Cyprus University of Technology

Faculty of Engineering and Technology

Doctoral Dessertation

Information-Theoretic Local Competition and Stochasticity for Deep Learning

Anastasios Antoniades
Academic Supervisor:Dr. Sotirios Chatzis

Limassol, March 2022

CYPRUS UNIVERSITY OF TECHNOLOGY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING, COMPUTER ENGINEERING AND INFORMATICS

Doctoral Dissertation

Information-Theoretic Local Competition and Stochasticity for Deep Learning

Anastasios Antoniades

Limassol, March 2022

Approval Form

Doctoral Dissertation

Information-Theoretic Local Competition and Stochasticity for Deep Learning

Presented by

Anastasios Antoniades

Supervisor: Dr. Sotirios Chatzis, Associate Professor

Signature _____

Member of the committee: Dr. Fragkiskos Papadopoulos, Associate Professor

Signature _____

Member of the committee: Dr. Andreas Nearchou, Professor

Signature _____

Cyprus University of Technology

Limassol, January 2022

Copyrights

The approval of the dissertation by the Department of Electrical Engineering, Computer Engineering and Informatics does not imply necessarily the approval by the Department of the views of the writer.

I am deeply grateful to my supervisor Dr. Sotirios Chatzis, without whom nothing would have been possible. I would like to thank my best friends Pavlos Yiannakou for all the constructive discussions we had from the beginning of our studies which motivated me to start and complete successfully this PhD project. Special thanks I would like to dedicate to Demetris Samouel who at difficult times of dead-ends, he was kindly steering me in the right direction and motivated me to keep it up. Finally, I would like to express my warmest gratitude to my family Stelios, Christiana, Charis and Irene, who have supported me along this journey.

A few words ...

from the first year of my studies at the University of Patras, first semester, first discussion in the first break I was fully aware of how fascinated I am by the mutual exchange of ideas and thoughts on topics related to science. I had the thirst and that desire to know what exactly is hidden behind by what my naked eye perceives. In order to satisfy that feeling, the doctorate was the only way. It was the only way to push my limits and get familiar with what is called uncertainty. Uncertainty is one of the most difficult obstacles that a researcher has to deal with and fortunately the friction with the field of Artificial Intelligence me those guarantees not just to overcome it but also to model it and to walk through the unknown by always keeping light with me.

ABSTRACT

Deep Learning has brought about a revolution in the capabilities of Artificial Intelligence systems in the last decade. The foundational operating principle of Deep Learning algorithms that differentiates them from previous approaches is the focus on learning strong representations of the modeled data.

The networks learn to extract these representations in a way that retains the salient information, while removing anything that is not useful for making correct inference.Unfortunately, deep networks suffer from vulnerability to adversarial attacks.That is, appropriately algorithmically crafted counter-examples, which are very easy for average humans to discern, completely foul state-of-the-art deep networks into misclassification. This problem implies that the representations deep networks learn to discern are actually brittle and contain information which is not salient enough to allow for strong generalization capacity under adverse conditions.

Recently, deep networks with Stochastic local winner-takes-all (LWTA) units have been proposed as a potent means of learning to extract data representations with one order of magnitude better generalization capacity in hard adversarial attack settings. This work builds upon this success, aiming to deal with the problem of learning diversified representations which is long-established.

For this purpose, we combine information-theoretic arguments with stochastic competition-based activations, that is Stochastic LWTA units. Under this framework, we leave the conventional deep architectures behind that are over-used in Representation Learning and based on non-linear activations and we replace these activations with sets of stochastically and locally competing linear units. In this setting, each network layer yields sparse outputs, determined by the outcome of the competition between units that are organized into blocks of competitors. We adopt stochastic arguments for the competition mechanism, which perform posterior sampling to determine the winner of each block.

We further endow the considered networks with the ability to infer the sub-part of the network that is essential for modeling the data at hand; we impose appropriate stick-breaking priors to this end. In order to make rich the information of the emerging representations, we resort to the context of information theory, and more specifically to the Information Competing Process (ICP). Then, under the stochastic Variational Bayes framework for inference, we have tied all the components together. We perform an experimental investigation with detail for our

approximation by using benchmark datasets on image classification.

As we show in our experiments, the resulting networks yield significant discriminative representation learning abilities. Additionally, the introduced paradigm allows for a principled investigation mechanism of the emerging intermediate network representations.

Keywords: Machine Learning, Bayesian Inference, Variational Bayes,Sampling methods,Information Theory,Variational Information Bottleneck,Local Winners Takes All,Representation Learning.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI:	Artificial Intelligence
VI:	Variational Inference
BBVI:	Black Box Variational Inference
BP:	BackPropagation
VGPs:	Variational Gaussian Process
VGPDS:	Variational Gaussian Process Dynamical Systems
PCA:	Principal Component Analysis
DNNs:	Deep Neural Networks
NN:	Neural Network
GP:	Gaussian Process
ELBO:	Evidence Lower Bound
EM:	Expectation Maximization
MAP:	Maximum a Postriori
KL:	Kullback Leibler (divergence)
MC:	Monte Carlo
ML:	Maximum Likelihood
RBF:	Radial Basis Function
ReLU:	Rectified Linear Unit
DP:	Diriclet Process
SGD:	Stochastic Gradient Descent
IBP:	Indian Buffet Process
LWTA:	Local Winner Takes All
BNN:	Bayesian Neural Networks
ICP:	Information Competing Process
RL:	Representation Learning

LIST OF PUBLICATIONS

Anastasios Antoniadis, Konstantinos Panousis and Sotirios P Chatzis. " Competing Mutual Information Constraints with Stochastic Competition-based Activations for Learning Diversified Representations". In: Association for the Advancement of Artificial Intelligence (AAAI) 2022.

Pavlos Nikolaides, Anastasios Antoniadis and Sotirios P Chatzis. " A Bayesian Optimization Approach For The Robust Unit Commitment Of Identical Generating Units ". In: The 12th Mediterranean Conference on Power Generation, Transmission, Distribution and Energy Conversion (MEDPOWER 2020), 2021 p. 264 – 269.

.

Chapter 1

Introduction

The English mathematician Alan Turing in 1936 introduced the Turing machine, a computational model that pushed the development of computing to new heights. Turing published a study in 1950 with the title"Computational Machinery and Intelligence," which is often mentioned as the cornerstone of modern artificial intelligence, more specifically the ability of a machine to display human potential such as reasoning, creativity, and learning.

Artificial Intelligence (AI) started the upward course, in the 1960s around the idea that we can decode and clarify smart human behaviors as a pattern of logical rules expressed by algorithms that machines could follow in order to present that behavior. The information given to the machine has been converted into symbols that the computer could handle by using a sequence of rules. This approximation has led to the design of knowledge systems that extract knowledge to solve problems.

Since 2000 AI has gradually peaked and came up from the ability of machines to learn from huge amounts of data. The algorithms behind these systems work through the correlation of the data being analyzed, thus giving the opportunity to the mechanisms to perform basic computer functions.

According to the human ability to learn, philosophers ask the question: "What is the way to evaluate inductive reasoning that leads to learning in terms of its correctness?". Accordingly, psychologists ask: "How the brain manages to store the results of the learning process and perceive patterns?" In the field of AI they simply ask: "How does a machine manage to create new models and learning patterns from specific examples and how accurate are these models in their application?"Using the aforementioned as a foundation ,could be given the following definition for Machine Learning, which is a key part of AI: Machine Learning is the ability of

a computing system to create models or patterns from a data set in order to predict the next movement. Machine learning in our era is one of the most evolving fields of science inspired by neuroscience and neurology and using tools from applied mathematics has succeeded in a very short time to achieve miracles in the field of science and AI. The essence of machine learning is summarized in the use of algorithms with general-purpose to identify patterns in the data in order to make decisions as accurately as possible. Machine learning in simple words is the application of Artificial Intelligence that gives to computers the ability to predict the outcomes automatically without the intervention of humans and is used in many aspects of our everyday life like marketing and sailing, health care,financial services, social media services, etc. Depending on the desired outcome, the algorithms are divided into the following categories.

Supervised learning is the process where a practitioner constructs a function that maps the inputs whose label is known to outputs of which the label is also known, with main purpose to generalize the process to inputs with unknown output. It is known as supervised learning due to the fact that the algorithm learning process from the training data set can be considered as a teacher supervising the learning process. Knows the correct answers, makes repeated predictions about training data and intervenes when there is a real need. The algorithm stops when it starts converging on a certain number and achieves an acceptable level of performance.The approach is used in problems such as **Classification** which is a method that maps the data to a specific category, in simple words classification tries to predict a discrete class label output for an example also can have discrete or real-valued input variables. **Regression** which is the problem that predicts a continuous quantity output for an example, however, is the task that approximating a function f from input variables X to a continuous output variable y. **Interpretation** examines the model's parameters and figure out at a general way how the model works.

Unsupervised learning where the algorithm creates a model with specific inputs without knowing the outputs.Unsupervised learning is when the input data (X) is known but there are no corresponding output variables. The ultimate goal of unsupervised learning is to model the distribution of data for the purpose to learn as much as is possible about the data. These are called unsupervised learning due to the fact that unlike supervised learning above there is no teacher and there are no correct answers. Algorithms are left to their own devices to discover and present the interesting data structure. The approach is used in **Association Analysis** the method that tries to discover the relationship between variables in large databases. It is aimed to recognize strong rules discovered in databases.**Clustering** what exactly does that method is

the natural grouping of data. In contrast to supervised learning, clustering algorithms only interpret the input data and find natural groups or clusters in feature space.

Semi-supervised learning when facing problems with very large datasets where only some of the data is labeled are called semi-supervised learning problems. These problems are between both unsupervised and supervised learning. An expressive example is when a photo archive where only some of the images are labeled, and the majority are unlabeled.Numerous real world machine learning problems fall into this category. This is due to the fact that it can be time-consuming to label data. Therefore the data which is with out label is easy and cheap to collect it. The structure of the inputs can be discovered and learned by unsupervised learning methods and also with the supervised learning which is used in order to to make predictions for the data with out label. That data are feeded back into the supervised learning algorithm as training data, and the model are used to generalize the unseen data.

Reinforcement learning It is the field that trying learning the optimal behavior in an unknown environment. This behavior is learned through interactions with the environment and observations.This is very close to children's behavior who try to explore the world around them and learn from their mistakes and their actions. In the absence of the supervisor,the learner should automatically discover the sequence of actions that maximize effectiveness. This process is like a trial and error search. The quality of the actions is measured by the immediate performance they return and also the delayed reward they may receive. As it can learn actions in an unknown environment that guide the algorithm to success without the assistance of a specific observe.

1.1 Statistical Inference

Statistical inference is a huge area that includes many statistical methods from analyzing data to drawing conclusions or inferences in business and research problems. It is a vitally important field in terms of applications in data science. Based on random sampling, statistical inference is the technique of making decisions about the parameters of a population. It enables us to estimate the relationship between independent and dependent variables. The main idea of statistical inference is to assess the uncertainty or variation from sample to sample. It gives us the ability to deliver a range of values for the true value of the population. Statistical inference is the procedure through which inferences are derived concerning a population and are based on characteristics evaluated from a sample of data drawn from that population. In

statistical inference, statements are made not merely about the particular subjects observed in research but also, more importantly, about the larger population of subjects from which the study participants were drawn. First of all it is of vital importance to discuss about the two main philosophies of statistical inferences which is the Bayesian inference and the Frequentist inference. In this thesis, we will focus on the Bayesian inference because have been observed better results from the time which achieve to set the right prior distribution over the hyperparameters of the model. Let mention below some of the main properties of each other.

1.1.1 Frequentist

The frequentist dominated statistical practice during the 20th century by using only conditional distributions of data given some specific hypotheses. It continues to be the most popular method used in scientific literature to this day — concepts such as confidence intervals and p-values belong to the frequentist paradigm. The substantial idea of frequentist statistics is regarding repeatability and gathering more data. In simple words, the interpretation of frequentist is the recurring and multi-occurring frequency of its occurrence. For instance, the probability of a coin to be head is 0.5 if we were to flip the coin numerous times. The frequentist approximation considered that any uncertainty in probabilistic estimates is due to the sampling error between the sample and the population so having large samples or samples that are closer to the actual population in frequentist approximations is a perfect scenario. Specifically, the frequentist approach does not depend on a prior distribution which there is a case to have a different form from one investigator to another due to its subjectivity which is the main argument that frequentists used against Bayesians. What exactly frequentist do is try to point estimate the parameters of the model.

1.1.2 Bayesian inference

Bayesian approach is just the procedure of inferring properties about a probability distribution or population from data using the Bayes theorem. The Bayesian approach models uncertainty by imposing a probability distribution over parameters so in that case, the performance of our model is dependent to a high degree on the prior and likelihood of observed data. There is a chance of high computationally intensive due to the integration of a large number of parameters.

However, in this era of big data and powerful computers, Bayesian methods[4] have undergone a tremendous renaissance in fields like machine learning. Below is a detailed analysis of the context of Bayesian inference. Bayesian inference can be considered as a way to get the right predictions from your data and it is particularly useful when you do not have as much data as you would like. Below is the definition of Bayes theorem which lay the foundation and the cornerstone in order to move smoothly from the mathematical formulation of Bayes theorem to the context of statistical inference.

$$P(\mathrm{A}|\mathrm{B}) = P(\mathrm{A})\frac{P(\mathrm{B}|\mathrm{A})}{P(\mathrm{B})}, \tag{1.1}$$

where A and B are events and $P(\mathrm{B}) \neq 0$

- $P(A|B)$ is a conditional probability: the likelihood of A occurring given that B is true.
- $P(A|B)$ is also a conditional probability: the likelihood of event B occurring given that A is true.
- $P(\mathrm{A}), P(\mathrm{B})$ the probabilities of observations which is independently of each other and is known as the marginal probability.

For the inversion procedure, all is needed is to guess the prior distribution $P(\mathrm{A})$ in order to estimate the posterior. Prior distribution is chosen by intuition about the likeness of underlying regularities with a known family of distributions. Once a reasonable choice of the prior is available, advantages associated with using the Bayesian approach emerge and will be made clear in the next sections.

Assuming that a hypothesis about the prior distribution is a set, possible routes on how to calculate the posterior distribution will be discussed.We will be discussed in detail how we can calculate the posterior distribution. Applying Bayes rule and considering the new information as $\mathbf{y}$, the posterior of the model is given by:

$$P(m_i|\mathbf{y}) = \frac{P(\mathbf{y}|m_i)P(\mathbf{y})}{P(m_i)} \tag{1.2}$$

$$P(\mathbf{y}) = \sum P(\mathbf{y}|m_i)P(m_i), \tag{1.3}$$

The first term in the numerator is the likelihood of our model which tell us how our data interacts with our parameters of a specific model and is very important quantity in Bayesian inference.

$$P(\mathbf{y}|m_i) = \int P(\mathbf{y}|\theta, m_i)P(\theta|m_i)d\theta, \tag{1.4}$$

The second term $P(m_i)$ is known as a prior and captures everything we know about the parameters of the model. The term in denominator $P(\mathbf{y})$ is a normalizing constant, also known as the marginal likelihood of the model specified by hyper-parameters θ, because it computes accurately the likelihood of observing the given data set given the modelling assumptions encapsulated in θ, which refers to the marginalization over the whole of parametric space in order to make predictions or to compare different kind of models.

Marginal likelihood has the form of intractable integration, which is very difficult to solve it analytically. Few years ago the problem of integration has been faced by constraining the models with simpler likelihood functions with specific formulas for prior distributions, which are known as conjugate priors with combination of likelihood function generated a tractable integral.

1.2 Sampling Methods for Bayesian Inference

The specialized computational tools enable the bayesian data analysis methods to be more flexible and more efficcient, apart from the simple models, explicit solutions for quantities relevant to Bayesian inference are not available. This limitation has motivated the specialists to develop numerous different approximation methods to overcome this obstacle. Due to the fact that the memory and capacity of computers have undergone an impressive advancement in recent years, the research community has taken advantage of this stuff and it used sampling techniques such as rejection sampling ,importance sampling and Monte Carlo, and more specific Markov Chain Monte Carlo which it was one of the most influential algorithms of the 20th century. It is widely used and has its roots in physics[5], and to the end of 1980, they started to have a significant impact in the field of statistics. These fundamental parts of sampling have been illustrated in detail by Bishop and Nabney in 2008. Sampling methods have historically been the main way of performing approximate inference. [1].

$$\mathbb{E}[f] = \int f(\mathbf{z}) p(\mathbf{z}) dz \tag{1.5}$$

This method allowed us to approximate the expectation by a finite sum

$$\hat{f} = \frac{1}{\mathrm{L}} \sum f(z^{(l)}) \tag{1.6}$$

1.2.1 Rejection Sampling

In order to implement rejection sampling[6] is needed to sample from the distribution $p(\mathbf{z})$ that is not one of the simplest and standard distributions examined, so the immediately sampling from $p(\mathbf{z})$ is complicated. Moreover let consider, as is often, that we can evaluate $p(\mathbf{z})$ easily for any value of $\mathbf{z}$, up to some normalizing constant Z, so that

$$p(\mathbf{z}) = \frac{1}{Z_p} \bar{p}(z) \tag{1.7}$$

where Z_p is not known and $p(\mathbf{z})$ can easily be estimated. In order to implement rejection sampling, we need a proposal distribution, from which we can draw samples easily from simpler distribution $q(\mathbf{z})$. The constant k are introducing whose value is selected in order to verify the $k\dot{q}(\mathbf{z}) \geq \bar{p}(\mathbf{z})\ \forall z$. This $kq(\mathbf{z})$ function is well known as comparison function. Every step of the rejection sampler include generating two random numbers. First of all, we generate z_0 from the distribution $q(\mathbf{z})$.. Secondly, by using the uniform distribution over $[0, k\dot{q}(z_0)]$ we generate a number u_0. This doublet of random numbers has a uniform distribution under the curve of the function $k\dot{q}(\mathbf{z})$. Finally, if $u_0 \geq p(z_0)$ then the sample is rejected, otherwise u_0 is holded.

1.2.2 Importance Sampling

By relying on the use of a proposal distribution $q(\mathbf{z})$ which it is easier to draw samples, importance sampling make an approaching framework for approximating expectations directly but does not supply a system for drawing samples from distribution $p(\mathbf{z})$. The expectation can be expressed in the form of a finite sum as below. 1.8[1]:

$$\mathbb{E}[f] = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z} = \int f(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z}) d\mathbf{z} \backsimeq \sum \frac{p(z^{(l)})}{q(z^{(l)})} f(z^{(l)}) \tag{1.8}$$

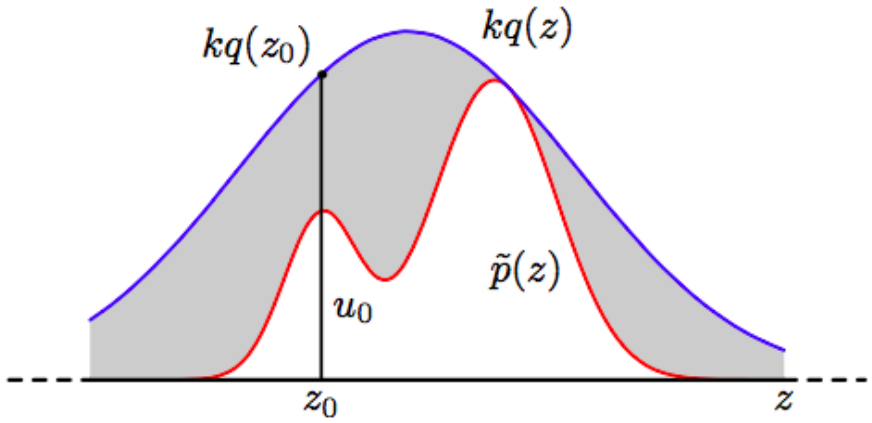


Figure 1.1: Samples are drawn from $q(z)$ and rejected if they drop in the grey field between the the scaled distribution $kq(z)$ and unnormalized distribution $\tilde{p}(z)$. The results are distributed according to $p(z)$, which is the normalized version of $\tilde{p}(z)$.[1]

$r_l = \frac{p(z^{(l)})}{q(z^{(l)})}$are called as importance weights, and they improve the bias introduced by sampling from the wrong distribution. Notice that, in contrast rejection sampling all of the samples generated are maintained.

1.2.3 Markov chains Monte-Carlo

As we mentioned above MCMC[7] is a sampling method that gives you the opportunity to find the posterior distributions of our parameters of the model. Especially, this type of algorithm generates Monte Carlo simulations in a way that based on the Markov property, then accepts these simulations at a certain rate to get the posterior distribution[8]. As with importance and rejection sampling,again sample have been sampled from a proposal distribution. MCMC method retained a record of the current state $\mathbf{z}(t)$, and the proposal distribution $q(\mathbf{z}|\mathbf{z}^{(\tau)})$ depends on this current state, so the sequence of samples z^1, z^2, z^3... can be illustrated as a Markov chain. The proposal distribution is selected to be simple that it is easier to draw samples from it. At each loop of the algorithm, produced a candidate sample z which is selected accordingly to a criterion from the proposal distribution. In simple words Metropolis algorithm, supposed the property of symmetry for the proposal distribution, which is $q(\mathbf{z}_A|\mathbf{z}_B) = q(\mathbf{z}_B|\mathbf{z}_A)$for all values of $\mathbf{z}_A$ and $\mathbf{z}_B$. The sample then accepted with probability.

$$\mathbf{A}(\mathbf{z}^\star, \mathbf{z}^\tau) = \min(1, \frac{\mathbf{z}^\star}{\mathbf{z}^\tau}) \tag{1.9}$$

if $\mathbf{A}(\mathbf{z}^{\star}, \mathbf{z}^{\tau}) \subset u, u \in (0,1)$then accepting the sample otherwise discard it. Notice that if the step from $z^{(\tau)}$ to $z^{\star}$ might lead to the increase in the value of $p(\mathbf{z})$, then the candidate point is certain to be kept. If the candidate sample is accepted, then $\mathbf{z}^{(\tau+1)} = \mathbf{z}^{\star}$, in other case the candidate point $\mathbf{z}$ is rejected, $\mathbf{z}(\tau+1)$ is set to $\mathbf{z}^{(\tau)}$ and another candidate sample is drawn from the distribution $q(\mathbf{z}|\mathbf{z}^{(\tau+1)})$ and we repeat this procedure. The Metropolis algorithm[9] when a candidate sample is rejected, the prior sample is involved instead in the final list of samples, causing a numerous copies of samples.

Despite all the advantages, the sampling methods have many drawbacks such as they are computationally intensive, you can use it only for small-scale problems, you are not sure about the independently of our sample and last but not least it was very slow so that disadvantages allowed the deterministic techniques to arise a new kind of method which is known as variational inference or variational Bayes which is making approximate inference for parameters in complex statistical models. Monte Carlo[10] provide a numerical approximation to the posterior using a set of samples. Variational approximation supply a locally optimal analytical and accurate solution to an approximation of the posterior.In the next chapter we will discuss with more detailed this method.

1.3 Variational Approximation

Variational inference has been proposed by M.Jordan in 1999 [11] is a popular method for approximate Bayesian inference which relies on the assumptions about the posterior over latent space. Variational Bayes (VB) is an optimization algorithm that is used for main reason to approximate Bayesian inference and is much more efficient than the typical sampling methods. VB can also be used in the frequentist framework in order to estimate the maximum likelihood when there are missing data. In the literature, you can find also the names Variational Bayes and Variational Inference are often used exchangeably. In simple words, variational inference converts the inference problem into an optimization problem. The name was taken from the variational distribution which is used for modeling the hidden variables and is one of the most popular deterministic approximate inference technique. The variational inference has the potential to confront problems with a large amount of data and high dimensionality instantly in order to provide an analytical approximation to the posterior probability of the hidden variables.

Approximate posterior with $q^{(\star)}$ and find it with an optimization approach, so suppose there

are lot of distributions out there in the world and there is a set of nice distributions, my goal is to find the mean and the covariance of my posterior. Firstly, tried to find the $q^\star$ which is the better distribution in the set of nice distributions which aforementioned. To achieve this it is needed to define a measure to tell us who is the closest distribution to the exactly posterior. At this point, the mean of Kullback–Leibler divergence emerges

$$q^\star = argmin f(q(\cdot), p(\cdot|y)) \tag{1.10}$$

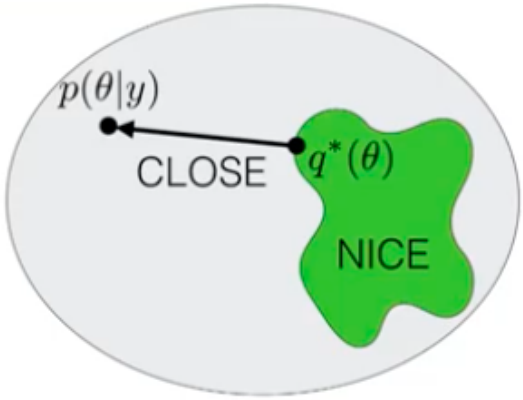


Figure 1.2: Illustration of how the variational approximation works with $q^\star_\theta$, p($\theta|y$) symbolize the variational distribution and exactly posterior respectively

and f is Kullback-Leibler divergence $KL(q||p)$ which is a kind of a distance between the exactly posterior and the approximate posterior, in simple words is a way to estimate the difference between two distributions.

$$\begin{aligned} KL(q(\cdot)||p(\cdot)) = \int q(\theta) \log \frac{q(\theta)}{p(\theta|y)} d\theta = \int q(\theta) \log \frac{q(\theta)p(y)}{p(\theta,y)} d\theta \\ = \log p(y) - \int q(\theta) \log \frac{p(\theta,y)}{q(\theta)} d\theta \end{aligned} \tag{1.11}$$

The general purpose is to minimize the KL divergence over q, as observed the first term does not contain q so the derivative will throw this term out because it affects nothing. The second term contains only things that we know $p(\theta,y)$ that just the prior times the likelihood those that exactly the inputs that we put to this problem the second term is known as a evidence lower bound or **ELBO** [12] and this is must be optimized.

$$
\begin{aligned}
\log p(x) &\geqslant \iint q(z)q(\theta)\log\frac{p(x|z,\theta)p(z)p(\theta)}{q(z)q(\theta)}dzd\theta \\
&= \iint q(z)q(\theta)\log p(x|z,\theta)dzd\theta \\
&+ \iint q(z)q(\theta)\log\frac{p(z)}{q(z)}dzd\theta \\
&+ \iint q(z)q(\theta)\log\frac{p(\theta)}{q(\theta)}dzd\theta \\
&= E_{q(z)q(\theta)}[\log p(x|z,\theta)] \\
&+ \int q(z)\log\frac{p(z)}{q(z)}dz + \int q(\theta)\log\frac{p(\theta)}{q(\theta)}d\theta \\
&= E_{q(z)q(\theta)}[\log p(x|z,\theta)] - D_{KL}[q(z) \parallel p(z)] - D_{KL}[q(\theta) \parallel p(\theta)] \\
&= \mathscr{L}(q)
\end{aligned}
\tag{1.12}
$$

characterizing KL divergence

if $q\star$ and p are high, we are happy
if $q\star$ is high but p is not, we pay a price
if $q\star$ is low, we do not care
if $KL = 0$ then distribution are identical

Due to the fact that KL has the main property of a distance which is $KL \geq 0 \implies \log p(y) \geq$ ELBO.The really important point here is that minimizing the KL is exactly equivalent to finding the$q\star$ to be the maximizer of the ELBO.

$$
q_\star = argmax f(q(\cdot), p(\cdot|y)) \tag{1.13}
$$

Which is exactly the same problem.

1.3.1 Mean-Field Variational Inference

The model's complexity regulates the optimization complexity. The difficulty to optimize a complex model is bigger than a simple model. Below will be discussed the mean-field Variational Inference which makes the assumption of independence between latent variables and focuses on the mean-field variational family[13], where each one of the latent variables is independent and are governed by a distinct factor in the variational density. A reliable

predictive model to be expressive enough needs to include numerous of latent variables. The previous formulation for the parametric form of the approximate distribution is better written as $q(\mathbf{Z}|\phi)$ where $\mathbf{Z} = [z_1, z_2, .., z_n]$. This imposes a difficulty on the training due to the dependence of the variables.This can be simplified with the mean-field approximation where has this specific property of latent variable in-dependency .A generic member of the mean-field variational family is

$$q(Z|\varphi_i) = \prod q(\zeta_i|\varphi) \tag{1.14}$$

However, introducing independency between the latent variables leads to a family of approximate distributions that are less expressive and will not include the posterior. Mean-Field Variational Bayes and the simple Variational Bayes limit their use to mainly conjugate models. Non-conjugate models can be trained with ad-hoc models of variational inference algorithms such as approximations. This includes models like Bayesian logistic regression,discrete choice models ,Bayesian generalized linear models, Bayesian item response models and non-conjugate topic models[14].

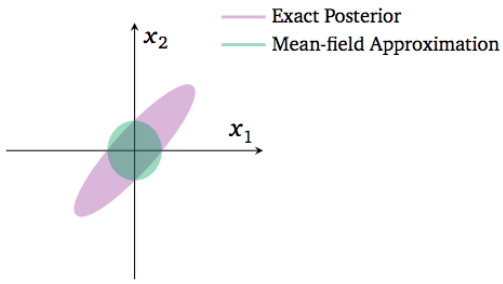


Figure 1.3: Illustration of the mean-field approximation to a two-dimensional Gaussian posterior[2]

Although we focus on drawing mean-field inference, researchers have dealt with more difficult families, a great way to enrich these families is to create dependencies between model variables.[15];[16], this is known as structured variational inference. A different way to expand these families is to consider mixtures of variational densities, additional latent variables within the variational family [1]. The two aforementioned methods equally improve the trueness of the approach but there is a trade-off. Structured and mixture-based variational families come with more difficulty to solve the variational optimization problems. In the next sections we will discuss techniques that enrich the Variational Bayes algorithm to train more expressive models.

1.3.2 Stochastic Variational Inference

Stochastic variational inference[17] is a way to approximate the posterior distributions which arise by sizeable datasets rapidly using stochastic optimization. Stochastic optimization is indispensable for the Stochastic variational inference in order to find the global variational parameters.Stochastic optimization[18][19] algorithms are using noisy estimates of the gradient with smaller step sizes at each time due to the fact that the estimation of gradients with Noise is very often cheaper to compute than the true gradient, and these kinds of estimations are able to assist algorithms to escape from local optima of complicated objective functions and take us to the context of fitting the global optima.

Assume $f(\lambda)$ as objective function and $B(\lambda)$ as a random function that has a gradient equal to expectation so that $\mathbb{E}_q[B(\lambda)] = \nabla_\lambda f(\lambda)$

The stochastic gradient algorithm, which belongs to the category of stochastic optimization algorithms optimize $f(\lambda)$ by iteratively following $B(\lambda)$.

$$\lambda^{(t)} = \lambda^{(t-1)} + \rho_t b_t(\lambda^{(t-1)}) \tag{1.15}$$

where b_t is an independent draw from the noisy gradient B. If the sequence of step sizes ρ_t satisfies:

$$\sum \rho_t = \infty \qquad \sum \rho_t^2 < \infty \tag{1.16}$$

then $\lambda(t)$ will converge to the optimal $\lambda^\star$ if f is convex. The aim of stochastic variational inference is to regularize and fit the global parameter λ that maximizes the evidence lower bound. We express L in terms of of the local and global variational parameters, we assume that the function $\phi(\lambda)$ return a local optimum of the local parameters

$$\nabla(\lambda, \phi(\lambda)) = 0 \tag{1.17}$$

determine the locally maximized ELBO to be the objective function when λ remain constant and the local variational parameters ϕ are define to be a local optimum $\phi(\lambda)$

$$L(\lambda) \triangleq L(\lambda, \phi(\lambda)) \tag{1.18}$$

The ELBO $L(\lambda)$ are optimized by stochastic variational inference by subsampling the data in order to create estimations of the natural gradient that have noise. We deconstruct $L(\lambda)$ into a

sum of local terms and a global term[20].

$$L(\lambda) = \mathbb{E}_q[\log p(\beta)] - \mathbb{E}_q[\log q(\beta)] + \sum_{n=1}^{N} max(\mathbb{E}_q[\log p(x_n, z_n|\beta)]) - \mathbb{E}_q[\log q(z_n)] \quad (1.19)$$

We assume a variable that its index is chosen randomly from the uniform distribution,$I \sim Unif(1,...,N)$. Determine $L_I(\lambda)$ as a random function of the variational parameters,

$$L_I(\lambda) = \mathbb{E}_q[\log p(\beta)] - \mathbb{E}_q[\log q(\beta)] + Nmax(\mathbb{E}_q[\log p(x_I, z_I|\beta)]) - \mathbb{E}_q[\log q(z_I)] \quad (1.20)$$

The the objective of $L(\lambda)$ is exactly the same to $\mathbb{E}[L_I(\lambda)]$. However, the natural gradient of L_I with respect to each parameter λ is unbiased but a noisy estimate of the natural gradient of the objective function. The process of estimate the natural gradient call L_I and then sampling a data point will provide us with the ability to estimate noisy gradients easiest for stochastic optimization. Now we are estimating the noisy slope

$$\nabla L_i = \mathbb{E}_q[\eta_g(x_i^{(N)}, z_i^{(N)}, \alpha)] - \lambda \quad (1.21)$$

$\{x_i^{(N)}, z_i^{(N)}\}$ are the dataset and $\eta_g(x_i^{(N)}, z_i^{(N)})$ is a function of random variables and observations. Lastly, we are using the aforementioned noisy natural gradients in an algorithm created by Robbins-Monroto in order to optimize the ELBO. We determine the intermediate global parameter $\hat{\lambda}_t$ to be the estimation of λ

$$\hat{\lambda} \triangleq \alpha + N\mathbb{E}_{\phi_i(\lambda)}[(t(x_i, z_i), 1)] \quad (1.22)$$

We are using the noisy gradient with step size ρ_t at each iteration to update the global variational parameter. Below is the update:

$$\lambda^{(t)} = \lambda^{(t-1)} + \rho_t(\hat{\lambda}_t - \lambda^{(t-1)}) \quad (1.23)$$

1.3.3 Automatic Variational Inference

In previous works, they used to use a variational inference algorithm which is needed the painstaking work of implementing a custom optimization approach. Variational family which

1: Initialize $\lambda^{(0)}$ randomly.
2: Set the step-size schedule ρ_t appropriately.
3: **repeat**
4: Sample a data point x_i uniformly from the data set.
5: Compute its local variational parameter,

$$\phi = \mathbb{E}_{\lambda^{(t-1)}}[\eta_g(x_i^{(N)}, z_i^{(N)})].$$

6: Compute intermediate global parameters as though x_i is replicated N times,

$$\hat{\lambda} = \mathbb{E}_{\phi}[\eta_g(x_i^{(N)}, z_i^{(N)})].$$

7: Update the current estimate of the global variational parameters,

$$\lambda^{(t)} = (1-\rho_t)\lambda^{(t-1)} + \rho_t\hat{\lambda}.$$

8: **until** forever

Figure 1.4: The algorithm of Stochastic Variational Inference

is appropriate for the model requires computing cost in order to compute objective function and compute the derivatives. ADVI overcomes this problem by generating automatically variational algorithms.

The main idea of ADVI is to achieve the automatic transformation of the problem into a space that is feasible to solve the variational optimization problem. This common space solves variational inference for all models in a large class. We will discuss specifically the procedure[21].

In the first step we will examine how to achieve to use a single variational family for all models.It transforms $\mathbf{p}(\mathbf{x}, \theta)$ to $\mathbf{p}(\mathbf{x}, \zeta)$, where the mapping from θ to ζ is built into the joint distribution. This method eliminate all the limitations on the latent variables θ. ADVI then specify the variational problem on the transformed variables in order to minimize this amount $KL(q(\zeta)|p(\zeta|\mathbf{x}))$ this transformation enable all the latent variables to defined on the same space.
Secondly reforms the gradient of the variational objective function as an expectation over q. Expressing the gradient as an expectation enable us to use the method of Monte in order to approximating it.
Thirdly, enables us to reparameterizes the gradient in terms of a Gaussian. In order to achieve this use another transformation in the variational family. This transformation this time enables ADVI to compute efficiently Monte Carlo approximations with samples from Gaussian. Finally, optimize the variational distribution, ADVI use noisy gradients.

1.3.4 Black-Box Variational Inference

This approximation method have been proposed by Rajeshr[22],with general purpose to overcome some limitations that emerges from the typical method o variational inference. Black Box Variational Inference is a new version of variational inference that dramatically decrease the analytic burden. The main idea behind method is the stochastic optimization of the ELBO by sampling from the variational posterior in order to calculate the noisy gradient. Key to its success is the variance reductions without models to reduce the variation of the noisy gradients. Black Box Variational Inference performs efficient for new models, while its needed negligible work from the practitians. Then, in order to optimize the variational parameter, we are using these stochastic gradients in a stochastic optimization algorithm. We shall see here that central to this idea are stochastic gradient estimators of the **ELBO**. Here we use the black-box variational inference (BBVI) as an umbrella term to refer to the techniques which rely on this idea. The goal in BBVI is to obtain Monte Carlo estimates of the gradient of the **ELBO** and to use it in order to achieve stochastic optimization to fit the variational parameters.
Let consider the ELBO:

$$\nabla_{\phi} L(\phi) = \mathbb{E}_{q(z;\phi)}[f(\mathbf{z})] \tag{1.24}$$

where

$$f(\mathbf{z}) = \nabla_{\phi} \log q(z;\phi)[\log p(\mathbf{x},\mathbf{z}) - \log p(\mathbf{z};\phi)] \tag{1.25}$$

The estimator include a control variate which is a random variable, reduce its variance and maintaining its expectation. A common choice for control variates is the weighted score function. Under the above formulation, the **ELBO** gradient becomes.

$$\nabla_{\phi} L(\phi) = \sum_{n=1}^{N} \mathbb{E}_{q(z;\phi)}[f_n(\mathbf{z}) - \varpi_n h_n(\mathbf{z})] \tag{1.26}$$

where we define

$$h_n(\mathbf{z}) = \nabla_\phi \log q(\mathbf{z}_n;\phi)\varpi_n = \frac{Cov(f_n(\mathbf{z}), h_n(\mathbf{z}))}{Var(h_n(\mathbf{z}))} \tag{1.27}$$

N is the number of samples.

1.4 Reparameterization Trick

The Auto-Encoding Variational Bayes proposed by [23] presents an innovative, differentiable, unbiased, and scalable estimator for the objective function in variational inference. The main idea of this estimator is the reparameterization trick. How they approximate posterior distributions which is used in KL divergence.When a neural network are trained, then inevitably are needed to execute the back propagation method across a stochastic node in the network, because the ELBO which is the objective function. Unfortunately, the back propagation is very difficult to execute across a stochastic node, because we cannot take the derivative of a random variable. Therefore, are used what is called the reparameterization trick in order to restructure the problem. It is worth to emphasize that the reparameterization trick can be implemented to different kinds of distributions and functions, and it is used in a lot of frameworks besides VAEs.

As we previously mentioned we are trying to find the better approximation of the posterior distribution, $p(z|x)$. We consider the $q^\star$ which is the optimal approximation of $p(z|x)$, we minimizing $q^\star$ by a loss function using gradient descent in other words, by using backpropagation to teach a neural network to represent $q^\star$. The problem that arise from this is that we cannot compute the derivative of one particular term in our loss function.The loss function can be written as follows

$$l_i(\theta,\phi) = -\mathbb{E}_{z\sim q(z|x_i)}[\log p_\phi(x_i|z)] + \mathbf{KL}(q(z|x_i||p(z)) \tag{1.28}$$

The term in the loss function is stochastic with respect to random variable z. The back propagation algorithm has a problem due to its difficulty to differentiate the random variable through the node. Take a glance to the left side of the diagram in the Figure below, where the stochastic node is acting as a bottleneck, preventing us from back propagating the derivatives from f to ϕ and x. The reparameterization trick tries to restructure the neural network in order to convey information from f to ϕ and x, as shown on the right side of the Figure 1.5 and allow the back propagation.

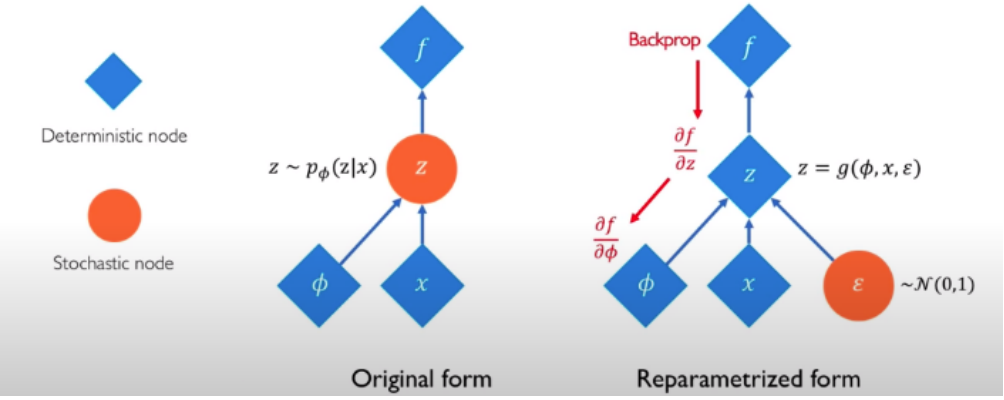


Figure 1.5: Reparameterization Trick

The reparameterization trick converts the random variable z into the following expression

$$z \sim N(\mu, \sigma^2) \longrightarrow z = \mu + \sigma \odot \varepsilon \text{ where } \varepsilon \in N(0,1) \tag{1.29}$$

The reparameterized version of z especially, μ is deterministic so the derivative of z is feasible. The standard deviation is stochastic but without a problem because it is no longer along the critical path in our backpropagation chain.

The method of reparametrization trick generating non-uniform random numbers by converting some base distribution, $p(\varepsilon)$, to a desired distribution, $p(z;\theta)$

$$p(\varepsilon) \longrightarrow g(\varepsilon;\theta) \longrightarrow p(z;\theta) \tag{1.30}$$

1.4.1 Gumbel-Softmax Trick

Reparametrization trick is a key idea which helped us to overcome problems that were previously considered very difficult.Unfortunately when we are dealing with discrete data this trick fails.The Gumbel-softmax trick[24] which essentially is the outcome of the parameterization of categorical distribution which is given with respect to the Gumbel distribution and the corresponding function is discontinuous, it can be made continuous using a continuous approximation dependent on a temperature parameter, which in the case of zero-temperature degenerates into non-continuous, original expression.Below we discuss the aforementioned components. The random variable G follow a standard Gumbel distribution if $G = -\log(-\log(U))$ with $U \sim \text{Uni}(0,1)$. Assumed X as a discrete random variable with $P(X = k) \propto \alpha_k$ and let $\{G_k\}_{k \leq K}$ be a sequence of standard Gumbel random variables.

$$X = \arg\max_k \left(\log \alpha_k + G_k\right). \tag{1.31}$$

In a simple words, a perspiration to sampling from a discrete distribution is to draw Gumbel noise by just transforming uniform samples and add it to $\log \alpha_k l$ which has to be known up to a normalizing constant and take the value k that produces the maximum.

1.4.2 Stochastic Processes

Firstly, the word stochastic comes from the Greek word 'stóhos' which means aim, guess.When the meaning of uncertainty and randomness came to the fore, it was needed to arise a field of science that defines and describes the meaning of uncertainty. When uncertainty was presented in our models the meaning of stochastic modeling have been flourished. In a simple words, it is a model for a process that include randomness. In the real world, uncertainty is observed in all the aspects of our life, so a stochastic model could represent anything. The antonym of stochastic is the deterministic model which predicts only a single outcome from a given set of circumstances. In contrast with a stochastic model which predicts a set of possible outcomes weighted by their probabilities. Deterministic models on every occasion have a set of unchangeable equations that describe the system exactly without changing the result at a time instead of stochastic modeling which generates different results at each time. Modeling a phenomenon as deterministic or stochastic it is purely the observer's choice. The choice depends on the purpose of the observer; the criterion for judging the choice is usefulness.

Definition:Stochastic process is a family of random variables X_θ, where the parameter θ is drawn from an index set θ. For example, let's say the index set is "time".For a continuous process, the random variables are denoted by X_t, and for a discrete process they are denoted by X_n. "Time" is one of the most common index sets another is vectors, represented by X_u, v, where u,v is the position.

Stochastic processes are discrete by the random variables X by their index set T, the range of possible values, and by the dependence relations among the random variables X. Gaussian Process, Dirichlet process, etc is some of the most widely used categories of stochastic processes. The use of these processes as models will explain in detail below.

1.5 Bayesian non parametric methods

A nonparametric Bayesian model is a model that the parameter space has infinite dimensions, in order to determine a nonparametric Bayesian model, is necessary to define first the prior

probability distribution on the space. A distribution on an infinite-dimensional space T is a stochastic process with paths in T. Such distributions are very difficult to define it than distributions on $\mathbb{R}^d$, but we have the ability to draw on a large arsenal of tools from stochastic process theory and applied probability. Non parametric Bayesian methods have seen rapid growth over the past 25 years. It is essentially to assume theoretically ,flexible models that are not limited to finite parameterizations. Bayesian non parametric models in a simple words are Bayesian models defined on infinite dimensional parameter space. The parameter space depends from the given problem in each case.

For instance in a regression task the parameter space is determined by continuous functions also in a density estimation problem the space can contains all the densities. Bayesian non parametric models handle just a finite subset of parameter dimensions in order to explain a finite sample of observations, with the set of dimensions selected depending on the sample, such that complexity of the model to be effective. In the below sections, we will mention and analyze some of the most important categories of the Bayesian non-parametric methods such as Gaussian process[25], Dirichlet process[26], its hierarchical extension, and lastly the Beta process. Bayesian nonparametric models have been applied to a numerous of machine learning problems such as image segmentation,classification, clustering, regression, latent variable modeling,sequential modeling, source separation and grammar induction.

1.5.1 Gaussian Process

Stochastic modelling is an interesting and challenging area of probability and statistics which in 1996 Rasmussen inspired by Neal's [27] in order to create a bridge between stochastic modelling and machine learning. The word "Gaussian" of its name symbolize that GP uses Gaussian distribution (or normal distribution) to characterize the random process. In order to characterize a single random variable with a Gaussian distribution, a mean value and a variance value are all we need. Therefore, to define a random process that comprises an infinite number of random variables is needed to upgrade the Gaussian distribution to a Gaussian random process. The word "Process" which is the second part of its name refers to the fact that GP is a random process. In simple words, a random process is a function f(·) that have the ability at any location x, f(x) will be random variable also in different locations x_i and x_j, the random variables $f(x_i)$ and $f(x_j)$ are correlated. Thirdly correlation strength between $f(x_i)$ and $f(x_j)$ depends on the distance between x_i and x_j. In general, as x_j moves away from x_i, their correlation strength decays. The main advantage is that we can interpret a random

process as an infinite collection of correlated random variables. Thus, introduced Gaussian process models which is a stochastic process with general purpose to bring together the two communities by construct GP's from neural networks with an infinite number of hidden units and there is an direct way of placing a prior over the underlying function f.

Let $\mathbf{f} = (f(x_1), ..., f(x_n))$ be an n-dimensional vector of function values evaluated at n points $x_i \in X$ and f is a random variable.

Definition: $p(\mathbf{f})$ is a Gaussian process if for any finite subset $x_1, ..., x_n \subset X$, the marginal distribution over that finite subset $p(\mathbf{f})$ has a multivariate Gaussian distribution.
Gaussian processes (GPs) can be characterized by two parameters a mean function, $\mu(x)$, and a covariance function, or kernel, $K(x, x_0)$[28][29].
where

$$f \sim \mathscr{GP}(\mu(\mathbf{x}), k_f(\mathbf{x}, \mathbf{x}^{'}))\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{1.32}$$

$$k_f(\mathbf{x}, \mathbf{x}^{'}) = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}^{'}) - \mu(\mathbf{x}^{'}))] \tag{1.33}$$

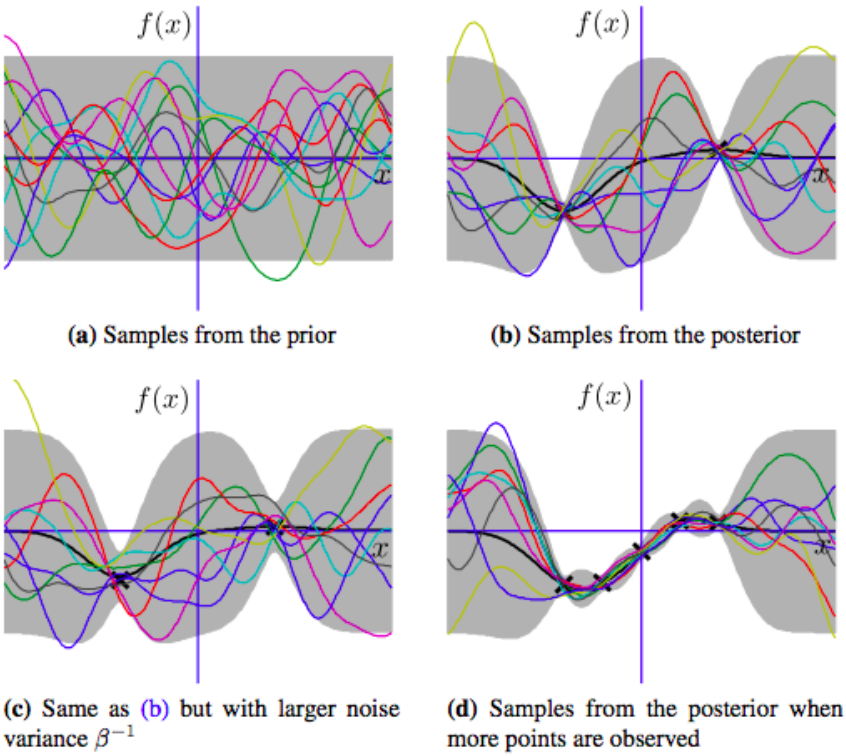
The Gaussian process has as output a normal distribution, which can be expressed in terms of mean which represent the most likely output and variance which illustrate the measure of the confidence of our model. Gaussian process have this powerful capacity to catch most of relations between inputs and outputs by using an infinite number of parameters and letting the complexity to determine through the means of Bayesian inference. In a simple words Gaussian process defines a distribution over functions p(f) which can be used to compute the posterior distribution in order to confront problems like Bayesian regression or classification[30]. Given the data $D = X, y$ and hyper-parameters $\theta(e.g., \theta = (\lambda, \sigma^2, f, \sigma^2))$, the log marginal likelihood[31] is

$$\log p(\mathbf{y}|\mathbf{x}, \theta) = \frac{-1}{2} y^T K^{-1} \mathbf{y} - \frac{1}{2} \log |k| - \frac{n}{2} \log 2\pi \tag{1.34}$$

The marginal log likelihood can be considered as a complexity penalty measure and is the combination of a data fit term. In the first term we have how well the current kernel parametrization explains the dependent variable the second term is a measure of complexity penalization and the finally the third term is a normalization constant.

The marginal likelihood is usually maximized through an optimization tool such as implemented in Carl Rasmussens MATLAB function minimize. These procedures using the partial derivatives with respect to θ:

$$\frac{\partial}{\partial \theta_i} \log p(\mathbf{y}|\mathbf{X},\theta) = \frac{1}{2}\mathbf{y}^T K^{-1} \frac{\partial K}{\partial \theta_j} K^{-1}\mathbf{y} - \frac{1}{2} tr(K^{-}1 \frac{\partial K}{\partial \theta_j}) \tag{1.35}$$



(a) Samples from the prior

(b) Samples from the posterior

(c) Same as (b) but with larger noise variance β^{-1}

(d) Samples from the posterior when more points are observed

Figure 1.6: A GP fitting the observed data. The black curve represent the GP mean.Standard deviations are illustrated as gray hue. The GP samples are depicted with different kind of colors. Note that even when the mean return to zero far from observations, the samples still fluctuate.

Undoubtedly, the biggest disadvantage of Gaussian process it is the disability to scale up well. Through the inferring of posterior is required to inverting the matrix $[K(X_t,X_t)+\sigma^2\mathbf{I}]$ which is computationally costly and intractable because inference scales cubic-ally and depends from the number of observations due to this fact increasing the speed of inference for Gaussian process regression is an issue of ongoing research. This means that in practice it is very difficult to work with more than a few thousand points. For this reason have been proposed sparse GP techniques[32] that can be used to greatly reduce computation time or to bound the computational cost of the matrix inversion by projecting into a predefined finite basis of functions drawn from the eigen-spectrum of the kernel [33]. In conclusion, the results show that Gaussian process regression is specifically beneficial when the underlying function is unknown.General purpose to bring together the two communities by construct GP's from

neural networks with an infinite number of hidden units.

1.5.2 Dirichlet Process

As we previously mentioned in the Gaussian process also Dirichlet Process (DP)[34] is a distribution over distributions. The Dirichlet process belong to the category of the stochastic processing with the characteristic that all samples if we sum it give us probability one. Stochastic processes are distributions over function spaces, with sample paths being random functions drawn from the distribution. DP gives us the ability to take advantage of its properties, which allow them to be interpreted as distributions over some probability space Θ it is a distribution over probability measures. Thus samples from a DP can be considered as random distributions. For distribution over probability measures to be a DP, its marginal distributions have to take on a specific form which we shall give below. This theory is built up at the measure theory and Dirichlet distributions, thus prerequisite knowledge is needed. Before we define it mathematically, we will try to give the intuition of the DP as an infinite-dimensional generalization of Dirichlet distributions. Assume a Bayesian mixture model consisting of K components:

$$\pi|\alpha \sim Dir(\frac{\alpha}{K},...,\frac{\alpha}{K}) \quad \theta_K^\star|H \sim H \quad z_i|\pi \sim Mult(\pi) \quad x_i|z,\theta_K^\star \sim F(\theta_{z_i}^\star) \tag{1.36}$$

where α is the pseudocount hyperparameter of the Dirichlet prior,π is the mixing proportion, H is the prior distribution over component parameters $\theta_k^\star$, and F(θ) is the component distribution parametrized by θ. There is a special way to parametrize Dirichlet prior over π and is obvious when the K increasing. We assume that the distribution G is distributed according to a DP, its marginal distributions have to be Dirichlet distributed [35]. Especially, let H be a distribution over θ and α be a positive real number. The vector $(G(A_1),...,G(A_r))$ is random due G is random and A_1,...,A_r is finite partition of θ, G is Dirichlet process with concentration parameter α, and base distribution H written G $\sim$ DP(α, H), if:

$$(G(A_1),...,G(A_r)) \sim Dir(\alpha H(A_1),...,\alpha H(A_r)) \tag{1.37}$$

for every finite partition A_1, ...,A_r of Θ. The parameters α and H have a significant role in the determination of the DP. The base distribution essentially is the mean of the DP: for any measurable set A $\subset \Theta$, we have E[G(A)] = H(A). From the other side, the concentration parameter can be understood as an inverse variance

$$V[G(A)] = \frac{H(A)(1-H(A))}{(\alpha+1)}. \tag{1.38}$$

1.5.3 Indian Buffet Process

The Indian Buffet Process (IBP) is a distribution that imposes to binary matrices which consisting by unbounded number of columns and N rows[36] which each column corresponds to a feature and each row corresponds to an object it is something which is very close to Chinese Restaurant Process (CRP) [37]. The IBP gives an effective way to determine non-parametric Bayesian models with latent variables. The IBP also allows each object to possess potentially any combination of infinitely many latent features. This setting has resulted to provide unprecedented flexibility in the model and assisting the development of a range of interesting applications such as choice behaviour, human similarity judgments, protein-protein interactions etc.

The IBP is defined as the limit of a corresponding distribution over matrices with K columns, as the number of columns $K \to \infty$. Let Z be a random binary $N \times K$ matrix, and denote entry (i, k) in Z by z_{ik}. For each feature k let μ_k be the prior probability that feature k is present in an object. We place a $Beta(\alpha_K, 1)$ prior on μ_k, with α being the strength parameter of the IBP. The full model is:

$$\mu_k \sim Beta(\frac{\alpha}{k}, 1) \; \forall \; k \tag{1.39}$$

$$z_{ik}|\mu_k \sim Bernoulli(\mu_k) \; \forall \; i,k \tag{1.40}$$

A different definition of the IBP where the feature probabilities are not integrated out and a specific ordering is imposed on the features is known as the stick-breaking construction for the IBP which fit perfectly with variational inference. This alternative representation gives the ability to ensure sparsity in the obtained representations, while at the same time allowing the emerge of components as new data appear. These representations help us to use new MCMC samplers are that are much easier in the implementation on general models than Gibbs sampling.

Let us assume observations, and a binary matrix $\boldsymbol{Z} = [z_{i,k}]_{i,k=1}^{N,K}$, K_i=1 which each entry therein, indicates the existence of feature k in observations. Taking the $K \to \infty$, we can build the following hierarchical representation:

$$u_k \sim Beta(\alpha, 1) \quad \pi_k = \prod_{i=1}^{k} u_i \quad z_{i,k} = Bernoulli(\pi) \tag{1.41}$$

where parameter $\alpha \geq 0$, controlling the sparsity and set k to be equal to the input dimension to avoid over-representation of features when to the input dimensionality to avoid the over-complete feature representation when $\lim K \to \infty$

Properties of Indian Buffet Process

Some properties of the binary matrix which arise from Indian Buffet Process:

1)The number of features chosen can arbitrary grows depending the data points.

2)The propensity to add new features decreases with data point as new features are added according to $Poisson(\alpha/n)$. As n increases, the Poisson distribution rate decreases.

3)From the other side, the old features get reused based on the principle "rich get richer" .

4)The non-empty columns are distributed as $Poisson(\alpha H_n)$ where $H_n = \sum 1/n$

Chapter 2

Bayesian Deep Nets

2.1 Neural Networks

The first attempts at artificial intelligence were made after the Second World War and were concerned with the imitation of the nervous system, ie they were neuromorphic. The first networks were called perceptrons, and the mathematical theory was first formulated by Mackalov in 1948. The first applications were made in 1950 at the US Air Force Laboratory at Cornell University. In 1968, Minsky wrote a scientific paper dealing with the limits of the first provisions of these neural networks, that is, studying what they could achieve. They discovered that in order to be able to do something useful and high performance it was necessary to add more levels, so we came to what is called deep neural networks whose definition refers directly to their structure.

The architecture of Artificial neural networks is consist of nodes and layers. Each artificial neuron or node connects to another and has a related weight.

The first neural network as we mentioned above is the Multi-Layer Perceptron (MLP) network, MLP was the cornerstone of NN and also for modern architectures. Below depicted a simple MLP with just one hidden layer which is suitable for classification and regression.

This neural network with dimension N_1 and input x, the output can be modeled as

$$\Phi_j = \sum_{i=1}^{N_1} \alpha(x_i w_{ij}^1) \quad f_\kappa = \sum_{j=1}^{N_2} g(\phi_j w_{jk}^2) \tag{2.1}$$

The parameter w means the connection between neurons f. Equation 1 represents the output of the hidden layer, which will be dimension N_2 also 1 represents the state of each node in

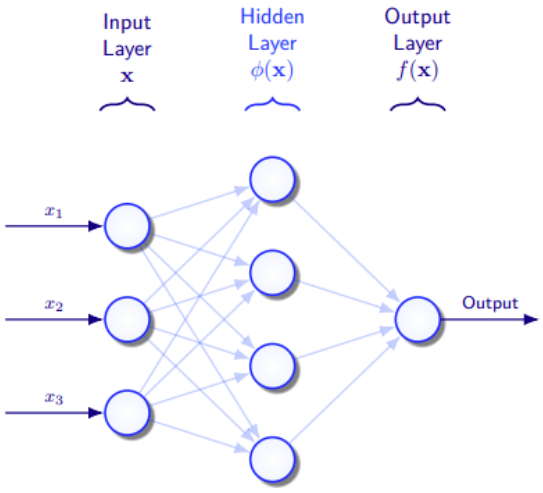


Figure 2.1: Neural Network

the hidden layer,equation 2 is a summation over the N_2 outputs from the prior hidden layer this is expressed as a transform followed by a non-linear transform $\phi(\cdot)$, which is called an activation function. For the primitive perceptron the activation function that was used for the first time was the $sign(\cdot)$ function although the utility of this function has stopped due to the fact that the derivatives are equal to zero so it is of vital importance to use more flexible activation functions such as the Sigmoid,Rectified Linear Unit (ReLU), Hyperbolic Tangent ,and Leaky-ReLU. When Sigmoid function is used expression 1 is equal with the logistic regression which means that the output of the network becomes the sum of multiple logistic regression models however when we are using regression model, the function that applied to the output $g(\cdot)$ will be the identity function,and for binary classification will be the Sigmoid.

The aforementioned Equations can be implemented efficiently by using matrix representations and very often is represented as such literature. This is managed by stacking the input vector in our data set as a column in X. Therefore propagation can be performed as

$$\Phi = \alpha(\mathbf{X^T W^1}) \tag{2.2}$$

$$F = g(\Phi \mathbf{W^2}) \tag{2.3}$$

While this matrix notation is more comprehensive, the choice to use the summation notation to describe the network here is deliberate. In the framework of frequentist NN learning, a MAP or MLE estimate is found through the minimization of a non-convex cost function J(x,y). Minimization of the function is performed through back-propagation, where the output of the

model is computed from partial derivatives for the parameters

This formula sometimes is very slow due to the fact that it need to read the whole dataset for each update. Simultaneously, due to the fact that all the dataset is needed to be stored, memory issues may occur. The training must be started from the beginning when new training examples appear, the training must be started from the beginning.

$$\theta_{t+i} = \theta_t - \alpha \nabla_\theta J(\theta) \tag{2.4}$$

The stochastic gradient descent (SGD)[38], is designed in order to overcome the aforementioned problems and to execute an update at each parameter for all training example of the dataset. The stochasticity is due to the fact that each example gives an estimation with noise as regards the average gradient for all examples. By assuming the input of the example as x and the output as y,the parameter update is

$$\theta_{t+i} = \theta_t - \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)}) \tag{2.5}$$

The expression above illustrate how the equation updates the weights through back propagate with α to represent the learning rate and the subscripts indicate the iteration in the training procedure. We are using the chain rule in order to find the derivatives of the parameters of the network. This leads to the preference of the ReLU activation function which is discontinuous and non-linear and so prevents the gradients from vanishing during training.

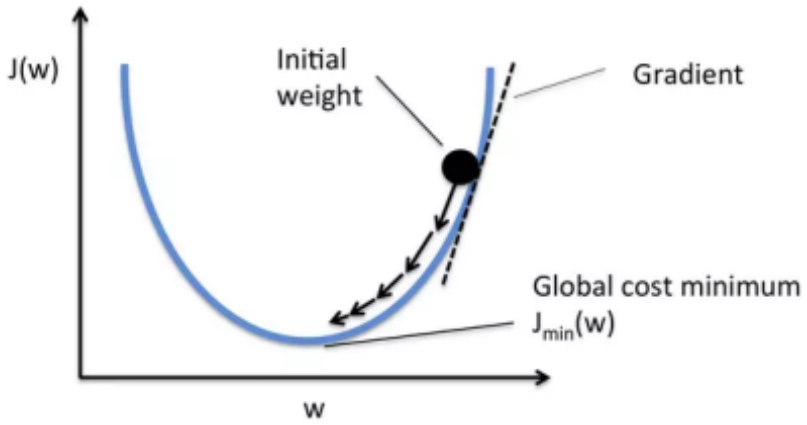


Figure 2.2: Gradient Descent Optimization

We assume three functions $f^{(1)}, f^{(2)}, and f^{(3)}$ in a chain, to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$[39]. The kind of this chain are the most commonly used structures of neural networks. In this

case, $f^{(1)}$is the first layer of, $f^{(2)}$is called the second layer, and so on. The length of the chain function gives the depth of the model, and based on that comes to the fore the terminology of deep learning.

Let discuss firstly the linear models, for example, the linear regression and logistic regression, are overused because they can be fit reliably and efficiently, either in convex optimization or closed-form. Unfortunately, linear models have obvious disadvantages and can not interact and model, models with more than two variables for this reason their ability is limited.

2.1.1 Overfitting

Overfitting occurs when our algorithm is modeling low-level patterns in the training set, in such a way that leads to incapability to generalize on unseen data and our test set is low performance. Learning details and unwanted information about the dataset cause noise in the training set to the degree that it negatively impacts the execution of the model on testing data. The model learns the noise in the training data is learned as concepts that are included in the model.

Overfitting is more common phenomenon in nonlinear and non-parametric models that is more flexible when learning a target function. However, a lot of non-parametric machine learning algorithms also include specific techniques in order to constrain the amount of the detail which model learns.

In contrast we have also the underfitting which refers to a model that is very poor as concern the performance so if we want to training the data or to generalize to new data this is very difficult. In a simple words:

• Overfitting: High performance on the training data, poor performance when generalize to new data.
• Underfitting: Low performance on the training data and also poor performance when generalize.

Due to these problems have been arisen methods in order to overcome it. The ultimate purpose of these methods is to prevent the models from overfitting but also be computationally efficient. Methods like early stopping, learning rate annealing and L_1, L_2, soft weight sharing[40], Dropout and DropConnect are some of the most importance. Figure 2.3

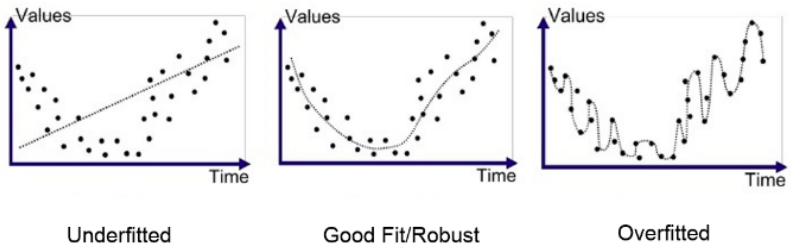


Figure 2.3: An illustration of how the data is modeled in the case of underfitting,well fitting and overfitting

Regularization Techniques

The meaning of 'regularization' refers to group of techniques that regularizes learning from specific features for conventional algorithms or neurons in the case of neural network algorithms.

It normalizes and moderates every weight that is linked to a neuron so that algorithms do not depend only on a few features or neurons to predict the result, we try to boost this technique in order to avoid the problem of overfitting.

In order to understand regularization is necessary to consider a simple case of linear regression. Mathematically, linear regression is stated as below: where y is the value to be predicted;

$$y = w_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n \tag{2.6}$$

$x_1, x_2, ..., x_n$ are features that decides the value of y; w_0 is the bias $w_1, w_2, ..., w_n$ are the weights attached to $x_1, x_2, ..., x_n$ relatively.

In order to train a regression model that predict with accurate the y value is needed the optimization of bias and weights.

To achieve this is needed to use an objective function and find the parameters by using the optimization method gradient descent algorithm.

$$RSS = \sum_{j=1}^{m} (y_i - w_0 - \sum_{i=1}^{n} w_i x_{ji})^2 \tag{2.7}$$

If the dataset has noise, it will tackle the aforementioned problem of overfitting, and learned parameters will not generalize well on unseen data. Avoided this by regularizing weights for better learning. The main three regularization techniques, namely Ridge Regression

(L2 Norm) Lasso (L1 Norm), and Dropout. L2 and L1 can be applied for any algorithms that involve weight parameters. Dropout is widely used in any neural network such as recurrent, convolutional in order to moderate the learning. Below we will take a glance at these techniques.

Ridge regression

Ridge regression is also called L2 norm or regularization.

$$Loss = \sum_{j=1}^{m} (y_i - w_0 - \sum_{i=1}^{n} w_i x_{ji})^2 + \lambda \sum_{i=1}^{n} w_i^2 \tag{2.8}$$

This technique presupposes to add the term of weight's square to the loss function and thus a new loss function generated which is denoted.

As we observed above, the original loss function is reformed by adding normalized weights here the weights are in the square form.

The parameter λ will be tuned by using a cross-validation dataset when $\lambda = 0$, it returns the residual sum of square as loss function which you chose initially. When the parameter λ is too high, the objective function will ignore the core loss function and will reduce the weight's square and will end up taking the value of the parameter as zero.

Using the aforementioned objective function we push the parameters to a learning procedure in order to minimize the loss function is of vital importance to push the parameters to be as small as possible. Thus, L2 norm resist weights from rising too high.

Lasso Regression

Also called lasso regression and denoted as below:

$$Loss = \sum_{j=1}^{m} (y_i - w_0 - \sum_{i=1}^{n} w_i x_{ji})^2 + \lambda \sum_{i=1}^{n} |w_i| \tag{2.9}$$

Lasso regression differs greatly from the ridge reggresion due to the utility of the absolute weight values for normalization instead from square weights. The lambda parameter behaves exactly the same way as behave in ridge regression what is needed is just the tuning.

Optimization algorithms penalize higher weight values as objective function only considers absolute weights.

Loss function in lasso regression along with the optimization algorithm pushing parameters very close to zero but not actually in zero, from the other side lasso eliminates the less important features and setting the respective weight values to zero. Thus, lasso also performs feature selection along with regularization.

Dropout

Another regularization technique which is used with the main purpose to dealt with the problem of overfitting, what it really doing is to randomly dropping different layer outputs at each iteration when this procedure applied, the output of the network layers is given by

$$r = m * \alpha(Wu) \tag{2.10}$$

where m is a binary mask and $*$ depicts element-wise product.

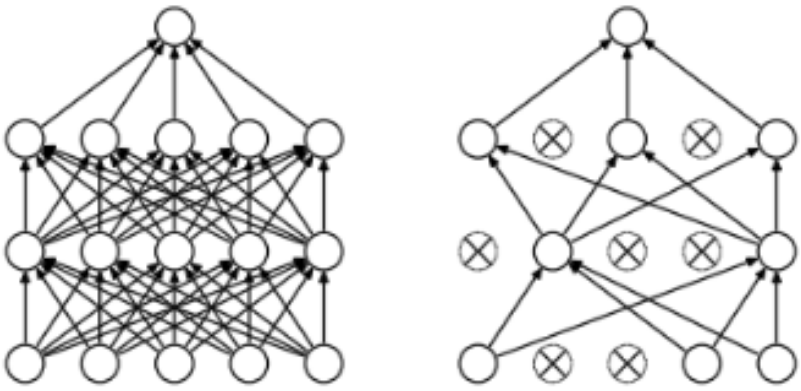


Figure 2.4: a)Two hidden layers Neural Network b)Dropout method

The mask is drawn independently from $m_j \sim Bernoulli(p)$ where p is the probability to keep the element and j is the element of the layer's.

Noise Injection

In several cases, we are using noise injection so, in that way we form a sort of data augmentation [41]. Also if inject random noise to the inputs layers and also to the hidden layers of the model during the training we achieve a significant increase of robustness [42]. Moreover, noise can be injected into the weights as proposed by Jim et al. (1996) [43] and Graves et. al (2011)[44]. Under the Bayesian approximation,the weights of the models are governed by

an amount of uncertainty and are represented via a probability distribution that models this uncertainty. The noise injection to the weights can be considered as a stochastic and practical way to reflect this uncertainty. The output units of the model could also be injected with noise which is known as label smoothing [45].

2.2 Deep Learning

Deep learning caused a revolutionary state in the field of machine learning due to its ability to handle huge number of layers and much better software tools, providing solutions to overcome problems. Deep learning is a part of machine learning. You can imagine it as an extension of machine learning[46].

Deep learning constitutes a new approach for data learning.Uses a specific family of models sequences of simple functions interconnected. These function chains are the neural networks. These sequences of functions can analyze a complex idea in a hierarchy of simpler ones. Each layer organizes the previous layer into more advanced and abstract concepts.

Deep neural networks map input-output (target) mapping through a deep sequence of simple data transformations, and these data transformations are learned through exposure to examples.

More specifically, the specifications of what a layer does in its input data are stored in its weights which form a set of numbers. Learning means finding the values for the weights of all the layers in a network so that it correctly matches the inputs to the outputs targets. It is a particularly difficult task, as modifying the value of one parameter affects the behavior of others.

In order for a neural network to control output it must be able to measure the deviation of the value of each output from the expected one.This process is performed from the loss function of the network.This function calculates the deviation between the predictions made by the network, and true targets, recording network performance.

How successfully the network worked or not, as it resulted from the value of the loss function, is used as feedback to change the values of the weights to reduce the loss score. This weight modification is performed by the optimizer, which essentially executes the algorithm backpropagation.

As the weights have been given random values the network implements a series of random transformations and the degree of loss is very high.With each example, the network adjusts the

weights better and reduces the degree of loss until a network with minimal loss occurs. The process of deep learning includes two phases: training and inference. Deep neural networks learn new skills during the training phase from existing data, and these skills are applied to unknown data in the inference phase.

Deep learning models are intended to find patterns among data with a logical structure in order to approach the way that people make decisions and draw conclusions so in that way to construct algorithms that can learn and make decisions on their own. This design makes deep learning models more capable than standard machine learning models. Oftentimes, deep learning algorithms require very large datasets in order to be successful, but once they have data, they can produce immediate results. An important advancement in the field of deep learning is called transfer learning, which involves the use of pre-trained models. These pre-trained models contribute to the need for algorithms to train large datasets. We mention a few examples of deep learning algorithms. **Convolutional neural networks**[47] is a type of neural network that has multiple layers. These layers extract features from data and analyze them. CNN's are mainly used for image processing, computer vision, and object detection. **Recurrent neural networks** [48] are using for sequential data and time-series data. They are well known for their "memory" as they take information from the previous layer to influence the next layer and eventually the output. Google Translate, image captioning, and Siri are some of the problems that RNN's are mainly used. **Autoencoders**[49] use neural networks, especially for representation learning. are used to mainly solve unsupervised learning problems and they replicate data from the input layer to the output layer. They are used for things such as pharmaceutical research and image processing.

In conclusion, the new era of Deep Learning has begun in 2006 by designing more complex architectures such as [50][51][52]. After an innovative breakthrough in speech and natural language processing in 2011, and also in image classification through the scientific competition ILSVRC in 2012 [53] so deep Learning won challenges beyond their conventional applications field.

However, deep learning models are tending to overfitting, which affects their generalization abilities[54]. When they provide a confidence interval they are prone to be overconfident about their predictions, also are very data-hungry, face difficulties in modeling uncertainty, and particularly prone as regards adversarial examples. This is a problem for applications such us medical diagnosis self driving cars or finance. Therefore, many approaches have been proposed to moderate this danger. Among them,the Bayesian neural networks which provides a framework to analyze and train uncertainty-aware neural networks, and more generally, to

support the development of learning algorithms.

2.3 Bayesian Neural Networks

The definition of Bayesian Neural networks(BNN) is slightly different across the machine learning literature, but a common undeniable definition is that a BNN is a stochastic artificial neural network that is trained using Bayesian inference[55].

Conventional ANNs such as convolutional, feedforward, recurrent networks are created by using a sequence of one input layer, a number of hidden layers, and one output layer. In the simplest architecture of feedforward networks, each layer l is a linear transformation, followed by a nonlinear operation s, which is known as activation function:

$$\mathbf{l_0} = \mathbf{x} \tag{2.11}$$

$$\mathbf{l_i} = s_i(\mathbf{W}_i\mathbf{l}_{i-1} + \mathbf{b}_i) \tag{2.12}$$

$$\mathbf{y} = \mathbf{l}_n \tag{2.13}$$

An ANN architecture represents a set of isomorphic functions to the set of $\theta = (\mathbf{W}, \mathbf{b})$ which is the parameters where W represents the weights of the network and b the biases.Deep learning is the procedure of regressing the parameters θ from the training data D. A usual approach is the minimum point estimation of the parameters $\hat{\theta}$, an individual value for each parameter by using the back-propagation algorithm. The log-likelihood is often defined as the cost function of the training set, often including the regularization term.This is known as Maximum A Posteriori (MAP) or Maximum Likelihood Estimation (MLE), when regularization is used.

The approximation of point estimate, which is a long-establish approach in deep learning, is relatively easy to use with algorithms and software packages but lags behind a lack of explainability. There is a possibility the final algorithm to generalize in overconfident and unexpected ways on data points out of training distribution [56][57]. The weakness of ANNs to say “I don’t know” is a problem for many applications. Of all the techniques that exist [58], stochastic neural networks have shown to be the most flexible and effective.

Stochastic neural networks are sub-categories of ANN which created by introducing stochasticity components into the network. This is achieved by adding to the network either a stochastic activation or stochastic weight in order to simulate multiple possible models θ with their

associated prior distribution $p(\theta)$. In this way, they can be considered as a particular case of ensemble learning.

The main incentive that ensemble learning relies on the observation that aggregating the predictions of a large set of average-performing but independent predictors can lead to better predictions than a single well-performing expert predictor. Stochastic neural networks could improve their accuracy in relation to corresponding point-estimate in a similar way, but this is not their purpose.

Stochastic neural network architecture has the main purpose of helping to estimate the quantity of uncertainty through modeling. Comparing the predictions of multiple sampled model parametrizations θ when different kinds of models agree, then the uncertainty is low if they disagree, then the uncertainty is high. This procedure could be expressed as follows:

$$\theta \sim p(\theta) \tag{2.14}$$

$$\mathbf{y} = \Phi_\theta(\mathbf{x}) + \varepsilon \tag{2.15}$$

where ε is the random noise due to the fact that the Φ is just an approximation.

Bayesian Neural Network is a stochastic artificial neural network similar to the ones existed in 1980 and 1990 trained by recruiting Bayesian inference.The ultimate aim of Artificial Neural Networks is to represent a function $y = \Phi(x)$. Bayesian statistics gives us a clear framework to understand and quantify the uncertainty which related with deep neural network predictions. The Bayesian paradigm in contrasts with the frequentist paradigm has an area of distinction which is hypothesis testing.

We considered as a first BNN was mentioned in [55]. This paper highlights the statistical properties of NNs by introducing the statistical interpretation of loss functions that are used. Proved that the procedure of minimizing the squared error term is similar to the procedure of finding the Maximum Likelihood Estimate (MLE) of a Gaussian. Was shown that by imposing a prior over the weights of the network, Bayes Theorem can be used in order to obtain an appropriate posterior.

Bayesian analysis has been proved that provide enough theory in order to address challenges such as overconfident predictions from typical NNs. In order to yield accurate approximations for the posterior, we must based on the approximate inference.

In order to create a BNN [59], the first step is to select the architecture of the neural network.

The second step is to select the stochastic model, before we observed the data we impose a prior distribution over the parameters which might have some hyperparameters, $p(\theta)$ is the prior confidence in the predictive power of the model $p(y|x,\theta)$. The selection of a BNN stochastic model is in some way equivalent to the selection of a loss function when training a neural network with point estimation. When we used BNN in order to predict, the distribution which is really interesting is the marginal probability distribution $p(y|x,D)$[60], which achieves to quantifies the model's uncertainty on its prediction.

$$p(y|x,D) = \int_{\theta} p(y|x,\theta')p(\theta'|D)d\theta' \tag{2.16}$$

The final prediction is summarized by a few statistics computed using a Monte-Carlo approach. Algorithm 1 summarizes the inference process for a BNN.

Algorithm 1 Inference procedure for a BNN

Define $p(\boldsymbol{\theta}|D) = \dfrac{p(D_{\boldsymbol{y}}|D_{\boldsymbol{x}},\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(D_{\boldsymbol{y}}|D_{\boldsymbol{x}},\boldsymbol{\theta'})p(\boldsymbol{\theta'})d\boldsymbol{\theta'}}$

for $i = 0$ **to** N **do**

  Draw $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|D)$

  $\boldsymbol{y}_i = \Phi_{\boldsymbol{\theta}}(\boldsymbol{x})$

end for

return $Y = \{\boldsymbol{y}_i | i \in [0,N)\},\ \Theta = \{\boldsymbol{\theta}_i | i \in [0,N)\}$

Figure 2.5: How the inference process can be implemented step by step

The $p(\theta|D)$ which is the posterior distribution, for complex models such as artificial neural networks is a high dimensional and highly non-convex probability distribution[61].

Below we will mention some practical advantages of using BNNs

· Firstly, BNN gives an inherent way to approximate the quantify the amount of uncertainty in deep learning since BNNs have better calibration than classical neural networks, their uncertainty is more reliable with the observed errors.

· Secondly, a BNN achieve to distinguishing the epistemic uncertainty p(θ| D) and the aleatoric uncertainty $p(y|x,\theta)$ [62][63]. This makes BNNs very efficient since they can learn from a relatively small dataset without the problem of overfitting.

· Thirdly, the Bayesian paradigm enables the analysis of learning methods. A number of methods that initially do not seem as Bayesian and can be understood as being approximate Bayesian, e.g., regularization or ensembling .

BNNs have found fertile ground in many different fields to quantify uncertainty such as network traffic monitoring, in computer vision, civil engineering,aviation,astronomy, electronics, and medicine. BNNs are also useful in active learning.

2.4 Adversarial Robustness

Despite their importance and great success, DNNs have some weaknesses. In particular, it has been observed that small perturbations in a DNN inputs may disorient the model and lead to a completely different result than expected.

The first report of the problem for DNN classification models was made in[54], where it was confirmed that with the introduction of carefully constructed noise in the original training images, adversarial examples emerge, which the classifier categorizes wrong. The noise input process for the purpose of constructing adversarial examples and consequently leading the DNN's to an intentional error is referred to as adversarial attack.

Example in Figure **2.5** while the initial classifier prediction is correct, the noise input in the original image produces the adversarial example which leads the classifier to an erroneous prediction with a high degree of confidence.

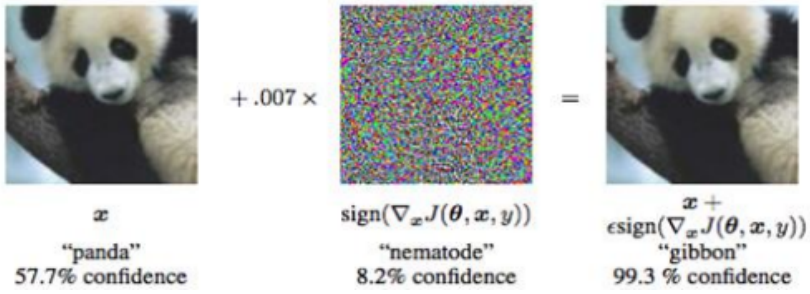


Figure 2.6: Noise input for Adversarial Example production

In addition, an important feature of adversarial examples is their similarity to the original sample, which suggests that the problem is mainly located in the inability of DNNs to "learn" the requested representations.

The existence of adversarial examples has been a major research problem in recent years, as more and more systems use Machine Learning algorithms, so they are vulnerable to such variants. Thus, methods are researched and developed to increase it robustness of Machine Learning models, methods to increase the resilience of models to competing examples. These

methods protect the classification models, so they are referred in the literature as adversarial defenses.

Essentially, through the specific methods it is sought to change the decision boundary of the model, so that it is closer to the real one while at the same time leaving less space available for the emergence of adversarial examples.

Adversarial Training is a technique for creating robust classifiers in which training in adversarial examples is performed. Its first successful application Adversarial Training in DNN appears in [64]. More specifically, training is carried out in data sets that contain authentic samples and adversarial examples in a ratio of 70% in favor of adversarial examples.

The simplest method for training model robustness against adversarially exambles is the one that seems most intuitive. Adversarial training is the basic idea in the machine learning literature is the basic technique for robust optimization. Is trying is to include and create adversarial examples in the training process. Since we know that the conventional way of training creates networks that are sensitive to adversarial examples what is trying to do is just train networks on a few adversarial examples.

The reasonable question that arises is which adversarial examples we should train.In order to get at an answer for this question. assume we generally want to optimize the min-max objective:

$$minimize_{\theta} \frac{1}{|S|} \sum_{x,y \in S} maxl(h_{\theta}(x+\delta), y) \tag{2.17}$$

2.4.1 White-Box Attacks

White-box attacks are the most popular category of attacks, in which the prospective attacker has access to all DNN information.Very often in the literature, these types of attacks are referred to as gradient-based attacks, since the most basic information in order to be able to apply them is the gradient of the cost function that controls the performance of the classification model under consideration.

Fast Gradient Sign Method

Fast Gradient Sign Method(FGSM)[65] is the first successful attempt to deal with the issue of adversarial examples in DNN. The results of this research are a consequence of the hypotheses

that were made and proved through tests. More specifically, the researchers claim that:

· All models, regardless of architecture, are sensitive to adversarial examples.

· The sensitivity of the Neural Networks is due to their strongly linear behavior in high dimensional spaces.

· Model training in adversarial examples is an additional effective means (regularization) in addition to other techniques such as Dropout, model averaging and prepossessing of training data.

The application of the method can be described by the following equation, in which the perturbation is equal to:

$$\eta = \varepsilon sign(\nabla_x J(\theta, x, y)) \tag{2.18}$$

where x is the image(input), θ is the parameters of the model, y is the output and finally J(·) is the cost function.

FGSM's goal was to create a method, in which adversarial examples are produced very quickly and not the production of the most efficient adversarial examples.This method possibly increased the model error to high levels. Finally, in terms of adversarial training, this has proven to be a successful method of increasing model performance in predicting adversarial examples. Adversarial training is performed using the following objective function:

$$J(\theta, x, y) = \alpha J(\theta, x, y) + (1 - a)(J(\theta, x + \varepsilon sign(\nabla_x J(\theta, x, y)) \tag{2.19}$$

where α is a hyperparameter that takes the value 0,5[39].

2.4.2 Black-Box Attacks

The research community dealing with the problem of adversarial examples, started with the hypothesis that the potential attacker has full access to the DNN that examined (white-box setting). However, in many cases this condition is not met as such access is usually limited. For this reason in recent years the interest has been concentrated around black-box setting 50 methods, in which the attacker has access only to the input and output (labels) or logits (the last layer of the network from which the vector is derived cumulative probability (softmax) for the classes examined before the final forecast). One of the first attempts in this direction was

[66], during which they are made white-box attacks on a substitute model, different from the candidate to be attacked, and transferred to the latter in a black-box[67] setting.

Gaussian Data Augmentation

Gaussian Data Augmentation(GDA)[68] is a defense technique in which Gaussian noise is added to the original dataset on which the classification model will be trained. This can be achieved in two ways: **1)** By applying the noise during the model training, as a result of which some augmentation examples are added to the original dataset so that its size can be expanded and **2)** By applying it noise during its forecasting process of the model, so the size of the original dataset remains the same but consists of noisy examples.

Regarding the methods which used to measure GDA performance,the following three are used:

· Robustness as defined in [69] · measuring the distance between original and adversarial training sets · the calculation of the local loss sensitivity metric $(\mathbf{l_F})$ [70] which is defined as:

$$l_F = \frac{1}{M}\sum_{i=1}^{M} \| \frac{\partial J(\theta, x, y)}{\partial x_i} \|_2 \tag{2.20}$$

where M is the total number of samples which used to verify the performance of the model.

Chapter 3

Stochastic LWTA(Local Winner Takes All) A promising new paradigm

As we previously mentioned neural network is consists of nodes and layers and learns how to find the patterns between inputs and outputs, in each node the inputs are multiplied with the weights and summed together, the result is known as the summed activation of the node. Then the value of the result is transformed via an activation function and defines the specific output or "activation" of the node. The easiest and simplest activation function is known as linear activation. In a network that is comprised only of linear activation functions the training becomes very easy unfortunately can not learn complex functions. Linear activation functions most of the time are used in regression problems that predict a quantity. Nonlinear activation functions are used much more often as they allow us to learn more complex structures in our dataset. Two conventionally widely used nonlinear activation functions are the hyperbolic tangent and sigmoid activation functions.

A very acute problem with both of them sigmoid and tanh is the fact that they have been used extensively from the specialists. This means that large values snap to 1.0 and small values snap to -1 or 0 for tanh and sigmoid respectively. Although, the functions are only really sensitive to changes around the mid-point of their input, such as 0.5 for sigmoid and 0.0 for tanh.

Large networks which using these functions with the property of non-liniarity are failing to infere useful gradient information. Error is backpropagated through the network and is used in order to update the weights. In each next layer the error decreases exponentially, due to the given the derivative of the chosen activation function. Deep networks fails to learn effective due to the problem which is known as a vanishing gradient.

Is necessary to use an activation function in order to use stochastic gradient descent with backpropagation of errors to train deep neural networks, that act and look like a linear function, but in fact, is a nonlinear function allowing to capture of complex relationships in the dataset.

To avoid easy saturation the function must also give more sensitivity to the activation sum input. The key is the utility of the rectified linear activation function(ReLU), a node that implements this activation function. Most of the times networks that are using the rectifier function for the hidden layers are referred to as rectified networks.

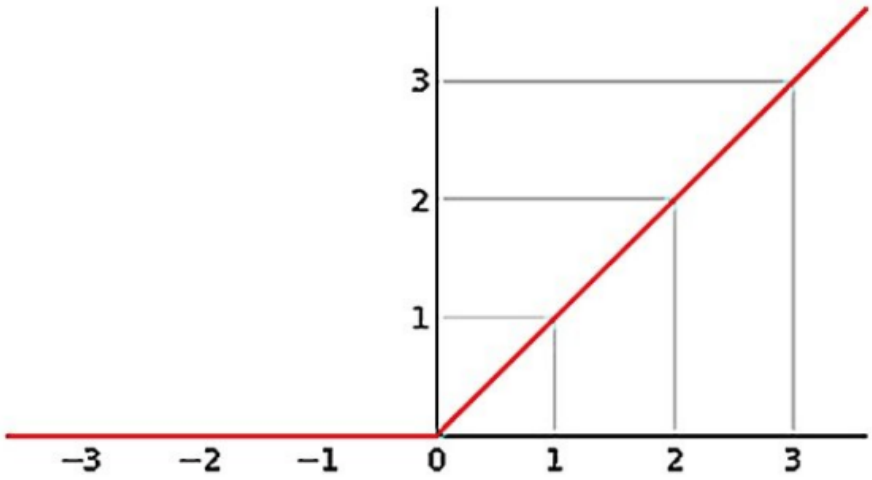


Figure 3.1: Rectified Activation Function.

Deep neural networks (DNNs)[71] as we mentioned to the previous chapter are models with a lot of flexibility that depict complicated functions as a combination of simpler. In spite of their high performance in a lot of applications, very often have to confront the problem of overparameterization that entails numerous weights, which a large part of them is actually unnecessary. This guides us to redundant computational costs and limits their scalability to hardware devices.

In addition, this fact cause in DNNs the tendency to be susceptible to the problem of overfitting which may be an obstacle if we try to predict unseen data.

The specialists have spent remarkable effort in order to deal with the problem of over-fitting in deep learning. Therefore, the range of regularization is restricted to effective training of all network weights. Dealing with redundancy in deep networks requires data-driven structure shrinkage and weight compression techniques.

A well-known type of solution for this purpose, is the training a condensed student network by

take advantage of previous trained full-fledged teacher network[72]. However, this paradigm have two main drawbacks: (i) Can not avoid overfitting and computational costs. On the contrary, the total training costs are increased related to the training costs of the student network and weight distillation (ii) the student teaching procedure itself entails a large deal of heuristics and assorted artistry in designing effective teacher distillation.

Chatzis [73] on 2018 considered that in order to face this problem is needed to introduce an extra set of auxiliary Bernoulli latent variables, which each of them indicates the utility of each component (in an "on/off" way).

Under this framework, by imposing stick-breaking priors over the weights they observed a sparsity [74]over the postulated auxiliary latent variables. Although, their research was limited to variational autoencoders fortunately showed promising results in a variety of benchmarks.

The deep networks have the main drawback of the over the utility of nonlinear units on each layer. This kind of function gives us an effective way to build hierarchical models as we mentioned in the introduction but unfortunately does not have the characteristic of biological plausibility. In neuro-science existing a reinforce belief with a lot of indications that neurons with similar functionalities and properties are aggregated in columns where the local competition takes place[75][76]. This is achieved through the resulting lateral inhibition mechanisms, according to which just a neuron that is in the block can be active in a steady state.

Having this as a guideline, numerous specialists have examined the scenario to develop deep networks which replace nonlinear functions with local competition mechanisms among simpler linear units.

As it has been proven, such local winner-takes-all (LWTA) networks can find out sparse and effective representations of their inputs [77] [78], and constitute universal function approximators, as powerful as networks with threshold or sigmoidal units [79]. Furthermore, this type of network organization has been shown that gives a lot of fascinating properties, including noise suppression ,automatic gain control, and prevention of catastrophic forgetting in online learning[80].

We consider that the ability to infer the posterior distribution of component (connection/unit) utility in the context of LWTA-based deep networks may offer considerable advantages such as computational efficiency. The construction of the model is based on nonparametric Stick-breaking priors which are Bayesian inference arguments we are implementing these arguments in a way that is matched perfectly to the distinctive characteristics of LWTA networks and in

that way, we give the opportunity to a data-driven mechanism will be intelligently adapted to the complexity of the model structure and infers the needed floating-point precision.

In order to construct algorithms to be efficient for training and inference, the model are based on stochastic gradient variational Bayes (SGVB)[81]. The name of the approach is Stick-Breaking LWTA (SBLWTA) networks. The evaluation of the approach is taking place to well-known benchmark datasets.

We introduce the architecture of deep networks where the output of each layer is obtained from competing blocks for linear units, and we are employing suitable arguments from nonparametric statistics to infer network component utility in a Bayesian sense. Below we depicted the design of the model.

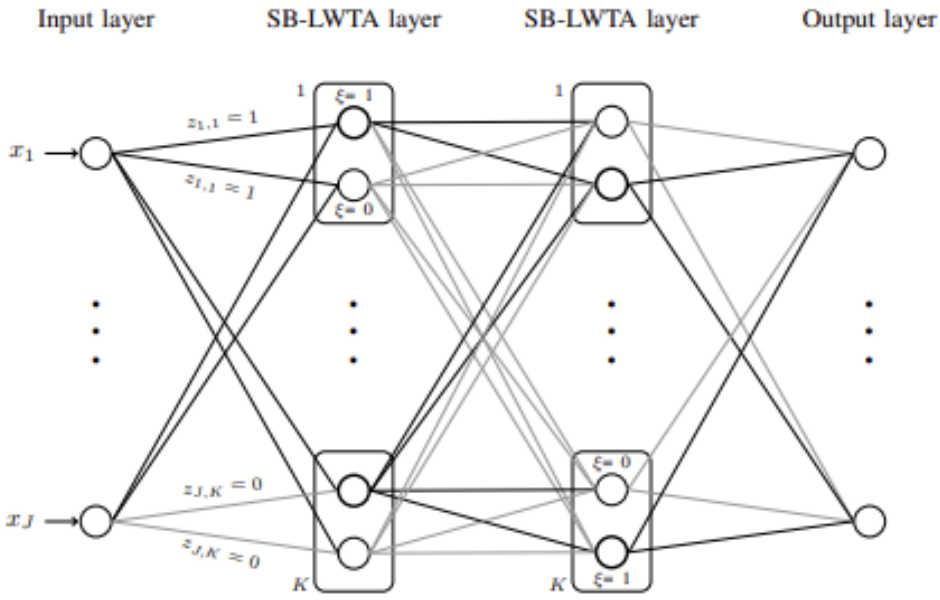


Figure 3.2: The rectangles illustrate the LWTA blocks also circles illustrate the competing units therein. The winner units are depicted with bold contours ($\xi = 1$) and bold edges signify retained connections (z = 1).

Inspired by the tendencies of neuroscience in biological systems to consider that neurons with similar functionality are concentrated in a neuron and local competition takes place for their activations gave the opportunity to Dr.Panousis to introduce the mechanism of Local Winners Takes All (LWTA). A mechanism that allows us to eliminate the problem of overfitting and also to address the adversarial context due to its efficiency of sparsely distributing input representations[82], while demonstrating automatic gain control, suppression noise, and robustness to catastrophic oblivion [80][83]. Thus, in each block only one neuron can be

active, while the rest are as if they do not exist [84][85] [76], leading to a Local Winner Takes All mechanism.

Activation functions which is commonly employed, such as ReLUs, constitute a mathematical tool that has been used extensively for training deep neural networks but they do not have biological plausibility. Following we describe the mathematical framework in more detail when locally competing units are employed via the LWTA mechanism.

We consider the inputs $\mathbf{x}_n \in \mathbb{R}^{N\times J}$ in a single layer of an LWTA-based network comprising K blocks with U competing units inside the blocks. Each block produces an output $\mathbf{y}_k$. Each linear unit in each block, computes its activation h_k^u and the output is decided through competition:

$$y_k^u = g(h_k^1 ..., h_k^u) \tag{3.1}$$

where g(·) is the competition function,the activation of each individual neuron follows the conventional inner product computation $h_k^u = w_{ku}^T \mathbf{x}$ where $\mathbf{W} \in \mathbb{R}^{J\times K\times U}$ is the weight matrix in this context. In the finally definition of LWTA network , the final output is:

$$y_k^u = \begin{cases} 1 & \text{if} \quad h_k^u \geq h_k^i \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

One more expressive version of the activation function have been proposed in the literature in order to overcome the restrictive binary output[80]

$$y_k^u = \begin{cases} h_k^u & \text{if} \quad h_k^u \geq h_k^i \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

Let assume a single layer of the model consisting of K blocks with U competing units therein. Each block by considering an input $X \in \mathbb{R}^{N\times J}$ with N examples comprising J features, each produces an output $\mathbf{Y}_k \in \mathbb{R}^{N\times K}$. The input layer is present at each block K, via different weights for each unit in this way the weights of the connections are illustrated into a three-dimensional matrix $\mathbf{W} \in \mathbb{R}^{J\times K\times U}$ also $\xi_n \in one-hot(U)^K$, which is the discrete latent vectors and the $one-hot(U)$ is an one-hot vector with U components,denoting the outcome of the local competition.In order to infer which layer connections must be retained, we borrow some concepts from the field of Bayesian Statistics. Especially, we begin by introducing binary latent variables $\mathbf{Z} \in [0,1]^{K\times J}$ simultaneously we impose a prior over the latent variables Z in

order to perform inference over it, in a way that promotes the preservation of the minimum necessary components. Following the above logical reasoning we will define the layer output $\mathbf{y}_n$ as below

$$[\mathbf{y}_n]_{ku} = [\xi_n]_{ku} \sum_{j=1}^{J} (w_{j,k,u} * z_{j,k}) * [x_n]_j \in \mathbb{R} \tag{3.4}$$

In the following expression, we assume that the winner indicator ξ_n are drawn from a Categorical distribution of the form:

$$q([\xi_k]) = Discrete([\xi_n]_k \mid softmax(\sum_{j=1}^{J} [w_{j,k,u}]_{u=1}^{U} * z_{j,k} * [x_n]_j)) \tag{3.5}$$

Regarding the utility latent variables, Z, are drawn from the Bernoulli distribution that read:

$$q(z_{j,k}) = Bernoulli(z_{j,k} \mid \tilde{\pi_{j,k}}) \tag{3.6}$$

Therefore, we impose an IBP prior distribution over the utility indicators and a Discrete prior distribution over the winner latent indicators, $[\xi]_k \sim Discrete(1/U)$:

$$u_k \sim Beta(\alpha, 1) \quad \pi_k = \prod_{i=1}^{k} u_i \quad z_{j,k} \sim Bernoulli(\pi_k) \tag{3.7}$$

while we resort to fixed-point estimation for the weight matrices W. The definition of the intermediate layer of the considered approach has been completed we also assume a variation relied on common feed forward layer as regards the output layer.

Finally we impose a distribution over the weight matrices $\boldsymbol{W}$. For simplicity we define a spherical prior over

$$\boldsymbol{W} \sim \prod_{j,k,u} \mathcal{N}(0,1)$$

and attempt to infer a posterior distribution of the form

$$q(\mathbf{W}) = \prod_{j,k,u} \mathcal{N}(\mu_{j,k,u}, \sigma^2_{j,k,u})$$

3.0.1 The Convolutional perspective of LWTA

However, we consider a variant of SB-LWTA which allows the processing of convolutional operations. This is very useful if we deal with problems with signals of 2-dimensions such as

images. However in contrast with the aforementioned to the grouping of linear units in LWTA blocks the convolutional perspective variant executes local competition among feature maps. Therefore each layer that contains kernels is confronted as an LWTA block. Convolutional SB-LWTA networks at each layer comprise multiple kernels of competing feature maps.

We consider the latent indicator variables, z, over whole kernels, that is full LWTA blocks. If the inferred posterior, $q(z_k = 1)$, over the k_{th} block is under a certain threshold then the block is as if it does not exist in the network. Especially, this setting allow us for entirely removing kernels and in that way the number of convolution operations are decreasing dramatically.

Therefore, this structure is very efficient due to the fact that relieves the model from the computational burden. Due to this rationale, a layer of the proposed convolutional variant represents an input, X_n, via an output tensor $\mathbf{Y}_n \in \mathbb{R}^{H\times L\times KU}$ obtained as the concatenation along the last dimension of the subtensors $[Y_n]_{k=1}^{K} \in \mathbb{R}^{H\times L\times U}$ defined below:

$$[\mathbf{Y}_n]_k = [\xi_n]_k((W_k * z) \star \mathbf{X}_n) \tag{3.8}$$

where "$\star$" represent the convolution operation and $[\xi_n]_k \in$ onehot(U). Local competition among feature maps within an LWTA block is implemented through a sampling procedure which is instructed from the feature map output, yielding:

$$q([\xi_{nk}]) = Discrete([\xi_n]_k | softmax(\sum [z_k W_k \star X_n]) \tag{3.9}$$

We impose a prior $[\xi_n]_k \sim Discrete(\frac{1}{U})$.We assume

$$q(z_k) = Bernoulli(z_k|\widetilde{\pi}) \tag{3.10}$$

with corresponding priors:

$$u_k \sim Beta(\alpha, 1) \quad \pi_k = \prod_{i=1}^{k} u_i \quad z_k \sim Bernoulli(\pi_k) \tag{3.11}$$

As I mentioned earlier imposed a spherical prior N(0, 1), and seek to infer a posterior distribution of the form $N(\mu, \sigma^2)$.This concludes the formulation of the convolutional layers of the proposed SB-LWTA model.

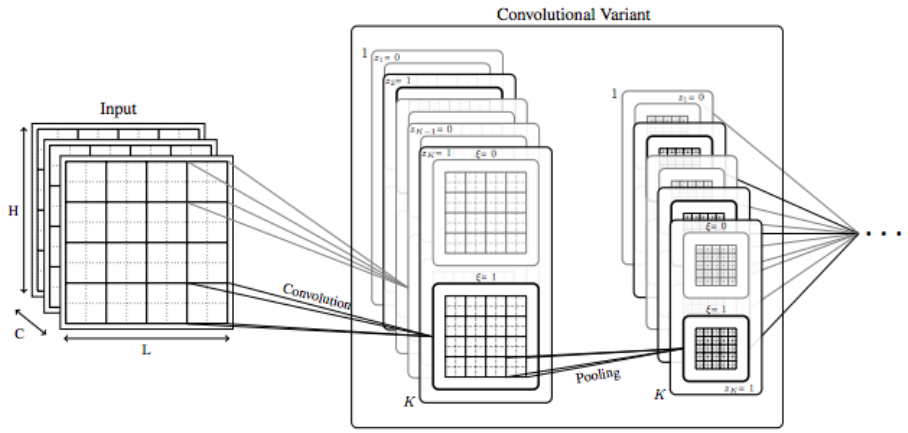


Figure 3.3: The convolutional approach of the method which bold frames denote the effective kernels (LWTA blocks of competing feature maps, with z = 1 and bold rectangles illustrate winner feature maps (with ξ = 1).

3.0.2 Training and inference algorithms

The training of the aforementioned model based on the maximization of the ELBO expression. More especially, we use Stochastic Gradient Variational Bayes (SGVB)[86] with the assist of the standard reparameterization trick for the postulated Gaussian weights, the Kumaraswamy reparameterization trick for the stick variables u ; and finally the Gumbel-Softmax relaxation trick for the introduced latent indicator variables, ξ and Z. Specifically, when dealing with stick variables which is Beta-distributed, it is observable the difficulty to use the reparameterization trick from the other side the postulated Gaussian weights is more feasible, in order to address this issue, we can approximate the variational posteriors $q(u_k) = Beta(u_k|a_k, b_k)$ via the Kumaraswamy distribution.

$$q(u_k; \alpha_k, b_k) = \alpha_k b_k u_k^{\alpha_k - 1}(1 - u_k^{\alpha_k})^{b_k - 1} \tag{3.12}$$

Samples from this distribution can be reparameterized as follows [87]

$$u_k = (1 - (1 - X)^{\frac{1}{b_k}})^{\frac{1}{\alpha_k}} \tag{3.13}$$

In the case of latent variables which in this case are Discrete performing back-propagation through reparameterized drawn samples becomes unachievable. All the posterior expectations in the ELBO are computed by drawing MC samples under the Normal, Gumbel-Softmax and

Kumaraswamy reparametrization trick, respectively. On this basis, ELBO maximization is performed with the utility of stochastic gradient techniques especially, we are using ADAM [23] with default settings. The objective function of the obtained ELBO is optimized by ADAM eventually.

ADAM optimize the resulting expression of the ELBO by the network which takes the form:

$$\begin{aligned} L(\phi) = \mathbf{E}_{q(\cdot)}[\log p(D|\mathbf{u},\mathbf{W},\mathbf{Z},\xi)] - \\ -\beta(KL[q(\mathbf{Z})||p(\mathbf{Z})] - KL[q(\mathbf{u})||p(\mathbf{u})] - KL[q(\xi)||p(\xi)] + \\ KL[q(\mathbf{W})||p(\mathbf{W})])] \end{aligned} \tag{3.14}$$

where we postulate:

$$q(\mathbf{u}) = \prod_k q(\mathbf{u}_k) \quad s.t : q(u_k) \approx Kumarasuamy(u_k|\alpha_k, b_k) \tag{3.15}$$

we consider ϕ as the target parameters which be optimize by algorithm. KL divergences which included in the ELBO separate into sums over the components of the latent variables, as a result of the mean-field aproximation which all the KL divergences are computed by MC sampling from the corresponding posteriors distributions. In this framework, we are using the reparameterization trick in order to reduce the variances. Hence, for the stick-variables u_k we obtain:

$$KL[q(u_k)|p(u_k)] = \mathbb{E}_{q(u_k)}[\log q(u_k) - \log p(u_k)] \approx \log q(\hat{u}_k) - \log p(\hat{u}_k) \;\; k = 1,...,K \tag{3.16}$$

where $\hat{u}_k$ is a reparameterized sample from the Kumaraswamy distribution with parameters α_k and bk.

As regards the component utility indicators z, the estimation of KL divergences is calculated as follows (the expression of KL divergence below is just for simple feedforward network):

$$KL[q(z_{j,k})|p(z_{j,k})] = \mathbb{E}_{q(z_{j,k}),q(u_k)}[\log q(z_{j,k}) - \log p(z_{j,k})] \tag{3.17}$$

$$\approx \log p(\hat{z}_{j,k}) - \log(\hat{z}_{j,k}|\prod_{i=1}^{j} u_i) \;\; j = 1,...,J \;\; k = 1,...,K \tag{3.18}$$

where the $\hat{z}_{j,k}$ samples are reparameterized via the Gumbel-Softmax trick with parameters $\widetilde{\pi}_{j,k}$ and 1-$\widetilde{\pi}_{j,k}$. We are using a similar process to estimate the terms of KL for the winning indicator variables ξ_{nku}:

$$
\begin{aligned}
KL(q(\xi_{n,k,u})|p(\xi_{n,k,u})) = \mathbb{E}_{q(\xi_{n,k,u})q(w)q(z)} = \\
\left[\log q\left(\xi_{n,k,u}|softmax\left(\sum_{j=1}^{J} w_{j,k,u} * z_{j,k} * x_{n,j}\right)\right)\right] - \log p(\xi_{n,k,u})] \\
\approx \left[\log\left(\hat{\xi}_{n,k,u}|softmax\left(\sum_{j=1}^{J} \hat{w}_{j,k,u} * \hat{z}_{j,k} * x_{n,j}\right)\right) - \log p(\hat{\xi}_{n,k,u})\right]
\end{aligned}
\tag{3.19}
$$

where the samples $\hat{\xi}_{nku}$ are reparameterized via the Gumbel-Softmax trick.The Gaussian weights yield:

$$
KL[q(w_{j,k,u})|p(w_{j,k,u})] = \mathbb{E}_{q(w_{j,k,u})}[\log q(w_{j,k,u}) - \log p(w_{j,k,u})] \approx \log q(\hat{w}_{j,k,u}) - \log p(\hat{w}_{j,k,u}) \tag{3.20}
$$

where $\hat{w}_{j,k,u} = \mu_{j,k,u} + \sigma_{j,k,u} \cdot \hat{\varepsilon}$, and $\varepsilon \sim N(0,1)$.

The expectation posterior of the $\log p(D|\mathbf{u},\mathbf{W},\mathbf{Z},\xi)$ is interpreted as the negative categorical cross-entropy between the class probabilities and the data labels which is generated one layer before the last which is the Softmax layer. The aforementioned posterior expectation is estimated with the samples that is already reparametrized. In this context, the task of optimize the targets ϕ, which comprises the a_k and b_k hyperparameters of the $Kumaraswamy(u_k|\alpha_k,b_k)$ posteriors, the $\mu_{j,k,u}$ and $\sigma_{j,k,u}$ hyperparameters of the $N(w_{j,k,u}|\mu_{j,k,u},\sigma^2_{j,k,u})$ posteriors, and the $\hat{\pi}_{j,k}$ hyperparameters of the $q(z_{j,k})$ posteriors.

SB-LWTA gives us two main advantages over conventional techniques in terms of inference for unseen data. Taking advantage of the inferred component utility latent indicator variables, we can naturally build a method for omitting the contribution of components that essentially is unnecessary. This is a unique benefit of the model due to the fact that compared to the alternatives obviates the need of an extensive heuristic search of hyperparameters values also. The fully Gaussian posterior distribution over the network weights, W, provides a way of reducing the floating-point bit precision level of the network implementation. Especially, the weights of the network which has each of them a posterior variance indicates a measure of uncertainty in their estimates. However, we can take advantage of this information of uncertainty in order to estimate which bits are really significant and to take away these which cause a high level of fluctuations under the approximation of the posterior sampling.

We perform the conventional forward propagation using the means of the weight posteriors in place of the weight values. In terms of winner selection, we inferred the posteriors $q(\xi)$ and then we select the unit with maximum probability as the winner, instead of just the sampling.

Finally, we maintain all network components the posteriors, q(z) of which exceed the imposed truncation threshold.

Chapter 4

Obtaining Stronger Representations From Deep Nets

Here we present our novel contribution. The work has been published in the conference of Association for the Advancement of Artificial Intelligence (AAAI) with the title ”Competing Mutual Information Constraints with Stochastic Competition-based Activations for Learning Diversified Representations”

4.1 Information theory

Telephones,radios, televisions, and, computers are devices that transmit, store and process information today they belong to the basic equipment of the average citizen in the industrialized world. undeniable the twentieth century witnessed the greatest revolution of information and communication technology since the invention of the printed book. This powerful development guided to the need for a well-established theory that would allow discussion about information with scientific precision. The roots of information theory are going back to eighteenth century when American inventor Samuel F.B.Morse and Alexander Ghrahabell discovered respectively the telegraph and the telephone. These discoveries attracted the interest of many remarkable scientists which with the assist of Fourier analysis they boosted up the theory and solved several practical mechanical problems of communication systems.

Modern information theory first appeared by Claude Shannon in early 1948 in the article “A Mathematical Theory of Communication” in the Bell System Technical Journal who

is considered as the father of the digital age, then that developed into a vigorous field of mathematics that motivated the growth of other scientific fields, such as biology, statistics, statistical mechanics, behavioral science, and neuroscience. The greatness of Shannon's idea was that he realized that in order to build a theory it is necessary to separate the signals of communication from the meaning of the messages that they convey and secondly he had realized that the amount of knowledge transmitted by a signal is not related to the size of the message. The potential of mobile phones and the growth of the internet is largely due to him.In order to have a solid background of information theory[88] is obligated to have a general idea about meanings such as compression,mutual information,KL distribution divergence,entropy and Data processing inequality which we will mention below with detail[89][90].

Compression To compress an information like text document, image or a sequence of bits, is to represent it in an abbreviated fashion, usually with the help of a code. We take our information, encode it using our short and efficient code and store it or transmit it, and, when we need to, decode the information back to its original form[91][89].Let us assume an information source that generates a sequence of symbols $X = X_1, ..., X_N$ which takes values in a measurable alphabet X. Due to the fact that we examine a probabilistic model for the information source automatically Xi are considered as random variables. The most important problem of source coding is the procedure of storing information as compact as is possible that contained in the source $x = x_1 ... x_N$. It is a very practical and interesting issue, which is also an instructive subject that allows us to gain the intuitions of 'uncertainty' and 'information' and are closely related to the definition of entropy which is very important for the rest of the chapter.

Mutual Information

The mutual information which illustrate the amount of dependency between 2 random variables that asre sampled simultaneously, in a simple words how much information two variables shares and mathematically defined as[92]. Let (X; Y) be two discrete random variables, distributed according to p(x; y) and with marginal distributions $p(x) = \sum_y p(x,y)$ and $p(y) = \sum_x p(x,y)$. The mutual information between X and Y is defined as

$$I(X,Y) \equiv I[p(x,y)] = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{4.1}$$

Using this definition and the above mentioned definitions, it is easy to obtain

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{4.2}$$

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{4.3}$$

It is very close to the meaning of entropy due to the fact that can be known as the reduction of uncertainty of a random variable if another is known so when we have a high value of mutual information that means a large reduction of uncertainty in contrast when the value is low that means a small reduction, in the case of zero value we have independencies between 2 variables.

Entropy:

$H(Y)$ is the entropy of a random variable X and can be computed through the

$$H(Y) \equiv H[p(y)] = -\sum p_y \log p_y \tag{4.4}$$

The mentioned function has as a threshold a multiplicative constant, the base of the logarithm is just setting the scale for this measure. If the base of the logarithm is chosen to be 2, the entropy is expressed in bits. In this case could be interpreted as the expected minimal number of 'yes'/'no' questions that is required in order to estimate the value of Y. In addition, the entropy emerging as the answer to several natural questions, such as "How much is the average length of the random variable?"In a simple words entropy which represent the "mess" in the variable X given the number of different values X can take[93].

The Data Processing Theorem

$X \leftrightarrow Y \leftrightarrow Z$ is a Markov chain which follow the $I(X;Z) \leq \min[I(X;Y), I(Y;Z)]$. In other words, if the only connection between X and Z is through Y, the information that Z gives about X cannot be bigger than the information that Y gives about X.

KL-divergence

Kullback-Leibler divergence which is a kind of a distance between the exactly posterior and the approximate posterior, in simple words is a way to estimate the difference between two distributions.

$$KL(q(\cdot)||p(\cdot)) = \int q(\theta) \log \frac{q(\theta)}{p(\theta|y)} d\theta \tag{4.5}$$

$$= \int q(\theta) \log \frac{q(\theta)p(y)}{p(\theta,y)} d\theta \tag{4.6}$$

$$= \log p(y) - \int q(\theta) \log \frac{p(\theta,y)}{q(\theta)} d\theta \tag{4.7}$$

Jensen-Shannon divergence

For two probability distributions p and q on the same alphabet L ,the Jensen-Shannon divergence of p and q is defined as:

$$JS(p\|q) = \frac{1}{2}(D(p\|\frac{p+q}{2}) + D(q\|\frac{p+q}{2})) \tag{4.8}$$

These measures are closely related to that of Shannon's Entropy, given in[94], which is defined loosely as a measure of the dispersion or "variance" of the individual distribution populations p, q.

4.2 Information Bottleneck

An important problem for humans was how they were able to extract relevant information from complex data in order to address problems such as filtering, learning, prediction etc. The problem of Information Bottleneck have been proposed by Tishby in 1999[95] as a way to quantify the trade-off between compression and prediction and the first implementation at [96][97]. Until 1999 information theory was not examined the relevance of information. Is an iterative algorithm which aim to find these representations that capture the relevant structure of the signal and ensure convergence of the algorithm.

It is important to mentioned that a significant property of the Information Bottleneck framework is that it can be implemented to a wide variety of data sets in exactly the same way. There is no need to adapt algorithms to the data each time, or define a distortion measure. When estimate the joint distribution the setup is completed. In addition, its architecture contain the ability to avoid the major problem of over-fitting without using any kind of specific technique to avoid it. Thus we avoiding the complexity of the model.

If we want to have a deep understanding of what exactly is Information bottleneck we must take a glance to the statistics field and focuses more specifically to the minimal sufficient statistic which in a simple words is a statistic functions T(x) that having the property to contain all information about the unknown parameter θ (and therefore $g(\theta)$) containing the sample $\mathbf{X} = (X_1, ..., X_n)$. In addition, sufficient statistical function T(X) has the significant advantage over X to usually has a much smaller dimension than the dimension n of $\mathbf{X} = (X_1, ..., X_n)$.In this sense, sufficient meaning brings a desirable compression of the data to be analyzed without loss of information about θ.

In real life the problems where $X = [X(i)]_{i=1}^{N}$ is an N-dimensional data, when you try to find the minimal sufficient statistic in such a way that its dimension does not depend on N.

Unluckily, it is very difficult almost impossible to find for all distributions except the ones that belonging to exponential family. In other words this is the reason why IB have been emerged,due to the inability of minimal sufficient statistic function.

Let us consider an information source X and we want to compress the this information and to convert to representation of the information, T, such that the amount of data that can be transmitted in a given period of time is minimal, so that communication is facilitated, while keeping the maximal amount of information about a specific characteristic of X. The random variable Y represents this characteristic. The mathematical form of the problem takes the following format [98]:

$$\min_{P_{R|X}} I(R;Y) - \beta I(R;X) \equiv \langle \log p(y|r) - \log p(y) + \beta \log p(r) - \beta \log p(r|x) \rangle \tag{4.9}$$

The β coefficient regulate the counterbalance between the importance of forgetting unuseful parts of X while creating R and the importance of keeping the useful parts for Y and represent the strength of the bottleneck. When β has high value compression is no longer significant and the we achieve the minimum by selecting R = X. The importance of compression is depicted by the tendency of β to zero.

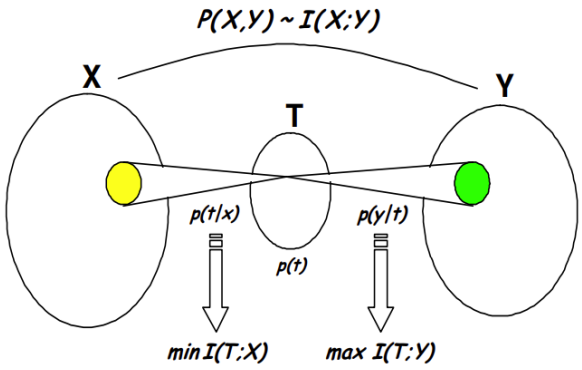


Figure 4.1: The information between X and Y is squeezed through the "bottleneck" representation, T.

4.3 Mutual Information and Representation Learning

Representations are very practical when you want to handle multiple modalities to transfer learned knowledge to tasks with few or no examples being given but a task representation exists. The difficulty of any information processing task depends on how the information is

represented. This is applicable also to daily life to machine learning and generally to computer science. For instance, it is just a simple mathematical operation for a person to divide 210 by 6 using long division. The task becomes much more difficult when instead of posed using the Roman numeral representation of the numbers. Most of the people asked to divide CCX by VI would begin by converting the numbers to the Arabic numeral representation, permitting long division procedures that make use of the place value system. In the framework of machine learning, a representation is better than another when makes a subsequent learning task easier. The choice of representation will usually depend on the choice of the next learning task. Mutual information (MI) is a needful tool in the framework of Representation Learning which is used in both the unsupervised settings, supervised settings, as well as in the semi-supervised typically relying on MI maximization[99][100][101]. Here, we focus on the supervised task, where commonly MI minimization is employed.

In spite of the success of the MI rationale to deep networks,there is still much space for improvement towards constructing networks with disentangled and diversified representations. The information-theoretic principles seem to be effective for representations with increased diversifying and discriminative capabilities. However, most of the methods that exist typically resort to the optimization of a single information constraint, significantly suppressing the variety of the emerging representations.

For this purpose, a detailed research attempt has been made for learning programs that rely on the idea of competition and collaboration between different neural representations.

Hu [102], recently proposed an innovative approach for diversifying the learned information by providing generalization and diversification capabilities in self-supervised and supervised settings. Under the proposed approach, different representation parts are constructed; these except the competition, give us the ability to collaborate in order to achieve a downstream task, by giving us the capability to learn the richest and discriminative representations.

Especially, let t indicate the downstream task that we want to achieve, e.g. x is the label of an input, and r illustrates the representation of input x. The purpose is to maximize the mutual information between the representation r and the target t, guiding to the objective function:

$$max[I(r,t)] \tag{4.10}$$

where $I(\cdot,\cdot)$ indicates the mutual information. We segregate reprsentation r into two different parts [z, y] with different constraints each one in order to achieve a considerable diversification

of the learned information, aiming to learn distinct pieces of information from the input x. Specifically, two different constraints are introduced, such that the "information capacity" of z is minimized, while the "information capacity" of y is maximized. The above process yields the following objective function:

$$max[I(r,t) + \alpha I(y,x) - \beta I(z,x)] \tag{4.11}$$

α, β are regularization constants. In order to boost the diversification of the learned information avoiding the domination of either term, z and y are constrained to be independent of each other, while trying to accomplish the downstream task only through their MI constraints. These modifications guide us to the ICP objective function:

$$max[I(r,t) + \alpha I(y,x) - \beta I(z,x) + I(z,t) + I(y,t) - \gamma I(z,y)] \tag{4.12}$$

where γ is also a regularization constant. The objective function which is optimized is based on the computation of the MI between different pairs of variables, due to the fact that different optimization methods are required since they aim to achieve different tasks. The above procedures can be found in [102]. Here we highlight that the objective constructions are differentiable and can be trained via backpropagation.

4.3.1 Information Competing Process

In spite of the significant improvement in the field most of the methods that exist what is exactly doing is to optimize a specific information constraint in order to learn "useful" and diverse representations; unfortunatly, this approximation fails to encourage the diversification among the representations that inferred from the latent units. Therefore, numerous of learning schemes have been invented in the field in order to deal this inadequacy; these are relied on the idea of competition and collaboration between different neural representations.[103]

Relied on recent advances in Representation Learning, we espouse the Information Competing Process(ICP) [102] in order to boost the diversification of information. The incentive behind ICP is that traditional mutual information approaches are often based on a single mutual information constraint, decreasing the diversification of the emerging representations. In contrast, the ICP rationale carries out a new diversifying objective function, separating the representation into two parts with different mutual information limitations each one the

structure of the function ensures that these parts "compete" in order to fulfill a task, but are then combined to accomplish said task in a synergistic way. The ICP objective function is :

$$max[I(r,t)] \tag{4.13}$$

where I$(\cdot,\cdot)$ means the mutual information. This constraint has as the ultimate goal in order to maximize the mutual information between the representation r and the target t. In order to enrich the information on representations distinguish instantly the representation r into two parts [z, y] with different constraints each other, and motivate the representations to learn discrepant information from the input x. Especially the information capacity of representation part z is constrained while at the same time the information capacity of representation part y is increased. To that effect, we have the following objective function:

$$max[I(r,t) + \alpha I(y,x) - \beta I(z,x)] \tag{4.14}$$

where $\alpha > 0$ and $\beta > 0$ are the regularization factors. In order to obstruct any one of the representation parts to dominating the task, we let z and y to fulfil the task t by uzing only the mutual information constraints I(z, t) and I(y,t). In addition, in order to ensure that the representations capture information via different constraints, ICP prevents z and y from knowing what each other learned for the downstream task, which is realized by enforcing z and y independent of each other. Those constraints instruct us in a competitive environment and enrich the information carried by representations. The ICP has an objective function that we could express as:

$$L_{ICP} = max[I(r,t) + \alpha I(y,x) - \beta I(x,z) + I(z,t) + I(y,t) - \gamma I(z,y)] \tag{4.15}$$

where I(z, x) and I(y, x) are called the mutual information minimization and maximization terms respectively, while I(z, t), I(y, t), I(r, t) are the inference terms and I(z, y) the predictability minimization term; γ are also regularization constant which in order to optimize them needed appropriate tractable bounds based on variational approximations for each task. All the aforementioned components, including the probabilistic feature extractors, task solvers, predictors and discriminators are implemented via deep neural networks, while the variational approximations are assumed to be Gaussian distributions.

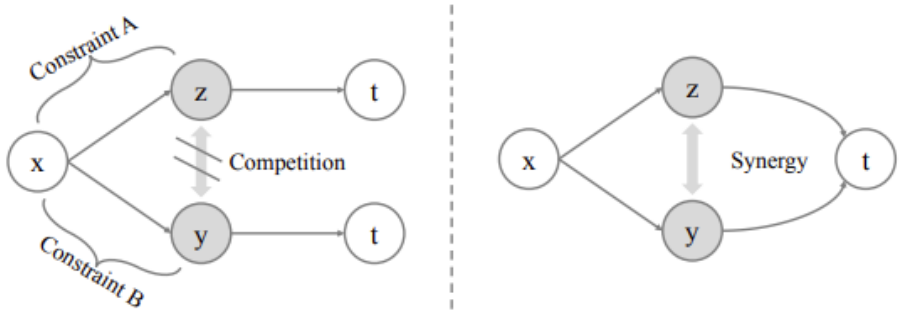


Figure 4.2: In the competitive step, the rival representation parts are forced to accomplish the downstream task solely by preventing both parts from knowing what each other learned under different constraints for the task. In the synergetic step, these representation parts are combined to complete the downstream task synthetically.

4.4 The Blahut Arimoto algorithm

Blahut Arimoto is an iterative computational algorithm that converge to maximum of the optimization problems with concepts like theoretical information capacity of a channel, the rate distortion function of a source or a source coding. Considered we have two convex sets A and B in $\mathbb{R}^N$ and we want to find the minimum distance between them. I choose a point a randomly and we trying to find the β that is closest to α. Then we stabilize β and find α. I repeat this process until my algorithm converges. At this point arises two reasonable questions is it this point the total minimum and what is going on in the case which p(x,y) is high-dimensional or is not-Gaussian. The way to deal with the aforementioned questions is to introduce the concept of deep learning and statistical learning by using variational scheme in order to maximize the lower bound above the information bottleneck method. In a simple words is intractable to apply Blahut Arimoto algorithm to deep neural networks[104].

4.5 Variational Information Bottleneck

There is a close connection between VIB and the well-known β-VAE formulation [105]. Both of them are based on information theoretic arguments but used in different contexts; the first one is for supervised while the second one for unsupervised learning.

The aproximation of variational information bottleneck[106] trying to maximize a lower bound on the IB objective function and this algorithm is related to the variational EM algorithm[107].

Input:

Source distribution $p(x)$.
Trade-off parameter β and convergence parameter ε .
A set of representative, given by T values.
Distortion measure $d : \mathcal{X} \times \mathcal{T} \to \mathcal{R}^+ , \forall x \in \mathcal{X}, \forall t \in \mathcal{T}$.

Output:

Value of $R(D)$ where its slope equals $-\beta$.

Initialization:

Initialize $R^{(0)} \leftarrow \infty$ and randomly initialize $p(t)$.

While True

- $P^{(m+1)}(t \mid x) \leftarrow \frac{P^{(m)}(t)}{Z^{(m+1)}(x,\beta)} e^{-\beta d(x,t)}$, $\forall t \in \mathcal{T}, \forall x \in \mathcal{X}$.
- $P^{(m+1)}(t) \leftarrow \sum_x p(x) P^{(m+1)}(t \mid x)$, $\forall t \in \mathcal{T}$.

$R^{(m+1)}(D) = D_{KL}[p(x)p^{(m+1)}(t \mid x) \| p(x)p^{(m+1)}(t)]$.

If $(R^{(m)}(D) - R^{(m+1)}(D)) \leq \varepsilon$
Break .

Figure 4.3: Pseudo-code of Blahut Arimoto

Tishby proved that the IB loss function can be optimized by applying iterative updates:

$$p_{t+1}(r|x) \propto p_t(r) exp[\frac{1}{\beta} \int_y p(y|x) \log \frac{p(y|x)}{p_t(y|r)}] \tag{4.16}$$

$$p_{t+1}(r) = \int_x p(x) p_{t+1}(r|x) \tag{4.17}$$

$$p_{t+1}(y|r) = \int_x p(y|x) p_{t+1}(x|r) \tag{4.18}$$

[108] when $p(x,y)$ is complicated, high-dimensional and non-gaussian these updates are very hard to find it analytically so approximation methods are necessary. Due to the positivity of the KL divergence, we can write,$\langle \log q(\cdot) \rangle \leq \langle \log p(\cdot) \rangle$ for any approximative distribution $q(\cdot)$.This allows us to formulate a variational lower bound for the IB objective function:

$$\hat{L}_{p(r|x),q(y|r),q(r)} = \frac{1}{N} \sum_{n=1}^{N} \langle \log q(y_n|r) + \beta \log q(r) - \beta \log p(r|x_n) \rangle \leq L_{p(r|x)} \tag{4.19}$$

where $q(y_n|r)$ and $q(r)$ are variational distributions, and we have replaced the expectation over $p(x,y)$ with the empirical expectation over training data. The constant term have omitted for simplicity, $H_Y = -\langle \log p(y) \rangle$. Setting $q(y_n|r) \leftarrow p(y_n|r)$ and $q(r) \leftarrow p(r)$ fully tightens the bound (so that $\hat{L} = L$), and guides to the iterative algorithm of Tishby. However, when these exact updates are not possible, one can instead choose a restricted class of distributions $q(y|r) \in Q_{y|r}$ and $q(r) \in Q_r$ for which inference is tractable. In this way, in order to maximize the objective $\hat{L}$ with respect to parameters Θ of the encoding distribution $p(r|x,\Theta)$, we repeat the procedure as follow until convergence:

- For fixed Θ, find $q^{new}(y|r), q^{new}(r) = argmax_{q(y|r),q(r)\in Q_y|r,Q_r} \hat{L}$
- For fixed $q(y|r)$ and $q(r)$, find $\Theta = argmax_{\Theta}\hat{L}$

The concept of Information Bottleneck has been extended analytically and studied in different kind of scenarios such as Gaussian variables, continuous variables through variational methods [109] ,geometric clustering and is used for learning invariant and disentangled representations in deep neural networks.

This method allows us to parameterize the information bottleneck model using a neural network and leverage the reparameterization trick for efficient training. As a result we observe that this approach gives us the opportunity to be able to model the mean and variance separately, thus ensuring the independence of these two important quantities also allows us to have much lower computational costs and of course we have much better results than pre-existing methods.

Applications such us image clustering ,neural code analysis word clusters for supervised and unsupervised text classification,natural language processing,galaxy spectra analysis, gene expression data analysis,protein sequence analysis are some of the main domains that could implemented by variational information bottleneck.

4.6 Proposed Approach

The main characteristic of the Representation Learning (RL) is the ability of extraction varified representations from the dataset. In spite of the huge research effort, the learning representations with such qualities remains an open research question which needs develop. Latterly mutual information constraints [110][100][111], have risen as a promising source

in order to help RL to be successful. Notably, the Information Bottleneck (IB) [95] has attracted a substantial amount of attention [106][112][113]. The essence of the IB theory lies in constraining the inferred representations, so as to minimize the information transmitted for the input while being maximally informative about the target outputs [95]. As regards deep learning, we can be found the first implementations of IB in [114].

It is remarkable that the major part of the proposed works do not examine the most important perspective of modern deep architectures which is the utility of the non-linearity, despite these advances. We consider that an extremely different paradigm of latent unit operation may allow the significantly improvement of the representation learning of deep networks.

The inspiration of the architectures is drawn from the biologically-plausible brain. The previous years it has been proven that neurons with same functionalities in the brain massing together in blocks, and a local competition takes place for their activation (Local Winner-Takes-All/LWTA). We consider that an extremely different paradigm of latent unit operation may allow to significantly enhance the representation power of deep networks. In each block, we have a winner that gets to convey its activation to neurons outside the block, while the rest are forced to silence. Simultaneously, a major operating principle of this mechanism seems to be stochasticity; the activations within a block may differ when presented with the same stimulus at different times [77][115][84].

The incorporation of the LWTA mechanism in deep architectures has been shown to yield important benefits [83][116], while exhibiting significant properties such as noise suppression, specialization, automatic gain control, robustness to catastrophic forgetting and adversarial attacks and compression. Despite these properties, LWTA-based networks have been scarcely examined in the RL literature, while most works fail to take into account the core principle of stochasticity that characterizes biological systems.

The aim of this work has purpose to deal with a long-established problem of learning diversified representations. For this purpose, we combine stochastic competition-based activations, namely Stochastic Local Winner-Takes All (LWTA) units with information-theoretic arguments. In this framework, we do not use the conventional deep architectures commonly used in Representation Learning, that are based on non-linear activations and we are using sets of locally and stochastically competing linear units.

Each layer of the network achives to increase the sparsity of the outputs, this is a result of the competition by between units that are organized into blocks of competitors. We are recruiting the stochastic arguments for the competition mechanism, which perform posterior sampling to

determine the winner of each block. Are considered networks with the ability to infer which sub-parts of the network is actually necessary in order to model the data we have, to achieve it we are using the stick-breaking priors to this end. Also to enrich the information of the rising representations, we resort to information-theoretic principles and specifically the Information Competing Process (ICP).

The combination between components is linked together under the framework of stochastic Variational Bayes for inference. We execute an experimental investigation in-depth, for our approach we are using datasets for image classification.

To train the model we use stochastic Variational Bayes. The objectives of the ICP are differentiable and can be trained using the reparametrization trick that we analyzed in chapter 1. However, the introduction of the Stochastic LWTA activations and the IBP-based mechanism entails the discrete latent variables ξ, z and Beta-distributed stick-variables u that are not readily amenable reparameterization trick. To this end, we utilize the proposed continuous relaxation of the Discrete distribution, founded on the Gumbel-Softmax trick[117][101] and the Kumaraswamy Beta-approximating distribution. These relaxations are employed during training in order to obtain unbiased low-variance estimates of the ELBO gradients. During inference, we draw samples from the trained posteriors for the winner selection variables and the respective component utility indicators.

4.7 Foundational Principles

Let us assume a dense layer of a conventional deep neural network comprising K hidden units. When presented with an input $x \in \mathbb{R}^J$, each hidden unit k performs an inner product computation, $h_k = w_k^T x = \sum_{j=1}^{J} w_{jk} x_j \in \mathbb{R} W \in \mathbb{R}^{J \times K}$. Let us assume a dense layer of a conventional deep neural K is the associated weight matrix of the layer. This response usually passes through a non-linear function $\sigma(\cdot)$, yielding $y_k = \sigma(h_k)$, $\forall$ k. In this way, the output of the final layer emerges from the concatenation of the non-linear activations of each unit.

Contrary to the framework of the LWTA, the hidden units which are singular nonlinear are replaced by U competing linear units which are grouped together in an (LWTA) block, and each layer has multiple blocks like these. Then, we signify by B the number of such blocks in a particular LWTA-based layer. The weight synapses are now structured as a three dimensional matrix $W \in \mathbb{R}^{J \times B \times U}$, revealing that, in this case, each input is now presented to each block b and each unit u therein. In this context, each linear unit u in each block

b computes its response, following the conventional inner product computation, such that $h_{b,u} = w_{b,u}^T = \sum_{j=1}^{J} w_{j,b,u} x_j \in \mathbb{R}$; then, competition takes places among the units in the block. The dominant principle of this theory is that among all of the U units which are in the block, just one can be the winner of the block. This unit can transmit its (linear) activation to the next layer, while all the rest are passed a zero value i.e convey to silence. It is obvious that this procedure ends up with a very beneficial property which is the sparsity among representations.

In each block, only one out of the U units produces a non-zero output of 1. In the literature which is related to LWTA, the competition process is usually deterministic, i.e., the winner with the highest linear activation is considered the winner at each time. Therefore, [118] has introduced the novelty to use stochastic arguments for the competition process. In this context, they introduced an appropriate set of discrete latent vectors $\xi \in onehot(U)^B$ to encode the outcome of the competition in each of the B stochastic LWTA blocks that constitute a stochastic LWTA layer.

This vector comprise B component subvectors; each component entails exactly one non-zero value at the index position that corresponds to the winner unit in each respective LWTA block. Further, in this work, we introduce an additional data-driven mechanism in order to endow the networks with the ability to infer which sub-parts of the network are essential for modeling the data at hand.

To this end, we introduce a matrix of auxiliary binary latent variables $Z \in [0,1]^{J \times B}$. Each entry therein is 1 if the j^{th} component of the input is presented to the bth block and zero otherwise. In this setting, the response of each unit is facilitated via the inner product operation between the effective network weights (as dictated by the latent indicators Z) and the input. We can now express the output y of a stochastic LWTA layer's $(b,u)^{th}$ unit, $y_{b,u}$, that is, the output of the u^{th} unit in the b^{th} block, as:

$$y_{b,u} = \xi_{b,u} \sum_{j=1}^{J} (z_{j,b} \cdot w_{j,b,u}) \cdot x_j \in \mathbb{R} \tag{4.20}$$

where $\xi_{b,u}$ denotes the u^{th} component of ξ_b, and $\xi_b \in one_hot(U)$ holds the b^{th} subvector of ξ. We postulate that the winner unit latent indicators $\xi_b, \forall b$ in 4.17 are obtained via a competitive random sampling procedure; this, translates to drawing samples from a Categorical distribution, where the probabilities are proportional to the intermediate linear computation that each unit performs. Accordingly, the higher the linear response of a particular unit in a particular block, the higher its probability of being the winner in said block; this yields:

$$q(\xi_b) = Categorical(\xi_b | softmax(\sum_{j=1}^{J} z_{j,b} \cdot [w_{j,b,u}]_{u=1}^{U} \cdot x_j)) \tag{4.21}$$

where $[w_{j,b,u}]_{u=1}^{U}$ denotes the vector concatenation of the set$[w_{j,b,u}]_{u=1}^{U}$. On the other hand, we postulate that the binary latent indicators Z are drawn from a Bernoulli distribution, operating in an "on"-"off" fashion, such that:

$$q(z_{j,b}) = Bernoulli(z_{j,b} | \tilde{\pi}) \tag{4.22}$$

where $\tilde{\pi}_{j,b}, \forall j$, b are trainable parameters. A graphical illustration of the proposed stochastic LWTA block is depicted in Fig. 4.4. Each stochastic LWTA layer comprises multiple such LWTA blocks as illustrated in Fig. 4.5(a). At this point, it is important to note a key aspect of the proposed approach; that is, stochasticity. In each layer, different stochastic representations arise due to the sampling procedure for both latent indicators ξ and Z. Even when presented with the same input, different subnetworks may be activated and different subpaths are followed from the input to the output, as a result of winner, ξ, and component utility, Z, sampling.

4.8 A Convolutional Variant

To take into consideration networks that employ the convolutional operation, we formulate a convolutional variant of the proposed Stochastic LWTA rationale. Now we will examine an input tensor $X \in \mathbb{R}^{H \times L \times C}$, where H, L, C are the height, length, and channels of the input. Under this framework, we are setting a group of kernels, each with weights $W_b \in \mathbb{R}^{h \times l \times C \times U}$, where h, l, C, U are the kernel height, length, channels, and a number of competing feature maps; each layer comprises B kernels. Analogously to the grouping and competition of linear units in dense layers, in this case, local competition is performed on a position-wise basis among feature maps. Each kernel is treated as an LWTA block with competing feature maps. The feature maps of each kernel compete to win the activation pertaining to each position; thus, we have as many competition outcomes within the kernel as the number of positions (in the definition of the feature maps).

This way, the latent winner indicator variables now pertain to the selection of winner feature maps on a position-wise basis. Further, we introduce analogous latent utility indicators $z \in [0,1]^B$ in order to infer which kernels (LWTA blocks) are necessary for modeling the available data. Thus, here, if $z_b = 0$, we omit whole blocks of competing feature maps. Under

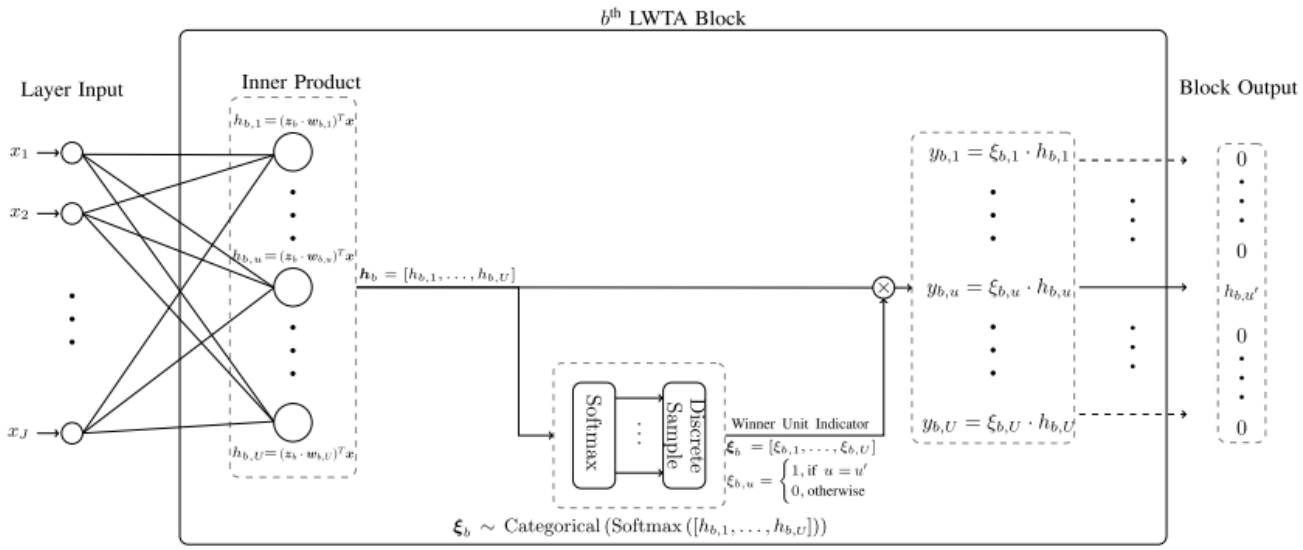


Figure 4.4: A detailed bisection of the b^{th} Stochastic LWTA block in an LWTA layer. Presented with an input $x \in \mathbb{R}^J$, each unit $u = 1, ..., U$ computes its activation $h_{b,u}$ via different weights $w_{b,u} \in \mathbb{R}^J, i.e., h_{b,u} = (z_b \cdot w^T b, u)x$. Here, z_b is the component utility indicator pertaining to the b^{th} block, which encodes which synapses leading to the b^{th} block the inference algorithm deems useful, and which not. The linear responses of the units are concatenated, such that $h_b = [h_{b,1}, ..., h_{b,U}]$, and transformed into probabilities via the softmax operation. Then, a Discrete sample $\xi_b = [\xi_{b,1}, ..., \xi_{b,U}]$ is drawn; this constitutes a one-hot vector with a single non-zero entry at position $u^{'}$, denoting the winner unit in the block. The winner unit passes its linear response to the next layer; the rest pass zero value.

this regard, each feature map $u = 1, ..., U$ in the b^{th} LWTA block (kernel) of a convolutional LWTA layer computes:

$$\mathbf{H}_{b,u} = (z_b \cdot \mathbf{W}_{b,u}) \star \mathbf{X} \in \mathbb{R} \tag{4.23}$$

Then, competition takes place among the U kernel feature maps for claiming the available positions, one by one. Specifically, the competitive random sampling procedure reads:

$$q(\xi_{b,h,l}) = Categorical(\xi_{b,h,l} | softmax(\mathbf{H}_{b,u,h^{'},l^{'}})) \tag{4.24}$$

where $[H_{b,u,h^{'},l^{'}}]_{u=1}^{U}$ denotes the concatenation of the set $H_{b,u,h^{'},l^{'}}{}_{u=1}^{U}$. In each kernel $b = 1, ..., B$ and for each position $h^{'} = 1, ..., H$, $l^{'} = 1, ..., L$,only the winner feature map contains a non-zero entry; all the rest feature maps contain zero values at these positions. This yields sparse feature

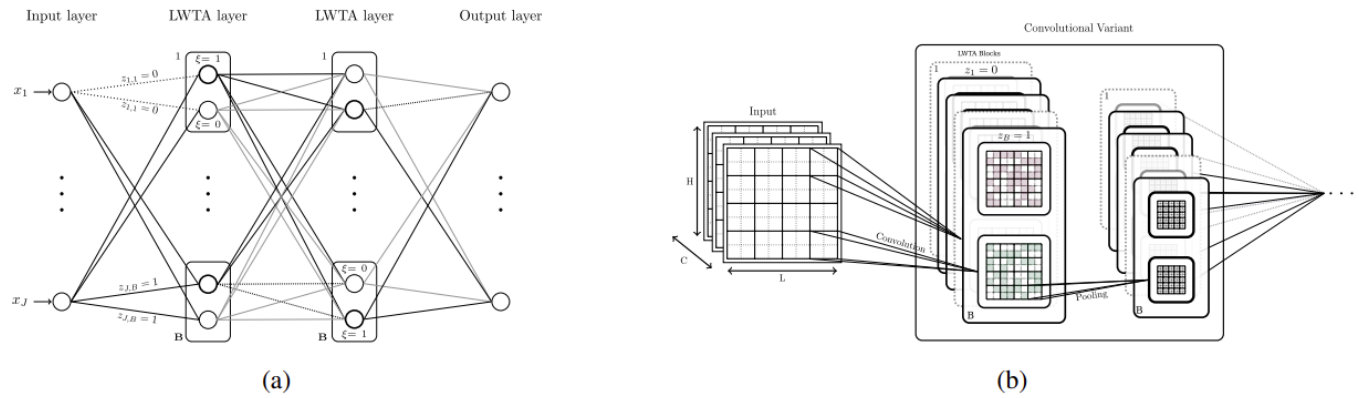


Figure 4.5: (a) A graphical representation of our competition-based modeling approach. Rectangles denote LWTA blocks, and circles the competing units therein.The winner units are denoted with bold contours ($\xi = 1$). Bold edges denote retained connections ($z = 1$). (b) The convolutional LWTA variant. Competition takes place among feature maps on a position-wise basis. The winner feature map at each position passes its output to the next layer, while the rest pass zero values at said position.

maps with mutually exclusive active pixels. Accordingly for the utility indicators:

$$q(z_b) = Bernoulli(z_b|\tilde{\pi}_b)\forall b \tag{4.25}$$

Thus, the output $Y \in \mathbb{R}^{H\times L\times B\cdot U}$ of a convolutional layer of the proposed stochastic LWTA-based networks is obtained via concatenation of the subtensors Y b,u that read:

$$\mathbf{Y}_{b,u} = \Xi_{b,u}((z_b \cdot \mathbf{W}_{b,u}) \star \mathbf{X}) \;\; \forall b,u \tag{4.26}$$

where $\Xi_{b,u} = [\xi_{b,u,h',l'}]_{h',l'=1}^{H,L}$. The corresponding illustration of the proposed stochastic convolutional LWTA block is depicted in Fig. 3. Convolutional stochastic LWTA-based layers comprise multiple such blocks, as shown in Fig. 2b.

4.9 Proposed Approaches

Let consider by t the output of the task, and by r the learned representation of x; in supervised learning, t corresponds to the target label pertaining to an observation x. Most of the approaches in information theory has as a main purpose to maximize a single constraint for example the MI between r which is the representation and t which is the label or target denoted as

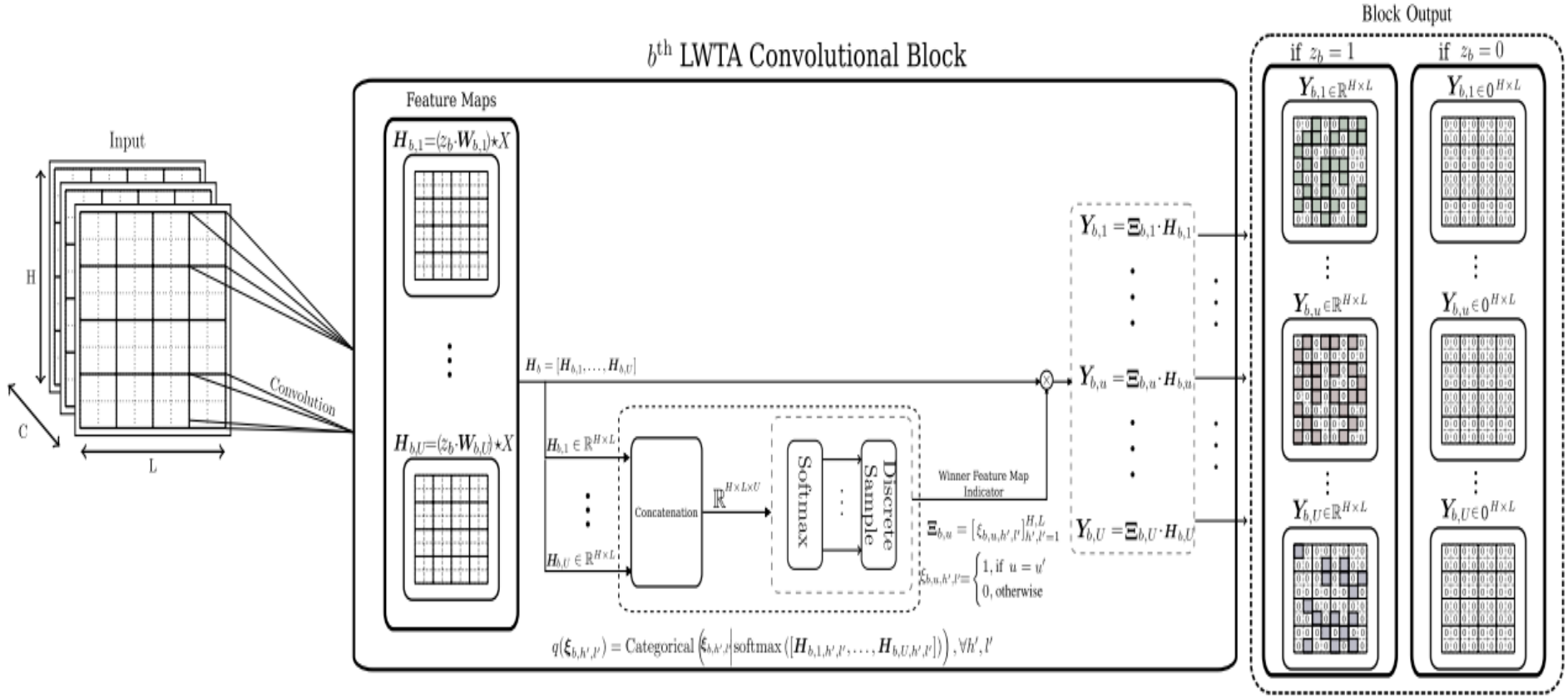


Figure 4.6: A detailed bisection of the b^{th} convolutional stochastic LWTA block. Presented with an input $X \in \mathbb{R}^{H \times L \times C}$,competition now takes place among feature maps on a position-wise basis.Only the winner feature map contains a non-zero entry in a specific position. This leads to sparse feature maps, each comprising uniquely position-wise activated pixels. The component utility indicators z_b encode which kernels/blocks the algorithm deems useful; when $z_b = 0$, whole blocks are omitted.

I(r, t) in contrary to ICP which takes a different path. The promoting of diversification of the emerging representations, r is explicitly split into two different parts, ζ, y; each imposed different constraints: the "information capacity" of ζ is minimized, while the "information capacity" of y is maximized. Under this regard, the objective reads:

$$max[I(r,t) + \alpha I(y,x) - \beta I(\zeta,x)] \tag{4.27}$$

where α, β are regularization constants. I(ζ, x) and I(y, x) are called the mutual information minimization and maximization terms respectively. It is important to note at this point that estimating the MI is very often quite difficult to calculate. Therefore it is necessary, to devise different optimization schemes for each considered term. For example, to minimize I(ζ, x), we introduce a variational approximation Q(ζ) to P(ζ) by giving the following tractable upper bound:

$$I(\zeta,x) \leqslant \mathbf{E}_{z\sim p(x)}[KL[P(\zeta|X)\|Q(\zeta)]] \tag{4.28}$$

where Q(ζ) is a standard Gaussian distribution. The parameters of this distribution, i.e., μ_ζ, σ_ζ , are founded on an amortization scheme, similar to the well-known VAE formulation [86]; therefore, for their estimation are employed deep neural networks. In this framework, before starting to define the full ICP objective, we highlight that all components stemming from the corresponding tractable bounds and optimization schemes described next, e.g., feature extractors, discriminators, and classifiers are originally implemented in [102] via ReLU-based DNNs. On this basis, this work represents an extremely different view, we are using deep networks which are constructed by novel stochastic LWTA arguments, instead of the over-used nonlinearities. Then, we complement this unique latent unit operation with a sparsity-inducing framework that allows determining the best-postulated sub-network configuration. As already discussed, a notable aspect of ICP is the introduction of auxiliary constraints in order to prevent either part from dominating the downstream task. To this end, the separated parts are individually allowed to accomplish t via dedicated MI constraints, namely I(ζ,t) and I(y,t); however, ζ and y are constrained from knowing what each other has learned. This property is achieved through an additional constraint, that is by minimizing I(ζ,y); this pushed y and ζ to be independent of each other. The objective function that we obtained reads:

$$L_{ICP} = max[I(r,t) + \alpha I(y,x) - \beta I(x,\zeta) + I(\zeta,t) + I(y,t) - \gamma I(\zeta,y)] \tag{4.29}$$

where γ is another regularization constant, $I(\zeta,t)$, $I(y,t)$, $I(r,t)$ are called the inference terms, and $I(\zeta,y)$ the predictability minimization term. $I(r,t)$ denotes the (synergy) between the separate parts, aiming to accomplish the downstream task in a synergistic way. Contrarily, the last three terms, i.e., $I(\zeta,t)$, $I(\zeta,y)$, $I(y,t)$ constitute the (competition) aspect of the approach. This conception allows for both competition as well as synergy of the different representation parts, enhancing the information carried by said representations. In the following, we briefly present the required optimization schemes for each term of the objective. In the case of $I(y,x)$ the KL divergence is divergent [102]; thus, we resort to maximization of the Jensen Shannon (JS) divergence. Its variational estimation yields:

$$JS[P(y|x)\|P(y)P(x)] = max[\mathbb{E}_{(y,x)\sim P(y|x)p(x)}[logD(y,x)] + \mathbb{E}_{(\hat{y},x)\sim P(y|x)p(x)}[log(1-D(\hat{y},x))]] \tag{4.30}$$

where D($\cdot$) is a discriminator, estimating the probability of an input pair; (y, x) is the positive pair sampled from $P(y|x)P(x)$ and $(\hat{y},x)$ is the negative pair sampled from P(y)P(x); $\hat{y}$ is a "disorganized" version of y [102]. For the inference term I(r, t), the following lower bound is derived:

$$I(r,t) \geq \mathbb{E}_{x\sim P(r|x)}[\mathbb{E}_{r\sim p(r|x)}[\int P(t|x)\log Q(t|r)dt]] \tag{4.31}$$

where $Q(t|r)$ is a variational approximation of $P(t|r)$, and $P(t|x)$ denotes the distribution of the labels. Thus, 4.28 essentially constitutes the cross-entropy loss. The expressions for the inference terms I(ζ,t) and I(y, t) are similar. Then, the term predictability minimization tries to push the parts of the representation to be independent. For this purpose, we introduce a predictor H that is used to predict y given ζ; by this way, we guide the variable extractor ζ from producing ζ values that can predict y[102]. The same process is followed for y to ζ. This yields:

$$minmax[\mathbb{E}_{j\sim p(j|x)}[H(y|\zeta) + \mathbb{E}_{y\sim p(y|x)}[H(\zeta|y)]] \tag{4.32}$$

A graphical illustration of the described optimization process of the ICP objective of 4.26 is depicted in Fig. 4. Therein, all DNN-based components are founded on the Stochastic LWTA and component utility arguments of Section 2.

4.10 Training & Prediction

The core training objective was defined in the previous section; this stems from the ICP rationale as expressed in 4.26; however, the existence of Stochastic LWTA activations and the component utility mechanism, necessitates the augmentation of the final objective via appropriate KL terms. Without loss of generality, we begin by considering a symmetric Categorical distribution for the latent variable indicators ξ_b; hence, $p(\xi_b) = Categorical(1/U) \forall b$ for the dense layers, and $p(\xi_{b,h',l'}) = Categorical(1/U), \forall b, h', l'$ for convolutional ones. Differently, for the latent utility indicators Z (or z) we do not impose a symmetric prior; instead, we turn to the non-parametric Bayesian framework and specifically to the Indian Buffet Process (IBP)[119]. This constitutes a sparsity-promoting prior; at the same time, its so-called stick-breaking process (SBP)[120] renders IBP amenable to ¨Variational Inference. The hierarchical construction reads:

$$p(z_{j,b}) = Bernoulli(\pi_b) \;\; \pi_b = \prod_{i=1}^{b} u_b \;\; u_b \sim Beta(\omega, 1) \tag{4.33}$$

where ω is a non-negative constant, controlling the induced sparsity. The SBP requires an additional set of latent stick variables u_b, $\forall b$. Since these are Beta-distributed, we assume a posterior of similar form: $q(u_b) = Beta(u_b|\tilde{a}_b, \tilde{b}_b)$, where $\tilde{a}_b, \tilde{b}_b$, $\forall$ b are trainable variational parameters. This yields:

$$L = L_{ICP} - KL[q(\xi)||p(\xi)] - KL[q(z)||p(z)] - KL[q(u)||p(u)] \tag{4.34}$$

For training, we perform Monte-Carlo sampling to estimate (4.31) using a single reparameterized sample for each latent variable. For the Gaussian distributed variables, e.g. ζ, we resort to the well-known Gaussian trick. For *xi* and Z, these are obtained via the continuous relaxation of the Categorical and Bernoulli distribution [117]. In the following, we describe the reparameterization trick for the ξ variables of dense layers; the cases for Z and for the convolutional variant are analogous. Let $\tilde{\xi}$ denote the probabilities of q(ξ) (Eqs.(4.18),(4.21)). Then, the samples $\hat{\xi}$ can be expressed as:

$$\hat{\xi}_{b,u} = softmax(\frac{\log \tilde{\xi}_{b,u} + g_{b,u}}{\tau}) \;\; \forall b, u \tag{4.35}$$

where $g_{b,u} = -\log(-\log V_{b,u}), V_{b,u} \sim Uniform(0,1)$ and $\tau \in (0, \infty)$ is a temperature constant that controls the degree of the approximation. Alike, the Beta distribution of the stick variables

u is not easily reparameterize; we get samples through the Kumaraswamy distribution for this kind of variables [121]; this constitutes an approximation of Beta and admits the following reparameterization trick:

$$\hat{u}_b = (1-(1-G)^{\frac{1}{\tilde{a}_b}})^{\frac{1}{\tilde{b}_b}} \tag{4.36}$$

where $G \sim Uniform(0,1)$ and $\tilde{a}_b$, $\tilde{b}_b$ are the variational parameters of the original Beta distribution. We can now compute each expectation term in the objective (4.31) via these samples. For example, we can write the KL divergence term for the latent variables ξ as:

$$KL[q(\xi_b)||p(\xi_b)] = \mathbb{E}_{q(\xi_b)}[\log q(\xi_b) - \log(p(\xi_b))] \approx \log q(\hat{\xi}_b) - \log p(\hat{\xi}_b), \forall b \tag{4.37}$$

During prediction time we draw samples directly from the trained posteriors q(ξ) and q(z), in order to determine the winner in each block of the network and to evaluate component utility respectively. Thus, each time we sample, even for the same input, a different sub-path may be followed according to the outcomes of the sampling processes. This leads to a stochastic alternation of the emerging representations of the network at each forward pass.

4.11 Experimental Evaluation

We perform a rigorous experimental evaluation in order to assess the efficacy of the proposed framework. To this end, we evaluate our model on image classification, where we consider two popular benchmarks, namely CIFAR-10 and CIFAR-100 [122] containing natural images with 10 and 100 classes respectively. We compare our approach to recent information-theoretic approaches to deep networks including VIB [106], DIM[111] and ICP [102]. To this end, we follow the same experimental setup of[102] and consider four different networks for both datasets: (i) VGG-16 [3], (ii)GoogLeNet[54], (iii) ResNet20 [123] and (iv) DenseNet40[124]. We employ some common data augmentation procedures, including random cropping and mirroring.

4.11.1 Experimental Setup

We consider two different setups To evaluate our approach,for our competition that relied on LWTA activations: (i) an architecture comprising LWTA blocks with $U = 2$ competing

units/feature maps, denoted as ICP_{LWTA-2}; and (ii) an architecture comprising LWTA blocks with $U = 4$ competing units/feature maps, denoted as - ICP_{LWTA-4}. In all cases, the total number of hidden units/features in each layer remain the same as in the original ReLU-based architectures. The comparability is ensuring (size-wise) by existing approximations. We also compare the performance when we employ the proposed, IBP-driven, network sampling mechanism, and when we omit it. For clarity, in the following, we denote our full model (which employs the IBP-based mechanism) as $ICP_{IBP\&LWTA-2}$ and $ICP_{IBP\&LWTA-4}$, respectively. In this context, to allow for compressing the model by exploiting the trained posteriors over the latent indicators z, we adopt the following rationale: After network training, we introduce a cut-off threshold $\tau = 0.001$. All components with trained utility posterior $\widetilde{\pi} \triangleq q(z = 1) \leq \tau$ are removed from the network; all rest are retained and used at prediction time. In the following tables, the compression metric corresponds to the ratio of the number of network components removed to the total number of network components. We choose an uninformative Beta prior for the IBP (4.30): Beta(1, 1); thus, $\omega = 1$. For the Gumbel-Softmax relaxation, we set the temperatures τ for the prior and posterior distributions to 0.5 and 0.67 respectively[24]. We draw a reparameterized sample from all the involved random variables through training , while we draw 5 different samples during inference (and proceed with Bayesian averaging). We execute a a lot of evaluations for each dataset and architecture, i.e., 5 runs, and report the best performing one. All experiments were run on a workstation with 2 $\times$ Quadro P6000 24GB GPUs and 64GB RAM.

4.11.2 Detailed Experimental Setup

To assess the efficacy of the proposed approach, we evaluate the performance on image classification, following the experimental setup of [102].Specifically, we consider four different networks, namely (i) VGG-16 [3], (ii) GoogLeNet [54], (iii) ResNet20 [125], and (iv) DenseNet40[124]. The original architectures are described in their respective papers; we consider CIFAR-10 and CIFAR-100 datasets [122]. To further diversify the emerging representations of the LWTA-based networks, we resort to recent information theoretics approaches and specifically to ICP [102]. The objective reads:

$$L_{ICP} = max[I(r,t) + \alpha I(y,x) - \beta I(z,x) + I(z,t) + I(y,t) - \gamma I(z,y)] \tag{4.38}$$

where α, β and γ are regularization constants. The original ICP approach uses Deep Networks that employ conventional based in Relu activation function. To allow for transparency and

comparability with these networks, we use the same regularization constants as found in the official code implementation of the ICP approach 1. We report the used values for each architecture and dataset in Table

Table 4.1: Values of the regularization parameters of the ICP objective for all architectures and dataset

Model	VGG-16	GoogLeNet	ResNet20	DenseNet40
cline2-5	$\alpha \parallel \beta \parallel \gamma$			
CIFAR-10	0.01 ‖ 0.1 ‖ 0.01			
CIFAR-100	0.001 ‖ 0.1 ‖ 0.001	0.01 ‖ 0.1 ‖ 0.01	0.01 ‖ 0.1 ‖ 0.01	0.001 ‖ 0.1 ‖ 0.001

Concerning the initialization of the parameters for each network layer, we use Xavier/Glorot initialization for the weights of the Fully Connected layers, and [125] initialization [123] for the weights of the Convolutional ones. For both types of layers, we set the initial estimates for the biases to zero. To avoid overflow/underflow in the computations, after each parameter update, we constrain the values of the parameters of the Kumaraswamy distribution[121] to lie in $[1e-4,1]$ and the parameters of the Bernoulli/Gumbel-Softmax posterior $\hat{\pi} \in [1e-4,1]$. This allows for a stable training regime, which avoids precision errors, without harming the parameter estimation process. In order to estimate model parameters, we draw a single (reparameterized) sample from all the involved random variables during training, while we draw 5 different samples during inference and proceed with Bayesian averaging.

4.11.3 Experimental Results

CIFAR-10.For CIFAR-10, in order to have transparency and as accurate results as possible in comparability we train the networks in a different number of optimization parameters and epochs, following the original implementation of ICP [102] .The obtained comparative results for all considered methods and networks can be found in Table 1. As we observe, our proposed method yields comparable classification accuracy to the best performing baseline ICP alternative, while at the same time allowing for powerful compression of the considered architecture. Apart from the computational benefits of the compression capabilities of the proposed approach, using the component omission mechanism allows for inferring the relevant parts of the input, further facilitating the diversification process of the emerging representations.

CIFAR-100.For CIFAR-100 the procedure is similar to the above on CIFAR-10. Table 2

depicted the results of the classification problem. In this set of experiments, the increased classification capabilities of the baseline ICP model, that is the conventional ICP model with ReLU-based nonlinearities, compared to the other information-theoretic approaches is more evident. Our proposed approach, once again follows this trend, yielding on-par or better classification accuracy with ICP while at the same time showing significant compression capabilities.

Random Seed Effect. A core aspect of the considered approach lies in its stochastic nature. Thus, to evaluate the overall robustness of our proposed method, we execute each experiment 5 times and report the means and standard deviations and these are presented in Table 3. We observe that our approach exhibits consistent performance in all cases, while in most occasions, the mean performance obtained by multiple runs outperforms the baseline ICP approach.

Here, we focus on the VGG-16 architecture described in the previous section, and assess the individual impact of each of the proposed components, i.e. LWTA and IBP, to the classification performance of the network. At the same time, we examine whether adoption of a deterministic LWTA scheme, as opposed to the adopted stochastic construction, would yield equal or inferior performance in these benchmarks.

The obtained comparative results are depicted in Table 4. Therein, the $ICP_{LWTA-*max}$ and $ICP_{IBP\&LWTA-*max}$ entries correspond to networks where the LWTA competition function picks the unit with greatest activation value, in a deterministic fashion; $*$ = 2, 4 denotes the number of competitors in each LWTA block. We take into consideration both, the use and omission of the IBP-based mechanism, respectively. As we notice, for U=2 and U=4 settings, the Stochastic LWTA approach outperforms deterministic LWTA units picking the greatest value. The mechanism that IBP is based on increases classification accuracy (in addition to compressing the network we perform inference with).

4.11.4 Representation Diversification

This work is aiming to enrich and sufficiently diversify the representations for deep networks. Below, we try to investigate the diversification capabilities of the proposed framework, both quantitatively and qualitatively. To execute this analysis, and due to space limitations, we focus on the CIFAR-10 architecture and the VGG-16 dataset.

To perform a qualitative evaluation, we visually compare the emerging intermediate representations from the hidden layers of employed ReLU-based networks (standard ICP) and

our approach. As we observe in Fig. 5, there exists a clear disparity between the proposed approach and the commonly employed nonlinearities. Clearly, the ReLU-based architecture allows for more "aesthetically pleasing" representations. However, the proposed networks yield significant diversification, as they split the resulting representations from each block to mutually exclusive parts. This constitutes a radically different RL scheme with significant diversification capacity for the emerging intermediate representations. For the quantification of the diversification properties, we turn to the commonly used Linear Separability metric. This metric constitutes a proxy for quantifying the disentanglement and MI between the emerging representations and the class labels ([111]). In this context, linear classification is usually considered, and for this we use the standard Support Vector Machine (SVM) approach. To this end, we hold the encoders of y and ζ fixed, and build separate SVM based classifiers on the two representation parts ζ , y, their combination (denoted as Total), as well as the output of the last convolutional layer of the ζ encoder (Conv), which stems either from ReLU units or LWTA blocks.

The results which are obtained are comparative and separable is illustrated in figure 4.10. We observe that the accuracy in the case of training a linear model SVM with representations a and b separately or the combination of these two or the last convolutional layer of the underlying encoder is much greater when inputs come from the networks of our own approximation (LWTA & IBP arguments). This suggests the existence of more diverse representations, that enrich the information and are able to be utilized by the classifier much better.

Below, we provide some further qualitative evaluations of the emerging intermediate representations when employing the described competition-based mechanism, namely Stochastic Local Winner- Takes-All. To this end, in Figs. 4.15-4.22 we visualize the outputs from different blocks of competing feature of the ResNet-20 and DenseNet40 architectures; we focus on various test examples of the CIFAR-10 dataset. Similar to the visualization of the VGG-16 architecture provided in the main text, we observe that the emerging feature maps result in almost mutually exclusive representations of the input. These visualizations suggest that our approach yields networks with significant diversification capacity.

4.11.5 Further Investigation

To gain some further insights into the behavior of the employed Stochastic LWTA activations and to assess that the competition mechanism does not collapse to singular, always-winning units, we examine the arising competition patterns in a random layer of the ζ encoder of the

VGG-16 architecture. The graphical illustration is provided in Fig. 9. As we observe, each class (y-axis) exhibits a unique distribution of unit activation. This suggests that the introduced mechanism succeeds in encoding diverse and salient discriminative information through the arising patterns.

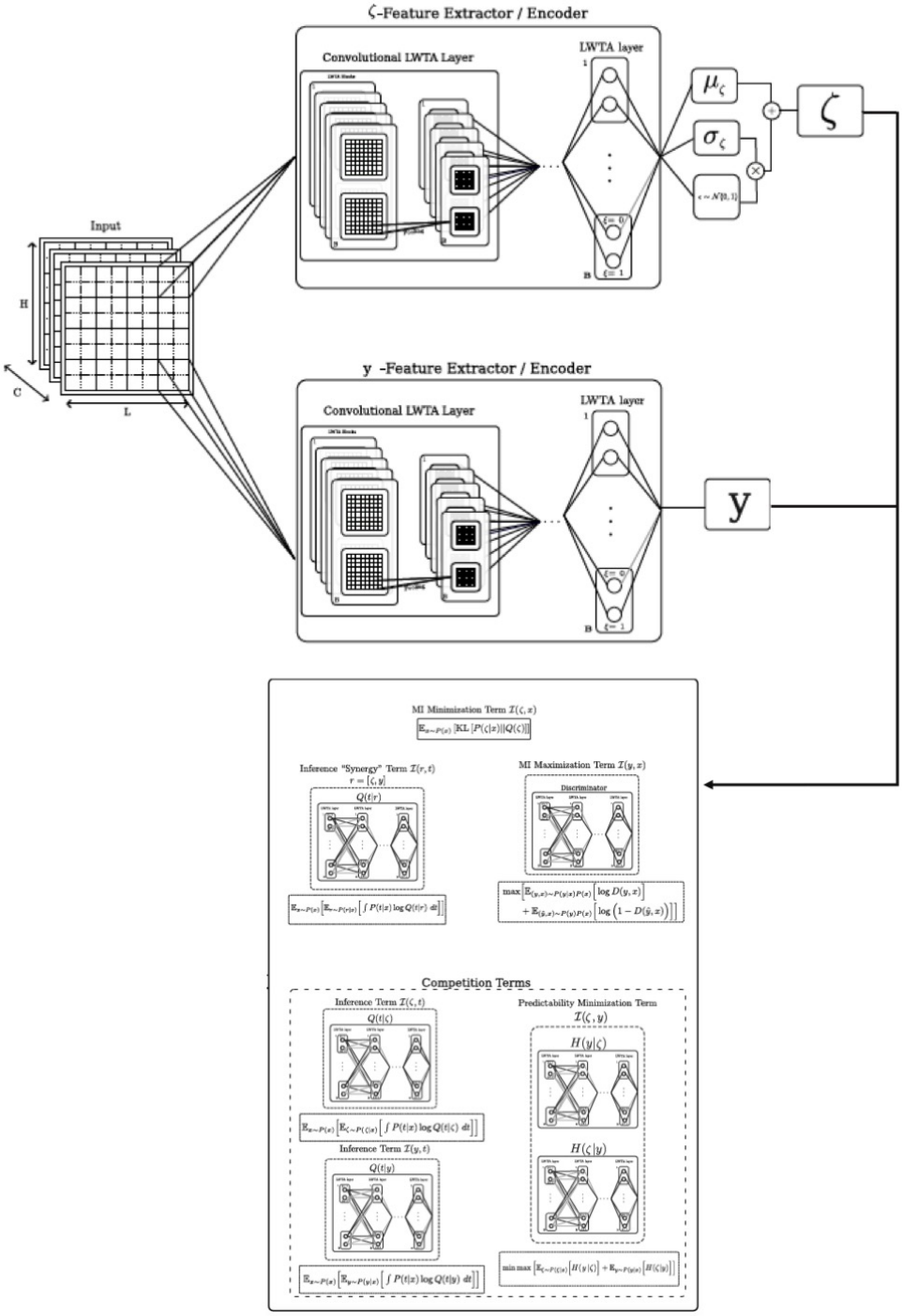


Figure 4.7: The ICP pipeline: r is splitting into two parts [ζ, y]. These two parts except the cooperation are entered into the competition process in order to accomplish the downstream task. This helps us to compute the MI between different pairs of variables; this is usually intractable and optimized through different schemes. All components of these, are implemented via LWTA and IBP-based DNNs.

Model	VGG-16	GoogLeNet	ResNet20	DenseNet40
	Error (%) || Compression (%)			
Baseline	6.67 || 0.00	4.92 || 0.00	7.63 || 0.00	5.83 || 0.00
VIB (Alemi et al. 2017)	6.81 || 0.00	5.09 || 0.00	6.95 || 0.00	5.72 || 0.00
DIM*(Hjelm et al. 2019b)	6.54 || 0.00	4.65 || 0.00	7.61 || 0.00	6.15 || 0.00
$\text{VIB}_{\times 2}$	6.86 || 0.00	4.88 || 0.00	6.85 || 0.00	6.36 || 0.00
$\text{DIM*}_{\times 2}$	7.24 || 0.00	4.95 || 0.00	7.46 || 0.00	5.60 || 0.00
$\text{ICP}_{\text{-ALL}}$	6.97 || 0.00	4.76 || 0.00	6.47 || 0.00	6.13 || 0.00
$\text{ICP}_{\text{-COM}}$	6.59 || 0.00	4.67 || 0.00	7.33 || 0.00	5.63 || 0.00
ICP	6.10 || 0.00	**4.26** || 0.00	6.01 || 0.00	4.99 || 0.00
$\text{ICP}_{\text{IBP \& LWTA-2}}$	**6.01** || **40.4**	4.31 || **35.2**	**5.94** || **37.1**	**4.78** || **32.0**
$\text{ICP}_{\text{IBP \& LWTA-4}}$	7.02 || 30.4	4.74 || 28.2	6.30 || 31.3	5.61 || 26.3

Figure 4.8: Table 1: Classification Error rate(%) on CIFAR-10

Model	VGG-16	GoogLeNet	ResNet20	DenseNet40
	Error (%) || Compression (%)			
Baseline	26.41 || 0.00	20.68 || 0.00	31.91 || 0.00	27.55 || 0.00
VIB (Alemi et al. 2017)	26.56 || 0.00	20.93 || 0.00	30.84 || 0.00	26.37 || 0.00
DIM*(Hjelm et al. 2019b)	26.74 || 0.00	20.94 || 0.00	32.62 || 0.00	27.51 || 0.00
$\text{VIB}_{\times 2}$	26.08 || 0.00	22.09 || 0.00	29.74 || 0.00	29.33 || 0.00
$\text{DIM*}_{\times 2}$	25.72 || 0.00	21.74 || 0.00	30.16 || 0.00	27.15 || 0.00
$\text{ICP}_{\text{-ALL}}$	26.73 || 0.00	20.90 || 0.00	28.35 || 0.00	27.51 || 0.00
$\text{ICP}_{\text{-COM}}$	26.37 || 0.00	20.81 || 0.00	32.76 || 0.00	26.85 || 0.00
ICP	24.54 || 0.00	**18.55** || 0.00	28.13 || 0.00	24.52 || 0.00
$\text{ICP}_{\text{IBP \& LWTA-2}}$	**24.35** || **32.0**	19.00 || **29.0**	**28.02** || **31.5**	**24.44** || **29.2**
$\text{ICP}_{\text{IBP \& LWTA-4}}$	25.44 || 22.0	20.12 || 26.5	29.34 || 21.2	25.07 || 24.4

Figure 4.9: Table: 2 Classification Error rate(%) on CIFAR-100

CIFAR-10				
	Mean (%) || Standard Deviation (%)			
Model	VGG-16	GoogLeNet	ResNet20	DenseNet40
$\text{ICP}_{\text{IBP \& LWTA-2}}$	6.07 || 0.04	4.35 || 0.04	6.00 || 0.06	4.87 || 0.05
$\text{ICP}_{\text{IBP \& LWTA-4}}$	7.12 || 0.08	4.84 || 0.06	6.42 || 0.10	5.70 || 0.08
CIFAR-100				
	Mean (%) || Standard Deviation (%)			
Model	VGG-16	GoogLeNet	ResNet20	DenseNet40
$\text{ICP}_{\text{IBP \& LWTA-2}}$	24.52 || 0.12	19.25 || 0.14	28.10 || 0.05	24.52 || 0.05
$\text{ICP}_{\text{IBP \& LWTA-4}}$	25.59 || 0.08	20.28 || 0.10	29.51 || 0.10	25.19 || 0.08

Figure 4.10: Table 3: Means and standard deviations of 5 different runs under different seeds for all datasets and architectures.

Model	VGG-16
Baseline	6.67
ICP	6.10
$\text{ICP}_{\text{LWTA-2}^{\max}}$	6.34
$\text{ICP}_{\text{LWTA-2}}$	6.23
$\text{ICP}_{\text{IBP \& LWTA-2}^{\max}}$	6.27
$\text{ICP}_{\text{IBP \& LWTA-2}}$	**6.01**
$\text{ICP}_{\text{LWTA-4}^{\max}}$	7.01
$\text{ICP}_{\text{LWTA-4}}$	6.85
$\text{ICP}_{\text{IBP \& LWTA-4}^{\max}}$	7.32
$\text{ICP}_{\text{IBP \& LWTA-4}}$	7.02

Figure 4.11: Table 4: Ablation Study: CIFAR-10 test-set using a VGG-16 (Simonyan and Zisserman 2015) [3] architecture.

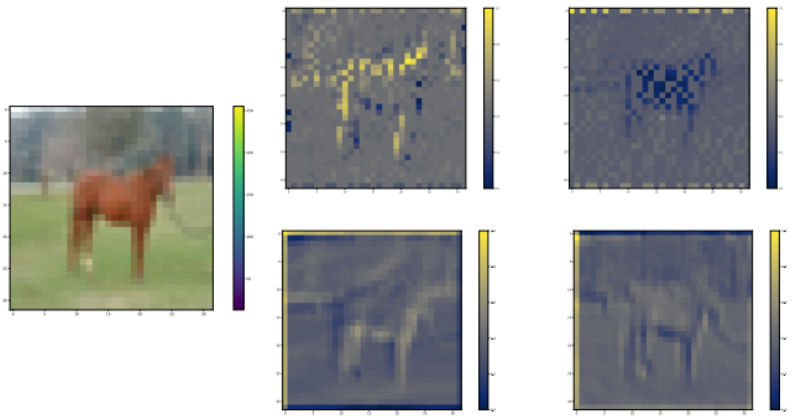


Figure 4.12: Feature map visualizations for a test example of the CIFAR-10 dataset, emerging from the first layer of the considered VGG-16 architecture. (Left) The original image, (Upper Row) a visualization of the outputs of the two competing feature maps of the first LWTA block (U = 2), and (Bottom Row) a visualization of the first two filters of the conventional ReLU-based approach. The latter exhibit significant overlap, contrary to the much diverse representations of our approach which are mutually-exclusive.

Method	Proxies			
	SVM(ζ)	SVM(y)	SVM(Total)	SVM(Conv)
ICP	91.5	91.9	92.9	31.2
$\text{ICP}_{\text{LWTA-2 \& IBP}}$	**91.9**	**92.4**	**93.4**	**32.2**

Figure 4.13: Results on linear separability using SVMs.

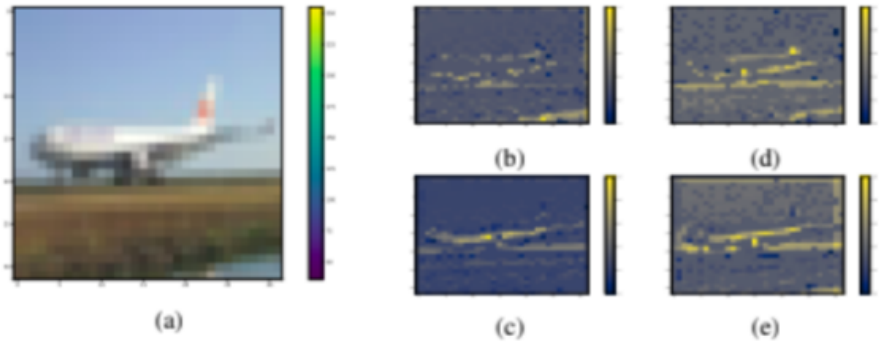


Figure 4.14: Feature map visualizations for a test example from class 0 of the CIFAR-10 dataset, emerging from the first layer of the considered ResNet-20 architecture. (a) The original image, (b)-(c) a visualization of the outputs of the two competing feature maps of the first LWTA block (U =2), and (d)-(e) a visualization of the ouputs of the second block.

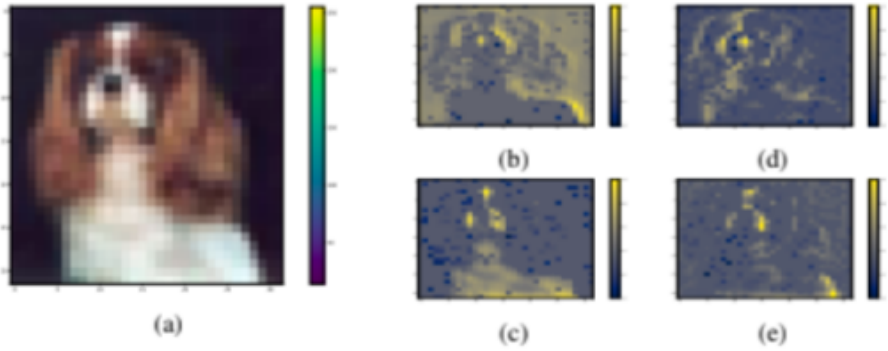


Figure 4.15: Feature map visualizations for a test example from class 5 of the CIFAR-10 dataset, emerging from the first layer of the considered ResNet-20 architecture. (a) The original image, (b)-(c) a visualization of the outputs of the two competing feature maps of the first LWTA block (U = 2), and (d)-(e) a visualization of the ouputs of the second block.

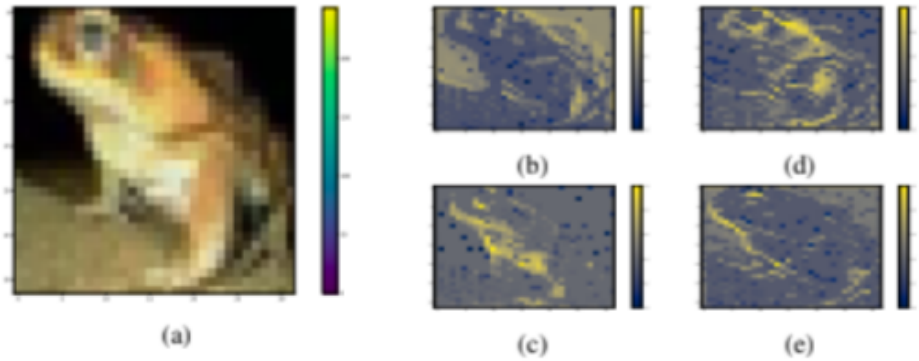


Figure 4.16: Feature map visualizations for a test example from class 6 of the CIFAR-10 dataset, emerging from the first layer of the considered ResNet-20 architecture. (a) The original image, (b)-(c) a visualization of the outputs of the two competing feature maps of the first LWTA block (U =2),and (d)-(e) a visualization of the ouputs of the second block.

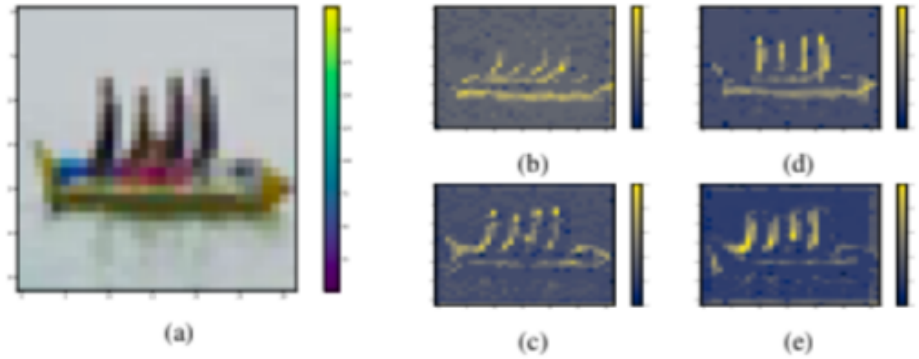


Figure 4.17: Feature map visualizations for a test example from class 8 of the CIFAR-10 dataset, emerging from the first layer of the considered ResNet-20 architecture. (a) The original image, (b)-(c) a visualization of the outputs of the two competing feature maps of the first LWTA block (U = 2),and (d)-(e) a visualization of the ouputs of the second block.

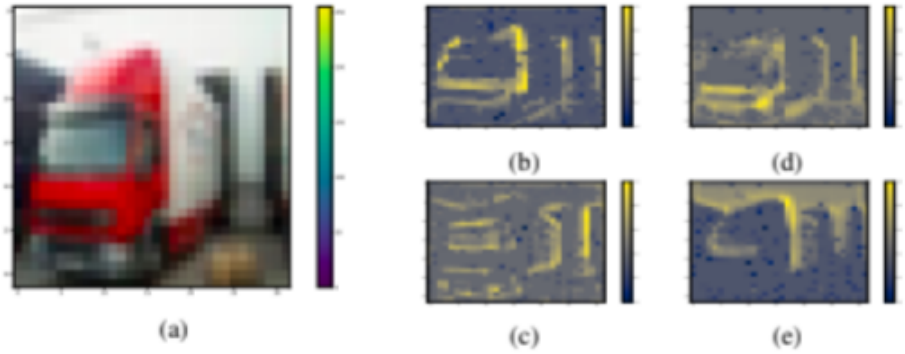


Figure 4.18: Feature map visualizations for a test example from class 8 of the CIFAR-10 dataset, emerging from the first layer of the considered ResNet-20 architecture. (a) The original image, (b)-(c) a visualization of the outputs of the two competing feature maps of the first LWTA block (U = 2), and (d)-(e) a visualization of the ouputs of the second block.

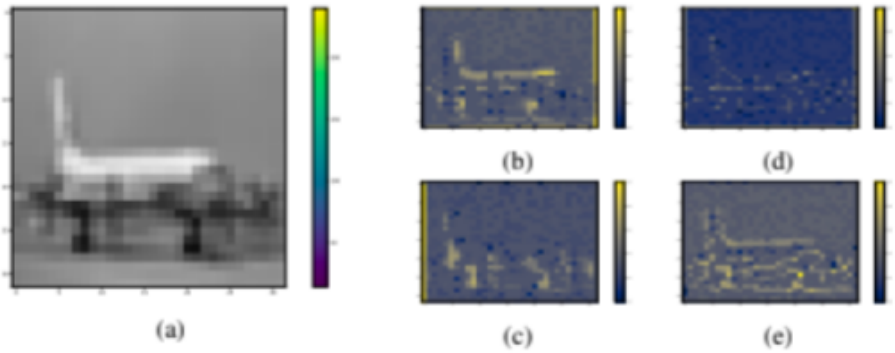


Figure 4.19: Feature map visualizations for a test example from class 9 of the CIFAR-10 dataset, emerging from the first layer of the considered ResNet-20 architecture. (a) The original image, (b)-(c) a visualization of the outputs of the two competing feature maps of the first LWTA block (U = 2), and (d)-(e) a visualization of the ouputs of the second block.

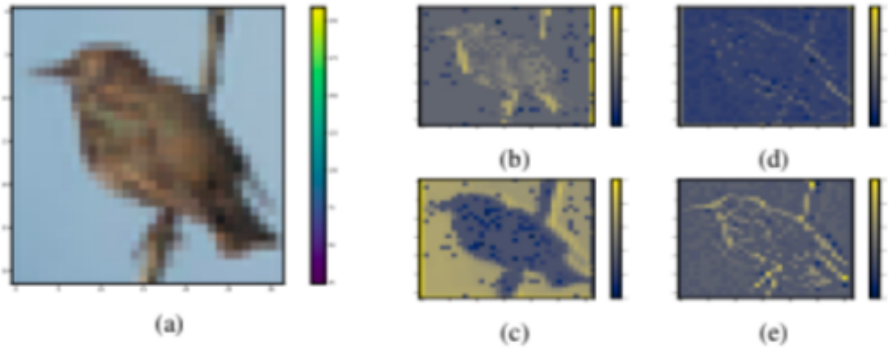


Figure 4.20: Feature map visualizations for a test example from class 2 of the CIFAR-10 dataset, emerging from the first layer of the considered DenseNet40 architecture. (a) The original image, (b)-(c) a visualization of the outputs of the two competing feature maps of the first LWTA block (U = 2), and (d)-(e) a visualization of the outputs of the third block.

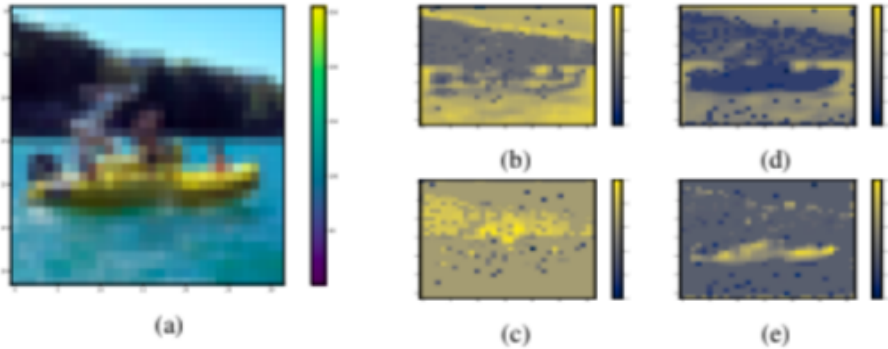


Figure 4.21: Feature map visualizations for a test example from class 8 of the CIFAR-10 dataset, emerging from the first layer of the considered DenseNet40 architecture. (a) The original image, (b)-(c) a visualization of the outputs of the two competing feature maps of the first LWTA block (U = 2), and (d)-(e) a visualization of the outputs of the third block.

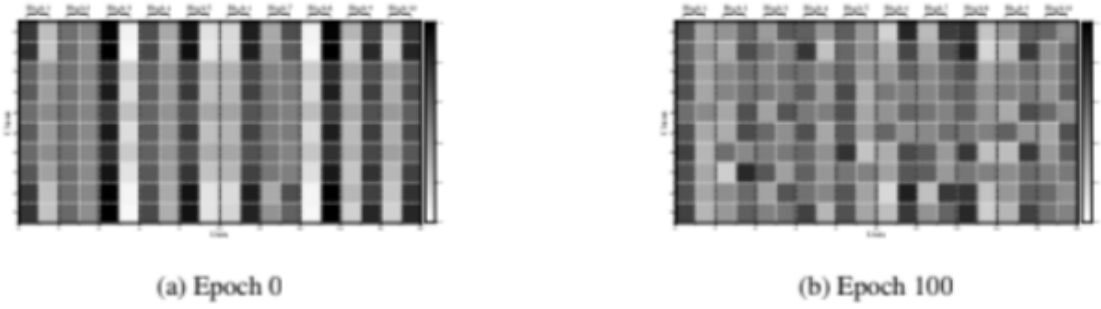


Figure 4.22: Changes in the probabilities of unit activation per class during training. Darker denotes probabilities closer to 1, while white denotes a 0 probability of activation.

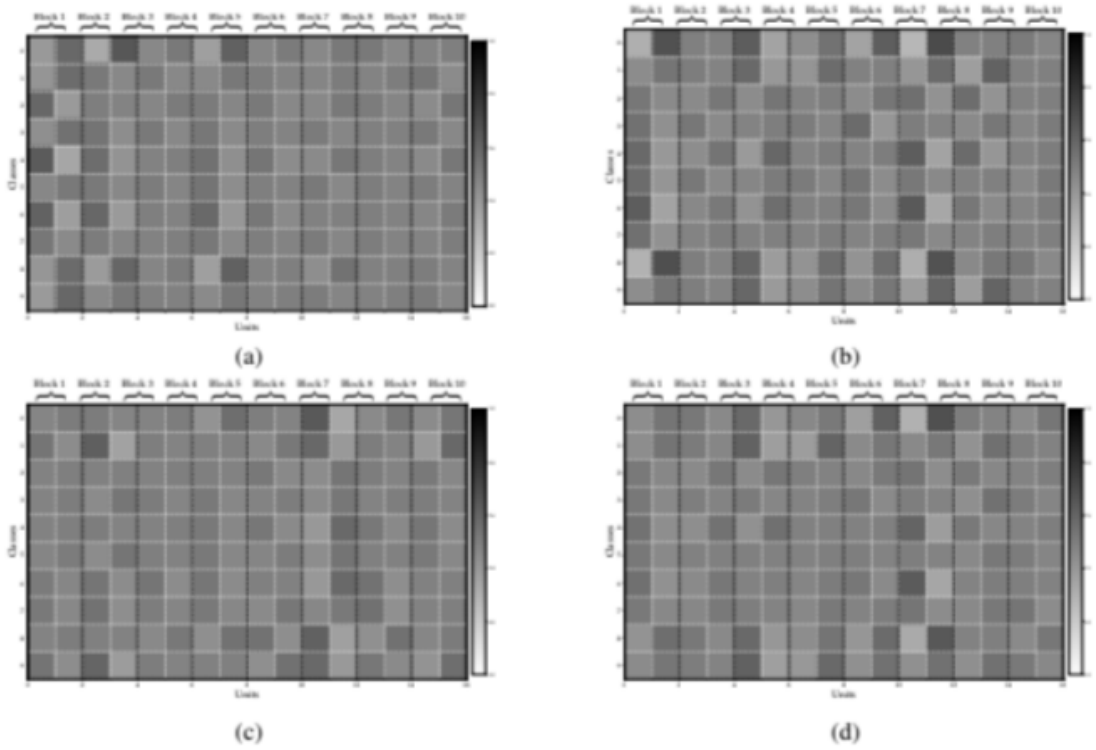


Figure 4.23: Probabilities of unit activation per class for various layers of the considered ResNet-20 architecture. Darker denotes probabilities closer to 1, while white denotes a 0 probability of activation.

Chapter 5

Conclusion

The deep neural networks from the beginning of their discovery address several problems, the most important being the generalization. This is due to the fact that the representations are not good enough and is difficult to generalize to unknown data also problems such as adversarial vulnerability occurs frequently.

Thus, we rely on a new innovation characterized by local stochasticity. We deepen in this approach with important innovations for these networks.

What we achieved was to combine the stochastic local with information-theoretic arguments in order to train the model which led us to many better-learned representations and secondly we constructed neural networks that are much better from the state of the art.

In this work, we attacked the problem of promoting diversified representations in Deep Learning. To this end, we introduced principled network arguments formulated by stochastic competition-based Local Winner-Takes-All activations. We combined these with network component utility mechanisms, which rely on the use of the IBP prior. Then, to further enrich the emerging representations, we employed information-theoretic arguments, founded on competing for mutual information constraints under the Information Competing Process.

This results in an efficient network training and prediction scheme, that significantly compresses the networks during prediction. We performed a thorough experimental evaluation, using benchmark datasets and several standard network architectures. We compared networks crafted using the proposed arguments against standard, ReLU-based constructions. Our experimental results provided strong empirical evidence of the efficacy of the proposed framework.

Specifically, in all cases, our approach yielded on-par or improved accuracy for significantly compressed networks. At the same time, our qualitative and quantitative analysis of the obtained representations showed our approach results in representations that: (i) visually appear much more diverse; and (ii) are more informative to a linear classifier trained on them, specifically an SVM used as a proxy to linear separability.

5.1 Future Work

In many real life situations, as we have shown in this thesis, we rely on mathematical models to help us to confront a variety of difficult tasks. Machine learning advances have enabled these models to develop and, in some cases, even outperform their human counterparts.In this setting, the question arises: Can we derive a perfect model that can address any real-world problem? An optimal model, in our opinion, is one that has the plasticity required to cope with uncertainty in a flexible manner. Bayesian inference is a tool for achieving that objective. It is capable of effectively counteracting the observable data epistemic uncertainty. We looked at a variety of ways to use Bayesian inference in model training in this thesis. As we have shown, our method can help models perform better in classification tasks. Specifically, Bayesian approaches were applied to the three key components of deep networks for the winner of each block the component utility mechanism and in the case in which we tried to enrich the representations. Machine learning is a dynamic field of research that developing and introducing new techniques continuously. For instance, new frameworks are developed like Transformers networks. Transformer is the first sequence model based purely on attention mechanism, with multi-headed self-attention replacing the recurrent layers most typically employed in encoder-decoder networks. The intersection between our contributions and these kind of networks we consider to have great potential for substantial breakthroughs in the future.

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[2] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, Apr 2017.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[4] Ulrich Paquet. Bayesian inference for latent variable models. Technical Report UCAM-CL-TR-724, University of Cambridge, Computer Laboratory, July 2008.

[5] N. Metropolis and S. Ulam. The monte carlo method. *J. Am. Stat. Assoc.*, 44:335, 1949.

[6] W. R. Gilks, N. G. Best, and K. K. C. Tan. Adaptive rejection metropolis sampling within gibbs sampling. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 44(4):455–472, 1995.

[7] Charles J. Geyer. Practical markov chain monte carlo. *Statistical Science*, 7(4):473–483, 1992.

[8] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[9] Christian P. Robert. The metropolis-hastings algorithm, 2016.

[10] Samik Raychaudhuri. Introduction to monte carlo simulation. In *2008 Winter Simulation Conference*, pages 91–100, 2008.

[11] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, November 1999.

[12] Hagai Attias. A variational baysian framework for graphical models. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 2000.

[13] Wei Han and Yun Yang. Statistical inference in mean-field variational bayes, 2019.

[14] Chong Wang and David M. Blei. Variational inference in nonconjugate models, 2012.

[15] L. K. Saul, T. Jaakkola, and M. I. Jordan. Mean field theory for sigmoid belief networks, 1996.

[16] David Barber and Wim Wiegerinck. Tractable variational structures for approximating graphical models. pages 183–189, 01 1998.

[17] Matt Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference, 2013.

[18] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[19] W. Nowak. Introduction to stochastic search and optimization. estimation, simulation, and control (spall, j.c.; 2003) [book review]. *IEEE Transactions on Neural Networks*, 18(3):964–965, 2007.

[20] Léon Bottou. On-line learning and stochastic approximations. 1999.

[21] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference, 2016.

[22] Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In Samuel Kaski and Jukka Corander, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 814–822, Reykjavik, Iceland, 22–25 Apr 2014. PMLR.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[24] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables, 2017.

[25] Mark Ebden. Gaussian processes: A quick introduction, 2015.

[26] Y. W. Teh and M. I. Jordan. Hierarchical Bayesian nonparametric models with applications. In N. Hjort, C. Holmes, P. Müller, and S. Walker, editors, *Bayesian Nonparametrics: Principles and Practice*. Cambridge University Press, 2010.

[27] Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT press, 1996.

[28] Mikhail Lifshits. Lectures on gaussian processes. 2012.

[29] R. M. Dudley. Review: I. a. ibragimov and yu. a. rozanov, gaussian random processes. *Bull. Amer. Math. Soc. (N.S.)*, 2(2):373–378, 03 1980.

[30] J.Q. Shi and T. Choi. *Gaussian process regression analysis for functional data*. 07 2011.

[31] Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

[32] Neil Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse gaussian process methods: The informative vector machine. volume 15, pages 609–616, 01 2002.

[33] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1177–1184, USA, 2007. Curran Associates Inc.

[34] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[35] Thomas S. Ferguson. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2):209 – 230, 1973.

[36] Zoubin Ghahramani and Thomas Griffiths. Infinite latent feature models and the indian buffet process. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2006.

[37] Trevor Campbell, Saifuddin Syed, Chiao-Yu Yang, Michael I. Jordan, and Tamara Broderick. Local exchangeability, 2021.

[38] Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[39] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[40] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression, 2017.

[41] Jocelyn Sietsma and Robert J.F. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67–79, 1991.

[42] Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. Analyzing noise in autoencoders and deep networks, 2014.

[43] Kam Chuen Jim, C. Lee Giles, and Bill G. Horne. An analysis of noise in recurrent neural networks: Convergence and generalization. *IEEE Transactions on Neural Networks and Learning Systems*, 7(6):1424–1438, 1996.

[44] Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[45] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[46] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, Jan 2015.

[47] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[48] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, Mar 2020.

[49] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2021.

[50] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, jul 2006.

[51] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

[52] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[54] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[55] Tom Charnock, Laurence Perreault-Levasseur, and François Lanusse. Bayesian neural networks, 2020.

[56] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017.

[57] Jeremy Nixon, Mike Dusenberry, Ghassen Jerfel, Timothy Nguyen, Jeremiah Liu, Linchuan Zhang, and Dustin Tran. Measuring calibration in deep learning, 2020.

[58] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks, 2018.

[59] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. Hands-on bayesian neural networks – a tutorial for deep learning users, 2022.

[60] Andrew Gordon Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1775–1784. JMLR.org, 2015.

[61]

[62] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105–112, 2009. Risk Acceptance and Risk Communication.

[63] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning, 2018.

[64] Huimin Wu, Zhengmian Hu, and Bin Gu. Fast and scalable adversarial training of kernel svm via doubly stochastic gradients, 2021.

[65] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale, 2017.

[66] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, 2016.

[67] Nina Narodytska and Shiva Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318, 2017.

[68] Juan F. Ramirez Rochac, Lily Liang, Nian Zhang, and Timothy Oladunni. A gaussian data augmentation technique on highly dimensional, limited labeled data for multiclass classification using deep learning. In *2019 Tenth International Conference on Intelligent Control and Information Processing (ICICIP)*, pages 145–151, 2019.

[69] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks, 2016.

[70] Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks, 2017.

[71] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[72] Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep?, 2014.

[73] Sotirios P. Chatzis. Indian buffet process deep generative models for semi-supervised classification, 2018.

[74] Hemant Ishwaran and Lancelot F James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001.

[75] Rodney J. Douglas and Kevan A.C. Martin. Neuronal circuits of the neocortex. *Annual Review of Neuroscience*, 27(1):419–451, 2004. PMID: 15217339.

[76] Anders Lansner. Associative memory models: from the cell-assembly theory to biophysically detailed cortex simulations. *Trends in Neurosciences*, 32(3):178–186, 2009.

[77] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[78] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

[79] Wolfgang Maass. Neural computation with winner-take-all as the only nonlinear operation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 2000.

[80] Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[81] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate bayesian inference. *ArXiv*, abs/1704.04289, 2017.

[82] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[83] Stephen Grossberg. Contour enhancement , short term memory , and constancies in reverberating neural networks. 1973.

[84] Per Andersen, Gary N. Gross, Terje Lomo, and Ola Sveen. *PARTICIPATION OF INHIBITORY AND EXCITATORY INTERNEURONES IN THE CONTROL OF HIPPOCAMPAL CORTICAL OUTPUT*, pages 415–466. University of California Press, 2020.

[85] Rodney J. Douglas and Kevan A.C. Martin. Neuronal circuits of the neocortex. *Annual Review of Neuroscience*, 27(1):419–451, 2004. PMID: 15217339.

[86] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[87] Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders, 2017.

[88] David H. Wolpert. Overview of information theory, computer science theory, and stochastic thermodynamics for thermodynamics of computation, 2018.

[89] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 1991.

[90] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[91] R.E. Blahut. *Principles and Practice of Information Theory*. Addison-Wesley series in electrical and computer engineering. Addison-Wesley, 1987.

[92] T. Avgeris, E. Lithopoulos, and N. Tzannes. Application of the mutual information principle to spectral density estimation. *IEEE Transactions on Information Theory*, 26(2):184–188, March 1980.

[93] E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106:620–630, May 1957.

[94] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[95] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle, 2015.

[96] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information, 2017.

[97] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2018.

[98] Albert E. Parker, Tomáš Gedeon, and Alexander G. Dimitrov. Annealing and the rate distortion problem. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 993–976, Cambridge, MA, USA, 2002. MIT Press.

[99] Anthony J. Bell and Terrence J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

[100] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[101] Ralph Linsker. Self-organization in a perceptual network. *IEEE Computer*, 21:105–117, 03 1988.

[102] Jie Hu, Rongrong Ji, ShengChuan Zhang, Xiaoshuai Sun, Qixiang Ye, Chia-Wen Lin, and Qi Tian. Information competing process for learning diversified representations, 2019.

[103] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hotloo Hao, Jürgen Schmidhuber, and Harri Valpola. Tagger: Deep unsupervised perceptual grouping, 2016.

[104] S. Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.

[105] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[106] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck, 2019.

[107] Radford M. Neal and Geoffrey E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, 1998.

[108] Matthew Chalk, Olivier Marre, and Gasper Tkacik. Relevant sparse codes with variational information bottleneck, 2016.

[109] Amichai Painsky and Naftali Tishby. Gaussian lower bound for the information bottleneck limit, 2017.

[110] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: Mutual information neural estimation, 2021.

[111] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization, 2019.

[112] Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation, 2017.

[113] Bin Dai, Chen Zhu, and David Wipf. Compressing neural networks using the variational information bottleneck, 2018.

[114] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation, 2019.

[115] Costas Stefanis. *INTERNEURONAL MECHANISMS IN THE CORTEX*, pages 497–526. University of California Press, 2020.

[116] G.A. Carpenter and S. Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21(3):77–88, 1988.

[117] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2017.

[118] Konstantinos Panousis, Sotirios Chatzis, and Sergios Theodoridis. Nonparametric Bayesian deep networks with local competition. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4980–4988. PMLR, 09–15 Jun 2019.

[119] Zoubin Ghahramani and Thomas Griffiths. Infinite latent feature models and the indian buffet process. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2006.

[120] Yee Whye Teh, Dilan Görür, and Zoubin Ghahramani. Stick-breaking construction for the indian buffet process. In *AISTATS*, 2007.

[121] Artur J. Lemonte, Wagner Barreto-Souza, and Gauss M. Cordeiro. The exponentiated kumaraswamy distribution and its log-transform. *Brazilian Journal of Probability and Statistics*, 27(1):31–53, 2013.

[122] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[123] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[124] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[125] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.