Revision: Semi Supervised Learning

Minor in AI - IIT ROPAR

5th May, 2025

What is Semi-Supervised Learning (SSL)?

Semi-Supervised Learning (SSL) is a powerful and pragmatic approach within machine learning that strategically combines a small amount of labeled data with a large pool of unlabeled data. In many real-world applications, acquiring labeled data is expensive, labor-intensive, and often requires domain expertise (such as radiologists for medical images, or linguists for language tasks). On the other hand, unlabeled data is plentiful and usually much easier to obtain—for example, raw texts from the web, untagged photos, or untranscribed speech.

SSL fills the space between two traditional learning paradigms: supervised learning and unsupervised learning. Supervised learning exclusively uses labeled examples, where each data point is paired with a corresponding label, such as an image annotated as "cat" or "dog." Unsupervised learning, in contrast, deals solely with unlabeled data and typically focuses on identifying structure within the data, such as clustering or dimensionality reduction. SSL, being a hybrid, harnesses the strengths of both—using the few labeled examples to guide the learning process while exploiting the structure inherent in the unlabeled data to enrich and reinforce the model's understanding.

The main objective of SSL is to achieve high performance without requiring large amounts of labeled data. By learning from both types of data, SSL models can uncover relationships, structures, and clusters that would remain hidden if only labeled data were used. This is especially crucial in domains where labeling is infeasible at scale. SSL models are designed to use unlabeled data not just as passive input but as a meaningful contributor to the training process, improving the generalization and robustness of the final model.

Motivation and Real-Life Examples

The practical motivation behind SSL is rooted in the imbalance between the availability of unlabeled and labeled data. In many domains, while there's no shortage of raw data, annotation is a bottleneck. To illustrate this, consider two concrete examples:

The first example is email spam detection. Suppose a user manually labels 100 emails as either "Spam" or "Not Spam." This is a small labeled dataset, as each labeling action consumes time and effort. Meanwhile, there may be an additional 10,000 emails available without any labels. A supervised model trained only on the 100 labeled samples would likely perform poorly due to limited data diversity. However, SSL makes it possible to improve model performance by also learning from the patterns, language structure, and metadata in the 10,000 unlabeled emails. Even without explicit labels, the model can identify recurring features common to spam messages and differentiate them from legitimate ones.

The second example comes from the medical field, where X-ray images must be reviewed and labeled by radiologists—a slow and expensive process. Suppose you have access to 500 labeled X-rays and 50,000 unlabeled ones. Instead of labeling all 50,000 (which might take months), SSL can help the model learn from the patterns present in both the labeled and unlabeled datasets. For instance, the model may notice shared structural patterns among images indicating pneumonia, even when labels are not provided. This enables the creation of a robust diagnostic tool with significantly reduced annotation costs.

Mathematical Formulation

To formally define Semi-Supervised Learning, let's assume we have two datasets. The first, denoted as

$$D_L = \{(x_i, y_i)\}_{i=1}^{l}$$

consists of l labeled samples where x_i represents the input features and y_i the corresponding label. The second dataset,

$$D_U = \{x_i\}_{i=l+1}^{l+u}$$

consists of u unlabeled samples. Together, the total dataset includes $l + u$ samples.

The learning task is to find a function $f(x)$—often a classifier or regression model—that generalizes well, not just over the labeled data but also in the context of the entire input space that includes the unlabeled data. This is usually done by minimizing a combined loss function that includes a supervised component (based on the labeled data) and an unsupervised component (based on patterns inferred from the unlabeled data). The inclusion of the unsupervised loss encourages the model to discover the structure in the data distribution, ensuring smoother decision boundaries and improved generalization.

Why SSL is Important

The significance of SSL lies in its ability to circumvent the need for large labeled datasets, which are often the main bottleneck in deploying machine learning systems. Labeling is expensive not just in monetary terms but also in the time and effort required from skilled professionals. For instance, in medical imaging, a single scan might take a specialist several minutes to evaluate, and massive datasets can require months of annotation.

Unlabeled data, by contrast, is ubiquitous. It exists in the form of logs, documents, emails, sensor readings, audio files, and more. These datasets are already collected in the course of normal business or operations, making them essentially "free" in terms of cost.

SSL unlocks the latent potential of these unlabeled datasets. It enables models to leverage both data types, reducing reliance on labels while improving performance. This is especially crucial in fields like:

Medical Imaging: Few labeled scans, massive hospital databases

Speech Recognition: Abundant audio, limited transcripts

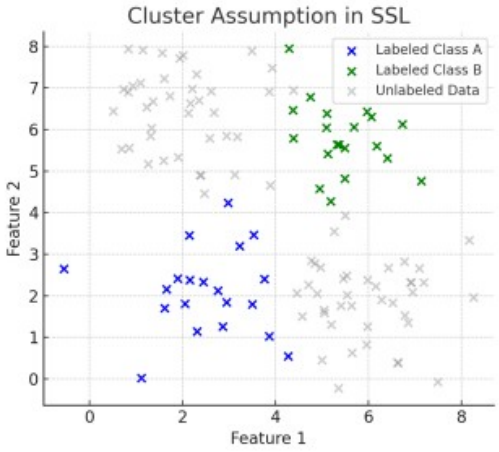
Text Classification: Rich corpora of raw documents

Autonomous Driving: Millions of road scenes, few annotated ones

Visualizing SSL

To visualize how SSL works, imagine a 2D space where data points are plotted based on their features. Labeled points might be red (class A) and blue (class B). The labeled data, being sparse, provide only a rough idea of the boundary between classes. Unlabeled points, though colorless, populate the space and form discernible clusters and patterns.

A model trained only on labeled points might draw a poor boundary, cutting across dense regions. SSL techniques use the unlabeled points to infer that the boundary should avoid cutting through dense clusters and instead run through sparse regions, aligning better with the true data distribution. This helps the model achieve low-density separation, leading to improved classification accuracy.



Comparing Learning Paradigms

SSL stands out when compared to traditional learning paradigms. In supervised learning, only labeled data is used, which limits scalability and adaptability when labeled data is scarce. In unsupervised learning, there is no label information at all, which restricts tasks to clustering or representation learning.

SSL, by utilizing a mix of few labeled and many unlabeled examples, offers a compromise. It provides the model with direct supervision where available and allows it to generalize patterns from the unlabeled data. This is particularly useful for tasks like spam detection, where collecting a few labeled samples is feasible, but labeling thousands is not.

Learning Type	Data Used	Example
Supervised	Only labeled	Image classification
Unsupervised	Only unlabeled	Customer segmentation
Semi-Supervised	Few labeled + many unlabeled	Spam detection with limited labels

Key Assumptions in SSL

Self-Training

Self-training is one of the simplest yet effective semi-supervised learning strategies. The central idea is to iteratively improve the model by using its own high-confidence predictions as ground truth labels for unlabeled data. Initially, a classifier is trained using the small set of labeled data. This classifier is then used to predict labels for the unlabeled data. From these predictions, only those with a confidence score above a specified threshold (e.g., 95%) are selected. These high-confidence predictions are treated as pseudo- labels, and the corresponding data points are added to the labeled dataset. The model is then retrained using this expanded dataset. This process is repeated over several iterations. Over time, the model becomes more confident and starts labeling more data correctly. However, care must be taken as poor pseudo-labels early on can mislead the training process. It is crucial to use a well-calibrated confidence threshold to mitigate noise in pseudo-labels.

Co-Training

Co-training assumes the data can be represented from two or more independent and sufficient views. Two separate models are trained on different feature subsets. For example, when classifying web pages:

- View 1: Page content (words like “goal”, “score”)
- View 2: Hyperlinks (e.g., links to ESPN)

Each classifier labels unlabeled examples for the other. This mutual learning process helps both models improve as long as the views are conditionally independent and each view alone is sufficient for classification.

Generative Model Assumption

This assumption is that the data are generated by underlying probabilistic distributions (e.g., Gaussians). If the model can fit a probability distribution to the labeled and unlabeled data (say, one Gaussian per class), it can assign a new point to the most likely distribution (i.e., class). This method is effective when the actual data-generating process aligns with the assumed distributions.

Cluster Assumption

This states that points in the same cluster likely share the same label. By clustering both labeled and unlabeled data (using, for instance, K-means), we can assign cluster-wide labels based on the few labeled samples.

Low-Density Separation

The principle here is that a good classifier should place its decision boundary in low-density regions—areas of the feature space with fewer data points. This reduces the chance of misclassifying similar samples. This is the rationale behind SSL techniques like Transductive SVMs.

Manifold Assumption

This powerful idea proposes that high-dimensional data often lie on a low-dimensional manifold. For example, images of handwritten digits live in a 784-dimensional space (28×28 pixels), but the variation is smooth and controlled (stroke, slant, thickness), forming a lower-dimensional structure. SSL models can propagate labels along this manifold, assigning similar labels to nearby points.

Summary Table of SSL Assumptions

Assumption	Key Idea	Example
Self-Training	Confident predictions are likely correct	Red fruit labeled as "Apple"
Co-Training	Independent views teach each other	Text + hyperlink views of a webpage
Generative Model	Data comes from known distributions	Gaussian clusters in 2D
Cluster	Same cluster → same label	Cat/dog image clusters
Low-Density	Boundaries avoid dense regions	Two moons toy dataset
Manifold	Labels vary smoothly along manifolds	Handwriting of digit "3"

Related Paradigms to SSL

Transfer Learning

Transfer Learning involves transferring knowledge learned in one domain (called the source domain) to a different but related domain (the target domain). Usually, this involves pretraining a model on a large labeled dataset (e.g., ImageNet), and then fine-tuning it on a smaller labeled dataset from the target domain (e.g., X-ray classification). The primary goal is to reuse features and model weights learned from one context to improve performance in another, especially when labeled data is limited in the target domain.

Weakly-Supervised Learning

Weakly-Supervised Learning focuses on using imperfect labels, rather than few labels. These imperfections might include:

- Noisy labels (e.g., auto-tagged tweets)
- Incompletely labeled data (e.g., videos labeled only by title)
- Coarse labels (e.g., document labeled with topics but no sentence-level tags)

While SSL assumes you have a few high-quality labels, Weak Supervision tolerates many low-quality labels. The model then learns to identify signal amidst the noise. SSL and Weak Supervision can be combined for greater flexibility in real-world tasks.

Positive and Unlabeled (PU) Learning

PU Learning is a special case of SSL where only positive examples are labeled, and the rest are unlabeled. There are no known negative examples. A common use case is spam detection, where spam emails are flagged, but non-spam (ham) emails are not explicitly labeled.

Meta-Learning

Meta-Learning, or "learning to learn," aims to enable a model to adapt quickly to new tasks with minimal data. Unlike SSL, which works on a single large task, Meta-Learning trains across many small tasks—for instance, classifying new classes using only 1 or 5 labeled examples per class.

Self-Supervised Learning

Self-Supervised Learning removes external labels entirely. Instead, it creates pseudo-labels or proxy tasks (called pretext tasks) directly from the data. Examples include predicting missing image patches (in models like SimCLR), predicting the next word (BERT), or reconstructing masked tokens.

Inductive vs Transductive Learning in SSL

Inductive SSL

In Inductive Learning, the goal is to learn a general function $f(x)$ that can be applied to any new, unseen input. This approach trains the model on both labeled and unlabeled data, using various semi-supervised learning (SSL) techniques such as pseudo-labeling, consistency regularization, entropy minimization, and data augmentation.

The defining feature of inductive SSL methods is **generalization**. That is, after training, the model is not restricted to a fixed test set—it can be deployed to make predictions on future data drawn from the same (or even a shifted) distribution. This makes inductive SSL highly practical in dynamic or streaming environments where new data continuously arrives and retraining is costly.

For instance, consider an email spam filter: the model is trained on a mix of labeled (spam/not spam) and unlabeled emails, and once trained, it must classify entirely new emails it has never seen before. Inductive methods are suitable here because they aim to capture the underlying patterns in the data and encode this knowledge into a reusable decision function $f(x)$.

Some popular techniques in inductive SSL include:

- **Self-training**, where a model iteratively labels unlabeled data and retrains on its own predictions.
- **Consistency-based methods**, like the Pi-model or Mean Teacher, which enforce that the model outputs similar predictions under input perturbations or model ensembling.
- **Contrastive learning**, where unlabeled data is used to learn high-quality feature representations by contrasting positive and negative pairs, often improving generalization in low-label regimes.

Inductive learning is a common choice in domains such as natural language processing, computer vision, and recommender systems, where deployment on unseen data is a core requirement.

Transductive SSL

In Transductive Learning, the model's objective is more restricted: instead of learning a general function for all future inputs, it focuses on labeling only the specific unlabeled data that is available during training. This makes transductive SSL more of a one-time inference task, where the model adapts specifically to the structure of the given dataset and doesn't need to generalize beyond it.

One of the clearest examples of transductive SSL is **label propagation on graphs**. Here, both labeled and unlabeled instances are nodes in a graph, with edges representing similarities. The algorithm spreads label information from labeled nodes to their neighbors in a way that respects the graph structure. Since the graph includes all test data, the solution is optimized for that specific set of unlabeled nodes, rather than learning a general function applicable to arbitrary new nodes.

This paradigm is well-suited for scenarios such as:

- Text classification on a fixed corpus, where the goal is to assign categories to a known set of unlabeled documents.
- Node classification in citation or social networks, where the entire graph structure is known in advance.
- Medical datasets where the set of test samples is fixed due to privacy constraints or limited data availability.

Transductive methods often benefit from strong structural assumptions (like smoothness or cluster assumptions) and can achieve high accuracy by tailoring the learning process to the known test set. However, their lack of generalizability can be a drawback if the model needs to handle new data in the future.

Comparison Table

Feature	Inductive	Transductive
Goal	Learn a general classifier $f(x)$	Predict labels for a fixed unlabeled set
Output	Deployable model	Only predictions, no reusable model
Generalization	Works on unseen data	Cannot generalize to new data
Examples	Pseudo-labeling, MixMatch	Label propagation, TSVM
Advantage	Real-world deployment possible	High accuracy on current batch
Limitation	May sacrifice some accuracy	Not usable on future data

Ladder Networks and Π-Models (Pi-Models)

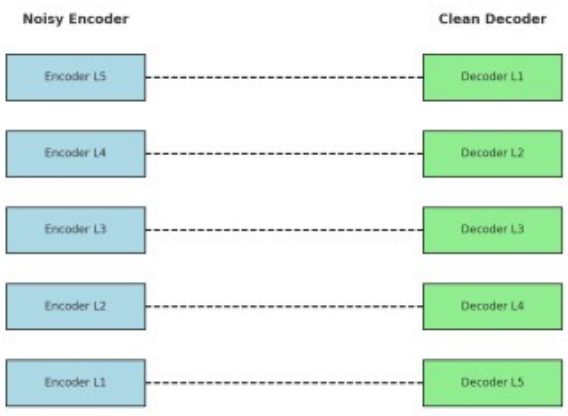
Ladder Networks

A Ladder Network is a deep neural network architecture that combines supervised and unsupervised learning by denoising internal representations. It consists of:

- A bottom-up encoder: adds noise to the input and passes it through the network
- A top-down decoder: reconstructs the clean version of each layer's activation
- Skip connections between encoder and decoder layers, forming a "ladder" shape

Each layer in the encoder learns noisy representations, while the decoder learns to recover the clean activations. Supervised loss is applied at the top (final output), and unsupervised loss is applied to each hidden layer. The total loss function is a combination of:

- Supervised loss (e.g., cross-entropy for classification)
- Layer-wise reconstruction loss, each weighted by a parameter λ_l



Π-Model (Pi-Model)

The Π-Model introduces consistency regularization. It is based on the idea that a model's prediction should remain stable under small perturbations of the input.

Mechanism:

- The same input x is passed twice through the same neural network with different augmentations or dropout noise.
- This results in two outputs: $f_1(x + \epsilon_1)$ and $f_2(x + \epsilon_2)$
- The model is trained to make these outputs match:

$$L_{unsup} = \|f_1(x) - f_2(x)\|^2$$

Variational Autoencoders (VAEs)

While traditional autoencoders learn to compress and reconstruct data, **Variational Autoencoders (VAEs)** bring a twist — they model the latent representation as a probability distribution.

The main idea is not just to reconstruct data, but to learn the *generative process* behind the data. This allows the model to generate new, similar samples by sampling from the learned distribution.

Intuition Behind VAEs

Imagine an encoder that compresses an input image (say, of a dog) into a summary z. The decoder then reconstructs the image from this summary. But unlike a regular autoencoder, VAEs treat z as a random variable — a point sampled from a normal distribution parameterized by a mean μ and variance σ^2.

This means the same image might map to slightly different z vectors each time, and the decoder can learn to reconstruct slightly different outputs from them. Over time, the model learns to generate diverse and realistic outputs.

VAE Loss Function

To train a VAE, the objective combines two components:

1. **Reconstruction Loss:** Measures how close the reconstructed output $\hat{x}$ is to the original input x. Common choices include mean squared error (MSE) or binary cross-entropy.
2. **KL Divergence:** Denoted $\mathrm{KL}(q(z|x) \parallel p(z))$, this term forces the learned latent distribution $q(z|x)$ to be close to a prior distribution, usually $p(z) = \mathcal{N}(0, I)$.

The total loss is:

$$\mathcal{L} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \mathrm{KL}(q(z|x) \parallel p(z))$$

Why Use a Distribution?

By modeling z as a distribution rather than a fixed vector, the decoder can sample an infinite number of latent codes and generate infinite variants of the data.

For example, once a VAE is trained on handwritten digits (MNIST), you can sample a $z \sim \mathcal{N}(0, 1)$ and decode it to produce a new digit image — even if it was not part of the original training data.

Diffusion Models: A Powerful Evolution

VAEs work in a single jump: $x \to z \to \hat{x}$. But this one-shot approach can struggle to capture complex details in high-quality data like realistic images.

Diffusion models solve this by taking a multistep approach. During training, noise is gradually added to the input data across many steps until the image becomes pure noise. Then, the model learns to reverse this noise — step-by-step — to reconstruct the original image.

This "denoising" process leads to much sharper, more detailed, and realistic outputs. Tools like **DALL·E 3** and **Stable Diffusion** are based on this concept.

Why Are They Called Denoising Models?

Because the model learns to recover a clean image from noisy inputs. The training phase involves adding noise, and the generation phase is the reverse: removing that noise gradually to form coherent outputs.

VAEs in Semi-Supervised Learning (SSL)

To apply VAEs in an SSL setting, we extend the VAE by adding a **classifier** to the architecture. Now, in addition to reconstructing the input, the model also predicts a class label y.

Architecture Overview

- The input x is passed through the encoder, yielding a latent variable z.
- z is used in two branches:
  - A decoder reconstructs x as $\hat{x}$.
  - A classifier predicts the label $\hat{y}$.

Training Strategy

For labeled data: Use all three components of loss:

1. Reconstruction loss ($x \to z \to \hat{x}$)
2. KL divergence ($z \sim \mathcal{N}(0, 1)$)
3. Classification loss ($z \to \hat{y}$)

For unlabeled data:

- Predict $\hat{y}$ from z.
- If confident, use $\hat{y}$ as a pseudo-label and compute classification loss.

Example: MNIST Digits

Assume you have:

- 500 labeled digit images (0–9)
- 50,000 unlabeled digit images

Train the VAE + classifier with labeled data. For unlabeled data, let the model assign a pseudo-label if it's confident. This helps the classifier generalize better and allows the decoder to generate new digits by sampling new z vectors.

Applications of VAE + SSL

- **Medical Imaging:** Train with a small labeled dataset and a large pool of unlabeled MRI or X-ray scans. Also useful for generating synthetic scans for augmentation.
- **Autonomous Driving:** Use unlabeled video frames to learn useful representations for pedestrians, traffic signs, etc.
- **Signature and Handwriting Analysis:** VAEs can learn from a few signature samples and generate realistic variations for verification systems.
- **Face Generation and Deepfakes:** Conditional VAEs can generate faces by age, gender, or expression, based on given labels.

Conditional VAEs: Controlled Generation

Regular VAEs sample z and generate data randomly. **Conditional VAEs (CVAEs)** extend this by also feeding in a label y into the decoder. This way, you can guide the generation process.

Example

To generate a digit "4":

- Sample $z \sim \mathcal{N}(0, 1)$
- Provide $y = 4$
- Pass $[z, y]$ into the decoder
- Output is a synthetic image of digit 4

This is particularly useful in *text-to-image generation*, *controlled synthesis*, and *attribute-guided sampling.*

Key Takeaways

- **Semi-Supervised Learning (SSL)** combines few labeled samples with many unlabeled ones to improve model performance.
- It is effective in reducing labeling effort while still achieving high accuracy.
- SSL is valuable in areas with expensive labeling like medical imaging, speech, and text processing.
- Assumes that data has structure: clusters, low-density regions, or low-dimensional manifolds.
- Popular approaches include self-training, co-training, generative models, clustering, and manifold learning.
- Advanced models like **Ladder Networks** and Π-Models use denoising and consistency to enhance learning.
- Related paradigms are Transfer Learning, Weak Supervision, PU Learning, Meta-Learning, and Self-Supervised Learning.
- SSL works in both **inductive** (generalizing to new data) and **transductive** (labeling current data only) settings.
- **VAEs (Variational Autoencoders)** combine reconstruction and generative modeling by learning probabilistic latent representations.
- In SSL, VAEs can be extended with a classifier, allowing the model to both reconstruct inputs and predict labels, even with limited supervision.
- **Unlabeled data** is leveraged by marginalizing over all possible labels, weighted by the model's predicted class probabilities.
- The **loss function** combines three parts: reconstruction loss, KL divergence, and classification loss — balancing generative and discriminative learning.
- **Conditional VAEs (CVAEs)** enhance control over generated outputs by incorporating label information directly into the decoder.