



Technische Universität München
TUM School of Engineering and Design
Data Science in Earth Observation

Joint Energy-based Model for Remote Sensing Image Processing

Author: Daniel Orozco

Master Thesis

Earth Oriented Space Science and Technology - ESPACE

Supervisors:

Prof. Dr. -Ing. habil. Xiaoxiang Zhu, Conrad Albrecht, Chenying Liu

18.02.2023



Technische Universität München
TUM School of Engineering and Design
Data Science in Earth Observation

Joint Energy-based Model for Remote Sensing Image Processing

Author: Daniel Orozco

Master Thesis

Earth Oriented Space Science and Technology - ESPACE

Supervisors:


Prof. Dr. -Ing. habil. Xiaoxiang Zhu, Conrad Albrecht, Chenying Liu

18.02.2023

Statement of Authorship

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

Munich, 18.02.2023 (date)

Author Daniel Orozco
.....
..... (signature)

Abstract

The rapid development of remote sensing technologies in recent years is creating an unprecedented amount of Earth Observation (EO) data. Although advances in deep learning are creating opportunities to exploit big amounts of data, most methods rely on densely accurately annotated sets to produce well-fit models. However, having large-amount and high-quality labeled data is impractical for modern problems mostly because of the excessive cost involved. Such limitations have been studied in the scope of Semi-supervised Learnings (SSLs), resulting in more cost-effective approaches where unlabeled samples are included to aid the generalization of models trained with limited labeled data. To this end, Joint Energy-based Models (JEM) is a novel approach that simultaneously optimizes a discriminative process and a generative process, where additional unlabeled samples can be incorporated in the generation branch to support the model training. Although the promising formulation of such models, the high complexity of its training and the sensibility to divergence are open challenges to consider. Also, current JEM methods are mainly applied for classification tasks, so their potential for improving segmentation tasks is still under discovery.

To solve the problems mentioned above, this thesis first investigates JEM training behavior from a theoretical perspective, finding in its definition the influential factors for divergence. Based on these observations, three regularization terms imposed on energy values and gradients are designed to constrain and alleviate the training complexity. Their effectiveness was tested in the experiments, indicating that all of them alleviate the divergence for remote sensing image classification tasks to some extent, yet the regularization on energy values of real samples performed the best. After that, we extended the JEM definition to segmentation tasks by means of certain assumptions on pixel independence. However, it didn't perform as well as expected, partly due to lack of spatial autocorrelation among pixels. Still, this work gives some valuable insights for future methodology design.

Acknowledgments

First of all, I would like to emphasize my sincere and extensive gratitude to my supervisor, Chenying Liu, for all the time dedicated to guide and mentor me, and for willingly share with me her expertise in the field . She encouraged me to pursue the excellence and made the process of my thesis a great experience full of learning and great science discussions.

Second, I would like to thank my supervisors, Conrad Albrecht and Prof. Zhu, for giving me the opportunity of working in the prestigious Future Lab AI4EO (Artificial Intelligence for Earth Observation) and guiding my work with valuable insights.

Third, I would like to thank all the people I met at DLR, for the warm welcome and the good moments we spend in the campus, all you are a great inspiration for my career.

Fourth, I would like to thank my friends in the ESPACE master and in Europe, for all the great experiences we have shared and for providing relief in stressful times. Thanks also to my friends in The Americas, for staying close to my heart.

Last but not least, I would like to thank all members of my loving family, for their unconditional support and constant encouragement. Especially, my profound gratitude to my brother for deeply caring and believing in me, to my parents for always supporting my dreams, and to my grandparents for their immeasurable love.

Contents

Abstract	iii
Acknowledgments	v
1. Introduction	1
1.1. Background and motivation	1
1.2. Thesis overview	3
2. Related works	5
2.1. Learning paradigms	5
2.1.1. Supervised learning	5
2.1.2. Unsupervised learning	5
2.1.3. Weakly-supervised learning	6
2.2. Semi-supervised learning	7
2.2.1. Discriminative models	8
2.2.2. Generative models	9
2.3. Energy-based models	10
2.3.1. EBM for generation tasks	10
2.3.2. Joint energy-based models	12
3. Joint Energy-based Models for Image Classification	15
3.1. Methodology	15
3.1.1. JEM architecture	15
3.1.2. Divergence	16
3.1.3. Regularization	19
3.1.4. Hyper-parameters	21
3.2. Experiments	22
3.2.1. Divergence	23
3.2.2. Regularization	30
3.2.3. Hyper-parameters	32

3.2.4. Semi-supervised classification	34
4. Joint Energy-based Models for Image Segmentation	39
4.1. Methodology	39
4.1.1. Differences between segmentation and classification	39
4.1.2. JEM for segmentation	40
4.2. Experiments	42
4.2.1. Individual branches analysis	45
4.2.2. Hyper-parameters	49
4.2.3. Divergence	52
5. Conclusions	53
5.1. Conclusions	53
5.2. Future lines	54
Acronyms	57
List of Figures	59
List of Tables	61
Bibliography	63
A. Appendix	71

Dedicado a mis preciosos padres, de su hijo que los ama.
(Dedicated to my precious parents, from their son who loves them.)

1. Introduction

"The future of remote sensing is unlimited in its possibilities for understanding our planet."

-Steve Wofsy-

1.1. Background and motivation

EO is of great importance for the monitoring and understanding of the Earth's changing systems. The interest in using satellites and other technologies in EO has been increasing over the years, leading to unprecedented variety in the goals and methods for mapping the Earth. Thus, a large amount of EO data is being generated every second, providing the assets to our society to make more sustainable and equitable decisions in the use of resources and management of ecosystems. To cope with the large amount of information available, the use of Deep Learning (DL) is allowing researchers to create agile pipelines and multi-domain frameworks to rapidly exploit most of the accessible information. Spanning over a wide range of applications including -but not limited to- disaster management and response [2], land use and land cover [70], soil moisture for agriculture [40], urban planning [13], effects of climate change [61], oceanography [52], and geology [48].

The majority of methods developed to this date, especially DL models with millions of parameters, rely on densely accurately annotated data since it ensures well-fit models with high accuracy and reduced need for expertise in data augmentation or transformations. Unfortunately, completely labeled sets come with high costs (for sophisticated labeling equipment and/or domain-specific human expertise) and require a long time to update. More cost-effective and common sets contain a large number of low-quality labels or a small number of high-quality labels, so research in the area of training models with weakly supervised information has gained attention [62].

Weakly-supervised Learning (WSL) is a paradigm that explores methods to overcome problems where the available data is incomplete, inexact, inaccurate, or a combination of those [80]. Inexact and inaccurate usually refer to a lack of quality in the labels, whereas incomplete refers to a small number of accurate labels. From the three types, incomplete data is of special interest for situations where the cost of a complete set of labels is enormous or can't be realized due to privacy and security reasons, especially in EO data where the high variability, in terms of viewing conditions and states of the dynamic systems being modeled, intrinsically limits the size of the labeled sets [66]. Researchers have developed a series of methods in the field of SSL to solve a such challenge, leveraging the capabilities of DL state-of-the-art, functional Deep Neural Networks (DNNs) architectures have been defined to gain knowledge from the unlabeled samples and thus assist the target parametric model with the complete set of information.

The existing SSL methods are mainly designed from two perspectives: discriminative and generative. Discriminative models are designed for learning a mapping that can accurately distinguish between categories or classes in the provided feature space, usually known for their simplicity and efficiency but incapable of modeling the feature space distribution [7]. Generative models discover patterns and the underlying structure of the feature space to learn its distribution, usually known for their capacity to generate realistic new samples but are computationally expensive [14]. It's clear that each approach is suited for a different disposition in the data distribution and labeling, thus the potential in the connection of both configurations leads to research in hybrid discriminative-generative models where generative models can assist the learning of supervised discriminative models. Following this idea, JEM, originated from statistical physics, presents a novel re-interpretation of the discriminative classifier where a generator process is hidden; it simultaneously optimizes the two processes and constructs a hybrid model capable of finding powerful class-distinctive criterion, estimation of distributions, and generation of congruent samples [18]. Such characteristics can be applied to robust scene classification and segmentation, and so its study is of most interest to overcome current challenges in SSL using EO data. Therefore, in this work, we focus on the implementation of JEM for image classification, investigate the inherent training instabilities, and ultimately extend the formulation and architecture to image segmentation.

1.2. Thesis overview

This thesis is focused on the study of JEM for image processing of remote sensing data. More specifically, our major contributions can be summarized into three aspects:

- Starting with state-of-the-art implementations of JEM for image classification, we study the reasons behind the training instability challenge.
- We present three regularization terms to relieve the training instability.
- We extend the definition and architecture to segmentation and explore the implications of our assumptions using different DNNs architectures.

Structured as follows:

- In chapter 1 we introduce the background and motivations of remote sensing applications for earth observation.
- In chapter 2 we conduct a literature review of machine learning paradigms, and modeling approaches. Narrowing to methods with potential for SSL, like JEM.
- In chapter 3 we follow the works of JEM on image classification and extend on details of the training process related to model divergence along with solutions to alleviate it.
- In chapter 4 we describe the derivations and changes in the JEM architecture needed to extend the problem to image segmentation and test the performance of such adaptations. Also, we present experimental results on the performance of the individual branches.
- In chapter 5 we summarize our findings and comment on possible future lines to continue our work.

2. Related works

*"All models are wrong, but some are useful."*¹

-George E. P. Box-

In this chapter, we present fundamental concepts and frameworks developed over the years by various researchers in the fields of remote sensing, computer vision, and deep learning. We introduce learning paradigms and how they are related and relevant to models like joint energy-based that are the foundation of our work.

2.1. Learning paradigms

2.1.1. Supervised learning

Supervised learning is a paradigm that, as its name indicates, is driven by a "supervisor" or "teacher" that advises the learning system. Given a set with pairs of samples and labels $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$, the system is said to be "taught" or trained with a portion of the set $S_t = \{(x_1, y_1), \dots, (x_m, y_m)\}$, $m \leq n$ so it can learn the mapping between input and output and accurately predict the labels \hat{y} of a new given set x' such as the training error and learner complexity are minimized [38, 45]. Common algorithms that uses this principle are Support Vector Machines (SVMs) [50] and nearest neighbor [4].

2.1.2. Unsupervised learning

Unsupervised learning is a paradigm where the only asset available for the system to learn from are the observation input samples $\chi = \{x_1, \dots, x_n\}$. It focuses on finding

¹Originally used in the scope of statistics, we emphasize in this chapter that all learning concepts, algorithms and models are useful under certain conditions.

patterns and building meaningful representations of the observed distribution [16, 10], with methods like principal components analysis (PCA) used in feature extraction [19], hidden Markov model [59] and K-means clustering [63].

2.1.3. Weakly-supervised learning

WSL is a paradigm that deals with "weak" labeled data, commonly placed in between supervised and unsupervised learning in the sense that in supervised learning exists a complete set of labels, whilst unsupervised has none. The term "weak" refers to the level of quality or preciseness in the labels, usually, it's categorized into three types (see Fig. 2.1): inexact when the labels are coarse-grained (low-resolution or low-detailed), inaccurate -also known as noisy- when the labeling process is unreliable and incomplete when only a subset -usually small- of the samples are labeled. It's worth noticing that, despite the difference between the categories, a combination of them is usually present in real-world applications [80, 42].

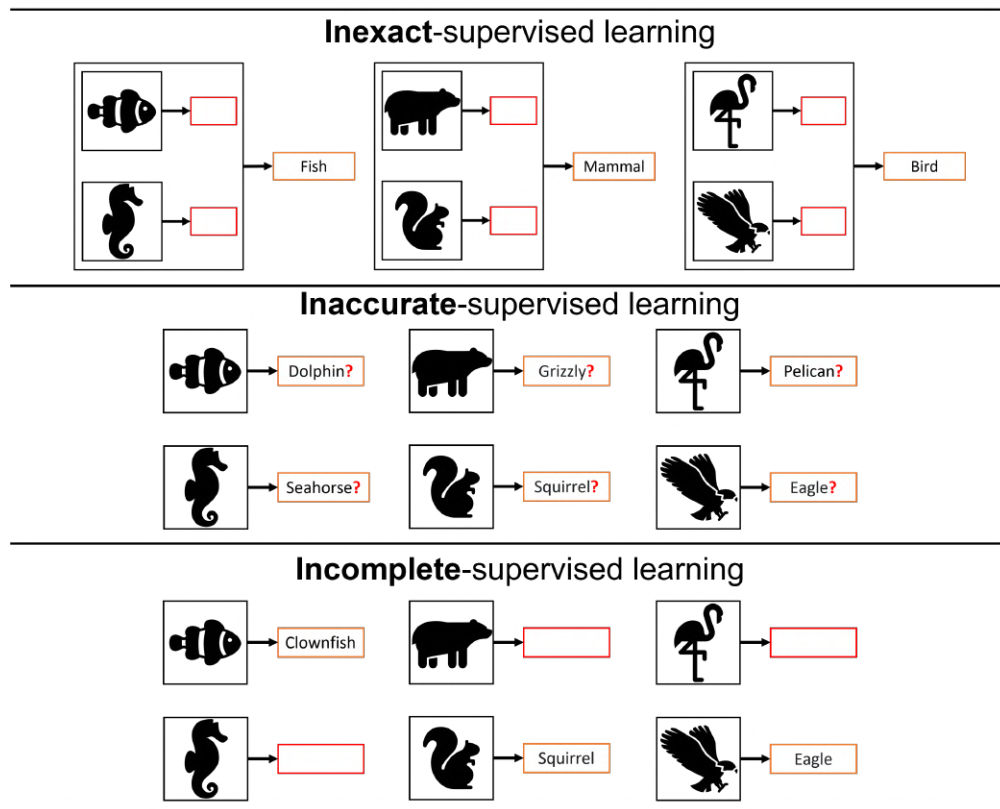


Figure 2.1. Weakly-supervised learning types.

Inexact-supervised learning is suited for problems where the dataset can be described in multiple levels of abstraction but its corresponding labels are incomplete and/or limited to some aggregations. Multi-instance learning [15, 82] is a technique that defines arrangement sets (named "bags") as a group of instances, where the label is associated with the entire bag; in cases like semantic segmentation, "bags" can be regarded as patch-wise or object-wise labels when no pixel-wise labels are available. Experiments such as molecules classification [49] and web mining [5] demonstrate the applicability of this technique.

Inaccurate-supervised learning concerns cases where the label's validity is in doubt. Faulty labels can be produced by errors in the labeling process such as instrument uncertainty, lack of expertise, environment noise, fuzzy domains, and others. One possibility to address this problem is to identify and correct the erroneous labels as demonstrated in a study of cancer diagnosis [22]. Also by including the trustworthiness of the label in the features, a binary classifier can be trained to detect fake news on social media [25]. Another approach is to mitigate the effects of noisy labels with the use of the unlabeled data to estimate its underlying distribution [78], following this concept Generative Adversarial Networks (GANs) can be used to aid the classifier with information to distinguish between noisy samples and real samples [77].

Incomplete-supervised learning focuses on techniques to exploit the limited -but regarded as ground-truth- labeled samples. If there's an entity (e.g. human expert or knowledge base) that could aid the prediction of the unlabeled samples, active learning approaches can be used to find the balance between prediction accuracy and the cost of querying this entity. For instance, transfer learning [55] where the entity is derived from a different dataset but with a related domain is being applied in language models (e.g. chatbots [73]) and computer vision [47]. When the problem lacks such an entity, then approaches mentioned in the following section 2.2 can be considered.

2.2. Semi-supervised learning

SSL is a paradigm that deals with learning problems where the data is partially labeled [83]. It can be applied as an extension to classification problems, given that the model is trained with both unlabeled samples $U = \{(x_i) | x_i \in \mathcal{X}^d, i = 1, \dots, u\}$ and labeled samples $L = \{(x_k, y_k) | x_k \in \mathcal{X}^d, y_k \in \Omega, k = 1, \dots, l\}$ where usually $u \gg l$ [21].

In recent years, the development of DNNs introduced new possibilities and strategies

to address these problems [41, 3, 71]. For instance, autoencoders [57] are a type of neural network that consists of an encoder and a decoder, the relation of the encoder design -finding a meaningful representation of the feature vector latent space- and the SSL manifold assumption -ensuring that the elements belonging to the same low-dimensional structure share the same label- enables this architecture to be used for feature extraction [14] with potential applications for image reconstruction [30], resolution enhancing [26] and audio source separation [64]

One of the biggest challenges of SSL relies on the uncertainty of up to which extent the unlabeled data result useful. Discriminative and generative models offer promising approaches suited to overcome these difficulties. To choose which method to use one should study the assumptions of the given problem and characteristics of the methods [21]. In the following section, we present the principles of the discriminative, generative, and hybrid discriminative-generative models and how these approaches can be applied to SSL.

2.2.1. Discriminative models

Discriminative models refer to those approaches that intend to produce and compute the optimal direct mapping $f : \chi \mapsto Y$ between input $\chi = \{X_1, \dots, X_n\}$ and output $Y = \{Y_1, \dots, Y_m\}$ [32]. Common machine learning algorithms widely used for classification construct decision boundaries, such as SVMs that define a hyperplane and decision trees that generate rules [7].

Some discriminative deep learning methods directly optimize for a loss function by using Convolutional Neural Networks (CNNs) or attention mechanisms [58]. The advantage of these approaches relies on the ability of the neural networks to identify the relevant features in several levels of abstraction, without the need for specific domain knowledge and despite the nature of the data. CNNs consists of convolutional layers, pooling layers, and fully connected layers allowing a hierarchical extraction of features [7]. Commonly used architectures like LeNet[39], AlexNet[36], GoogLeNet[68], ResNet[23] and DenseNet[29] are broadly applied in object tracking[9] and image segmentation[51], among others.

In the context of SSL, discriminative models exploit the unlabeled data with methods like SVMs [50] trained from labeled samples which kernel function is derived from unlabeled samples [33], committees where several models are generated using the labeled samples and uses the disagreement of the models with the unlabeled samples

to select the one with higher confidence [81], Pseudo-Label assigns labels to the unlabeled samples by selecting the one with the highest belonging probability allowing the training process to conduct simultaneously this assignment of labels and other supervised-fashion optimization [41] showing to be beneficial for hyperspectral classification [74] and automatic speech recognition [27].

2.2.2. Generative models

Generative models enclose the approaches that intend to produce a probability density model $P(\Omega)$ of all system variables (input and output) $\Omega = \{X_1, \dots, X_n, Y_1, \dots, Y_m\}$ and employ it in the optimization criteria. Given this density model and its full joint distribution $p(x, y)$, inferences and predictions can be made by mathematical manipulations (like conditioning or marginalizing) [32] and also to draw samples $\{\hat{x}, \hat{y}\}$ [14]. Generative models in principle calculate $P(Y|X)$ using Bayes' rule by estimating the distributions $P(X|Y)$ and $P(Y)$ [17]. For instance, Gaussian mixture models estimate the probability density function by assuming that the samples are generated from a mixture of Gaussian distributions thus widely used for clustering [60].

The fundamental concept of SSL establishes the condition that information of the posterior distribution $p(y|x)$ is present in the marginal data distribution $p(x)$. And so by using the unlabeled instances of the dataset to gain information of the $p(x)$, also information of the $p(y|x)$ is revealed and used to assist the learning [14]. The development of deep learning generative models produced some methods designed to solve SSL problems, for instance, Variational Autoencoders (VAEs) are based on autoencoders, where the encoder is trained to map the input data $p(x)$ into a latent space $p(z)$ while at the same time, the decoder converts the latent space $p(z)$ into a distribution $p(z|x)$ from where sampling can be done [14, 17]. This model has been used for speech emotion classification [37] and molecule design [46], among others.

GANs are another typical kind of generation models, but composed of two networks trained simultaneously as a minimax optimization problem formulated as:

$$\min_G \max_D \mathcal{L}_{GAN}(D, G) = \mathbb{E}_{x \sim p_x(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

The generator network G , designed to generate samples as close as possible to the original distribution by its parameters θ_G , minimizes the objective function \mathcal{L}_{GAN} . And the discriminator network D , trained to distinguish between "real" (i.e. original distribution) and "fake" (i.e. synthetically generated) samples by its parameters θ_D ,

maximizes the objective function \mathcal{L}_{GAN} [14, 51, 17]. Some adaptation to GANs [20] enable its use in tasks such as classification [31], small object detection [43] and language generation [44].

Both discriminative and generative models have their own advantages and challenges, recently hybrid discriminative-generative models have been studied to present a complementary approach [11]. For instance, JEM is a novel approach of this kind that applies simultaneous optimization to a discriminative classifier and a generative process, showing promising results for semi-supervised classification [18, 79]. In the following sections, we present the details of Energy-based Models (EBM) and JEM.

2.3. Energy-based models

EBM belongs to the deep generative branch and it originates from statistical physics, the main idea is to define a probability density to embody the states' distribution of a system by mapping any datapoint, $x \in \chi$, to a scalar value with an energy function $E(x) : \chi \mapsto \mathbb{R}$ [53]. Thus, to parametrize an EBM, the chosen energy function should map realistic instances of x to low values, and unrealistic ones to high values. Moreover, the energy function also defines a probability distribution, through a Boltzmann distribution, to express the probability density $p(x)$ as

$$p(x) = \frac{\exp(-E(x))}{Z}, \quad (2.1)$$

where $Z = \int_{\chi} \exp(-E(x))$ (2.2) is the normalizing constant (also known as the partition function) [18, 12]. Unfortunately, estimating normalized densities is often intractable, so computing Z represents a challenge [18].

2.3.1. EBM for generation tasks

To dismiss the explicit use of the Z constant, one can rely on the potential of the energy function for sampling. In this view, let $\theta \in \Theta$ be a set of parameters such that the parametric model $p_{\theta}(x)$ is a good approximation of the data distribution $p(x)$ [53]. Then, training of EBM can be accomplished by maximizing the expected log-likelihood function $\mathcal{L}(\theta|x)$.

With the formulation of the log-likelihood derivative, expressed as

$$\frac{\partial \mathcal{L}(\theta|x)}{\partial \theta} = \frac{\partial \log p_{\theta}(x)}{\partial \theta} = \mathbb{E}_{p_{\theta}(x')} \left[\frac{\partial E_{\theta}(x')}{\partial \theta} \right] - \frac{\partial E_{\theta}(x)}{\partial \theta}, \quad (2.3)$$

there is no simple solution for sampling from $p_\theta(x)$. In order to estimate the first term, Markov chain Monte Carlo (MCMC) [8] algorithms have been implemented for samplers. For instance, the Gibbs sampler was used with Restricted Boltzmann Machines [28], and more recent approaches use neural networks and a Stochastic Gradient Langevin Dynamics (SGLD) [72] sampler that reduces the sampling procedure time [12]. The SGLD sampler is formulated as follows

$$\begin{aligned} x_0 &\sim \mathcal{U}(-1, 1) \\ x_{t+1} &= x_t - \frac{\alpha}{2} \frac{\partial E_\theta(x_t)}{\partial x_t} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \alpha), \end{aligned} \quad (2.4)$$

where x_0 is the initial sample, t represents the iteration, α is the step-size and ϵ is Gaussian noise. Samples then will be generated from the energy function distribution as $t \rightarrow \infty$ and $\alpha \rightarrow 0$ [72, 12] as shown in Fig 2.2.

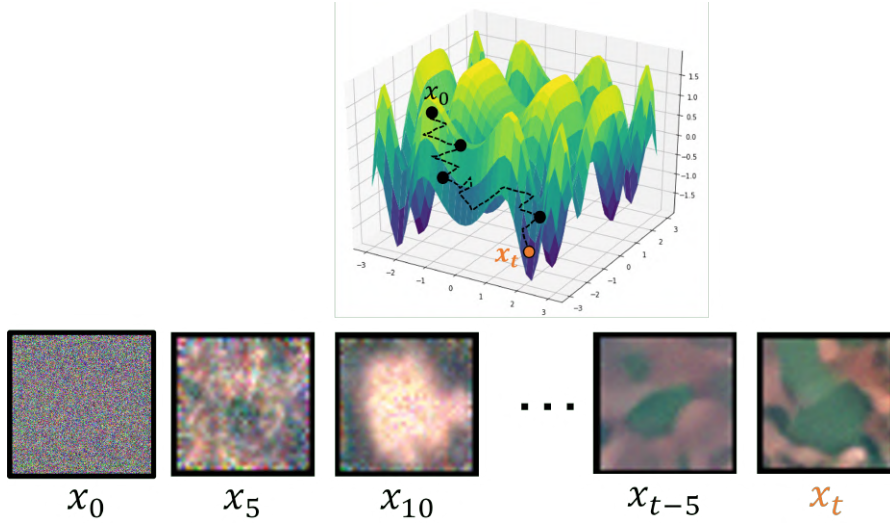


Figure 2.2. Generation of samples using SGLD.

The flexibility of EBM have been exploited in generation tasks (images [12] and scene graphs [67]), hybrid generation-classification [18, 75] and out-of-distribution classification [18], among others. However, training EBM is a challenging task because of the SGLD hyper-parameters effect in high computational complexity and poor training stability [53, 18, 12].

2.3.2. Joint energy-based models

JEM re-interprets the standard modern classifier as a EBM [18]. In classification problems a parametric function $f_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^K$, with D input dimensions and K classes, maps every point $x \in \mathbb{R}^D$ to real-valued numbers known as logits, where a single logit that corresponds to the y^{th} , $y \leq K$ class can be represented as $f_\theta(x)[y]$. Usually, logits are used to parameterize a categorical distribution $p_\theta(y|x)$ in a discriminative way with the Softmax transfer function:

$$p_\theta(y|x) = \frac{\exp(f_\theta(x)[y])}{\sum_{y'} \exp(f_\theta(x)[y'])}, \quad (2.5)$$

JEM re-interprets the logits to define the energy-based model of the joint distribution $p_\theta(x, y)$ as:

$$p_\theta(x, y) = \frac{\exp(f_\theta(x)[y])}{Z(\theta)}, \quad (2.6)$$

with $Z(\theta)$ being the unknown normalizing constant. By marginalizing out y with $\sum_y p_\theta(x, y)$, the unnormalized density $p(x)$ is as:

$$p_\theta(x) = \frac{\sum_y \exp(f_\theta(x)[y])}{Z(\theta)}. \quad (2.7)$$

Moreover, comparing (2.7) with (2.1), the logits can be re-used to define the energy function as follows:

$$E_\theta(x) = -\log \sum_y \exp(f_\theta(x)[y]). \quad (2.8)$$

To benefit from the generative characteristics of EBM and retain the classifier performance, the training of the f_θ is proposed to find the appropriate optimization of the involved distributions by maximizing the likelihood of the joint discriminative-generative model:

$$\log p_\theta(x, y) = \log p_\theta(x) + \log p_\theta(y|x), \quad (2.9)$$

where the term $\log p_\theta(y|x)$ can be optimized using cross-entropy the same as traditional supervised classification tasks, and $\log p_\theta(x)$ using SGLD (2.4) with (2.8) as the energy function to draw samples. Since the second term doesn't require labels, JEM has the potential to explore information from unlabeled data for semi-supervised learning purposes. So, two simultaneous processes can be distinguished in the JEM configuration, a discriminative one that learns from the samples x and labels y , and the generative

one that learns from the entire distribution of samples x using the encoded information of the classifier in the energy function.

Tested with natural images it shows comparable performance in semi-supervised classification compared against Virtual Adversarial Training and outperforms it in other data domains, proving that JEM requires much less domain-specific knowledge for SSL [79]. Further work tested JEM on EO datasets, reaching the same level of performance as other classification-only and generation-only architectures, showing tangible improvement for semi-supervised classification and model calibration when trained with $\leq 1\%$ of labeled samples, also proved the robustness of JEM, especially with datasets that have significant differences between the training and test sets and investigated the potential for applications like land cover mapping where the unnormalized log-likelihood value serves as a proxy of confidence in the predictions [6]. Although, instability was also faced during training in all mentioned experiments; also, the difficulty to apprehend some classes from the data distribution and the linear relationship between sampling time and image size, makes applications like remote sensing an open challenge for JEM [79, 6].

The potential of JEM has been tested with natural and real-world imagery achieving competitive performance against other standalone models and outperforming other hybrid models. Except for solving the problem of lack of labeled data, it also shows the additional advantages of improved model calibration characteristics, and out-of-distribution detection, among others, [6]. Nevertheless, the model still faces the training instability of EBM, so it requires human intervention and thorough tuning of the hyperparameters with unclear effect of those in the model [18]. So in this work, we study the factors in the formulation of JEM to determine how its behavior and definition contribute to the stability or instability of the general training process from a theoretical perspective. Based on that, we also formulate potential techniques and architectures specially designed for JEM model training in different tasks.

3. Joint Energy-based Models for Image Classification

"Innovation thrives on experimentation and exploration."

-Elon Musk-

In this chapter, we explore the behavior of JEM training by doing a deep study of the reasons behind divergence and evaluate the effectiveness of possible solutions using three regularization terms. We study the influence of hyper-parameters and use a pre-emptive approach to contribute to the divergence problem of JEM. Also, we test the adaptable and versatile capabilities of JEM on semi-supervised remote sensing image classification.

3.1. Methodology

3.1.1. JEM architecture

As stated in the previous chapter, JEM model training simultaneously optimizes the classification and generation branches. As shown in Fig. 3.1, the classification branch uses an encoder to extract the relevant features from the training samples and computes the Cross-Entropy with the provided labels as the branch loss. Whereas the generation branch uses the energy function defined in (2.8) and SGLD in an iterative process to generate samples that follow the gradient of the energy, then uses the difference between the energy values of training samples and generated samples as the branch loss. The weights of the model are then adjusted using back-propagation of the combined losses as follows:

$$\mathcal{L}_{TOTAL} = \mathcal{L}_{CLA} + \mathcal{L}_{GEN}, \quad (3.1)$$

with classification loss as \mathcal{L}_{CLA} and generation loss as \mathcal{L}_{GEN} .

Such architecture has the potential for semi-supervised classification just by adapting the use of the samples. Let x^U be the unlabeled samples and x^L the set of labeled samples, then only labeled data in x^L go through the classification branch, while all the data from both x^L and x^U can be fed into the generation branch. Thus, those samples without a label can also make a difference in model training.

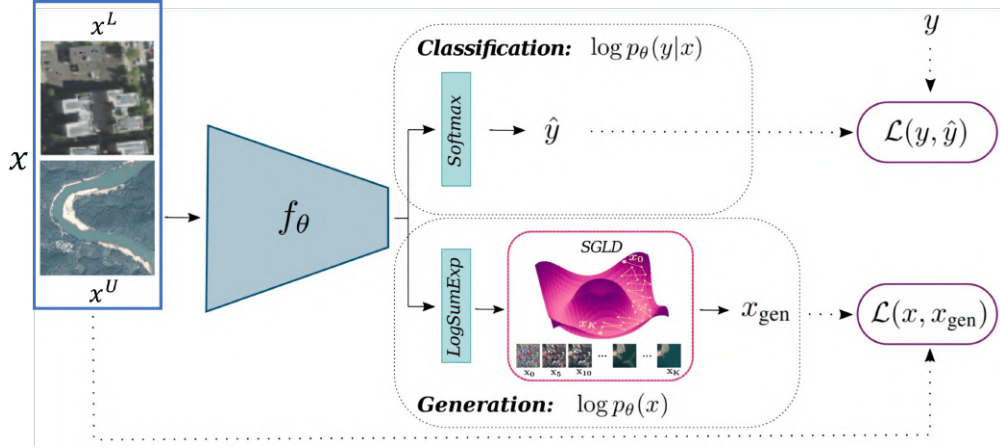


Figure 3.1. JEM for semi-supervised learning overview [6].

However, known characteristics of JEM are the instability of the training process and the sensibility of the model to divergence. In this work, we analyze in a dedicated section this challenge and present relevant aspects for the reduction of optimization complexity and reliable training.

3.1.2. Divergence

The training loss is a measure that indicates how well the model is able to fit the training data. Generally, the optimization process is designed to minimize the training loss defined by a distance-like form between predictions and training samples/labels. With such formulation the loss is bounded, since it's expected to decrease until the value gets as close as possible to the minimum value of 0, so preventing it to continue decreasing until $-\infty$. However, there is no such constraint in the generation loss of JEM since it's designed as a proxy of the realisticity of the samples, for the case of both realistic and unrealistic samples the loss is likely to contain negative/positive values which are extremely large in quantity. In such cases, the model training falls into a local minimum and can learn no more useful features or even fallacious features from the data. This is

the so-called **divergence** problem in JEM model training. Correspondingly, we force the interruption of training when the absolute loss value is higher than 1×10^8 in the experiments. We found that model training encounters the divergence problem very often, especially when using JEM for remote sensing data. Therefore, to get well-trained models and further expand the potential of JEM for EO, it is of big importance to analyze the behavior of the loss and explore potential solutions for the divergence. In the following, we first investigate this problem from a theoretical perspective and then formulate intuitive interpretations of it.

Let us examine in detail the individual loss functions for each of the two branches:

$$\mathcal{L}_{CLA} = CrossEntropy(y, \hat{y}) = - \sum_k y[k] \cdot \log \hat{y}[k], \quad (3.2)$$

$$\mathcal{L}_{GEN} = E_{\theta}(x) - E_{\theta}(x'), \quad (3.3)$$

where $E_{\theta}(\cdot)$ is the energy function defined in (2.8), x is a (real) training sample, x' is a generated sample by SGLD, y is the corresponding one-hot label of the training sample and \hat{y} is the predicted label probabilities derived from the logits $f_{\theta}(x)$ by applying the softmax function as follows:

$$\hat{y}[k] = softmax(f_{\theta}(x)[k]) = \frac{e^{f_{\theta}(x)[k]}}{\sum_{k'} e^{f_{\theta}(x)[k']}} \quad (3.4)$$

with $f_{\theta}(x)$ ranging between $[-\infty, \infty]$ and $\hat{y}[k]$ normalized to the range $[0, 1]$.

In the classification branch, since the logarithmic function $-\log(\cdot)$ is monotonously decreasing, then the expression $-\log \hat{y}[k]$ in (3.2) will have 0 as the lower limit at $\hat{y}[k] = 1$. The one-hot label $y[k]$ can only have two values: 0 or 1. Thus, (3.2) is overall bounded above 0, consequently, its minimization won't result in values reaching $-\infty$.

Whereas in the generation branch, the logits are transformed to a single unbounded scalar value using the $\log \sum \exp$ operator in the energy function definition (2.8). Thus, the energy values don't have a theoretical numerical boundary (i.e., in the range of $[-\infty, \infty]$). As mentioned in 2.3, the energy function maps realistic samples to low values and unrealistic ones to high values. To do so, (3.3) minimizes the energy values of real samples and simultaneously maximizes those of fake samples by directly subtracting two unbounded energy values. This would result in the unbounded definition of the generation loss where the minimization can lead to values reaching $-\infty$. Therefore, (3.3) can easily cause divergence during training.

Based on Fig. 3.2, we present an intuitive explanation of the reasons behind divergence. Generation tasks are mainly composed of two processes, that is, a generative

process to produce fake samples and a discriminative process to distinguish real samples from fake ones. The novelty of JEM is the re-interpretation of the classifier where the energy function of a generative model can also exploit the logits by mapping high/low energy values to fake/real samples, thus instinctively acting as a discriminative process (I. Discriminative process in Fig. 3.2). Also, the iterative SGLD process uses the gradient of the energy function to generate new samples that are close to the distribution of the real ones, thus instinctively acting as a generative process (II. Generative process in Fig. 3.2).

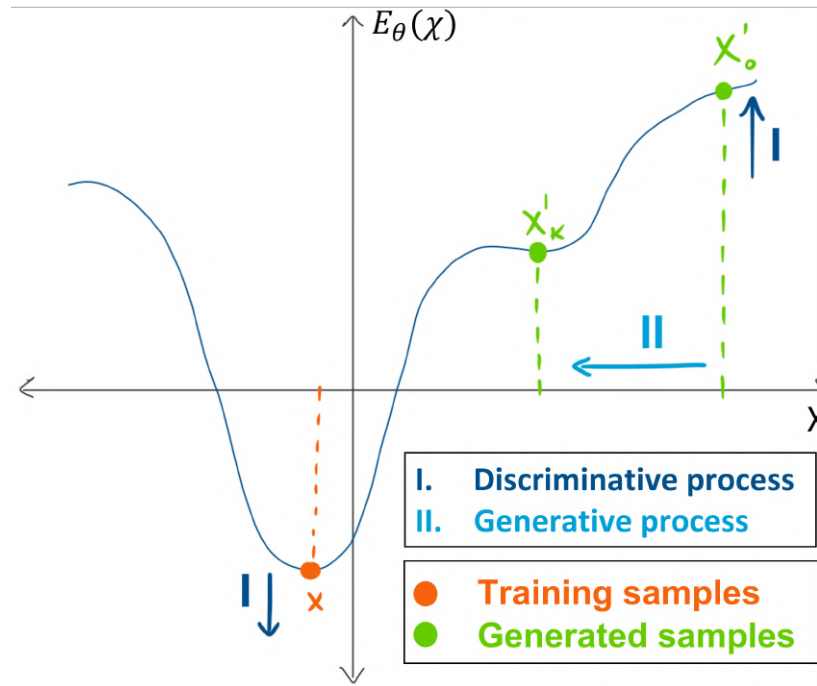


Figure 3.2. Landscape of energy values.

To conclude, this interpretation presents a dual process of optimization that in our judgment shows a competing behavior in the energy landscape. In the discriminative process, the model attempts to optimize its capability to separate the generated samples from the ones included in the training set by assigning low values to training samples and high values to generated samples, as a way to correctly classify the data. While at the same time in another generative process, the optimization is such that the generated samples distribution gets as close as possible to those of the samples from the latent space in the training set, so forcing the energy function to assign lower values to generated samples (which counteracts with the other optimization process), in order to

contribute to the generative facet of the model. We consider this competing behavior in the unbounded landscape as a possible explanation for the sensibility of the model to divergence. By regulating one of the optimization processes we expect the overall complexity will reduce and so divergence could be intuitively alleviated to some extent.

In the following, we introduce regularization as a way of imposing boundaries to the divergence problem and present new formulations for the generation loss of JEM. Also, we explore the influence of hyper-parameters in the training process stability and how they impact the model divergence.

3.1.3. Regularization

An intuitive way of constraining the values of (3.3) is to apply the absolute operator on it. However, by disregarding the sign in the difference, the discriminative process is omitted since the capability to assign positive values to fake samples and negative values to real samples are removed with the absolute value operator. This alteration to the formulation breaks the design of the dual optimization problem so it's not a valid approach.

Instead, we resort to constraining the values generated by the energy function, since its contribution produces the unbounded values in the total loss. Moreover, if boundaries are applied to (3.3) it can help constrain the landscape of energy values, thus potentially improving the stability of JEM model training. To do so, we utilize regularization, a technique that adds a penalty term to the objective function of an optimization problem, imposing the model to change its behavior and complexity [69]. In JEM, this could be applied in the \mathcal{L}_{GEN} to constrain the values as follows:

$$\mathcal{L}_{GEN_REG} = \mathcal{L}_{GEN} + \alpha \cdot REG, \quad (3.5)$$

where α is a hyper-parameter that regulates the strength of the regularization term REG .

There are many types of regularization, in our work we utilize the L2 regularization that is based on the L2-norm (also known as Euclidean norm). We stated in the previous section the unbounded characteristic of the energy function, so first, we can impose the regularization directly on the energy values. Specifically, for the training/real samples x :

$$REG_r = \sum E_\theta(x)^2, \quad (3.6)$$

and to the energy values of the generated/unreal samples x' :

$$REG_u = \sum E_\theta(x')^2, \quad (3.7)$$

the effectiveness of the regularization terms is then linked with the behavior itself of the energy values. We expect the term REG_r to be more stable as the training samples remain constant during training.

Another possibility is to impose the regularization to the gradients of the energy function, although the computation of gradients is a complex operation so it must be assessed the implications on training time. During training the values of the generated samples are expected to have high variations because the constant improvement of the model's generation ability will consequently modify frequently the quality of generated samples, and also due to the generation process random initialization and added randomness in the SGLD iterations; whereas the training samples aren't modified and so have stable values. Based on that, in our work, we apply L2 regularization to the gradients of the training samples x and it's defined as:

$$REG_g = \sum \nabla E_\theta(x)(x)^2, \quad (3.8)$$

Using regularization, the optimizer aims to minimize not only the difference between energy values but also the regularization term. Thus, it can be interpreted, in the view of the dual optimization and the energy landscape, that the discriminative process will be partially fixed and close to a maximum/minimum in one of the vertical directions. In the case of the L2 for energy values of training samples (left diagram in Fig. 3.3) it will be fixed to a maximum and L2 for energy values of generated samples (right diagram in Fig. 3.3) to a minimum, both tho the value of 0. Whereas in the case of the L2 for energy gradients of training samples, it will be fixed similarly to the L2 for energy values of training samples but with respect to an arbitrary value, not necessarily 0. As a consequence, these regularization terms can help to abate the overall optimization complexity, eventually constraining the energy function values and alleviating the divergence. Nevertheless, as mentioned above, compared to generated samples, training samples are supposed to keep unchanged all the time. So do the corresponding energy values. Thus, the two regularization terms imposed on training samples are expected to show more effectiveness.

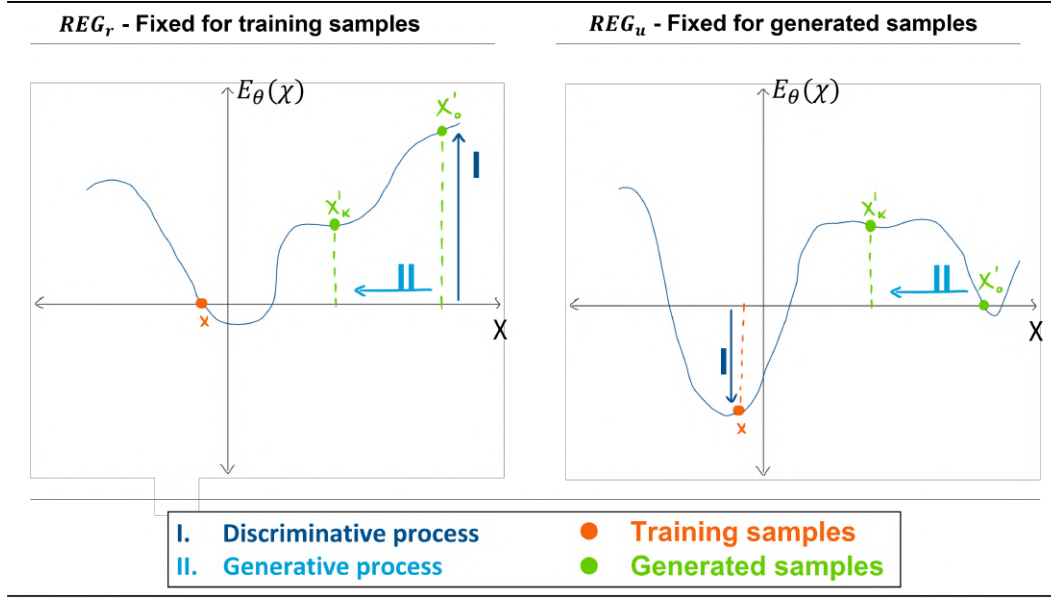


Figure 3.3. Constrained landscape of energy values with regularization.

3.1.4. Hyper-parameters

As we already stated, there are many processes involved in the JEM training, so it contains many hyper-parameters that regulate and influence the general learning process. We study their impact on the overall training and how they relate to the divergence. Specifically, we identify two hyper-parameters that affect the training behavior with direct influence on the divergence in this section, namely: learning rate and SGLD. The need of fine-tuning these parameters adds complexity to the model and reflects the lack of generalization in the model since each application will have different values that stabilize the training.

Learning rate. It determines the rate at which the weights of the network are adjusted to optimize the model [65]. A larger learning rate means that the model will make more drastic adjustments to the weights, usually, it leads to faster convergence but it also poses the risk of missing the absolute optimum. Whereas, with a smaller learning rate the adjustments will be more gradual, leading to slower convergence and so increasing the training time but can't ensure finding the optimal weights. When close to divergence, there are drastic changes in energy values in a few iterations, changing the learning rate will influence how controlled is the optimization and so modify the fluctuation of the energy function, but with drastic consequences on the training time.

SGLD. It's a stochastic algorithm used in machine learning and artificial intelligence optimization, implemented in the deterministic function of the generation branch that produces new samples from the latent space in the training set [72]. The goal of the generative function is to produce samples as close as the ones from the training data, the number of steps in the SGLD algorithm affects the quality of the generated samples. It's expected that a higher number leads to more accurate estimates of the distribution in the latent space, producing improved generated samples, with the energy values being closer to the observed samples, and so potentially reducing the overall complexity of the dual optimization problem by improving the generation process, but with the drawback of increasing the training time.

3.2. Experiments

Datasets. The training of the model uses two publicly available datasets: the CIFAR-10 Dataset [35] and the EuroSAT Dataset [24]. For both Datasets, only the RGB channels are used. The **CIFAR-10** Dataset consists of 60,000 labeled color natural images of size 32x32 pixels, it's divided into 10 mutually exclusive classes (e.g. airplane, dog, frog, etc.). For the experiments, we used a split of 80% for training, 15% for test, and 5% for validation with a well-balanced distribution of classes. This dataset is used as a demonstration by official JEM codes. We include some reproduced results on it only for comparison purposes with EO data. The **EuroSat** is an EO Dataset and consists of 27,000 labeled multi-spectral (13 bands) satellite images of size 64x64 pixels, it's divided into 10 land use and land cover classes (e.g. annual crop, highway, river, etc.). For the experiments, we used a split of 80% for training and 20% for test and validation with a well-balanced distribution of classes.

Implementation specifics. As presented in [18] we use the Wide-ResNet-28-10 architecture, with the following common main parameters in the experiments: 200 epochs, Adam optimizer [34] and no batch normalization. All the remaining values of the parameters used in the default configuration of the model can be found in Table 3.1. The Pytorch [56] open-source framework is used for all implementations.

In the following, we present the results of the JEM implementation based on [18] and our adaptations for an EO dataset. We encounter divergence in the very first experiments, so we start with a section dedicated to the study of its behavior to get an understanding of the reasons and implications for the model training. Later, we experiment with regularization and hyper-parameters to find a suitable configuration

Parameter	Value
Learning rate	1×10^{-5}
Batch size	64
Number of channels	3
Decay epochs	160, 180
Decay rate	0.3
Sigma	0.03
Buffer size	10000
Reinit frequency	0.05
SGLD number of steps	40
SGLD learning rate	1.0
SGLD std	0.01

Table 3.1. Common hyper-parameters used for JEM classification experiments.

that constrains and regulates the divergence. At the end, we present the potential of JEM for the task of semi-supervised classification.

3.2.1. Divergence

As shown in Table 3.2, divergence was found using the default configuration of the model with both datasets. For the EuroSAT dataset, it happens very early in training, whilst CIFAR-10 completes around five times more epochs. We believe the complexity of remote sensing data accentuates the sensibility of the model to diverge because it has more varied and abstract features as compared to natural images that are object-centric and contain more regular shapes and patterns. Since the focus of our work is in EO, in the following experiments we mainly present the results on the EuroSAT dataset.

Dataset	Completed epochs	Completed epochs (%)
CIFAR-10	51	25.5%
EuroSat	10	5%

Table 3.2. Completed epochs for JEM classification with different datasets.

When inspecting the values of the total loss and the individual contribution of its

two terms across training time, we corroborated that it's the unbounded formulation of the generation loss term the one that causes the divergence, as can be seen in Fig. 3.4 where its contribution corresponds to at least five orders of magnitude of the total loss in the last iteration. Also, when the behavior in the generation process causes the divergence it can be seen how also the classification process starts to get affected, eventually, both process collapses so the complete training process fails.

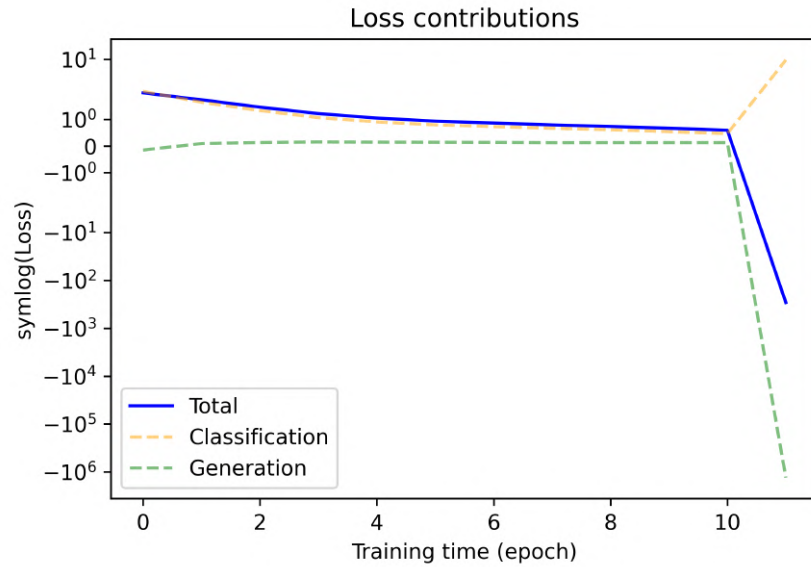


Figure 3.4. Loss contributions across training in JEM for classification. ¹

Investigating the role of the energy function and its behavior during training, we noticed that three types of trend in the energy values can be observed (Fig. 3.5): jitter in the beginning, stability for some time, and an abrupt growth towards the end. The first behavior can be interpreted as the model beginning to learn features and adapting its weights, the stability occurs when the model has found some major set of parameters to optimize, and the sudden rise when the model falls into a local minimum and causes the divergence.

Analyzing the divergence behavior, intuitively one could formulate that it's possible to avoid the divergence by **detection** during training of those occasions where the model might be heading towards divergence and before that happens, load a previous

¹We use the function *symlog* from the Python matplotlib library for representation purposes. It's designed for plots to create a logarithmic scale that is symmetrical around zero.

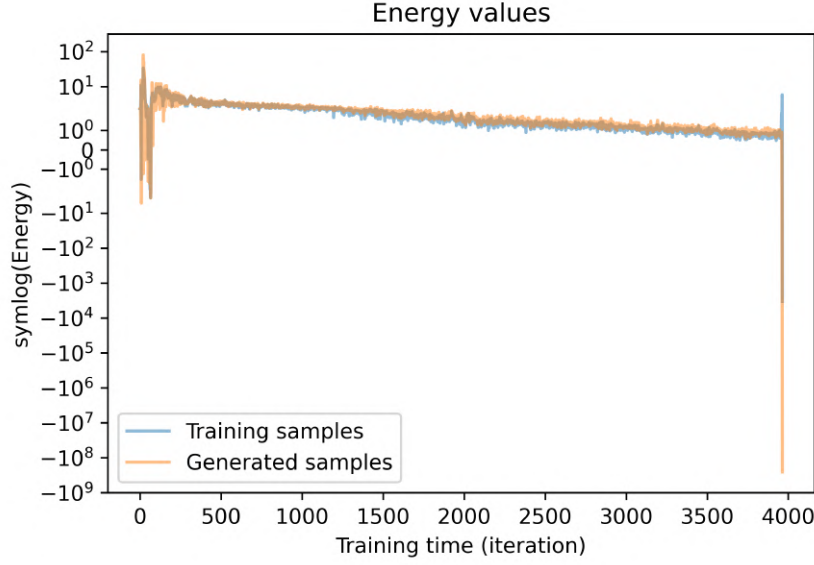


Figure 3.5. Energy values across training in JEM for classification. ¹

state of the model and **restart** the training from that point with different random settings in order to elude the bad local minimum. In the following subsection, we test a pre-emptive approach that explores this idea with the goal of alleviating the need for human intervention to resume training once the model diverges.

3.2.1.1. Detection and restart

Detection. In the training process we use two statistical measures as indicators of the propensity of the model to divergence: rolling standard deviation and the number of Interquartile Range (IQR) outliers. The rolling standard deviation of the absolute energy value differences between training and generated samples can detect short-term variations, thus coping for the volatile characteristic of the divergence. On the other hand, since the training samples won't change during training, we expected their energy values will remain constant. So an outlier in the energy values of training samples means the optimization of the generative process is starting to deviate from the constant range and assigning values out of the expected proportion. The energy value of training samples is directly related to the generation branch loss. Once it starts to exponentially increase, so does the loss until the model diverge.

We then oversee the training using thresholds, one for the rolling standard deviation

α_{RSD} and the other for the number of outliers α_{OTL} . We found empirical values based on detailed observation of divergence cases, specifically the behavior in the corresponding statistical measures. The thresholds are set by finding inflection points over the trends, as shown in Fig. 3.6, 3.7 we see for both the rolling standard deviation and outliers this happens around iteration 35.

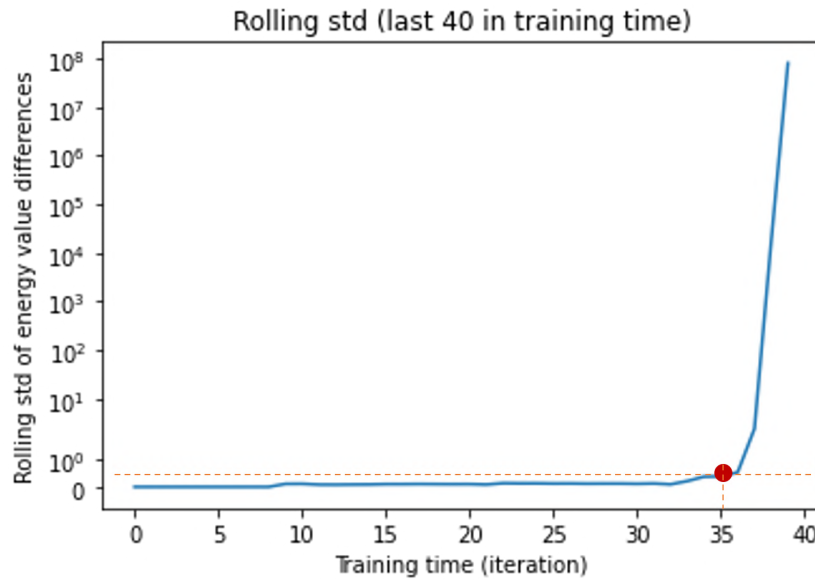


Figure 3.6. Rolling standard deviation of energy values difference in training iterations close to divergence.

Since the goal of detection is to anticipate the divergence, we set the thresholds according to the values in a few iterations before the inflection points. A characteristic of the divergence is how instantaneously it happens, thus to set the thresholds is a critical step since the difference of energy values between contiguous iterations can be immense (e.g. the last two iterations in Fig. 3.5 have energy values with a difference in nine orders of magnitude). The finally chosen thresholds are listed in Table 3.3.

Threshold	Value
Rolling standard deviation α_{RSD}	0.3
Number of outliers α_{OTL}	15

Table 3.3. Thresholds for divergence detection.

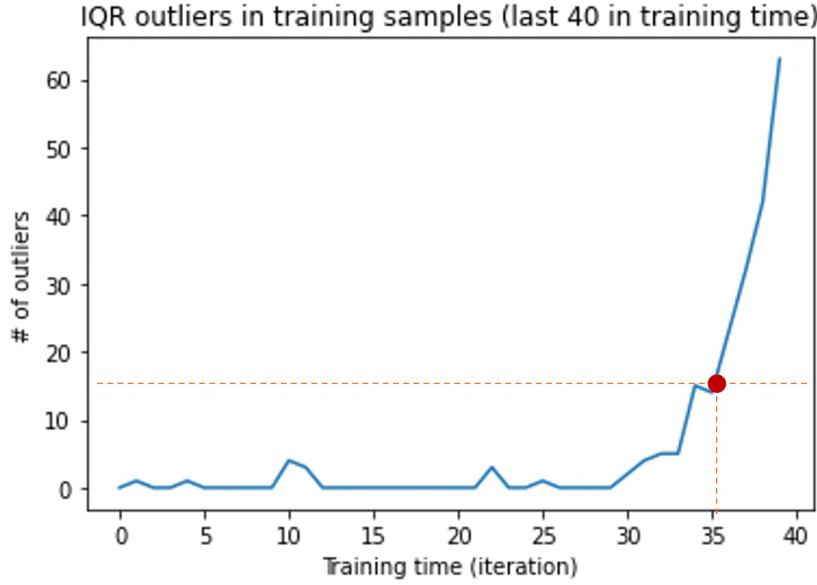


Figure 3.7. Number of Interquartile Range outliers for training samples in training iterations close to divergence.

Restart. We identified three factors that influence the viability of the method to find and load a model from a previous state that is capable of continuing with the training, namely: the size of the jump, replay buffer, and random seed.

The replay buffer and random seed are related to the behavior in the sample generation process. We found that by loading the saved replay buffer from the same previous iteration as the model results more beneficial than restarting the training with an empty buffer. Also, since the generation process includes some stochastic steps, we found that is more beneficial to set a new random seed when restarting the training.

The main idea of this step is to restart training with a saved model. For this, it's imperative to save several states of the model during training. A checkpoint encapsulates the state of a system at a given time. The system we refer to in this section consists of the learning model that consists of the implemented Wide-ResNet object, the optimizer that is the implemented Adaptive Moment Estimation (Adam) object, and the replay buffer which is an array that contains the generated samples. To save a checkpoint we use a state-dictionary PyTorch object.

The frequency of saving and amount of checkpoints depends mainly on the available memory resources. Let γ be the size of training data and η the batch size, an epoch

is completed when the model has been trained with the complete training set and the number of iterations needed is determined by γ/η . If the amount of memory is unlimited or vast, a checkpoint could be saved every iteration, a more realistic scenario is to save a fixed amount of l checkpoints every k iterations so balancing the usage of memory and having enough dispersed checkpoints. An idea for better administration of memory resources is to overwrite the saved checkpoints once a limit is reached. In our experimental settings, we have training size $\gamma = 21,600$ and batch size $\eta = 64$, thus approximately 340 iterations in one epoch; we consider that with $l = 10$ saved checkpoints every $k = 20$ iterations, two-thirds of an epoch will be covered and won't represent a burden to the memory capacity. By testing different jump size values (see Table 3.4) we found that loading a model with a minimum of 30 iterations back is needed to enable the training to restart and continue for more epochs.

Jump size	Completed epochs after training restarted	Replay buffer
30	> 10	Previous state
	< 1	Random initialized
50	> 10	Previous state
	< 1	Random initialized
70	< 1	Previous state
	< 1	Random initialized
100	> 10	Previous state
	< 1	Random initialized

Table 3.4. Number of completed epochs after restart of training with different jump sizes and replay buffers.

This whole process is summarized in Algorithm 1.

Altogether we demonstrated that real-time tracking and pre-emptive detection of divergence is possible. However, restarting training with a state from previous iterations doesn't produce any improvements in the accuracy of the model, specifically, we corroborated that it remained roughly constant in later epochs (see Fig. 3.8, 3.9). It might be the case that by the moment the first divergence warning is activated the model is already biased and headed toward a local minimum in the energy function and

Algorithm 1 Pre-emptive detection of divergence and restart of training

```

 $\alpha_{RSD} \leftarrow 0.3$ 
 $\alpha_{OTL} \leftarrow 15$ 
 $jump\_size \leftarrow 30$ 
 $\omega \leftarrow 10$   $\triangleright$  window size for the rolling standard deviation

 $current\_rstd \leftarrow RollingStd(|x - x'|, \omega)$ 
 $num\_outliers \leftarrow \sum_x IQR\_Outlier(x)$ 
if ( $current\_rstd \geq \alpha_{RSD}$ ) & ( $num\_outliers \geq \alpha_{OTL}$ ) then
    Divergence warning!
     $previous\_replay\_buffer \leftarrow LoadBuffer(jump\_size)$ 
     $previous\_optimizer \leftarrow LoadOptimizer(jump\_size)$ 
     $previous\_model \leftarrow LoadModel(jump\_size)$ 
     $RestartTraining(previous\_replay\_buffer, previous\_optimizer, previous\_model)$ 
end if

```

frequent divergence happens in the consequent epochs after this, for this reason, the majority of experiments couldn't complete more than 1 epoch after restart of training (see Table 3.4). So, adding a gradient threshold to detect the possible divergence earlier in training, when the model isn't biased yet, could be a potential improvement to the method. Nevertheless, repeatedly loading previous models with adjustment to random factors isn't a reliable approach to handle the divergence, thus the need to address it from the theoretical perspective with regularization.

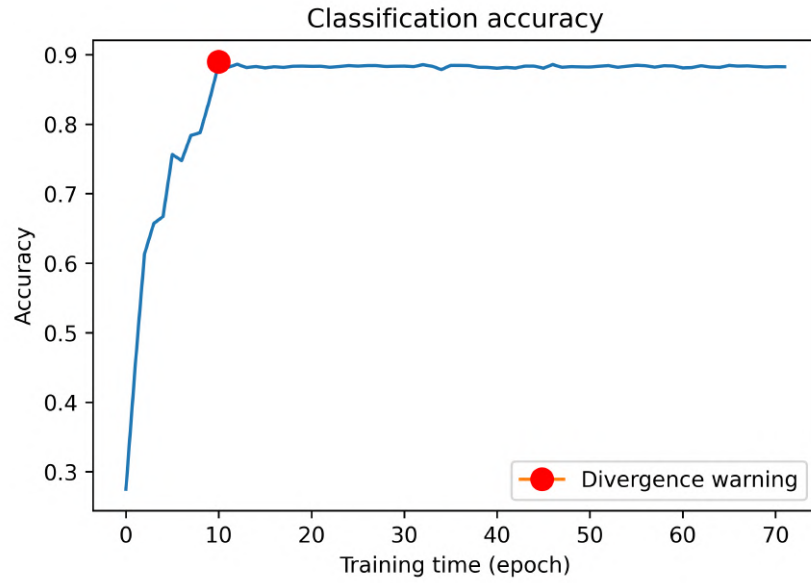


Figure 3.8. Classification accuracy across training for divergence detection and restart.

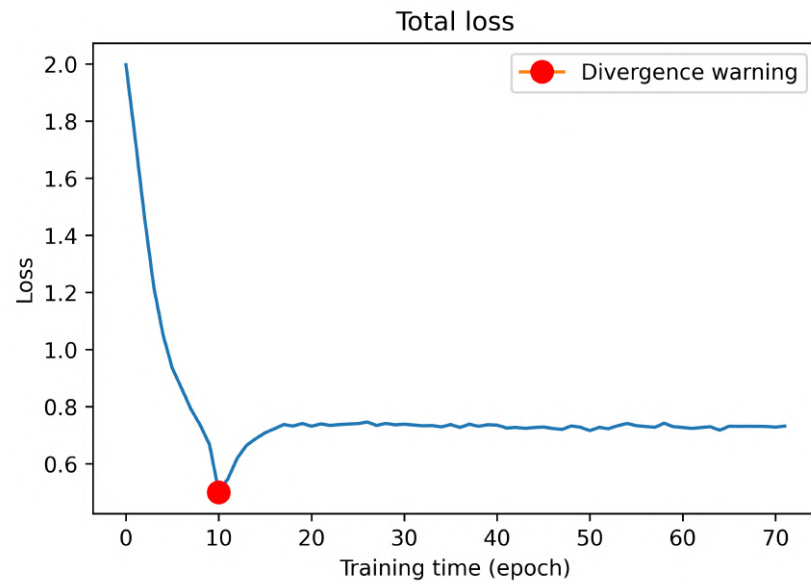


Figure 3.9. Total loss across training for divergence detection and restart.

3.2.2. Regularization

In this section, we test three regularization terms via experiments. Overall the regularization proved to be effective in enabling the model to train for more epochs compared

to the experiment without regularization (see Table 3.5). By setting the constraint to the energy values of samples used for training, the results show improvement in the number of completed epochs, whereas when the constraint is applied to the generated samples the divergence still occurs before reaching the 200 epochs set. The range of energy values for the generated samples is expected to fluctuate over time since the weights of the model will also be changing over time, hence trying to apply a boundary, to this range already unbounded by essence, won't result in the desired adjustment of the total loss. The contrary happens for the samples used in training, since these all belong to real images from satellites, it's expected that despite the difference between classes, the energy values assigned to them will be bounded by the reality or the physical characteristics of its top-level category. Also, the regularization for energy gradients of training samples enabled the model to be trained more epochs but didn't completely avoid the divergence in the 200 epochs set. We see in Fig. 3.5 that divergence happens rapidly during very few iterations, at some point the value in the gradients might be already too high such that its norm surpasses the threshold and thus fail to fully prevent the divergence.

Regularization		Completed epochs		Best classification accuracy (validation)
Weight	Type	#	%	
0.2	Energy of training samples REG_r	200	100%	95.16%
	Energy of generated samples REG_u	155	77.5%	63.42%
	∇ Energy of training samples REG_g	188	59%	89.82%

Table 3.5. L2 regularization on energy values of training and generated samples and on energy gradients of training samples.

From the regularization types we tested, we found that all of them successfully allows the model to train more epochs but with different implications. We see in Table 3.6 that regularization on energy gradients is computationally more expensive as the estimation alone of gradients represents an increase of 110 seconds in training time.

Furthermore, we faced the importance of constraining the loss function but keeping

Regularization type	Enables more training epochs	Complete training	Average time per epoch (seconds)
Energy of training samples REG_r	✓	✓	288
Energy of generated samples REG_u		✗	289
∇ Energy of training samples REG_g			398

Table 3.6. Regularization types implications.

the original formulation. In early attempts to solve the divergence, we disregarded the double optimization problem and formulated intuitively an approach to impose the absolute value operator to the energy values difference in (3.3) so the generation branch could be optimized by minimizing the difference itself to values close to zero and thus ignoring the sign. With this approach, we managed to train the model for more epochs (see Table 3.7), however, the similarity of generated samples to training samples got severely affected (see Fig. 3.10), so we manually interrupted the training as we detected this isn't a reliable approach. Actually, in this case, the discriminative process in the generation branch is removed to some extent. So only the classification branch is trained. That's why we can achieve higher classification accuracy yet very badly generated samples. Proving that the formulation of the dual optimization problem must be preserved, so reliable approaches are like the regularization that just adds a penalty term to the original loss formulation.

Trained epochs (before manual interruption)	Best classification accuracy (validation)
45	94.84%

Table 3.7. Classification accuracy with absolute value operator in generation branch loss.

3.2.3. Hyper-parameters

In this section we investigate the impacts of hyper-parameters in experiments. Table 3.8 shows the amount of completed training epochs for different learning rate values using

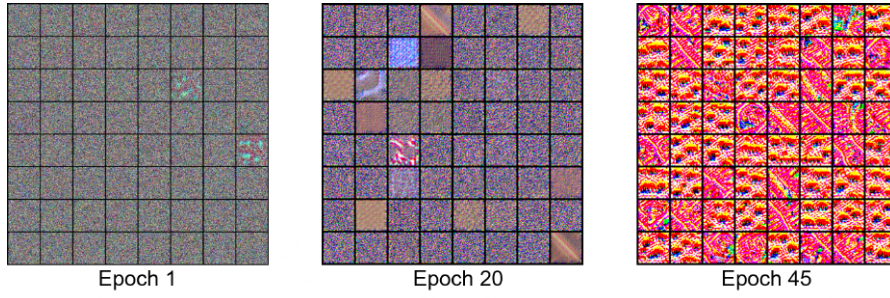


Figure 3.10. Generated samples with absolute value operator in generation branch loss.

the same model and parameters. It can be seen that a smaller learning rate resulted in an increase in epochs that led to better classification accuracy in the evaluation set. So, proving to produce a more controlled optimization that can be beneficial to stabilize the training but not enough to solve the divergence completely and complete the 200 epochs set.

Learning rate	Completed epochs	Completed epochs (%)	Best classification accuracy (validation)
1×10^{-4}	10	5%	84%
1×10^{-5}	86	43%	88.6%

Table 3.8. Completed epochs for different learning rates.

We discovered that increasing the number of SGLD steps won't produce remarkably distinct results in the number of completed epochs, while taking considerably more time to train (see Table 3.9). Also, the quality of the generated samples also doesn't differ much (see Fig. 3.11). Though, this value becomes relevant when testing the capabilities of generating new samples of a trained model.

From the regularization types and hyper-parameters previously explained in detail, we found that the best configuration and combination to successfully address the divergence is: the learning rate of 1×10^{-5} , 40 SGLD steps and regularization on energy values of training samples with 0.2 weight. A summary of the consolidated results is presented in table 3.10.

Number of SGLD steps	Average time per epoch (seconds)	Completed epochs	Completed epochs (%)
40	2310	10	5%
100	5080	9	4.5%

Table 3.9. Completed epochs and average time per epoch for different number of SGLD steps.

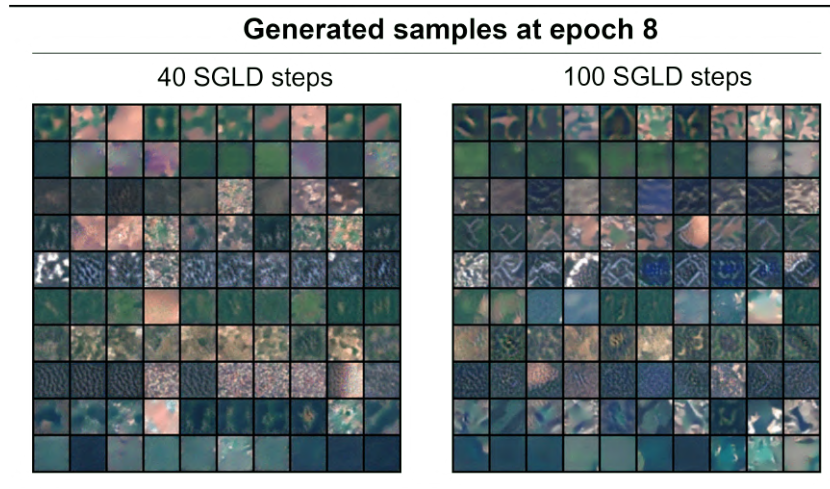


Figure 3.11. Generated samples for different number of SGLD steps.

3.2.4. Semi-supervised classification

The work presented in [79] explores the characteristics of JEM using datasets of natural images. We reproduced the published code with the specified configurations and the same dataset (CIFAR-10). We got comparable results with those in [79] for the task of image classification on natural images. After the model converged to acceptable results, the training was halted as the model diverged in the 51st epoch (as shown in Table 3.11). Since our focus was on remote sensing data, we considered this experiment as just a learning opportunity to get used to the implementation and characteristics of the model.

Complementary results can be found in appendix A, including classification accuracy across training A.1, total loss across training A.2 and generated samples across training A.3.

Learning rate	Regularization		Completed epochs		Best classification accuracy (validation)
	Type	Weight	#	%	
1×10^{-4}	Energy of training samples	0	10	5%	84%
		0.5	7	3.5%	54%
		1	6	3%	40%
1×10^{-5}	Energy of training samples	0	86	43%	88.6%
		0.2	200	100%	95.16%
		0.5	31	15.5%	66.8%
	Energy of generated samples	0.75	30	15%	63%
		0.2	155	77.5%	63.42%
		∇ Energy of training samples	0.2	118	59%

Table 3.10. Completed epochs for different hyper-parameters and regularizations.

Completed epochs	Mean accuracy
51	90.3%

Table 3.11. CIFAR-10 classification accuracy.

After some adjustments of the existing implementation to match the specifications of the remote sensing dataset and finding a configuration for stable training, we got comparable results to those in [6] for the task of image classification. For our experiments, we used the image size of 32 pixels, we consider that by using half of the size with respect to [6] part of the information is lost, therefore some difference in the results is expected. Moreover, this adds to the mentioned challenge of JEM where the time of the sampling method is directly related to the image size [6], a possible investigation in the ratio between information loss and training time could be conducted to find an optimal configuration.

Completed epochs	Mean accuracy
200	95.16%

Table 3.12. EuroSat classification accuracy.

Complementary results can be found in appendix A, including classification accuracy across training A.4, total loss across training A.5, generated samples across training A.6 and classification confusion matrix A.7.

Also, we tested the performance of JEM in the view of its potential for semi-supervised learning by comparing the uncertainty associated with the predictions of different experiments: 100% of labeled data, 80% of labeled data, and 20% of labeled data. For this we used model calibration, which is a figure of merit of a model apart from classification scores, it indicates how informative a model can be regarding the uncertainty associated with the predictions. A perfectly calibrated model matches exactly the confidence of its output with the expected accuracy (represented by dashed lines in Fig. 3.12). The Expected Calibration Error (ECE) [54] is a metric that allows us to quantify this calibration information and compare different models, defined as:

$$ECE = \sum \frac{n}{N} * |acc(e) - conf(e)|, \quad (3.9)$$

where for a given bin e , n is the number of samples in e , N is the total number of samples, $acc(e)$ is the accuracy of the model on the samples in e and $conf(e)$ is the average confidence of the model on the samples in e .

The lower the ECE the better calibration is, meaning that the confidence of the model output is closer to the expected accuracy. We can see in Fig. 3.12 that JEM semi-supervised models are robust since the calibration is close to the ideal case, even accentuated with fewer available labels in the experiments having lower ECE values while still being able to support the classification process (see Table 3.13). Although, the JEM semi-supervised models didn't reach the classification accuracy of the supervised models, we consider the truly potential can be exploited in extreme labeling scenarios (<5% labels). Nevertheless, in Table 3.13 and Fig. 3.12 we can also see how the definition of JEM and the relation between its two branches results in better-calibrated models in comparison with supervised classifier models, proving the benefit of JEM for semi-supervised classification.

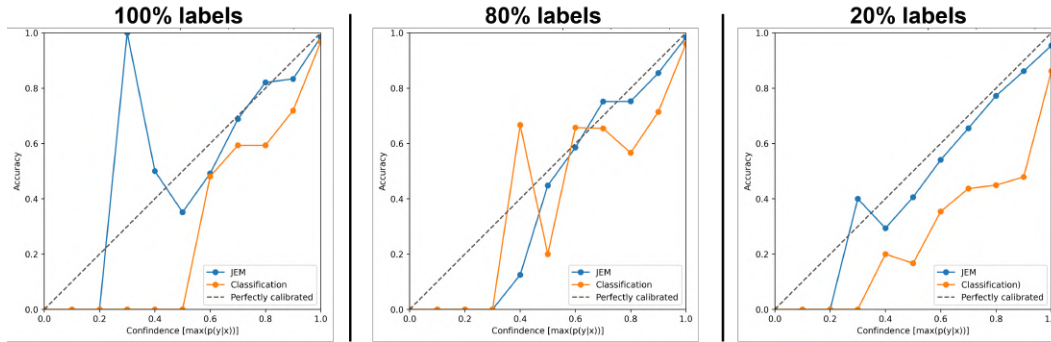


Figure 3.12. EuroSat supervised and semi-supervised classification models calibration.

Mode (% labels)	Configuration	Mean accuracy	ECE
Supervised (100% labels)	JEM	95.16%	1.08%
	Classification	94.86%	3.37%
Semi-supervised (80% labels)	JEM	94.06%	0.71%
	Classification	94.42%	4.10%
Semi-supervised (20% labels)	JEM	80.76%	0.91%
	Classification	81.02%	15.05%

Table 3.13. EuroSat supervised vs semi-supervised classification accuracy and ECE.

4. Joint Energy-based Models for Image Segmentation

"Without assumptions, there can be no progress."

-Stephen Hawking-

In this chapter, we present the assumptions and formulation of the image segmentation problem, also the derivations and changes in the architecture needed to extend from image classification. We test the design with different architectures and comment on the effects on model training and results.

4.1. Methodology

4.1.1. Differences between segmentation and classification

Image segmentation corresponds to the task of assigning a semantic label to each pixel in an image [76] $f_\theta : \mathbb{R}^{W \times H \times D} \rightarrow \mathbb{R}^{W \times H}$, with D input dimensions, W pixels width, H pixels height. Different from classification where a single label is assigned to the whole image $f_\theta : \mathbb{R}^{W \times H \times D} \rightarrow \mathbb{R}$. As shown in Fig. 4.1 and Fig. 4.2.

For the task of classification, the proposed architecture in [18] is a Wide-ResNet-28-10, which is a wide deep residual network mainly consisting of a residual block that acts as an encoder where the networks learn the most important input information in the so-called residual functions.

Neural networks designed for segmentation essentially follow an encoder-decoder architecture, where the decoder transforms the latent representation (formed by the encoder) back into the original input space. This process of reconstruction usually depends on a gradual increase in dimensions. In our work, we used U-Net, which is a convolutional network architecture designed specifically for image segmentation

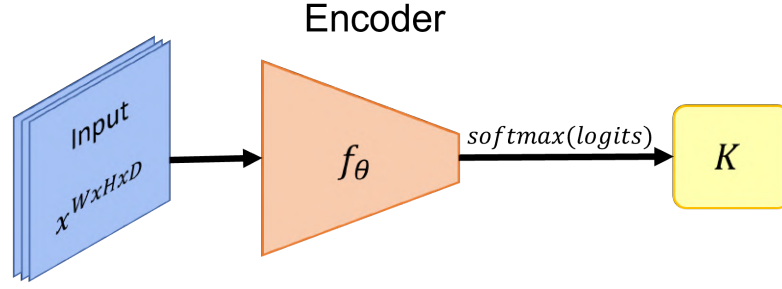


Figure 4.1. Encoder architecture for classification.

tasks. The extension of the architecture and its composition can be seen in the following figure.

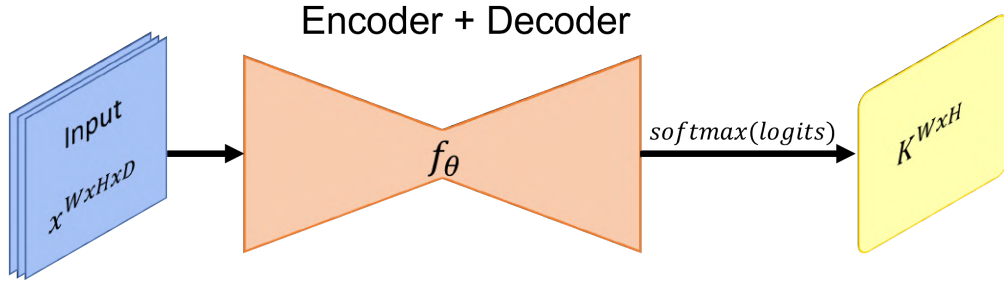


Figure 4.2. Encoder-decoder architecture for segmentation.

In JEM the formulation of the energy function is set for classification problems, where the logits are 1-dimensional because of the encoder architecture it uses. As we stated at the beginning of this section, in segmentation problems we have 2 more dimensions, so we can't use directly the formulation of JEM for it. In the following section, we present then the needed assumptions and derivations to adapt JEM for segmentation.

4.1.2. JEM for segmentation

In the classification architecture mentioned above, the encoder produces a $1 \times K$ -dimensional output $\vec{z} = f_{\theta}(x) = [z_1, \dots, z_K]$, commonly called logits, that represents the unnormalized predicted values (that after normalization results in probabilities) of the input image for each class k_i . The formulation of the energy function in JEM (2.5) uses the Softmax function on the logits to transform the joint distribution $p_{\theta}(x, y)$ to the unnormalized density $p_{\theta}(x)$ by marginalizing out y , in all these formulations the dimension of the logits \mathbb{R}^K enables the energy function to produce a single scalar

value for the entire input x as its definition states. The encoder-decoder adaption for segmentation produces logits as a $W \times H \times K$ -dimensional vector:

$$\vec{z} = f_\theta(x) = \left[\begin{bmatrix} \{w_1, h_1, z_1\} & \dots & \{w_i, h_1, z_1\} \\ \dots & \dots & \dots \\ \{w_1, h_j, z_1\} & \dots & \{w_i, h_j, z_1\} \end{bmatrix}, \dots, \begin{bmatrix} \{w_1, h_1, z_k\} & \dots & \{w_i, h_1, z_k\} \\ \dots & \dots & \dots \\ \{w_1, h_j, z_k\} & \dots & \{w_i, h_j, z_k\} \end{bmatrix} \right], \quad (4.1)$$

so the formulation of the energy function needs to be adapted to handle the extra dimensions. First, for simplicity in the equations, let N be the dimension that represents the entire pixel space ($W \times H$) of a single input x and $f_\theta(x)^n[y_n]$ the single logit that corresponds to the y^{th} , $y \leq K$ class and n^{th} , $n \leq N$ pixel. In order to conglomerate the information of each pixel to represent the whole image, we rely on the assumption that pixels are independent of each other and so the probabilities can be multiplied, subsequently, the formulation of the categorical distribution $p_\theta(y|x)$ results in:

$$p_\theta(Y|X) = \prod_n p_\theta(Y_n|X) = \prod_n \frac{\exp(f_\theta(X)^n[Y_n])}{\sum_{Y'} \exp(f_\theta(X)^n[Y'])} = \frac{\exp(\sum_n f_\theta(X)^n[Y_n])}{\prod_n \sum_{Y'} \exp(f_\theta(X)^n[Y'])}. \quad (4.2)$$

Then the proposed energy based model of the joint distribution $p_\theta(X, Y)$ can be expressed as:

$$p_\theta(X, Y) = \frac{\prod_n \exp(f_\theta(X)^n[Y_n])}{Z(\theta)} = \frac{\exp(\sum_n f_\theta(X)^n[Y_n])}{Z(\theta)}, \quad (4.3)$$

and by marginalizing out Y with $\sum_Y p_\theta(X, Y)$, the unnormalized density $p(X)$ as:

$$p_\theta(X) = \frac{\prod_n \sum_Y \exp(f_\theta(X)^n[Y])}{Z(\theta)}, \quad (4.4)$$

where $Z(\theta)$ is the normalizing constant as in (2.2).

Moreover, resulting in the re-use of the logits in the energy function definition for segmentation:

$$E_\theta(X) = -\log \prod_n \sum_Y \exp(f_\theta(X)^n[Y]), \quad (4.5)$$

from where samples are generated.

Similar to the classification task in JEM, the training of the f_θ is proposed to find the appropriate optimization of the involved distributions by maximizing the likelihood of the joint distribution $\log p_\theta(X, Y) = \log p_\theta(X) + \log p_\theta(Y|X)$, where the term $\log p_\theta(X)$ can be optimized using SGLD (2.4) enabling the generation of fake samples. In the classification task the term $\log p_\theta(Y|X)$ is usually optimized using cross-entropy, in our

work we present the results of adding dice loss to it. For segmentation tasks, where the predictions are dense label maps at pixel level, the dice coefficient is defined as:

$$Dice(Y, \hat{Y}) = 2 \cdot \frac{\sum(\hat{Y} \cdot Y)}{\sum \hat{Y}^2 + \sum Y^2 + \epsilon}, \quad (4.6)$$

measures the overlap between predicted \hat{Y} and ground truth Y label maps and thus motivates the model to produce segmentation maps with high similarity. Here, ϵ is a small constant value that ensures numerical stability by avoiding division by zero.

Extending the definition of (3.2) to:

$$\mathcal{L}_{CLA} = CrossEntropy(Y, \hat{Y}) + Dice(Y, \hat{Y}) \quad (4.7)$$

Resulting in the following proposed architecture of JEM for segmentation:

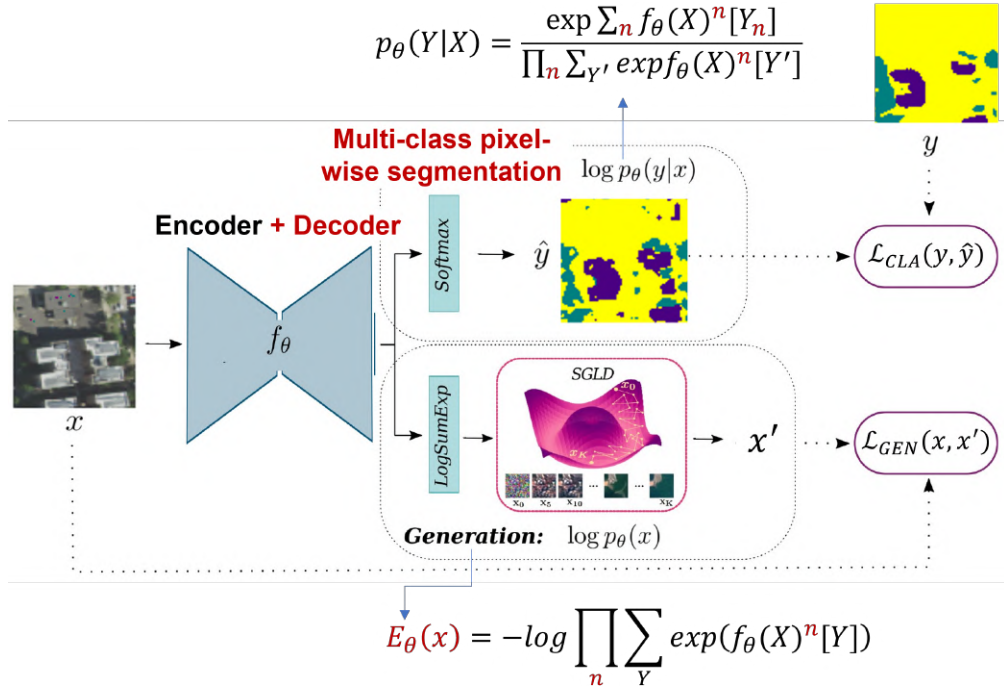


Figure 4.3. JEM for segmentation overview.

4.2. Experiments

Datasets. The **AutoGeoLabel** NYC Dataset [1] consists of 7,140 color images of size 256x256 pixels and pixel-labeled maps with values divided into 3 classes (vegetation,

buildings, and bare land) belonging to New York City. For the experiments, we used a split of 85% for training and 15% for test and validation.

Implementation specifics. Mainly based in [18] and other published architectures [23], with the following common main parameters in the experiments: 200 epochs, Adam optimizer [34] and no batch normalization, and using pixel-wise accuracy as evaluation metric. All the remaining values of the parameters used in the default configuration of the model can be found in Table 4.1. The Pytorch [56] open-source framework is used for all implementations.

Parameter	Value
Learning rate	1×10^{-4}
Batch size	16
Number of channels	3
Number of classes	3
Decay epochs	160, 180
Decay rate	0.3
Sigma	0.03
Buffer size	3000
Reinit frequency	0.05
SGLD number of steps	20
SGLD learning rate	1.0
SGLD std	0.01

Table 4.1. Common hyper-parameters used for JEM segmentation experiments.

In the following, we present the results of the JEM adaptation to segmentation. In the first experiments, we detected some limitations in the performance of the model, so in the second section, we analyze the individual behavior of the two branches and find a configuration that boosts the learning to some extent. We also encounter divergence, so at the end, we present the differences in the behavior of the energy values compared with the classification task and how this affects our attempts of regularization.

Following the encoder presented in [18] and extending the architecture by adding a decoder, in our work we used a Wide-ResNet (encoder) + U-Net (decoder) and Vanilla U-Net (encoder-decoder). From the changes in architecture and implementation with respect to the classification experiments, we identify two main differences: the addition

of a decoder to the architecture and the 8 times increase in image size in the training samples.

To test the implementation of the needed adaptations for segmentation, we set an experiment using all the training samples with the original size of the images and the Vanilla U-Net architecture, after several epochs we stopped the training as we noticed the training loss was constantly increasing over time, the validation accuracy was considerably low and fluctuating, and the quality of generated samples was far from looking alike remote sensing images. We decreased the complexity of the experiment and set the image size to half of the original (128x128 pixels) and one-quarter of it (64x64 pixels), unfortunately, the training behavior didn't improve much in these experiments (see Fig 4.4).

Architecture	Training size (samples)	Validation size (samples)	Sample size (pixels)	Best segmentation accuracy (validation)
Vanilla U-Net	6,000	1,140	256x256	29.67%
			128x128	55.94%
			64x64	56.62%

Table 4.2. Segmentation results with U-Net and different sample sizes.

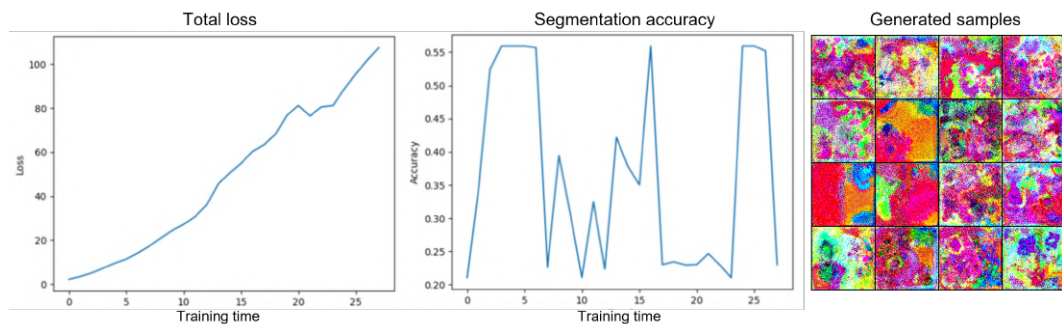


Figure 4.4. Vanilla U-Net segmentation experiment with 128x128 pixels image size.

In order to determine what is causing the undesired training behavior, we analyzed in detail the behavior of the segmentation and generation branch when operating individually, and present the results in the following section.

4.2.1. Individual branches analysis

Segmentation branch. The U-Net is widely used in segmentation and proven to work with various datasets. In order to check the performance of each branch, here we set experiments with JEM but only using the segmentation branch with 6,000 training samples and 1,140 validation samples. We corroborated that the used network predicts segmentation maps with acceptable accuracy and similarity (see Fig. 4.5), so the segmentation branch alone performs well with the adaptations and our dataset. Moreover, to determine a balance between information loss and reduced training time, in Table 4.3 we see that the decrease in sample size by a factor of 4 reduces the time by 43 seconds and the accuracy by 4.84% and a factor of 2 reduces the time by 33 seconds and the accuracy by 1.76%, so we consider that the image size of 128x128 pixels produces enough reduction in the training time while having a considerably small decrement in accuracy, being the best balance for this reduced experiment and can serve as an indicator for the general behavior of the segmentation branch. Then, for the following experiments we used 128x128 pixels as the minimum image size to reduce the complexity of the training while preserving enough spatial information.

Sample size (pixels)	Average time per epoch (seconds)	Best segmentation accuracy (validation)
64x64	11	83.83%
128x128	21	86.91%
256x256	54	88.67%

Table 4.3. Segmentation branch results with Vanilla U-Net and different sample sizes.

Also, with the preliminary segmentation maps results in the experiments with the segmentation branch alone, we noticed that the predicted maps were accurate in establishing the structures and shapes of the input, but need some improvement to assign correctly the class to the objects. Thus, we added to the segmentation loss the Dice metric and got a slight improvement in the average accuracy (see Table 4.4) and more similarity in the produced segmentation maps as seen in Fig. 4.6.

Generation branch. To test the implications of the network adaptations in the energy function and the generative process, we set experiments with JEM but only using the generation branch. We noticed that Vanilla U-Net for the segmentation task has limitations in the generation process as the quality of the generated samples was far

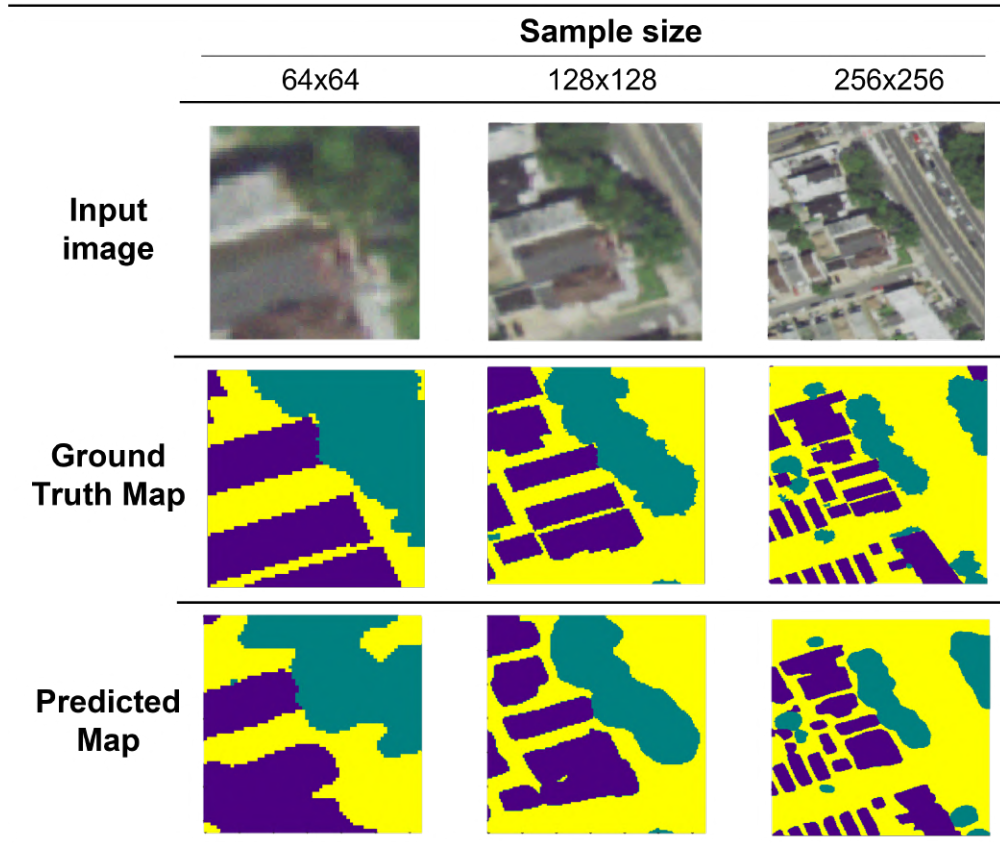


Figure 4.5. Segmentation prediction maps with different sample sizes.

Sample size (pixels)	Segmentation Loss	Best segmentation accuracy (validation)
64x64	Cross-Entropy	83.83%
	Cross-Entropy + Dice	84.40%
128x128	Cross-Entropy	86.91%
	Cross-Entropy + Dice	87.05%
256x256	Cross-Entropy	88.67%
	Cross-Entropy + Dice	88.72%

Table 4.4. Segmentation branch results with different loss metrics.

from optimal. The generated samples didn't look like remote sensing images but like random noise instead (see 4.4). As we already mentioned, the architecture for the

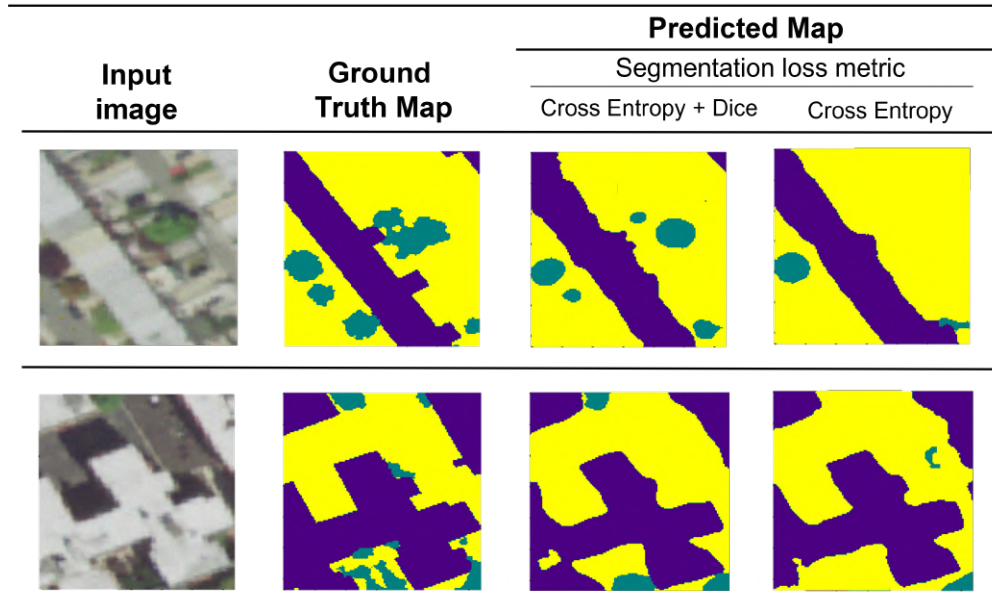


Figure 4.6. Segmentation prediction maps with different loss metrics.

segmentation task is composed of an encoder and a decoder, where the extraction of features is carried out by the encoder. So first, we tested the capability of the different encoders in Table 4.5 for the generative process to assess the capabilities of the networks to extract meaningful features from our dataset. Notice that here we test all encoders in a classification fashion, that is, no decoder is used. All encoders are based on CNNs architecture. Wide-ResNet is an extension of the ResNet architecture where the encoder consists of multiple Residual Blocks, the difference being Wide-ResNet uses more filters in each block. Whereas the U-Net encoder consists of a series of convolutional and max-pooling layers.

Encoder
U-Net
Wide-ResNet
ResNet18
ResNet34
ResNet50

Table 4.5. Encoders for image segmentation.

We set experiments with the minimum complexity, using only 1 training sample to identify the effectiveness in recovering relevant information and producing high-quality samples that look close to the input, the results are presented in 4.6. We determined that the Wide-ResNet encoder was the best architecture for the generation branch alone. The U-Net and ResNet encoders perform a few more down-sampling operations in comparison to Wide-ResNet, so more spatial information is lost which is harmful for image reconstruction, thus Wide-ResNet has an advantage in this sense. Also, we found that the fine-tuning of the learning rate hyper-parameter is of great importance for this branch since it directly affects the convergence of the model and varies for each of the tested architectures.


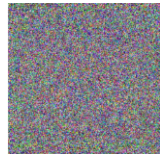
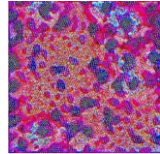
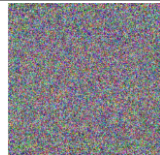
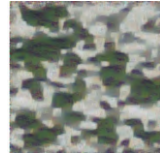
Training sample	Encoder	Learning rate	Generated sample
	ResNet18/34/50	1×10^{-2}	
		1×10^{-3}	
		1×10^{-4}	
		1×10^{-5}	
	U-Net	1×10^{-3}	
		1×10^{-4}	
		1×10^{-5}	
	Wide-ResNet	1×10^{-3}	
		1×10^{-4}	

Table 4.6. Generation branch results with different encoders and learning rates.

As the focus of this chapter is on the task of image segmentation, with the observations from the encoder experiments, we tested again the performance of the generative process using the segmentation architecture Wide-ResNet (encoder) + U-Net (decoder). We see in Fig. 4.7 that the capabilities of the generative branch are still limited, as the appearance of the generated samples is not yet close to the expected remote sensing look. Although this architecture shows some improvement in the generation capabilities compared with the Vanilla U-Net, we consider that this branch limitation influences the overall performance of the model in the image segmentation task. In the derivations to adapt from classification to segmentation we part from the given assumption that there is no spatial autocorrelation between the pixels, with our experiments we consider this assumption might be too strong for EO data where neighboring pixels definitely show to be autocorrelated.

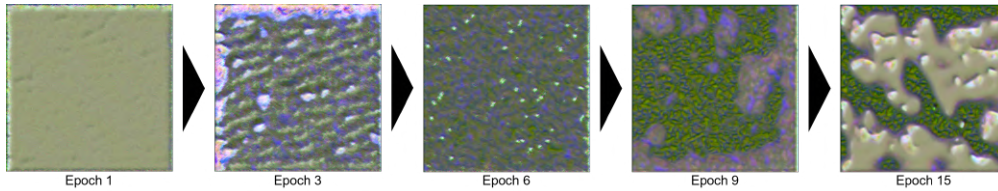


Figure 4.7. Generated samples with Wide-ResNet + U-Net.

4.2.2. Hyper-parameters

After the experiments from the individual operation of the branches and given the unstable character of the first experiments with the combined branches, we experimented with fine-tuning the hyper-parameters. Specifically: the learning rate and buffer size, for these experiments we used the best configuration from the individual branch experiments: Wide-ResNet (encoder) with a sample size of 128x128 pixels, 10 samples for training and 5 samples for validation, and adding the Dice metric to the segmentation branch loss.

In the experiments with the generation branch alone, we found that the network architecture is sensible to the learning rate, for the Wide-ResNet using 1×10^{-4} got the best results, so it became the default value of our experiments. In the classification experiments (Table 3.8 decreasing the value to 1×10^{-5} resulted in a more controlled learning process and thus an increase in the accuracy; in the segmentation experiments decreasing the learning rate also resulted in a more controlled learning process but,

even with several more epochs, it never reached neither surpassed the results of the experiment that uses the default learning rate value. We consider that in a more complex task, such as segmentation, inspecting the individual behavior of the branches is key to finding the best learning rate value for the architecture in use.

The iterative process of SGLD uses a buffer to store the generated samples, and in its initialization, random samples are produced using a uniform distribution with values between $[-1, 1]$ and samples from the buffer. The lower the buffer size, the higher the probability of using generated samples from recent SGLD iterations in the random initialization, we can see in Table 4.7 that -independent of the learning rate- the minor buffer sizes have the highest accuracy, as the iterative generation process benefits when it's initialized with the uniform distribution and more realistic samples. In addition, this hyper-parameter depends also on the size of the training set, so it is important to experiment and inspect the generation branch to find the value that produces the best results. From the hyper-parameters experiments, we found the best combination for our experimental configuration is to use a learning rate of 1×10^{-4} and buffer size 3,000 for more stable training but without improvement in the average accuracy.

Learning rate	Buffer size	Best segmentation accuracy (validation)
1×10^{-4}	3,000	67.88%
	6,000	59.48%
	9,000	39.25%
	30,000	38.26%
	60,000	38.37%
1×10^{-5}	3,000	39.02%
	6,000	38.86%

Table 4.7. Segmentation results with Wide-ResNet + U-Net, different learning rates, and buffer sizes.

Once we found a stable combination of architecture and hyper-parameters, we tested the learning capabilities of the model considering different levels of information complexity. For this, we performed a new series of experiments using the complete JEM (both branches in operation) with different sizes for training and validation sets. Despite the average improvement of 10% in the best validation accuracy compared with the initial experiment where Vanilla U-Net was used (Table 4.2), the model didn't

reach satisfactory results, as it can be seen in Fig. 4.8 the predictions represent some of the big features but the model clearly needs further training especially for the finer spatial features. In general, we observed that the dual optimization process might carry enough complexity in the training, as the complete configuration (both branches) couldn't match the individual branch performance. More robust pixels aggregations for the energy function formulation are needed to improve the model and amplify the capabilities of JEM for segmentation.

Architecture	Sample size (pixels)	Training size (samples)	Validation size (samples)	Best segmentation accuracy (validation)
Wide-ResNet	128x128	10	5	67.88%
+		80	20	62.43%
U-Net		2,000	20	47.50%

Table 4.8. Segmentation results with Wide-ResNet + U-Net and different sizes for training and validation sets.

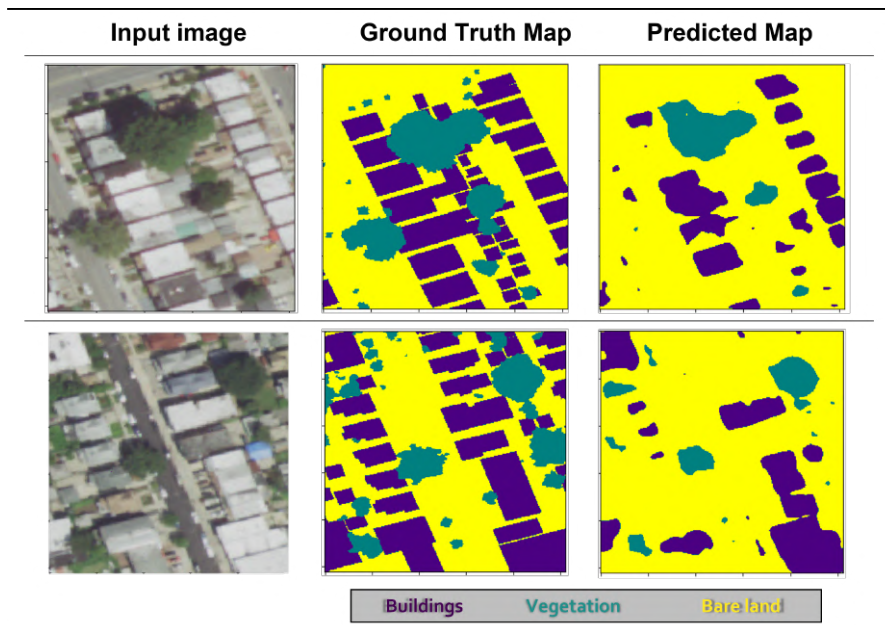


Figure 4.8. Segmentation prediction maps with Wide-ResNet + U-Net.

4.2.3. Divergence

In the segmentation experiments, we also faced divergence, unfortunately, the regularization techniques we used in the classification experiments to alleviate the complexity of the dual optimization problem didn't work for the given configuration. Inspecting the formulation of the energy function and compared with the one in classification, the 2 additional dimensions in the logits impose a substantial increase in the whole magnitude of the energy values. Consider now that every pixel in the image is aggregated to have a single scalar energy value for the entire sample, thus for the 128x128 image size, we have around 16,000 values to aggregate and so the energy values of segmentation images are in general 4 orders of magnitude bigger than the energy values of classification, as we can see in Fig. 4.9. When applying the L2 regularization, the energy value is squared, so the order of magnitude is consequently quadratic increased which leads to immediately surprising the divergence threshold. In this case, more powerful regularization is required.

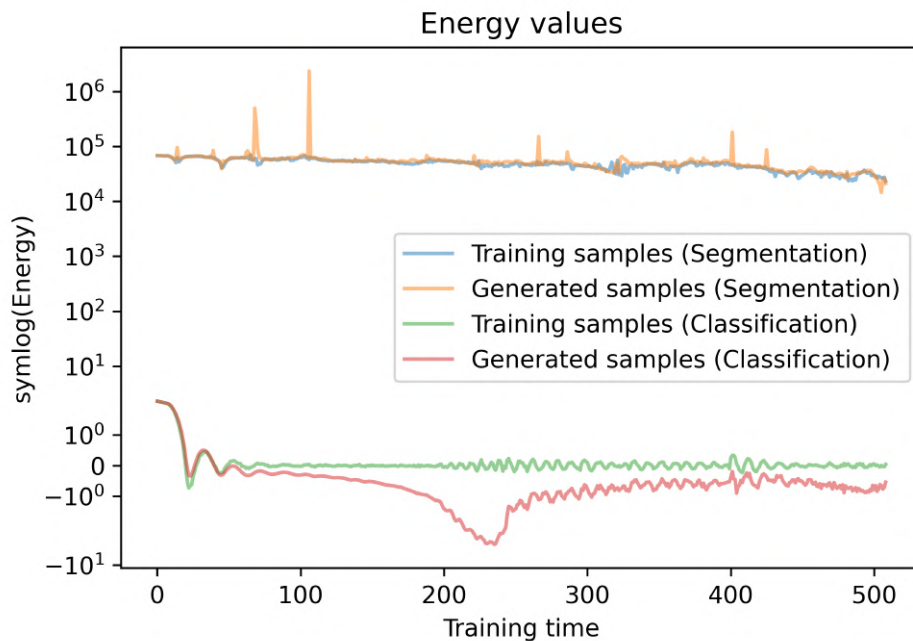


Figure 4.9. Energy values for segmentation and classification. ¹

¹We use the function *symlog* from the Python matplotlib library for representation purposes. It's designed for plots to create a logarithmic scale that is symmetrical around zero.

5. Conclusions

"...Success is not final, failure is not fatal: It is the courage to continue that counts."

-Winston Churchill-

5.1. Conclusions

In this work, we focus on the behavior of JEM and specifically on applications to EO data. From the formulation of the loss optimization and energy function in JEM, we established that the generation process represents the most significant contributor to the instability of the training. The unbounded contribution of the generation loss skews the model towards divergence where no more meaningful learning is possible. Although some works [18, 79] mention the empirical adjustment of hyper-parameters to postpone the divergence, we present in our work the connection between the definition of learning rate and the number of SGLD steps and the concept of divergence to demonstrate such approaches don't offer a reliable procedure to avoid divergence, since it is not based on the theory of it but depends on experimental experience. A known drawback of the divergence is the need for human intervention to reinstate the training. Following it, we utilize an empirical method for pre-emptive detection of divergence and conclude that restarting the training with a previous state of the model isn't enough to avoid the divergence. So knowing the rationale behind the divergence, we formulate regularization terms as naturally suited approaches to bound the generation loss that successfully regulates the training behavior while keeping the basis of the JEM formulation. From the point of view of the energy function landscape, we found an intuitive explanation for the potential of regularization is the easement of the competition in the dual optimization problem. We discovered that applying the regularization to the energy values of training samples is a more beneficial approach, as the sample generation process induces many stochastic steps so the

energy values would be naturally volatile across training. We found a configuration of regularization and hyper-parameters that are suited for the classification task on remote sensing images, reaching comparable results with works [6] on the EuroSAT dataset and confirming the big potential for WSL as JEM models trained in a semi-supervised fashion resulted better calibrated than a full supervised model.

When transitioning from an image classification task to an image segmentation one we found hindrances in the model, mainly related to the complexity added by the additional two dimensions in the feature space and the assumption to aggregate the pixels. In the implications analysis of adapting a deep neural network classification architecture to a segmentation one, we discovered the segmentation branch alone with the Vanilla U-Net architecture performs adequately, also we increased the average accuracy and similarity of predicted segmentation maps by extending the formulation of the segmentation loss with the Dice term; whereas the generation branch is restricted in the quality of fake samples with closest remote-sensing-like aspect using a Wide-ResNet encoder. We sum up the model's limitations to the validity of the assumption we did in the theoretical derivations from classification to segmentation where we assumed the pixels to be independent, especially in remote sensing images where spatial autocorrelation is a common characteristic.

With this work, we contribute to the understanding of JEM giving insights into the divergence problem. Also, we continue the work in the open research questions of reliable training for EBM and opportunities of JEM for segmentation. We consider this work to be relevant as advances in image processing of remote sensing data are of big importance in the field of EO.

5.2. Future lines

One of the biggest open challenges in JEM is the instability of the training, so exploration in further regularization terms can be beneficial to increase the effectiveness in alleviating the divergence. Another implication of the training instability is the constant need for human intervention. Thus, further investigation into methods like the detection and restart we present, can provide robust ways to detect divergence in advance and avoid it. Especially, big improvements can be done in the restart of the training, where studies from a theoretical perspective of model training can establish a robust technique to load a model that isn't yet biased and can continue learning.

While JEM have been proven to work for classification on different remote sensing datasets, more comprehensive approaches to extend for segmentation are of interest for applications like detailed land cover and land use and change detection. Specifically, a potential way is to investigate statistically robust definitions to aggregate the energy values of all pixels into a single representative singular value of the image, for instance including measures of spatial autocorrelation like imposing Markov Random Field (MRF) on the energy value aggregation process. Also, improvements in the generation process to control the feedback sensibility in the network to bad-looking samples are an interesting area to explore because of the potential to regulate the general learning process and prevent divergence.

Acronyms

CNNs Convolutional Neural Networks.

DL Deep Learning.

DNNs Deep Neural Networks.

EBM Energy-based Models.

EO Earth Observation.

GANs Generative Adversarial Networks.

JEM Joint Energy-based Models.

MCMC Markov chain Monte Carlo.

SGLD Stochastic Gradient Langevin Dynamics.

SSL Semi-supervised Learning.

SVMs Support Vector Machines.

VAEs Variational Autoencoders.

WSL Weakly-supervised Learning.

List of Figures

2.1. Weakly-supervised learning types.	6
2.2. Generation of samples using SGLD.	11
3.1. JEM for semi-supervised learning overview [6].	16
3.2. Landscape of energy values.	18
3.3. Constrained landscape of energy values with regularization.	21
3.4. Loss contributions across training in JEM for classification.	24
3.5. Energy values across training in JEM for classification.	25
3.6. Rolling standard deviation of energy values difference in training iterations close to divergence.	26
3.7. Number of Interquartile Range outliers for training samples in training iterations close to divergence.	27
3.8. Classification accuracy across training for divergence detection and restart.	30
3.9. Total loss across training for divergence detection and restart.	30
3.10. Generated samples with absolute value operator in generation branch loss.	33
3.11. Generated samples for different number of SGLD steps.	34
3.12. EuroSat supervised and semi-supervised classification models calibration.	37
4.1. Encoder architecture for classification.	40
4.2. Encoder-decoder architecture for segmentation.	40
4.3. JEM for segmentation overview.	42
4.4. Vanilla U-Net segmentation experiment with 128x128 pixels image size.	44
4.5. Segmentation prediction maps with different sample sizes.	46
4.6. Segmentation prediction maps with different loss metrics.	47
4.7. Generated samples with Wide-ResNet + U-Net.	49
4.8. Segmentation prediction maps with Wide-ResNet + U-Net.	51
4.9. Energy values for segmentation and classification.	52
A.1. CIFAR-10 classification accuracy across training.	71
A.2. CIFAR-10 total classification loss across training.	72

A.3. CIFAR-10 generated samples across training.	72
A.4. EuroSat classification accuracy across training.	73
A.5. EuroSat total classification loss across training.	73
A.6. EuroSat generated samples across training.	74
A.7. EuroSat classification confusion matrix.	74

List of Tables

3.1. Common hyper-parameters used for JEM classification experiments. . .	23
3.2. Completed epochs for JEM classification with different datasets.	23
3.3. Thresholds for divergence detection.	26
3.4. Number of completed epochs after restart of training with different jump sizes and replay buffers.	28
3.5. L2 regularization on energy values of training and generated samples and on energy gradients of training samples.	31
3.6. Regularization types implications.	32
3.7. Classification accuracy with absolute value operator in generation branch loss.	32
3.8. Completed epochs for different learning rates.	33
3.9. Completed epochs and average time per epoch for different number of SGLD steps.	34
3.10. Completed epochs for different hyper-parameters and regularizations. .	35
3.11. CIFAR-10 classification accuracy.	35
3.12. EuroSat classification accuracy.	35
3.13. EuroSat supervised vs semi-supervised classification accuracy and ECE.	37
4.1. Common hyper-parameters used for JEM segmentation experiments. .	43
4.2. Segmentation results with U-Net and different sample sizes.	44
4.3. Segmentation branch results with Vanilla U-Net and different sample sizes.	45
4.4. Segmentation branch results with different loss metrics.	46
4.5. Encoders for image segmentation.	47
4.6. Generation branch results with different encoders and learning rates. .	48
4.7. Segmentation results with Wide-ResNet + U-Net, different learning rates, and buffer sizes.	50
4.8. Segmentation results with Wide-ResNet + U-Net and different sizes for training and validation sets.	51

Bibliography

- [1] C. M. Albrecht, F. Marianno, and L. J. Klein. "Autogeolabel: Automated label generation for geospatial machine learning." In: *2021 IEEE International Conference on Big Data (Big Data)*. IEEE. 2021, pp. 1779–1786.
- [2] F. Alidoost and H. Arefi. "Application of deep learning for emergency response and disaster management." In: *Proceedings of the AGSE Eighth International Summer School and Conference*. University of Tehran. 2017, pp. 11–17.
- [3] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel. "Mixmatch: A holistic approach to semi-supervised learning." In: *Advances in neural information processing systems* 32 (2019).
- [4] N. Bhatia et al. "Survey of nearest neighbor techniques." In: *arXiv preprint arXiv:1007.0085* (2010).
- [5] M.-A. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon. "Multiple instance learning: A survey of problem characteristics and applications." In: *Pattern Recognition* 77 (2018), pp. 329–353.
- [6] J. Castillo-Navarro, B. Le Saux, A. Boulch, and S. Lefèvre. "Energy-Based Models in Earth Observation: From Generation to Semisupervised Learning." In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–11. doi: 10.1109/TGRS.2021.3126428.
- [7] N. K. Chauhan and K. Singh. "A review on conventional machine learning vs deep learning." In: *2018 International conference on computing, power and communication technologies (GUCON)*. IEEE. 2018, pp. 347–352.
- [8] S. Chib. "Chapter 57 - Markov Chain Monte Carlo Methods: Computation and Inference." In: ed. by J. J. Heckman and E. Leamer. Vol. 5. *Handbook of Econometrics*. Elsevier, 2001, pp. 3569–3649. doi: [https://doi.org/10.1016/S1573-4412\(01\)05010-3](https://doi.org/10.1016/S1573-4412(01)05010-3).

- [9] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu. "Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 4836–4845.
- [10] P. Dayan and G. E. Hinton. "Using Expectation-Maximization for Reinforcement Learning." In: *Neural Computation* 9.2 (Feb. 1997), pp. 271–278. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.2.271. eprint: <https://direct.mit.edu/neco/article-pdf/9/2/271/813557/neco.1997.9.2.271.pdf>.
- [11] G. Druck, C. Pal, A. McCallum, and X. Zhu. "Semi-supervised classification with hybrid generative/discriminative methods." In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, pp. 280–289.
- [12] Y. Du and I. Mordatch. *Implicit Generation and Generalization in Energy-Based Models*. 2019. DOI: 10.48550/ARXIV.1903.08689.
- [13] A. Dubey, N. Naik, D. Parikh, R. Raskar, and C. A. Hidalgo. "Deep learning the city: Quantifying urban perception at a global scale." In: *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I* 14. Springer. 2016, pp. 196–212.
- [14] J. van Engelen and H. Hoos. "A survey on semi-supervised learning." In: *Machine Learning* 109 (2020), pp. 373–440. DOI: 10.1007/s10994-019-05855-6.
- [15] J. Foulds and E. Frank. "A review of multi-instance learning assumptions." In: *The knowledge engineering review* 25.1 (2010), pp. 1–25.
- [16] Z. Ghahramani. "Unsupervised learning." In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures* (2004), pp. 72–112.
- [17] H. GM, M. K. Gourisaria, M. Pandey, and S. S. Rautaray. "A comprehensive survey and analysis of generative models in machine learning." In: *Computer Science Review* 38 (2020), p. 100285. ISSN: 1574-0137. DOI: <https://doi.org/10.1016/j.cosrev.2020.100285>.
- [18] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. *Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One*. 2019. DOI: 10.48550/ARXIV.1912.03263.
- [19] D. Groth, S. Hartmann, S. Klie, and J. Selbig. "Principal components analysis." In: *Computational Toxicology: Volume II* (2013), pp. 527–547.

- [20] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye. "A review on generative adversarial networks: Algorithms, theory, and applications." In: *IEEE transactions on knowledge and data engineering* (2021).
- [21] M. F. A. Hady and F. Schwenker. "Semi-supervised Learning." In: *Handbook on Neural Information Processing*. Springer, 2013, pp. 215–239. doi: 10.1007/978-3-642-36657-4_7.
- [22] D. Hao, L. Zhang, J. Sumkin, A. Mohamed, and S. Wu. "Inaccurate Labels in Weakly-Supervised Deep Learning: Automatic Identification and Correction and Their Impact on Classification Performance." In: *IEEE Journal of Biomedical and Health Informatics* 24.9 (2020). PMID: 32078570; PMCID: PMC7429345, pp. 2701–2710. doi: 10.1109/JBHI.2020.2974425.
- [23] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.
- [24] P. Helber, B. Bischke, A. Dengel, and D. Borth. "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.7 (2019), pp. 2217–2226.
- [25] S. Helmstetter and H. Paulheim. "Weakly supervised learning for fake news detection on Twitter." In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE. 2018, pp. 274–277.
- [26] A. A. Heydari and A. Mehmood. "SRVAE: super resolution using variational autoencoders." In: *Pattern Recognition and Tracking XXXI*. Vol. 11400. SPIE. 2020, pp. 87–100.
- [27] Y. Higuchi, N. Moritz, J. Le Roux, and T. Hori. "Momentum pseudo-labeling: Semi-supervised asr with continuously improving pseudo-labels." In: *IEEE Journal of Selected Topics in Signal Processing* 16.6 (2022), pp. 1424–1438.
- [28] G. E. Hinton. "Training Products of Experts by Minimizing Contrastive Divergence." In: *Neural Computation* 14.8 (2002), pp. 1771–1800. doi: 10.1162/089976602760128018.
- [29] G. Huang, Z. Liu, L. Maaten, and K. Weinberger. "Densely Connected Convolutional Networks. Computer Vision and Pattern Recognition." In: *arXiv preprint arXiv:1608.06993* (2016).

- [30] R. Huang, C. Liu, G. Li, and J. Zhou. "Adaptive deep supervised autoencoder based image reconstruction for face recognition." In: *Mathematical Problems in Engineering* 2016 (2016).
- [31] S. A. Israel, J. Goldstein, J. S. Klein, J. Talamonti, F. Tanner, S. Zabel, P. A. Sallee, and L. McCoy. "Generative adversarial networks for classification." In: *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*. IEEE. 2017, pp. 1–4.
- [32] T. Jebara. *Machine learning: discriminative and generative*. Vol. 755. Springer Science & Business Media, 2012.
- [33] T. Joachims et al. "Transductive inference for text classification using support vector machines." In: *ICML*. Vol. 99. 1999, pp. 200–209.
- [34] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980.
- [35] A. Krizhevsky, G. Hinton, et al. "Learning multiple layers of features from tiny images." In: (2009).
- [36] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks." In: *Communications of the ACM* 60.6 (2017), pp. 84–90.
- [37] S. Latif, R. Rana, J. Qadir, and J. Epps. "Variational autoencoders for learning latent representations of speech emotion: A preliminary study." In: *arXiv preprint arXiv:1712.08708* (2017).
- [38] E. G. Learned-Miller. "Introduction to supervised learning." In: *I: Department of Computer Science, University of Massachusetts* (2014), p. 3.
- [39] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [40] C. S. Lee, E. Sohn, J. D. Park, and J.-D. Jang. "Estimation of soil moisture using deep learning based on satellite data: A case study of South Korea." In: *GIScience & Remote Sensing* 56.1 (2019), pp. 43–67.
- [41] D.-H. Lee et al. "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks." In: *Workshop on challenges in representation learning, ICML*. Vol. 3. 2. 2013, p. 896.
- [42] Y.-F. Li, L.-Z. Guo, and Z.-H. Zhou. "Towards safe weakly supervised learning." In: *IEEE transactions on pattern analysis and machine intelligence* 43.1 (2019), pp. 334–346.

- [43] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan. "Perceptual generative adversarial networks for small object detection." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1222–1230.
- [44] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun. "Adversarial ranking for language generation." In: *Advances in neural information processing systems* 30 (2017).
- [45] B. Liu and B. Liu. *Supervised learning*. Springer, 2011.
- [46] Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt. "Constrained graph variational autoencoders for molecule design." In: *Advances in neural information processing systems* 31 (2018).
- [47] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang. "Transfer learning using computational intelligence: A survey." In: *Knowledge-Based Systems* 80 (2015), pp. 14–23.
- [48] Z. Ma and G. Mei. "Deep learning for geological hazards analysis: Data, models, applications, and opportunities." In: *Earth-Science Reviews* 223 (2021), p. 103858.
- [49] O. Maron and T. Lozano-Pérez. "A framework for multiple-instance learning." In: *Advances in neural information processing systems* 10 (1997).
- [50] D. Meyer and F. Wien. "Support vector machines." In: *R News* 1.3 (2001), pp. 23–26.
- [51] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. "Image segmentation using deep learning: A survey." In: *IEEE transactions on pattern analysis and machine intelligence* (2021).
- [52] S. Mittal, S. Srivastava, and J. P. Jayanth. "A survey of deep learning techniques for underwater image classification." In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).
- [53] E. Nijkamp, M. Hill, T. Han, S.-C. Zhu, and Y. N. Wu. "On the Anatomy of MCMC-Based Maximum Likelihood Learning of Energy-Based Models." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020), pp. 5272–5280. DOI: 10.1609/aaai.v34i04.5973.
- [54] M. Pakdaman Naeini, G. Cooper, and M. Hauskrecht. "Obtaining Well Calibrated Probabilities Using Bayesian Binning." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (Feb. 2015). DOI: 10.1609/aaai.v29i1.9602.
- [55] S. J. Pan and Q. Yang. "A survey on transfer learning." In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

- [56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library." In: *Advances in neural information processing systems* 32 (2019).
- [57] W. H. L. Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli. "Autoencoders." In: *Machine learning*. Elsevier, 2020, pp. 193–208.
- [58] D. Qiao, G. Liu, T. Lv, W. Li, and J. Zhang. "Marine vision-based situational awareness using discriminative deep learning: A survey." In: *Journal of Marine Science and Engineering* 9.4 (2021), p. 397.
- [59] L. Rabiner and B. Juang. "An introduction to hidden Markov models." In: *ieee assp magazine* 3.1 (1986), pp. 4–16.
- [60] D. A. Reynolds et al. "Gaussian mixture models." In: *Encyclopedia of biometrics* 741.659-663 (2009).
- [61] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, et al. "Tackling climate change with machine learning." In: *ACM Computing Surveys (CSUR)* 55.2 (2022), pp. 1–96.
- [62] *Semi-supervised learning* / [edited by] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien. eng. Adaptive computation and machine learning series. Cambridge, Massachusetts: MIT Press, 2010. ISBN: 1-282-09618-4.
- [63] K. P. Sinaga and M.-S. Yang. "Unsupervised K-means clustering algorithm." In: *IEEE access* 8 (2020), pp. 80716–80727.
- [64] A. Singh and T. Ogunfunmi. "An Overview of Variational Autoencoders for Source Separation, Finance, and Bio-Signal Applications." In: *Entropy* 24.1 (2022). ISSN: 1099-4300. DOI: 10.3390/e24010055.
- [65] L. N. Smith. "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay." In: *arXiv preprint arXiv:1803.09820* (2018).
- [66] M. Sudmanns, D. Tiede, S. Lang, H. Bergstedt, G. Trost, H. Augustin, A. Baraldi, and T. Blaschke. "Big Earth data: disruptive changes in Earth observation data management and analysis?" In: *International Journal of Digital Earth* 13.7 (2020), pp. 832–850.

- [67] M. Suhail, A. Mittal, B. Siddiquie, C. Broaddus, J. Eledath, G. Medioni, and L. Sigal. "Energy-based learning for scene graph generation." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 13936–13945.
- [68] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.
- [69] Y. Tian and Y. Zhang. "A comprehensive survey on regularization strategies in machine learning." In: *Information Fusion* 80 (2022), pp. 146–166. issn: 1566-2535. DOI: <https://doi.org/10.1016/j.inffus.2021.11.005>.
- [70] A. Vali, S. Comai, and M. Matteucci. "Deep learning for land use and land cover classification based on hyperspectral and multispectral earth observation data: A review." In: *Remote Sensing* 12.15 (2020), p. 2495.
- [71] K. Vesely, M. Hannemann, and L. Burget. "Semi-supervised training of deep neural networks." In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE. 2013, pp. 267–272.
- [72] M. Welling and Y. W. Teh. "Bayesian learning via stochastic gradient Langevin dynamics." In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 681–688.
- [73] T. Wolf, V. Sanh, J. Chaumond, and C. Delangue. "Transfertransfo: A transfer learning approach for neural network based conversational agents." In: *arXiv preprint arXiv:1901.08149* (2019).
- [74] H. Wu and S. Prasad. "Semi-supervised deep learning using pseudo labels for hyperspectral image classification." In: *IEEE Transactions on Image Processing* 27.3 (2017), pp. 1259–1270.
- [75] X. Yang. "Hybrid Energy-based Models for Image Generation and Classification." PhD thesis. Georgia State University, 2022. DOI: 10.57709/32208392.
- [76] X. Yuan, J. Shi, and L. Gu. "A review of deep learning methods for semantic segmentation of remote sensing imagery." In: *Expert Systems with Applications* 169 (2021), p. 114417.
- [77] Y. Zhang, H. Lian, G. Yang, S. Zhao, P. Ni, H. Chen, and C. Li. "Inaccurate-Supervised Learning With Generative Adversarial Nets." In: *IEEE Transactions on Cybernetics* (2021).

- [78] Z.-Y. Zhang, P. Zhao, Y. Jiang, and Z.-H. Zhou. "Learning from incomplete and inaccurate supervision." In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 1017–1025.
- [79] S. Zhao, J.-H. Jacobsen, and W. Grathwohl. "Joint energy-based models for semi-supervised classification." In: *ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning*. Vol. 1. 2020.
- [80] Z.-H. Zhou. "A brief introduction to weakly supervised learning." In: *National Science Review* 5.1 (Aug. 2017), pp. 44–53. issn: 2095-5138. doi: 10.1093/nsr/nwx106. eprint: <https://academic.oup.com/nsr/article-pdf/5/1/44/31567770/nwx106.pdf>.
- [81] Z.-H. Zhou and M. Li. "Semi-supervised learning by disagreement." In: *Knowledge and Information Systems* 24.3 (2010), pp. 415–439.
- [82] Z.-H. Zhou and J.-M. Xu. "On the relation between multi-instance learning and semi-supervised learning." In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 1167–1174.
- [83] X. Zhu and A. B. Goldberg. *Introduction to Semi-Supervised Learning*. Vol. 3. Synthesis Lectures on Artificial Intelligence and Machine Learning 1. Morgan & Claypool Publishers, 2009, pp. 1–130. doi: 10.2200/S00196ED1V01Y200906AIM006.

A. Appendix

JEM for classification complementary results

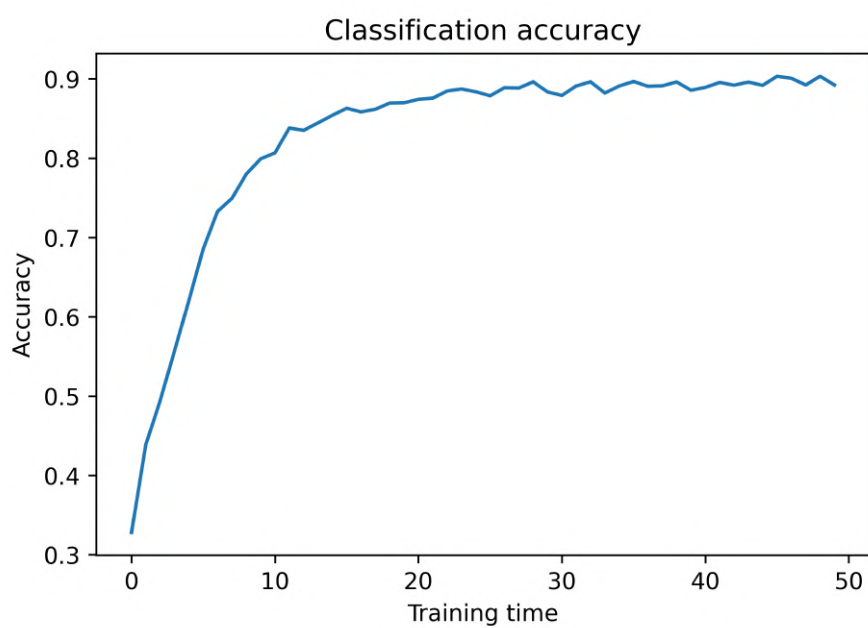


Figure A.1. CIFAR-10 classification accuracy across training.

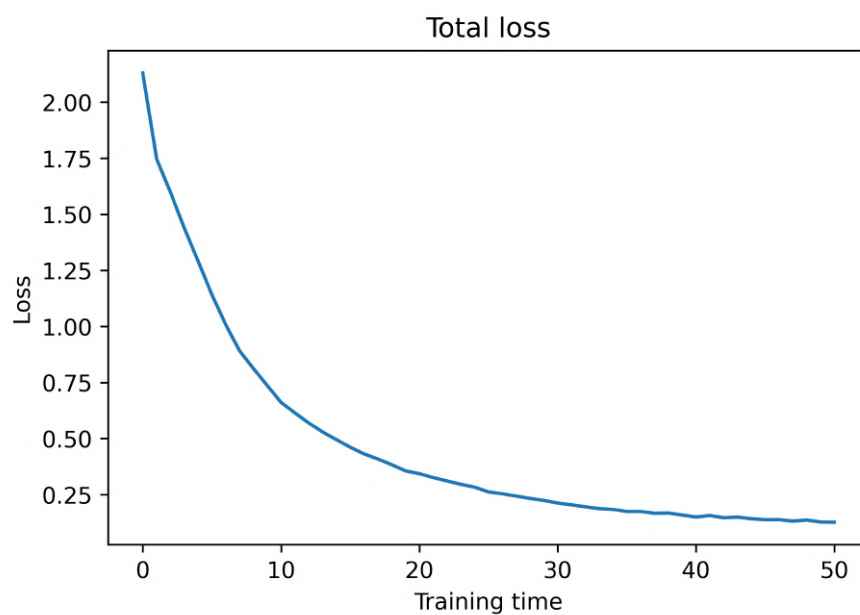


Figure A.2. CIFAR-10 total classification loss across training.



Figure A.3. CIFAR-10 generated samples across training.

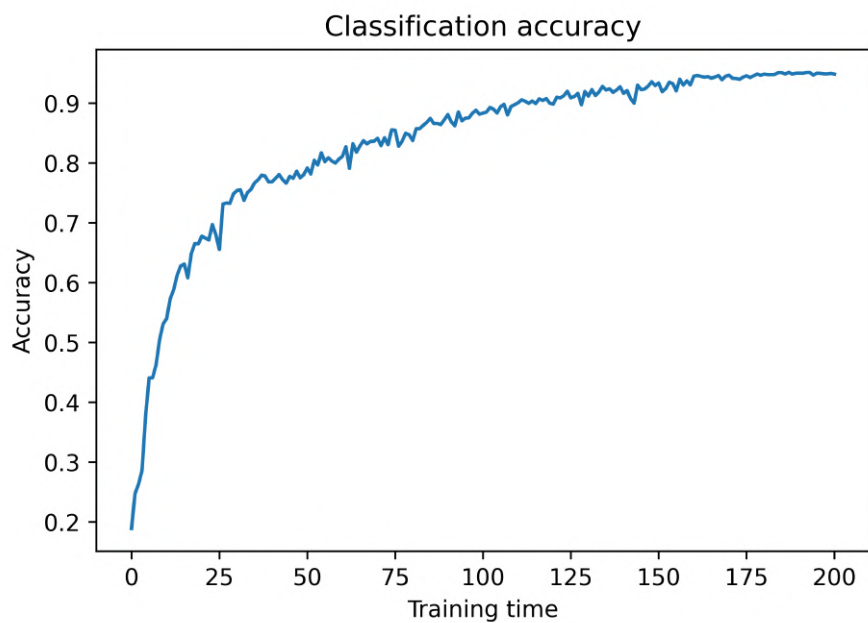


Figure A.4. EuroSat classification accuracy across training.

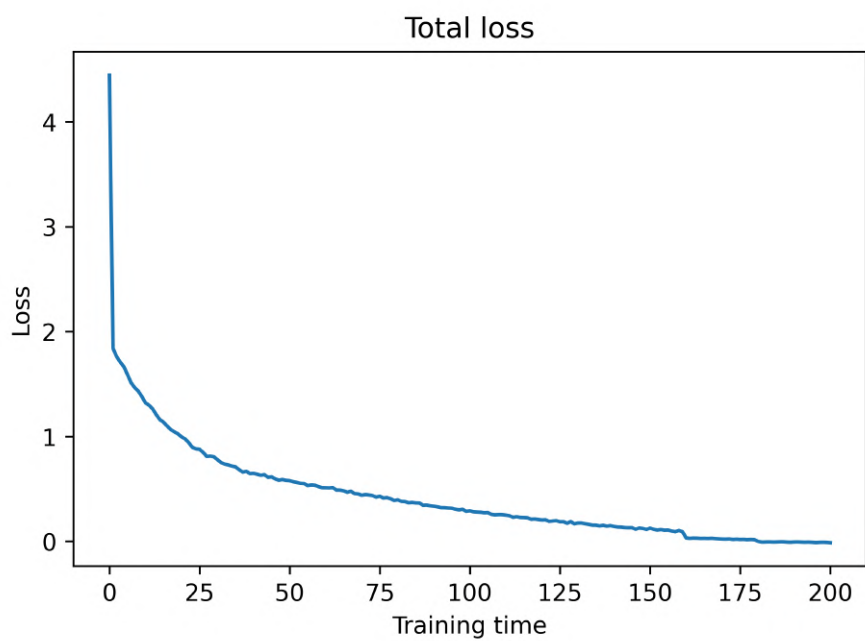


Figure A.5. EuroSat total classification loss across training.

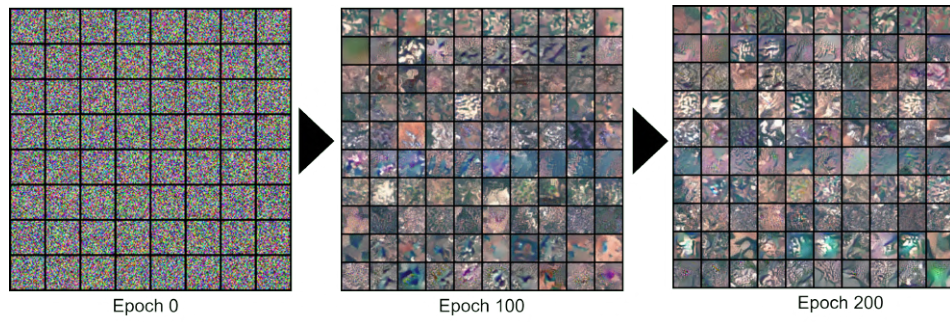


Figure A.6. EuroSat generated samples across training.

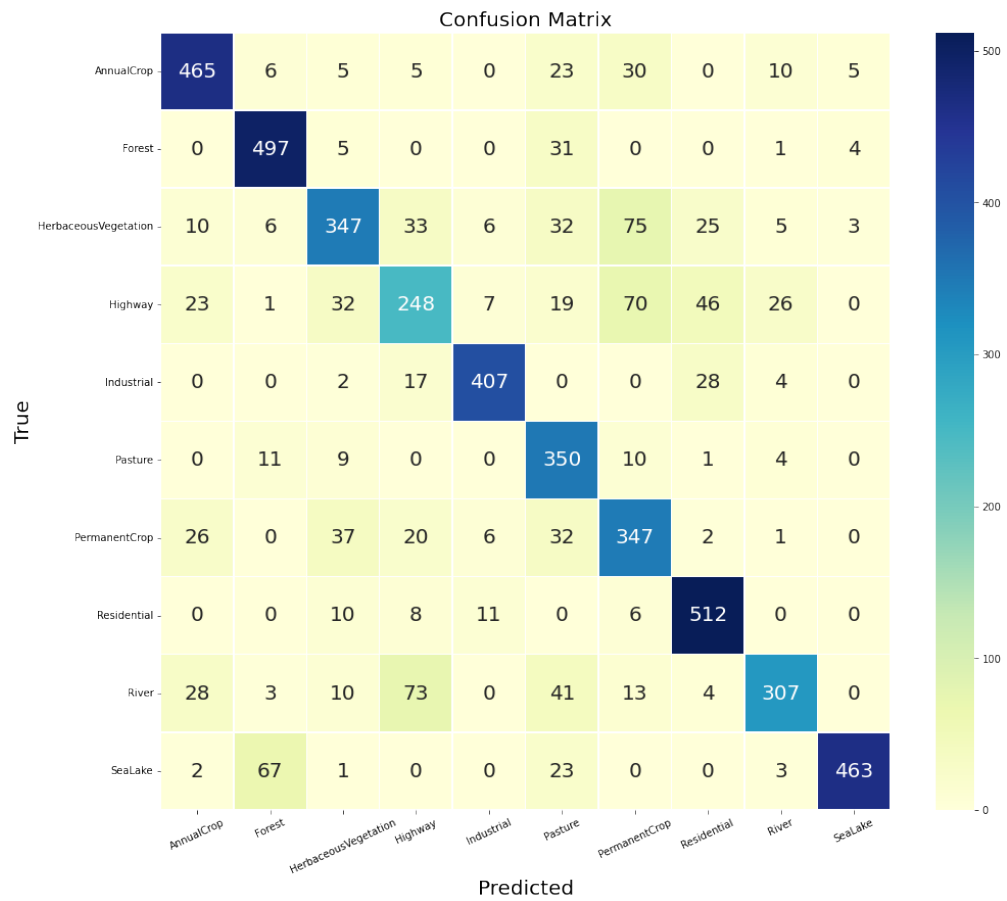


Figure A.7. EuroSat classification confusion matrix.