GIM but with a supervised loss function, achieve similar results to their respective end-to-end trained counterparts (*CPC* and *Supervised*). When classifying phones, *CPC* does not reach the supervised performance (64.9% versus 77.7%). *GIM* achieves 62.5%, while *Greedy Supervised* accomplishes 73.4%. Thus, in contrast to the vision experiments (Section 4.1), we see similar differences in performance between the greedily trained models (*GIM* and *Greedy Supervised*) when compared to their respective end-to-end optimized counterparts (*CPC* and *Supervised*).

Overall, the discrepancy between better-than-supervised performance on the speaker task and less-than-optimal performance on the phone task suggests that the features extracted by GIM and CPC are biased towards sequence-global tasks.

### 4.2.3 *Ablation Studies*

The local greedy training enabled by GIM provides a step towards biologically plausible optimization and improves memory efficiency. However, the last autoregressive module aggregates its inputs over multiple patches employing Backpropagation Through Time (BPTT), which puts a damper on both benefits. We conduct two ablation studies in which we test whether we can restrict the flow of gradients through time. By doing so, we increase the (temporal) locality of the GIM training and thus its biological plausibility and memory efficiency.

In order to limit the flow of gradients through time, we modify the autoregressive module. In general, the autoregressive module $g_{ar}$ takes into account the current input $z_t$, as well as the hidden state of the previous time-step $h_{t-1}$, in order to produce its output $c_t$, i.e. $c_t = g_{ar}(z_t, h_{t-1})$ (omitting the module-index $m$ here for brevity). In the standard GIM model, we block the flow of gradients to the previous module, such that $c_t = g_{ar}(\text{GradientBlock}(z_t), h_{t-1})$. Thus, gradients are still allowed to flow through time and can influence the calculation of the hidden states in previous time-steps. Now, we restrict this gradient flow in two different ways.

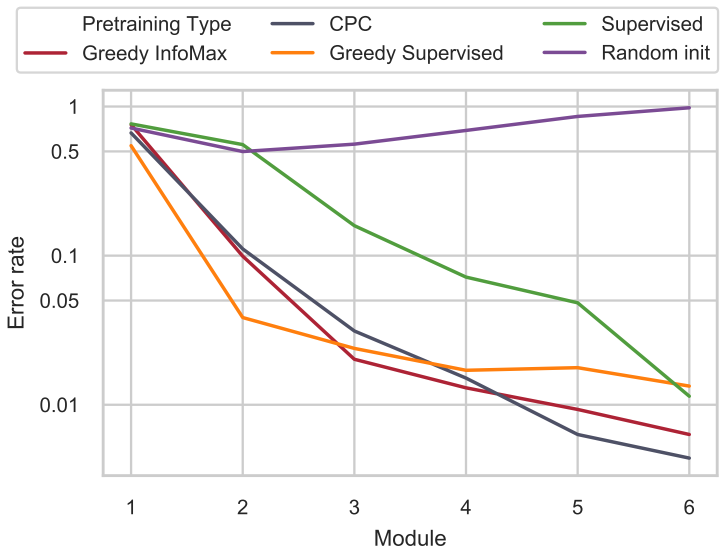


Figure 4.4: Speaker Classification error rates on log scale (lower is better) for intermediate representations (convolutional modules 1 to 5), as well as for the final representation created by the autoregressive layer (module 6 – corresponding to the results in Table 4.3). For comparison: A linear classifier trained on top of the raw audio data achieves an error rate of 0.98.

In the ablation *GIM without BPTT*, we remove BPTT by blocking the flow of gradients between time-steps, such that $c_t = g_{ar}(\text{GradientBlock}(z_t), \text{GradientBlock}(h_{t-1}))$. Thus, the gradients derived from the loss at time-step $t$ do not influence the calculation of the hidden state of the previous time-step $h_{t-1}$. For the ablation *GIM without* $g_{ar}$, we remove the autoregressive module entirely. Here, the linear classifier is applied to the representation created by the last convolutional module (i.e. $z_t$).

In Table 4.4, we present the performance of the ablated models. Together, these two ablations indicate a crucial difference between the tested downstream tasks. For the phone classification task, we see a steady decline of the performance when we reduce the modeling of temporal dependencies, indicating their importance for solving this task. When classifying the speaker identity, on the other hand, reducing the modeling of temporal dependencies in the ablated models barely influences their performance.

Together with the image classification results from Section 4.1, where no autoregressive module was employed either, this indicates that the GIM approach performs best on downstream tasks where temporal or context dependencies do not need to be modeled by an autoregressive module. In these settings, GIMs performance is on par with the CPC model, which makes use of end-to-end backpropagation, a global objective, and BPTT.

### 4.2.4 *Intermediate Module Representations*

The greedy layer-wise training of GIM allows us to train arbitrarily deep models without ever running into a memory constraint. We investigate how the created representations develop in each individual module by training a linear classifier on top of each module and measuring their performance on the speaker identification task. With results presented in Figure 4.4, we observe that each GIM module improves upon the representation of their predecessor. Interestingly, CPC exhibits equivalent performance throughout all intermediate modules despite these modules relying solely on the error signal from the global loss function on the last module. This is in stark contrast with the supervised end-to-end model, whose intermediate layers perform significantly worse than their greedily trained counterparts. This suggests that, in contrast to the supervised loss, the InfoMax principle “stacks well”, such that the greedy, iterative application of the InfoNCE loss performs similar to its global application.

# 5 *Related Work*

We have studied the effectiveness of the self-supervised CPC approach (Oord et al., 2018; Hénaff et al., 2019) when applied to gradient-isolated modules, freeing the method from end-to-end backpropagation. In this chapter, we will provide an overview of existing, biologically plausible learning algorithms for ANNs, as well as representation learning approaches that make use of similar concepts as our proposal, i.e. MI maximization and context prediction.

## 5.1 *Biologically Plausible Learning Algorithms*

It is implausible that the biological brain uses a mechanism similar to backpropagation for the adaptation of synapses. As argued in Section 2.2.3, this is mostly due to two aspects of the algorithm: the necessity for symmetric, reciprocal weights and the representation and optimization of a global objective function. Correspondingly, there are two lines of work that try to develop more biologically plausible learning rules. The first attempts to optimize a global loss by developing alternative ways to assign credit to individual neurons across the network. The second seeks to remove the necessity for a global objective function by providing more local error representations.

### 5.1.1 *Alternative Solutions to the Credit Assignment Problem*

Backpropagation is highly effective in solving the credit assignment problem: given the error in the final output, which connections in the network contributed to this error, and how should they be changed to improve performance? It does so by backpropagating the gradients of the loss to all layers in the network, which requires the weights to be symmetrical (see Section 2.2.1). However, such symmetrical weights are rarely found in the brain. This finding motivates a line of research that tries to optimize a global loss by assigning credit to the connections in the network in different ways.

LeCun (1986) proposed an alternative way for assigning credit in neural networks named Target Propagation. In this method, each hidden layer is updated using targets instead of gradients, eliminating the requirement for symmetric weights. Ideally, these targets should be designed in a way such that a network that realizes its targets concurrently minimizes its loss in the final layer. One possibility is to use (approximate) inverse transformations to backpropagate output activations which optimize the loss. Difference Target Propagation (Lee et al., 2015) develops this idea further and adds a layer-wise reconstruction error to these targets, which can stabilize imprecise inverses. While these approaches were shown to perform well on MNIST, they do not match the performance of backpropagation on more complex datasets such as CIFAR-10 and ImageNet (Bartunov et al., 2018). Ororbia et al. (2018) advanced the idea of Target Propagation further by choosing targets that are within the set of possible representations of the layer to be trained. This Local Representation Alignment technique allows for more efficient updates of the layer's parameters but has yet to be successfully applied to more complex tasks as well.

An alternative set of research attempts to find ways to backpropagate gradients without the need for symmetrical weights. In the method called Feedback Alignment (Lillicrap et al., 2016) and its variant Direct Feedback Alignment (Nøkland, 2016), backpropagation was shown to still work when using random weights in the backward-pass. Similarly to Target Propagation, this approach was shown to not scale to larger problems (Bartunov et al., 2018). However, its performance can be improved tremendously by choosing the feedback weights' magnitudes at random while keeping their signs fixed to the feedforward weights (Xiao et al., 2019). Surprisingly, such networks can attain the performance of networks trained with standard backpropagation on ImageNet and MS COCO. Another alternative to propagating gradients backward was proposed by Kohan et al. (2018). In their algorithm, Error-Forward Propagation, the output of the network is fed back into the input layer, effectively reusing the feedforward connections to deliver error feedback. This method does not only remove the necessity for symmetric weights but eliminates the architectural constraint in which backpropagation requires the existence of some backward connectivity pattern for every neuron. It requires a settling process for each step during the training, however, and it remains unclear whether it can be applied to large-scale problems.

Another approach that attempts to find a more biologically plausible way to assign credit to neurons is Equilibrium Propagation (Scellier and Bengio, 2017). It computes the gradients of the objective function similarly to backpropagation but propagates the errors implicitly using only local perturbations. It does so by dividing its training process into two phases. In the "negative" phase, the input is presented and an energy-based network settles to an energy-minimum. In the ensuing "positive" phase, the output units are perturbed towards the target and the network settles to a new energy-minimizing state. Synaptic updates bring the energy-minimizing state of the negative phase closer to that in the positive phase, thereby making the network a better predictor of the target. As both phases make use of the same neural computation, this method improves over the biological plausibility of backpropagation. Nonetheless, it still requires symmetric weights, which puts a damper on this benefit. Additionally, the practical applicability of this approach still needs to be demonstrated. So far, experiments were only conducted on small-scale datasets, and the negative phase generally takes a long time to settle to a fixed point representing the network's prediction.

Overall, the discussed methods employ a global supervised loss function and focus on finding more biologically plausible ways to assign credit to neurons. This is in contrast to our proposed algorithm, which makes use of backpropagation within individual modules, but optimizes only local, self-supervised losses.

### 5.1.2 *Local Error Representation*

The second aspect reducing the biological plausibility of backpropagation is that it typically optimizes a global objective function. First, a neural network propagates its input through all layers to produce the final output. Then, this output is compared to some target value, and the resulting error is used to adjust the connectivity strengths throughout the network (see Section 2.2.1). This is in stark contrast to biological synapses, which are adjusted predominantly based on local signals. This discrepancy has motivated a second line of research that aims to develop more biologically plausible learning algorithms by training with locally represented errors.

There exist several approaches that train layers greedily and thus locally, by optimizing specifically designed supervised loss functions. One such approach is Predsim which was developed by Nøkland and Eidnes (2019). Here, individual layers are trained using a combination of a supervised cross-entropy loss (which is similar to the greedy supervised baseline in our experiments) and a newly developed similarity matching loss. This similarity loss enforces representations to be clustered according to their class label. This can be vaguely interpreted as another way to extract coherent features (see Section 2.1.2), where in this case one tries to preserve the coherence between the input and its corresponding label. An alternative supervised loss function that can be applied to individual layers was developed by Elad et al. (2018). Following the information bottleneck theory, their loss function is designed to maximize the MI between the outputs of a layer and the targets while minimizing the MI between the inputs and outputs.

Balduzzi et al. (2015) propose a local learning algorithm in which each layer is optimized using truncated feedback. Essentially, they prune the backpropagation of gradients in such a way that each layer is updated solely based on the influence that it has on its immediate neighbor. Balduzzi et al.

thoroughly back up their method by a theoretical analysis of backpropagation, which allows them to show that their local training still follows the same error gradients. However, it is only applicable to networks with one-dimensional outputs.

Decoupled Neural Interfaces (Jaderberg et al., 2017) constitute an entirely different approach for representing errors locally. In this method, a global loss is optimized by using predicted synthetic gradients in place of the true backpropagated gradients.

These methods all enable a network to be trained locally, on a layer-wise level, and thus to enjoy the same asynchronous training benefits as our proposed method. However, in contrast to Greedy InfoMax, they rely on labeled data for the optimization of their parameters.

The most notable methods for unsupervised layer-wise training are deep belief networks (Hinton et al., 2006; Bengio et al., 2007). These networks essentially stack Restricted Boltzmann Machines, which are trained greedily using Contrastive Divergence. After this pretraining, the network is optimized globally using a supervised loss. Lee et al. (2009) showed that similarly constructed convolutional deep belief networks applied on the TIMIT dataset can achieve very good performance for multiple audio classification tasks.

## 5.2 *Representation Learning*

Our proposal makes use of two key concepts related to representation learning: MI maximization and context prediction. In the following, we will provide an overview of existing methods relying on these concepts.

### 5.2.1 *Information-Theoretic Approaches*

As argued in Section 2.1.2, information theory can provide a useful framework for representation learning. At the moment, the most promising approach in this setting is to maximize the Mutual Information (MI) between representations of coherent parts of the input, i.e. between parts that are predictive of one another. Recently, several papers successfully applied variants of this method to different domains.

The Contrastive Bidirectional Transformer (Sun et al., 2019), for example, achieved state-of-the-art performance on video captioning and action anticipation by learning representations that maximize the MI between a video and its aligned text. Instead of maximizing the MI between inputs from different modalities, Deep InfoMax (Hjelm et al., 2019) maximizes the MI between local and global representations of images. Bachman et al. (2019) extended this method by simultaneously maximizing the MI between several independently augmented copies of each image and multiple features scales. Using this approach, they hold the current state-of-the-art for unsupervised pretraining on ImageNet using standard linear evaluation of their features. Similarly, Tian et al. (2019) maximize the MI between different views, such as different color channels, of an image. They show that the performance generally scales with the number of employed views.

The approach that we follow in our proposal is Contrastive Predictive Coding (CPC) (Oord et al., 2018). Here, the MI between temporally nearby representations is maximized, which biases the model towards learning to represent slow features (Wiskott and Sejnowski, 2002). Hénaff et al. (2019) showed that this approach could achieve state-of-the-art semi-supervised performance on ImageNet with as little as 13 labeled instances per class. Poole et al. (2018) unite some of these recent works under a common framework. They highlight that the objective in CPC, InfoNCE, exhibits low variance at the cost of high bias and propose new lower bounds that allow for balancing this bias-variance trade-off.

Most recently, Tschannen et al. (2019) argue that the success of the presented methods might only be loosely coupled to their information-theoretic background. Instead, they show that the performance of these models depends on inductive biases given by the feature extractor architecture and on the parameterization of the MI estimators. Nonetheless, they acknowledge that the preservation of coherences in the created features plays a vital role in successful representation learning.

All methods considered so far achieve convincing results for features extraction, but they all rely on models that are optimized end-to-end using a global objective. In contrast to this, we show that we can employ a similar MI maximization approach to extract useful features using a stack of greedily optimized modules, i.e. using only local objectives and no end-to-end backpropagation.

An information-theoretic approach for greedy, layer-wise representation learning has, to the best of our knowledge, only been explored in the context of total correlation (Ver Steeg and Galstyan, 2015). By maximizing this measure related to MI, representations are enforced to be maximally informative about the data. In contrast to GIM, this approach has only been applied to small-scale problems, is restricted to discrete random variables and the number and cardinality of latent factors to be used in a representation needs to be specified beforehand.

### 5.2.2 *Context Prediction Methods*

Besides the information-theoretic approaches, several context prediction methods have been explored for unsupervised representation learning. A prominent example in language processing is Word2Vec (Mikolov et al., 2013), in which a model is trained to produce word embeddings by predicting words given their context. Likewise, Doersch et al. (2015) study such an approach for the visual domain.

Hyvarinen and Morioka (2016) propose to make use of the non-stationary structure in time series. Their Time-Contrastive Learning approach trains a feature extractor in such a way that it enables a linear classifier to discriminate between different segments in temporal data. Thus, the features extractor needs to represent the temporal structure of the data, in particular, the underlying distributions of different segments. Their approach provides the first identifiable result for nonlinear Independent Component Analysis (ICA). Similarly, graph neural networks use contrastive principles to learn unsupervised node embeddings based on their neighbors (Nickel et al., 2011; Perozzi et al., 2014; Nickel et al., 2015; Kipf and Welling, 2016; Veličković et al., 2018).

Inversely to the MI maximization approaches described above, Schmidhuber (1992) proposed a method where individual features are trained to minimize their predictability by other features. This forces them to extract independent factors that carry statistical information but entails the risk of neurons latching onto local independent noise sources in the input.

Similar to the information-theoretic approaches for representation learning, these methods focus on providing global objectives with which to update one coherent model with end-to-end backpropagation. In contrast to this, our proposed method makes use of a context prediction method (CPC) in order to train modules greedily with local losses.

# 6 *Discussion & Future Work*

We presented Greedy InfoMax (GIM), a novel self-supervised representation learning algorithm that employs a stack of modules trained greedily with local objectives. We demonstrated that each module improves upon the output of its predecessor, and that the top module can achieve competitive performance to the same architecture that is optimized using end-to-end backpropagation. In this chapter, we will provide a discussion on the biological plausibility of GIM and its computational ramifications, and elaborate on possible future work.

## 6.1 *Biological Plausibility*

Since humans' cognitive abilities exceed those of present-day computers in many fields, taking lessons from the brain's inner workings might help us to enhance the capabilities of algorithms. Following this idea, we developed GIM, which is inspired by the apparent gap between the standard way of optimizing neural networks, backpropagation, and neuroscientific evidence on how the brain learns.

We argue that our method is indeed more biologically plausible than backpropagation. First of all, we do not employ a global objective for the optimization of our network. Instead, connections in the network are updated based on local information. This resembles learning processes in the brain more closely, where synapses are primarily modulated by the activations of their pre- and post-synaptic neurons and thus by the neurons local to the synapse. Secondly, we do not employ labeled data but use a self-supervised loss instead. Similarly, the brain does not require annotated data to do feature learning. Thus, GIM may hint at how the brain can learn based on statistical regularities in the given data. Additionally, GIM is compatible with established neuroscientific theories as depicted in Section 2.3.

Although GIM provides a step forward, the local loss function that we employ is not entirely biologically plausible. The InfoNCE loss involves negative sampling: it contrasts the future input against random samples from all other possible inputs. It remains an open question whether the brain could implement such a sampling strategy. Alternatively, it would be interesting to investigate whether a more biologically plausible loss function for preserving the MI can achieve similar performance to the InfoNCE loss.

A second point to consider is that our proposal focuses on providing a more local way for optimizing neural networks, but we still employ backpropagation within the individual modules. Thus, we neglect the second aspect that reduces the biological plausibility of backpropagation – the necessity for symmetric weights. These symmetric connections are used in the backpropagation algorithm to propagate error feedback to all connections. This enables it to solve the credit assignment problem (Bengio et al., 2015): if there is an error in the final output, which connections should be changed to improve the performance? We argue that this problem might, in fact, not be relevant in networks that are optimized locally using GIM. Instead of requiring connections to improve on a global objective, we train them to create features that are informative about neighboring features in position or time. In our current approach, this still requires symmetric weights in order to propagate the feedback within a module from the local loss function to the respective connections to be updated. It remains an open question whether we could optimize individual layers towards a similar objective, but even more locally in order to remove this restriction. Alternatively, since more biologically plausible methods such as Direct Feedback Alignment or Error-Forward Propagation appear to work well for small scale architectures (see Section 5.1.1), we suspect that it is possible to use these instead of backpropagation for assigning credit within modules.

## 6.2 *Computational Considerations*

In this section, we will discuss the computational properties of our proposed self-supervised learning algorithm, GIM. This includes its memory and computational complexity, as well as some outlook on possible future work, e.g. how it might be extended for massively parallelized training.

GIM can reduce the GPU memory consumption for the training of a neural network considerably. Since it allows each module to be trained separately, the memory complexity can be reduced by the factor of the depth of the network as we have argued theoretically in Section 3.2 and shown empirically in our experiments.

At the same time, GIM increases the computational complexity in comparison to the end-to-end optimized CPC approach. When training with local objectives, we need to draw samples and calculate the loss in each module separately. On top of that, when training modules iteratively, the input data needs to be loaded for each module individually.

However, GIM allows for more efficient parallelization of the training process, which could help to counteract the increased computational complexity in practice. Since modules can be trained separately, the training process can be parallelized easily, e.g. by training each module on a separate GPU. This scheme could be implemented efficiently, as virtually no communication between modules and thus GPUs is required.[1]

[1] For the most efficient implementation, one would have to store the outputs of each module as a dataset to train the next module on. This data-saving and -loading cycle could run in parallel with the training of the modules such that it does not influence the computations on the GPUs.

Such parallelization would be most useful for the training of very deep neural networks that would otherwise be constrained by their memory or run-time requirements. So far, we conducted experiments on comparatively small models that were divided into maximally six modules. Further experiments need to determine whether GIM scales to deeper architectures that are divided into a larger number of separately trainable modules. If this was the case, GIM could provide an easy and efficient way for the massively parallelized training of neural networks.

While parallelization could provide a tremendous speed-up for the training of neural networks, GIM also allows for training very deep networks that would otherwise not fit into GPU memory. As argued before in Section 3.2, GIM's memory footprint can be reduced considerably by training modules iteratively, one after another. However, our experiments indicate that such asynchronous training can lead to impaired results (see Section 4.1.3). Future work needs to examine how to circumvent this problem. One potential solution could be to interleave the training of the individual modules. In our experiments, we used the most extreme setting in which one module was trained until convergence before the training of the next module started. Detaching to this extent is not necessary for reducing the memory footprint of GIM, however. As long as we restrict the training process to train one module at a time, we are free to switch around which module to update at each time-step. This should allow for achieving the same downstream performance as the simultaneously trained model while reaping the memory benefits of the iterative training. However, switching modules to be updated increases the amount of communication necessary as modules need to be interchanged between the GPU and memory.

A second possible solution to circumvent the decreased performance of the iterative training would be to tackle the overfitting that we observed in our experiments. They revealed that especially the higher modules suffer from this problem when they are trained on top of converged predecessors. This behavior might be caused by the different inputs that the higher modules receive during simultaneous vs. iterative training. When training all modules simultaneously, the lower modules provide noisier inputs to the higher modules as they are still adjusting their weights as well. These noisy inputs might regularize the training of the higher modules and reduce overfitting. Thus, it might help to adjust the training of the higher modules to be more similar to the simultaneous training case, e.g. by inducing noise on their inputs in the early stages of training. Alternatively, it might be possible to circumvent overfitting by applying other methods commonly applied for this problem, such as dropout (Srivastava et al., 2014) or batch normalization (Ioffe and Szegedy, 2015).

A different aspect that was revealed by our experiment is that the representations created by GIM (and CPC) are biased towards global features: they perform better on the global tasks of speaker and image classification. In the local task of phone classification, however, both approaches lag behind the supervised model. Oord et al. (2018) showed that different sampling strategies, e.g. drawing negative samples from sequences of the same speaker only, can influence the model's performance on the downstream task. It would be interesting to extend this approach for GIM. Its local training enables us, for instance, to apply different sampling strategies to different modules. Thus, we could potentially influence the amount of abstraction of the features in each module by choosing different sampling strategies, which in turn could lead to improved downstream performance.

Last but not least, it would be interesting to investigate whether other information-theoretic approaches for representation learning could be applied in a similar greedy training regime as GIM. After all, they all rely on a similar concept: they maximize the MI between coherent representations, which in turn maximizes the information preserved by the model. We argued that this information preserving property of the InfoNCE loss allowed us to apply it greedily to individual modules. Thus, it would be interesting to test this hypothesis by applying other MI maximization approaches, such as Deep InfoMax (Hjelm et al., 2019) or Contrastive Multiview Coding (Tian et al., 2019), to the training of individual modules.

# 7 Conclusion

We presented Greedy InfoMax, a novel self-supervised greedy learning approach. The relatively strong performance demonstrates that deep neural networks do not necessarily require end-to-end backpropagation of a supervised loss to learn useful features for perceptual tasks. Our proposal enables greedy self-supervised training, which makes the model less vulnerable to overfitting, reduces the vanishing gradient problem, and enables memory-efficient asynchronous distributed training. While the biological plausibility of our proposal is limited by the use of negative samples and within-module backpropagation, the results provide evidence that the theorized self-organization in biological perceptual networks is at least feasible and effective in artificial networks, providing food for thought on the credit assignment discussion in perceptual networks (Bengio et al., 2015; Linsker, 1988).

# References

Artola, A. and Singer, W. (1993). Long-term depression of excitatory synaptic transmission and its relationship to long-term potentiation. *Trends in neurosciences*, 16(11):480–487.

Asl, M. M. (2018). Propagation delays determine the effects of synaptic plasticity on the structure and dynamics of neuronal networks. *ResearchGate*.

Atick, J. J. and Redlich, A. N. (1990). Towards a theory of early visual processing. *Neural Computation*, 2(3):308–320.

Bachman, P., Hjelm, R. D., and Buchwalter, W. (2019). Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*.

Balduzzi, D., Vanchinathan, H., and Buhmann, J. (2015). Kickback cuts backprop's red-tape: biologically plausible credit assignment in neural networks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., and Lillicrap, T. (2018). Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in Neural Information Processing Systems*, pages 9368–9378.

Becker, S. (1996). Mutual information maximization: models of cortical self-organization. *Network: Computation in neural systems*, 7(1):7–31.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160.

Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., and Lin, Z. (2015). Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Bi, G.-q. and Poo, M.-m. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24):10464–10472.

Bi, G.-q. and Poo, M.-m. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual review of neuroscience*, 24(1):139–166.

Bliss, T. V. and Lømo, T. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *The Journal of physiology*, 232(2):331–356.

Bridle, J. S., Heading, A. J., and MacKay, D. J. (1992). Unsupervised classifiers, mutual information and phantom targets. In *Advances in neural information processing systems*, pages 1096–1101.

Caporale, N. and Dan, Y. (2008). Spike timing–dependent plasticity: a hebbian learning rule. *Annual Review of Neuroscience*, 31:25–46.

Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223.

Cox, D. D. and Dean, T. (2014). Neural networks and neuroscience-inspired computer vision. *Current Biology*, 24(18):R921–R929.

Cramer, R. and Fehr, S. (2011). The mathematical theory of information, and applications (version 2.0).

Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337(6203):129–132.

Debanne, D., Gähwiler, B., and Thompson, S. M. (1994). Asynchronous pre-and postsynaptic activity induces associative long-term depression in area ca1 of the rat hippocampus in vitro. *Proceedings of the National Academy of Sciences*, 91(3):1148–1152.

Debanne, D., Gähwiler, B. H., and Thompson, S. M. (1998). Long-term synaptic plasticity between pairs of individual ca3 pyramidal cells in rat hippocampal slice cultures. *The Journal of Physiology*, 507(1):237–247.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430.

Elad, A., Haviv, D., Blau, Y., and Michaeli, T. (2018). The effectiveness of layer-by-layer training using the information bottleneck principle. *OpenReview*.

Foster, D. J. and Wilson, M. A. (2006). Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440(7084):680.

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127.

Ganchev, T., Fakotakis, N., and Kokkinakis, G. (2005). Comparative evaluation of various mfcc implementations on the speaker verification task. In *Proceedings of the SPECOM*, volume 1, pages 191–194.

Gopnik, A., Meltzoff, A. N., and Kuhl, P. K. (1999). *The scientist in the crib: Minds, brains, and how children learn.* William Morrow & Co.

Gustafsson, B., Wigstrom, H., Abraham, W., and Huang, Y. (1987). Long-term potentiation in the hippocampus using depolarizing current pulses as the conditioning stimulus to single volley synaptic potentials. *Journal of Neuroscience*, 7(3):774–780.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Hebb, D. O. (1949). The organization of behaviour.

Hénaff, O. J., Razavi, A., Doersch, C., Eslami, S., and Oord, A. v. d. (2019). Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hirsch, J., Barrionuevo, G., and Crepel, F. (1992). Homo-and heterosynaptic changes in efficacy are expressed in prefrontal neurons: An in vitro study in the rat. *Synapse*, 12(1):82–85.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Trischler, A., and Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization. *Proceedings of the 7th International Conference on Learning Representations*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., and Sugiyama, M. (2017). Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1558–1567. JMLR. org.

Hyvarinen, A. and Morioka, H. (2016). Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In *Advances in Neural Information Processing Systems*, pages 3765–3773.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

Iriki, A., Pavlides, C., Keller, A., and Asanuma, H. (1989). Long-term potentiation in the motor cortex. *Science*, 245(4924):1385–1387.

Jacobsen, J.-H., Smeulders, A., and Oyallon, E. (2018). i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. (2017). Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1627–1635.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kinney, J. B. and Atwal, G. S. (2014). Equitability, mutual information, and the maximal information coefficient. *Proceedings of the National Academy of Sciences*, 111(9):3354–3359.

Kipf, T. N. and Welling, M. (2016). Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning*.

Kohan, A. A., Rietman, E. A., and Siegelmann, H. T. (2018). Error forward-propagation: Reusing feedforward connections to propagate errors in deep learning. *arXiv preprint arXiv:1808.03357*.

Krause, A., Perona, P., and Gomes, R. G. (2010). Discriminative clustering by regularized information maximization. In *Advances in neural information processing systems*, pages 775–783.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Laughlin, S. B. (1987). Form and function in retinal processing. *Trends in Neurosciences*, 10(11):478–483.

LeCun, Y. (1986). Learning process in an asymmetric threshold network. In *Disordered systems and biological organization*, pages 233–240. Springer.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. (2015). Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 498–515. Springer.

Lee, H., Pham, P., Largman, Y., and Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104.

Levy, W. and Steward, O. (1983). Temporal contiguity requirements for long-term associative potentiation/depression in the hippocampus. *Neuroscience*, 8(4):791–797.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7:13276.

Lillicrap, T. P. and Santoro, A. (2019). Backpropagation through time and the brain. *Current opinion in neurobiology*, 55:82–89.

Linsker, R. (1988). Self-organization in a perceptual network. *Computer*, 21(3):105–117.

Magee, J. C. and Johnston, D. (1997). A synaptically controlled, associative signal for hebbian plasticity in hippocampal neurons. *Science*, 275(5297):209–213.

Marblestone, A. H., Wayne, G., and Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Frontiers in computational neuroscience*, 10:94.

Mazzoni, P., Andersen, R. A., and Jordan, M. I. (1991). A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, 88(10):4433–4437.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2015). A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A Three-Way model for collective learning on Multi-Relational data. In *ICML*, volume 11, pages 809–816.

Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. In *Advances in neural information processing systems*, pages 1037–1045.

Nøkland, A. and Eidnes, L. H. (2019). Training neural networks with local error signals. In *Proceedings of the 36th International Conference on Machine Learning*.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Ororbia, A. G., Mali, A., Kifer, D., and Giles, C. L. (2018). Conducting credit assignment by aligning local representations. *arXiv preprint arXiv:1803.01834*.

Panayotov, V. (2014). Kaldi pretrained model on LibriSpeech SAT and DNN. http://www.kaldi-asr.org/downloads/build/6/trunk/egs/librispeech/. [Online; accessed 29-July-2019].

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Poole, B., Ozair, S., van den Oord, A., Alemi, A. A., and Tucker, G. (2018). On variational lower bounds of mutual information. In *NeurIPS Workshop on Bayesian Deep Learning*.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. (2011). The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society.

Rao, R. P. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Salimans, T. and Bulatov, Y. (2017). Gradient checkpointing.

Scellier, B. and Bengio, Y. (2017). Equilibrium propagation: bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24.

Schmidhuber, J. (1992). Learning factorial codes by predictability minimization. *Neural Comput.*, 4(6):863–879.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, S., Sjöström, P. J., Reigl, M., Nelson, S., and Chklovskii, D. B. (2005). Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS biology*, 3(3):e68.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Sun, C., Baradel, F., Murphy, K., and Schmid, C. (2019). Contrastive bidirectional transformer for temporal representation learning. *arXiv preprint arXiv:1906.05743*.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Tian, Y., Krishnan, D., and Isola, P. (2019). Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*.

Toyoizumi, T., Pfister, J.-P., Aihara, K., and Gerstner, W. (2005). Spike-timing dependent plasticity and mutual information maximization for a spiking neuron model. In *Advances in neural information processing systems*, pages 1409–1416.

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M. (2019). On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, LIX(236):433–460.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2018). Deep graph infomax. *arXiv preprint arXiv:1809.10341*.

Ver Steeg, G. and Galstyan, A. (2015). Maximally informative hierarchical representations of High-Dimensional data. In *Artificial Intelligence and Statistics*, pages 1004–1012. jmlr.org.

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., et al. (2019). Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind Blog*.

Werfel, J., Xie, X., and Seung, H. S. (2004). Learning curves for stochastic gradient descent in linear feedforward networks. In *Advances in neural information processing systems*, pages 1197–1204.

Whittington, J. C. and Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262.

Whittington, J. C. and Bogacz, R. (2019). Theories of error back-propagation in the brain. *Trends in cognitive sciences*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wilson, M. A. and McNaughton, B. L. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172):676–679.

Wiskott, L. and Sejnowski, T. J. (2002). Slow feature analysis: unsupervised learning of invariances. *Neural Comput.*, 14(4):715–770.

Xiao, W., Chen, H., Liao, Q., and Poggio, T. (2019). Biologically-plausible learning algorithms can scale to large datasets. *Proceedings of the 7th International Conference on Learning Representations*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

# *Figures*

# Tables

# Acronyms

| | | |
|---|---|---|
| *AI* | Artificial Intelligence | 8 |
| *ANN* | Artificial Neural Network | 8, 9, 18, 19, 22, 23, 38 |
| *BPTT* | Backpropagation Through Time | 18, 23, 26, 35, 36 |
| *CPC* | Contrastive Predictive Coding | 10, 11, 17, 18, 30, 31, 34–38, 42, 43, 45, 47 |
| *GIM* | Greedy InfoMax | 10, 11, 26, 28–32, 34–37, 41, 42, 44–47, 60 |
| *GRU* | Gated Recurrent Unit | 34 |
| *ICA* | Independent Component Analysis | 43 |
| *KL* | Kullback-Leibler | 13 |
| *MI* | Mutual Information | 10, 11, 13–18, 23, 24, 26, 27, 30, 38, 40–43, 47 |
| *NCE* | Noise Contrastive Estimation | 17 |
| *RNN* | Recurrent Neural Network | 18, 23 |
| *STDP* | Spike-Time Dependent Plasticity | 24 |

# Appendices

## A Additional Material for the Background Section

### A.1 Proof of Proposition 2.2

*Proof.* For the lower bound, we see that if $P_X(x) = 1$ for some $x \in \mathcal{X}$, then $\forall x' \neq x : P_X(x') = 0$ (by the properties of probability distributions). By convention, for $x \in \mathcal{X}$ with $P_X(x) = 0$, the argument in the summation of the entropy is zero. Together with $P_X(x) \log \frac{1}{P_X(x)} = 0$ for $P_X(x) = 1$, we get that $H(X) = 0$. If $H(X) = 0$, then for any $x$ with $P_X(x) > 0$, it must be that $\log \frac{1}{P_X(x)} = 0$, which is true for $P_X(x) = 1$.

For the proof of the upper bound, we use Jensen's inequality. The log-function is strictly concave on $\mathbb{R}_{\geq 0}$, and thus

**Jensen's Inequality**
Let $f : \mathcal{D} \to \mathbb{R}$ be a concave function and $n \in \mathbb{N}$. For any $p_1, \dots, p_n \in \mathbb{R}_{\geq 0}$ with $\sum_{i=1}^n p_i = 1$ and for any $x_1, \dots, x_n \in \mathcal{D}$ it holds that

$$\sum_{i=1}^n p_i f(x_i) \leq f\left(\sum_{i=1}^n p_i x_i\right).$$

If f is *strictly* concave and $p_1, \dots, p_n > 0$, then equality holds iff $x_1 = \ldots = x_n$.
*Proof omitted.*

$$\begin{aligned}
H(X) &= \sum_{x \in \mathcal{X}} P_X(x) \log \frac{1}{P_X(x)} \\
&\leq \log \sum_{x \in \mathcal{X}} P_X(x) \frac{1}{P_X(x)} \\
&= \log \sum_{x \in \mathcal{X}} 1 \\
&= \log |\mathcal{X}| .
\end{aligned}$$

Since the log-function is *strictly* concave, we may restrict the sum to all $x$ with $P_X(x) > 0$. Then, equality holds iff $\log \frac{1}{P_X(x)} = \log \frac{1}{P_X(x')}$, and thus $P_X(x) = P_X(x')$ for all $x, x' \in \mathcal{X}$. □

### A.2 Proof of Proposition 2.4

*Proof.* The lower bound follows from proposition 2.2 and definition 2.3.

For the proof of the upper bound, we get:

$$\begin{aligned}
&H(X|Z) - H(X) \\
&= \sum_{x \in \mathcal{X}, z \in \mathcal{Z}} P_{XZ}(x,z) \log \frac{P_Z(z)}{P_{XZ}(x,z)} - \sum_{x \in \mathcal{X}} P_X(x) \log \frac{1}{P_X(x)}
\end{aligned}$$

By marginalization we can reformulate as

$$= \sum_{x \in \mathcal{X}, z \in \mathcal{Z}} P_{XZ}(x,z) \log \frac{P_X(x) P_Z(z)}{P_{XZ}(x,z)}$$

Due to the convention that the terms in the summation are set to zero when $P_{XZ}(x,z) = 0$, we can restrict the summation to those pairs $(x,z)$ with $P_{XZ}(x,z) > 0$

$$= \sum_{\substack{x \in \mathcal{X}, z \in \mathcal{Z}: \\ P_{XZ}(x,z) > 0}} P_{XZ}(x,z) \log \frac{P_X(x) P_Z(z)}{P_{XZ}(x,z)}$$

Since the log-function is strictly concave on $\mathbb{R}_{\geq 0}$, we can make use of Jensen's inequality:

$$\begin{aligned}
&\leq \log \sum_{\substack{x\in\mathcal{X},z\in\mathcal{Z}:\\ P_{XZ}(x,z)>0}} P_{XZ}(x,z)\frac{P_X(x)P_Z(z)}{P_{XZ}(x,z)}\\
&= \log \sum_{\substack{x\in\mathcal{X},z\in\mathcal{Z}:\\ P_{XZ}(x,z)>0}} P_X(x)P_Z(z)\\
&\leq \log \sum_{x\in\mathcal{X},z\in\mathcal{Z}} P_X(x)P_Z(z)\\
&= \log\left(\sum_{x\in\mathcal{X}} P_X(x)\sum_{z\in\mathcal{Z}} P_Z(z)\right)\\
&= \log 1 = 0\,,
\end{aligned}$$

where for the first inequality, equality holds iff $\forall(x,z)$ with $P_{XZ}(x,z) > 0 : P_X(x)P_Z(z) = P_{XZ}(x,z)$. For the second inequality, equality holds iff $\forall(x,z)$ with $P_{XZ}(x,z) = 0$: $P_X(x)P_Z(z) = 0$. It follows that $H(X|Z) = H(X)$ iff $\forall(x,z)$ : $P_X(x)P_Z(z) = P_{XZ}(x,z)$, i.e. iff $X$ and $Z$ are independent. □

## A.3 *Derivation of the Backpropagation Algorithm*

In the backpropagation algorithm weights are updated in the negative direction of the derivative of the loss function with respect to the weight to be updated:

$$\Delta w_{ij} = -\eta\frac{\partial\mathcal{L}}{\partial w_{ij}}\,. \tag{1}$$

We can calculate this derivative using the chain rule:

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial w_{ij}} &= \frac{\partial\mathcal{L}}{\partial o_j}\frac{\partial o_j}{\partial w_{ij}} &(2)\\
&= \frac{\partial\mathcal{L}}{\partial o_j}\frac{\partial o_j}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}}\,. &(3)
\end{aligned}$$

For the last term, we see that only one term of the summation depends on $w_{ij}$, such that

$$\frac{\partial z_j}{\partial w_{ij}} = \frac{\partial\sum_{k=1}^{n} w_{kj}o_k}{\partial w_{ij}} = \frac{\partial w_{ij}o_i}{\partial w_{ij}} = o_i\,. \tag{4}$$

The second term in Equation (3) can be rewritten as the derivate of the activation function:

$$\frac{\partial o_j}{\partial z_j} = \frac{\partial\varphi(z_j)}{\partial z_j}\,. \tag{5}$$

The first term in Equation (3) can be readily evaluated if $j$ is an output neuron. If $j$ is a hidden neuron, however, we need to make use of the chain rule again to get:

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial o_j} &= \sum_{m\in M}\frac{\partial\mathcal{L}}{\partial o_m}\frac{\partial o_m}{\partial z_m}\frac{\partial z_m}{\partial o_j} &(6)\\
&= \sum_{m\in M}\frac{\partial\mathcal{L}}{\partial o_m}\frac{\partial o_m}{\partial z_m}w_{jm}\,, &(7)
\end{aligned}$$

where $M$ contains all the neurons receiving inputs from neuron $j$. By defining the error term $\delta_j$ as

$$\delta_j = \frac{\partial \mathcal{L}}{\partial o_j} \frac{\partial o_j}{\partial z_j} \; . \tag{8}$$

we arrive at the final update rules:

$$\Delta w_{ij} = -\eta \frac{\partial \mathcal{L}}{\partial w_{ij}} \tag{9}$$

$$= -\eta o_i \delta_j \tag{10}$$

where

$$\delta_j = \begin{cases} \frac{\partial \mathcal{L}}{\partial o_j} \frac{\partial \varphi(z_j)}{\partial z_j} & \text{if } j \text{ is an output neuron} \\ \left(\sum_{m \in M} w_{jm} \delta_m\right) \frac{\partial \varphi(z_j)}{\partial z_j} & \text{if } j \text{ is an inner neuron} \end{cases} \tag{11}$$

## B Experimental Setup

We use PyTorch 1.0 (Paszke et al., 2017) for all our experiments.

### B.1 Vision Experiments

In our vision experiments, we employ the ResNet-50 v2 architecture (He et al., 2016b), in which we remove the max-pooling layer and adjust the first convolutional layer in such a way that the size of the feature map stays constant. Thus, the first convolutional layer uses a kernel size of 5, a stride of 1, and a padding of 2. Additionally, we do not employ batch normalization (Ioffe and Szegedy, 2015) and remove the forth residual block of the architecture. For the greedy training with GIM, we split the remaining architecture according to the residual blocks into three separately trained modules.

We train our model on 8 GPUs (GeForce 1080 Ti) each with a minibatch of 16 images. We train it for 300 epochs using Adam and a learning rate of 1.5e-4 and use the same random seed in all our experiments.

For the self-supervised training using the InfoNCE objective, we need to contrast the predictions of the model for its future representations against negative samples. We draw these samples uniformly at random from across the input batch that is being evaluated. Thus, negative samples can be drawn both from the same or from different images at random patch locations. We found that including the positive sample (i.e. the future representation that is currently to be predicted) in the negative samples did not hurt the final performance. For each evaluation of the InfoNCE loss, we use 16 negative samples and predict up to $k = 5$ rows into the future. For contrasting patches against one another, we spatially mean-pool the representations of each patch.

Before applying the linear logistic regression classifier on the output of the third residual block, we spatially mean-pool the created representations of size $7 \times 7 \times 1024$ again. Thus, the final representation from which we learn to predict class labels is a 1024-dimensional vector. We use the Adam optimizer for the training of the linear logistic regression classifier and set its learning rate to 1e-3. We optimized this hyperparameter by splitting the labeled training set provided by the STL-10 dataset into a validation set consisting of 20% of the images and a corresponding training set with the remaining images.

### B.2 *Audio Experiments*

The detailed description of our employed architecture is given in Table A1. We train our model on 4 GPUs (GeForce 1080 Ti) each with a minibatch of 8 examples. Our model is optimized with Adam (Kingma and Ba, 2014) and a learning rate of 2e-4 for 1000 epochs. We use the same random seed for all our experiments. Overall, our hyperparameters were chosen to be consistent with Oord et al. (2018).

For the supervised baselines, we observed strong overfitting which we countered by early stopping after 100 epochs on the phone classification task and after 300 epochs in the speaker classification task.

| Layer | Output Size (Sequence Length × Channels) | Parameters Kernel | Stride | Padding |
|---|---|---|---|---|
| Input | $20480 \times 1$ | | | |
| Conv1 | $4095 * \times 512$ | 10 | 5 | 2 |
| Conv2 | $1023 * \times 512$ | 8 | 4 | 2 |
| Conv3 | $512 * \times 512$ | 4 | 2 | 2 |
| Conv4 | $257 * \times 512$ | 4 | 2 | 2 |
| Conv5 | $128 \times 512$ | 1 | 2 | 1 |
| GRU | $128 \times 256$ | - | - | - |

Table A1: General outline of our architecture for the audio experiments. Every convolutional layer is followed by a ReLU activation function.
* For applying the InfoNCE objective on these layers, we randomly sample a time-window of size 128 to decrease the dimensionality.

Similarly to the vision experiments, we take the negative samples uniformly at random from across the batch that is currently evaluated. Again this may include the positive sample. In our audio experiments, we use a total of 10 negative samples and predict up to $k = 12$ time-steps into the future.

We train the linear logistic regression classifier using the representations of the top, autoregressive module. For the speaker classification task, the representations are first average-pooled over all time-steps of the sequence, while no pooling is applied for the phone classification. Again, we employ the Adam optimizer but select different learning rates than before. For this hyperparameter search, we split the training set provided by Oord et al. (2018) into two random subsets using 25% of the samples as a validation set. In the speaker classification experiment, we used a learning rate of 1e-3, while we set it to 1e-4 for the phone classification experiment.