SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I2P122

Volume 13 Issue 2, 291-306, February 2026

Original Article

Robust Adaptive Learning for Real-World Data with Changing Features and Imperfect Labels

Akula Suneetha[1], K. Ratna Babu[2], Rama Vasantha Adiraju[3], V. V. Jaya Rama krishnaiah[4], Desamala Prabhakara Rao[5], Vullam Naga Gopi Raju[6]

[1]*Department of Computer Sciences and Engineering, KKR & KSR Institute of Technology and Sciences, Guntur, Andhra Pradesh, India.*
[2]*Computer Engineering, M.B.T.S. Government Polytechnic, Guntur, Andhra Pradesh, India.*
[3]*Department of Electronics and Communication Engineering, Aditya University, Surampalem, Kakainada, Andhra Pradesh, India.*
[4]*Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, A.P., India*
[5]*Department of Electronics and Communication Engineering, Chalapathi Institute of Technology, Mothadaka, Guntur, Andhra Pradesh, India.*
[6]*Department of Computer Science and Engineering, Chalapathi Institute of Technology, Mothadaka, Guntur, Andhra Pradesh, India.*

[1]*Corresponding Author : akulasuneetha25@gmail.com*

Received: 25 November 2025 Revised: 26 December 2025 Accepted: 28 January 2026 Published: 20 February 2026

Abstract *- In many machine learning tasks, environments are complex and unstable. Features in the data change over time. The labels in the data contain errors. These issues make model training very difficult. This paper focuses on a situation in which both problems happen at the same time. Learning in such a case is very challenging. It is even harder when the data in the new stage is limited. A new method termed ALDN is proposed. This stands for Adaptive Learning for Dynamic Features and Noisy Labels. The method has two main stages. The first stage learns a model from clean data. The second stage uses that model to help train a new model. The feature space has changed in this second stage. To do this, a tool titled optimal transport is used. This helps map the old model into the new feature space. Then, a special rule is applied to align the behavior of the old model with the new one. This rule is named a regularizer. It helps the new model learn better, even with noisy labels. Two versions of the method are presented. One uses a direct rule to keep the models close. The other uses an indirect rule that compares predictions. It is proven that the method works well in theory. The method is tested through many experiments. Comparisons are made with other identified methods. Better performance is shown in most cases. It works well even when the features change, and the labels are not reliable. This paper presents a useful way to deal with both dynamic features and noisy labels. It reuses past models and adapts them to new data. This helps when there is little clean data and the new data is messy. The method is practical and backed by strong theory and experiments.*

Keywords *- Learning, Adaptive, ALDN.*

1. Introduction

In the real world, many machine learning systems work in complex and changing environments [1]. These environments are not stable. Data collected over time looks different. The features change. The quality of labels gets worse [2]. These problems create new challenges for learning. The main issue becomes more serious when we cannot store all past data. In many systems, there are limits on storage and computing power. So, only keep the model trained earlier and some newly collected data. When new data is collected, its feature space do not match the feature space of the past data. Some features disappear, and new ones appear [3]. This mismatch causes errors. Due to the change in the data collection process, the labels lose accuracy. This leads to noisy labels. This paper addresses a key problem involving simultaneous changes in both features and labels. The difficulty of this problem comes from the fact that three important parts of data change at the same time [4]. These are the volume of the data, the feature space, and the label correctness. When these parts change together, traditional learning methods no longer work well. There are three main challenges in this setting. First, it is not realistic to collect and store all data before training. Second, using only the shared features between past and current data is not enough. Ignoring vanished and new features leads to a loss of important information. Third, noisy labels make it hard to train an

accurate model. It hurts the prediction performance and reliability of the learned model. Some existing methods try to handle changing feature spaces [5]. Others try to deal with noisy labels. The method is provided that works well in theory and in practice, even when the feature space changes and labels are noisy.

To solve this problem, introduce a new approach termed ALDN. The method works in two stages. The first stage is titled the Previous stage or P-stage. In this stage, a model is trained using clean and full-featured data. This model is saved for future use. The second stage is named the Current stage or C-stage. In this stage, the feature space has changed. Some features disappeared, and some new features appeared [6]. The data collected in this stage contains noisy labels. To deal with the change in feature space, a technique termed optimal transport is used. This helps in mapping the learned model from the P-stage to the new feature space of the C-stage [7]. This mapped model is used as a starting point for learning a new model in the C-stage. It is referred to as the prior model. To make the learning process stable, a regularizer is added. This regularizer makes sure that the new model is not too different from the prior model. It acts as a stability constraint. This helps to reduce the negative effects of noisy labels and changing features [8].

Two different implementations of this method are presented. The first version uses a direct steadiness constraint. This version is named ALDN-D. It directly forces the new model to stay close to the prior model. The second version is termed ALDN-ID. It uses an indirect regularization constraint. It compares the predictions of the new model and the prior model [9]. This way, the model is allowed to differ in some parts but still remain steady in general. Finally, the approach is tested on various datasets. These include synthetic datasets and real-world data, like activity recognition tasks. The results show that the method performs better than existing methods. It works well even when the data is limited, the features changed, and the labels are noisy [10]. The Key Contributions of this paper are:

- The paper introduces a new and practical learning problem focused on data with dynamic features and noisy labels, an area that has not been widely studied.
- A novel method named ALDN is proposed, which reuses past models and applies a regularizer to maintain consistency during training, supported by theoretical guarantees.
- Experimental validation demonstrates that the method outperforms others across multiple settings, including real-world scenarios.

This work is useful for many practical applications. These include systems that deal with wearable sensors, like health monitoring or human activity recognition [11]. It is useful for systems that monitor the environment, for example, those used during natural disasters. In these situations, it is usually hard to collect enough clean and reliable data. Models trained earlier become valuable and are reused using the approach proposed in this paper. The ALDN method does not require storing all previous data [12]. It only needs the model trained in the P-stage. Changing features and noisy labels make it difficult to learn a good model. But the proposed method handles both of these problems. It works even with a small amount of labelled data. This makes the solution very practical and useful [13]. The use of optimal transport and reliability regularization makes the model both stable and adaptive.

2. Related Work

This section reviews the two major challenges in the paper learning with dynamic features and learning with noisy labels. These are common in real-world settings, but few methods handle both together. Many real applications deal with changing features. In streaming settings, features appear or disappear over time. Earlier works, such as [14, 15], focus on feature selection when features arrive one by one. In these methods, the number of samples remains fixed, but the features stream continuously. More complex settings allow both the data and feature space to change. Zhang [16] introduced the clue of a trapezoidal data stream, as new features keep adding over time. Hou et al. [17] proposed OPID, which handles overlapping features between different stages. Zhang and Zhou [18] proposed FESL, which assumes a short overlap between old and new features. C. Hou [19] extended this by working with overlapping features during feature addition. Some studies consider fully dynamic streams. In these, features appear and disappear randomly. Examples include the works of B J Hou [20], C.Hou and C. Xu [21], and Koby [22]. More related methods are like [16], which modifies the Passive-Aggressive algorithm to fit the trapezoidal setting. Qian [21] extends OLSF to support weak supervision. The original PA algorithm was introduced by Tongliang [23], and it tolerates some label noise.

All these methods mostly ignore label noise. The solid theoretical support is lacking. That limits the use in more complex real-world scenarios. Label noise happens when the data labels are wrong. This is common in real data. Some methods use heuristics to manage noisy labels. Examples include the work of Benoit [24], J Zhou [25], B Han [26], Jiaheng [27], and Bo Han et al. [28]. The Passive-Aggressive algorithm [23] is used in this setting. Most heuristic methods do not come with theoretical guarantees. B Frenay discussed this gap in [29]. Other methods use statistical models. A common tool is the noise transition matrix. Giorgio et al. [30] propose a loss correction method using this matrix. Curtis et al. [31] improve label certainty using confident learning. Theoretical guarantees are provided in works like Natarajan et al. [32] and H chen et al. [33]. To use the transition matrix, some methods assume the presence of anchor points. These are samples that clearly belong to a single class. Fernay [29], Han [26], and Patrini [30] used this approach to adjust the loss and correct noise during training.

These methods work well, but only when the feature space stays constant. Some recent works focus on feature-dependent label noise. Here, the label noise depends on the input features. Xia et al. [34], Berthon et al. [35], and B. Han et al. [36] worked in this direction. But these methods are more complex and require additional assumptions. Other approaches consider weak supervision. Jacob [37] offers a review of such methods. Deep learning tries to manage noisy labels. Goldberger and Yuncheng Li et al. [38] added a noise adaptation layer. Yuncheng Li et al. [38] used a distillation technique to clean noisy data. Deep learning models usually need large, clean datasets. Most of the previous work looks at either dynamic features or noisy labels. Few works handle both. The method in this paper solves both problems together and provides theory and results to support it.

3. Proposed Approach

In many real-world learning tasks, data evolves over time. For instance, the features available in an earlier stage of data collection might not remain the same later. Some features vanish completely, while new ones appear. This situation becomes even more difficult when the labels provided for newly collected data are noisy-that is, some of them are incorrect. To address this dual challenge, the paper proposes a two-stage learning method named ALDN.

The ALDN method helps train machine learning models when both the features and labels of data change or degrade across different time periods. The process is divided into two main stages. The first is named the P-stage or the previous stage. In this stage, clean data with a stable set of features and accurate labels is assumed. Using this clean data, train a reliable model. The second is the C-stage or the current stage. In this stage, the feature space has changed, and the data available is both limited in size and corrupted by noisy labels.

Since the model learned from the P-stage is trained on different features than those in the C-stage, it cannot be directly reused. To solve this issue, the ALDN approach maps the old model into the new feature space using a carefully designed transformation. Once mapped, the prior model is used as a guide for learning the new model in the C-stage. To describe the method more precisely, introduce specific notations for both stages.

In the P-stage, the input data is represented by a matrix. $X_P \in R^{n_1 X d_1}$. Here, n_1 is the number of samples and d_1 is the total number of features. The corresponding labels are stored in a matrix $Y_P \in R^{n_1 X C}$. Here, C is the number of classes. These labels are assumed to be clean and accurate. The feature matrix X_p is divided into two parts: X_p^v.

It contains features that will vanish in the next stage and X_p. It includes the features that survive. Therefore, take $X_P = \left[X_P^V, X_p^s\right] A = \pi r^2 A = \pi r^2 X_p = [X_P^V \quad A = \pi r^2$ and the total number of features is. $d_1 = d_v + d_s$ Here is the number of vanished features and d_s is the number of survived features.

In the C-stage, the data is represented by a matrix. $X \in R^{n_2 \times d_2}\, A=\pi r^2 A = \pi r^2 A = \pi r^2$, here $d_2 = d_s + d_n$. This reflects that some features from the P-stage survived. d_s and new features d_n appeared. The matrix X is similarly divided as X =$[X^s, X^n]$, here X^s contains the survived features and X^n contains the new ones. The labels in this stage are noisy and stored in Y $\in R^{n_2 \times C}$. The true labels Y $\in R^{n_2 \times C}$. are not accessible during training. To model the classification problem, define two sets of classifier parameters. The model trained during the P-stage is represented by$W_p \in R^{d_1 \times C}$. It is splited as $W_P = \begin{bmatrix} W_p^v \\ W_p^s \end{bmatrix}$

Here, the components correspond to the vanished and survived features, respectively. In the C-stage, the classifier that the learner wants to learn is represented by $W_p \in R^{d_2 \times C}$ and is split similarly to $W_P = \begin{bmatrix} W^s \\ W^n \end{bmatrix}$, representing weights for the survived and new features. The purpose of ALDN is to use the well-trained model, W_P from the P-stage to help train a new model W for the C-stage. Despite the difference in feature space and the presence of label noise. The prior model, denoted as W_o is expected to be close to the final model W. The closeness is enforced through a regularization term added to the loss function during training.

To make the model trained in the P-stage useful for the C-stage, first deal with a key challenge: the feature space has changed. The classifier W_P was trained on the feature space of dimension $d_1 = d_v + d_s$, but now the new data in the C-stage lies in a different feature space of dimension $d_2 = (d_s + d_n)$. Since some features vanished and new ones appeared, the direct application of the old model W_P. To the new data is not possible. To solve this, a transformation is proposed that maps the model W_P into the new feature space of the C-stage. The approach is to learn a mapping matrix T $\in R^{d_n \times d_v}$ that transfers the weights associated with vanished features into the space of new features. Once this mapping is learned, the prior model $W_0 \in R^{(d_s + d_n) \times c}$ for the C-stage. It is constructed by concatenating the weights for the survived features and the mapped weights for the new ones. Mathematically, the prior model is defined as:

$$W_0 = \begin{bmatrix} W_p^s \\ d_s \times T W_p^v \end{bmatrix} \tag{1}$$

Here, the scalar d_s is used to scale the mapping. TW_p^v represents the transformation of the vanished weights into the new feature space. This problem is approached by using the concept of Optimal Transport (OT). OT is a powerful mathematical tool that finds the most efficient way to convert

one distribution into another. In this case, it is used to connect the vanished features in the P-stage with the new features in the C-stage. To apply OT, treat the vanished features X_p^v and the new features X^n as two separate distributions. Let us denote $\mu_n \epsilon \Delta^{d_v}$ as the Marginal distribution over vanished features. $\mu_n \epsilon \Delta^{d_n}$ as the Marginal distribution over new features, both distributions are set as uniform to treat each feature equally. So, take $\mu_v = \frac{1}{d_v}1$ and $\mu_n \frac{1}{d_v}1$, here 1 is a vector of all ones. Next, define a cost matrix G ∈ $R^{d_n \times d_v}$. Each element G^{ij} in this matrix represents the cost of transforming the j-th vanished feature into the i-th new feature. The objective is to find a transport plan T that minimizes the total cost:

$$min_T \langle T, G \rangle \; subject\ to\ T1 = \mu_n, T \cdot 1 = \mu_V, T \geq 0 \qquad (2)$$

This is a standard optimal transport problem. The result is a matrix T that shows the way vanished features are mapped to new ones. To calculate the cost matrix G, use the survived features as a bridge. Since survived features X_p^s and X^s, they are available in both stages and share the same meaning. Use them to reconstruct both vanished and new features. Specifically, solve two sparse reconstruction problems:$A = \pi r^2$

$$min_{U_p} \left\| X_p^v - X_p^s U_P \right\|_F^2 + \lambda \sum_{i=1}^{d_v} \left\| (U_p) \right\| \qquad (3)$$

$$min_u \| X^n - X^s U \|_F^2 + \lambda \sum_{j=1}^{d_n} \left\| (U_p) \right\| \qquad (4)$$

Here, $U_p \in R^{d_s \times d_v}$ and $U \in R^{d_s \times d_n}$ are the coefficient matrices are used to reconstruct the vanished and new features from the survived features. The ℓ_0 Pseudo-norm encourages sparsity in the reconstruction. Once it is obtained U_p, and U, compute the cost matrix G as the pairwise Euclidean distances between the columns of U_p and U. This reflects the difference or similarity between vanished and new features in terms of relationships with survived features. Finally, using the cost matrix G, solve the optimal transport problem to get the mapping matrix T. This matrix is then used to transform the model W_P into the prior model W_o for the C-stage. A major benefit of this process is that they do not need to store the original data X_p from the P-stage. Instead, only need to store W_P and the reconstruction coefficients U_p. It saves a lot of memory and makes the method more practical. This mapped model W_o plays a central role in the rest of the training process. It is expressly in helping to learn a good model in the C-stage despite the presence of label noise and limited data.

In the C-stage, the available labels are noisy. For many samples, the observed label does not represent the true class. To address this, introduce a technique to estimate the underlying noise in the labels. This is done using a noise transition matrix. The noise transition matrix M ∈ $R^{c \times c}$ indicates the probability of one class label flipping into another. Formally, each element $M_{i,j}$ represents the probability that a sample from true class i is labelled incorrectly as class.

$$M_{i,j} = \Pr[y = e_j, x = Q_i] \text{s j} \qquad (5)$$

Here, y is the observed noisy label, and y is the true label. The problem is that M is not given in advance. Estimate it if it is assumed that for each class, there exists at least one anchor point. An anchor point is a sample that belongs to one class with absolute certainty. In mathematical terms, if a sample x_i satisfies:

$$M_{i,j} = \Pr[y = e_c, x = Q_i] = 1 \qquad (6)$$

Then x_i is an anchor point for class C. Using this assumption, estimate the row of the transition matrix corresponding to class C by observing the model's prediction on the anchor point:

$$M_{c,j} \approx \Pr[y = e_j, x = Q_i] = 1 \qquad (7)$$

In practice, finding anchor points using only the noisy and limited data of the C-stage is unreliable. Therefore, the proposal to use the prior model W_0 to guide the search for anchor points. First, the model W_0 is used to train a new model W using the noisy data (X, Y), with an added regularizer to keep $\dot{W}$ close to W_0:

$$min -_w \frac{1}{n_2} \sum l\,(f(W, x_i), y_i) + \lambda C(W, W_0) \qquad (8)$$

Here, ℓ is the loss function (like cross-entropy) and C(W, W_0)is a constant constraint, either direct or indirect. After training W, use it to estimate the anchor point for each class i by selecting the sample $x \in X$ that gives the highest predicted probability for class i:

$$x_\iota = argmax_{x \in X} \Pr(\ddot{y} = e_\iota, x) \qquad (9)$$

The estimated row of the transition matrix is simply the predicted label distribution at this point:

$$M_{i,j} = \Pr[y = e_j, x = Q_i] \qquad (10)$$

This process gives us an estimated noise transition matrix M, which is then used to correct the loss during model training. The entire process makes use of the prior knowledge encoded in W_0 to improve both the training and the label correction in the noisy C-stage environment. Without W_0, it would be much harder to find good anchor points due to the small amount of noisy data. Once the prior model W_0. is available, and the noise transition matrix M is estimated. The next step is to train a new classifier W using the noisy data in

the C-stage. Because the labels in this stage are not clean, directly applying a traditional loss function, such as cross-entropy, leads to poor results. ALDN-D uses a direct constraint based on the difference between model weights. ALDN-ID uses an indirect constraint based on the model predictions. A corrected version of the loss function is used based on the inverse of the noise transition matrix M. This surrogate loss estimates the expected loss as if clean labels were available. Given the predicted probability vector $y_i = f(W, x_i)$ for sample x_i and the observed label y_i, the surrogate loss is defined as:

$$l(f(W, x_i), y_i; M) = \sum_{c=1}^{c} l(f(W, x_i), y = e_c)\, X\, [M^{-1}]_{y_i, e} \quad (11)$$

This loss reweights each class based on the probability that the observed label originated from that class under the noise model. In this version, the reliability constraint directly penalizes the distance between the new model W and the prior model W_0. The loss function is:

$$L_{ALDN-D}(w) = \frac{1}{n_2} \sum l(f(W, x_i), y_i; M) + \lambda\, ||W - W_0||_F^2 \quad (12)$$

This formulation confirms that W stays close to W_0 in the parameter space. It is simple and efficient to compute. It assumes that the structure of W_0 remains highly relevant. It does not always be true when the data has changed greatly. Here, instead of directly comparing the weight matrices, the predictions of the two models are compared. This indicates that agreement is enforced at the output level. The loss function is:

$$L_{ALDN-ID}(w) = \frac{1}{n_2} \sum l(f(W, x_i), y_i; M) + \lambda\, ||XW - XW_0||_{2,1} \quad (13)$$

The $l_{2,1}$-norm first computes the l_2-norm of each row, then sum them using the l_1-norm. This promotes sparsity in the difference, meaning most predictions of W and W_0. are kept similar, but a few are allowed to differ. This method is more flexible than ALDN-D. It is useful when the new data distribution is not exactly aligned with the old one. It allows the new model to deviate when necessary. To train the model in the C-stage, the objective functions are defined that are minimized.

Both formulations involve two terms. A noise-corrected loss using the surrogate function ℓ, which depends on the estimated noise transition matrix M. A constancy constraint that helps reuse the prior model W_0. Because these objectives are differentiable, efficient optimization is possible using the gradient descent algorithm. Let the total objective be:

$$J(W) = \sum_{i=1}^{n_2} l(f(W, x_i), y_i; M) + \lambda\, X\, C(W, W_0) \quad (14)$$

Here, $C(W, W_0)$is the reliability constraint, either direct or indirect, depending on whether we use ALDN-D or ALDN-ID. Minimize this function iteratively:

$$W^{(t+1)} = W^{(t)} - \alpha\, X\, \nabla J(W^{(t)}) \quad (15)$$

Here, α is the learning rate and ∇J(W) is the gradient of the objective with respect to the weights W. Let us denote the prediction vector as:

$$y_i = softmax(x_i W)\ here\ y_{ij} = \frac{e^{x_i w_j}}{\sum_{k=1}^{c} e^{x_i w_k}} \quad (16)$$

The cross-entropy loss corrected using matrix M becomes:

$$l(f(W, x_i), y_i; M) = \sum_{c=1}^{C} \log(y_{ic})\, X [M^{-1}]_{y_i, c} \quad (17)$$

Let us define:

$$J_1(W) = \frac{1}{n_2} \sum_{i=1}^{n_2} l(f(W, x_i), y_i; M) \quad (18)$$

The gradient of $J_1(W)$involves computing the derivatives of the softmax and cross-entropy functions. Since this is standard in classification tasks, follow the procedure used in modern deep learning:

$$\frac{\partial J_1(W)}{\partial W} = X \cdot \times (\hat{Y} M^{-1} - Y^{clean}) \quad (19)$$

Here, $\hat{Y} \in R^{n_2 \times C}$ is the matrix of softmax predictions. Y^{clean} represents the clean label distribution reconstructed via anchor points and matrix inversion. For ALDN-D (Direct Constraint):

$$C(W, W_0) = \|W - W_0\|_F^2 \Rightarrow \nabla C(W, W_0) = 2(W - W_0) \quad (20)$$

So, the total gradient becomes:

$$\nabla J(W) = \nabla J_1(W) + 2\lambda(W - W_0) \quad (21)$$

For ALDN-ID (Indirect Constraint), let Z = XW and $Z_0 = XW_0$. The constraint is:

$$C(W, W_0) = ||Z - Z_0||_{2,1} \quad (22)$$

The gradient of the $l_{2,1}$-norm is computed using subgradients:

$$\frac{\partial \|Z - Z_0\|_{2,1}}{\partial Z} = D, here\ D_{i,:} = \frac{Z_{i,:} - Z_{0,i,:}}{\|Z_{i,:} - Z_{0,i,:}\|_2 + \dot{o}} \quad (23)$$

Here, $D \in R^{n_2 \times C}$ is a matrix of row-normalized differences. $\dot{o}$ is a small constant to avoid division by zero.

Then, the gradient with respect to W is:

$$\nabla C(W, W_0) = X \cdot D \tag{24}$$

So, the full gradient becomes:

$$\nabla J(W) = \nabla J_1(W) + \lambda XD \tag{25}$$

Once the gradient is calculated, update the weights using the following rule:

$$W^{(t+1)} = W^{(t)} - \alpha \times \nabla J(W^{(t)}) \tag{26}$$

This process is repeated until convergence, typically when the change in loss is small or the gradient becomes close to zero. Both implementations of ALDN are optimized efficiently using the gradient descent method. The use of the prior model provides a strong starting point that improves robustness against label noise and feature drift. Let us revisit the noise transition matrix M, as given in Equation (5). This formulation assumes that noise only depends on the clean label y. But in the feature-dependent case, revise the transition as:

$$M_{i,j}(x) = P\tilde{r}[y = e_j \; y = e_i, x] \tag{27}$$

The probability of a class being corrupted into another depends on the input x itself. In such cases, a fixed noise matrix is no longer sufficient. Instead, it needs to estimate the way noise behaves across different regions of the feature space.

The ALDN framework is extended to handle this case by using a local manifold-based assumption. That is, nearby samples in the feature space should exhibit similar noise behaviour. To capture this, define a sample-wise noise transition matrix $M^{(i)}$ for each sample x_i. To make the learning feasible and robust, the paper applies manifold regularization on the estimated noise matrices. Define a graph over the input samples where the weight between sample i and j is given by:

$$A_{i,j} = \exp[-\frac{\|x_i - x_j\|^2}{\sigma^2}] \tag{28}$$

This graph reflects the closeness of the samples in feature space. Enforce that the estimated noise transition matrices for nearby samples should be similar. Specifically, add a penalty term:

$$\sum_{i,j} A_{i,j} \times \left\|M^{(i)} - M^{(j)}\right\|_F^2 \tag{29}$$

This term makes certain that samples close in the feature space are nearly identical transition matrices, promoting smoothness and structure. The extended loss function becomes:

$$L_{feat-noise}\left(W, \{M^{(i)}\}\right) = \frac{1}{n_2}\sum_{i=1}^{n_2} \ell(f(W, x_i)^{\sim}, y_i; M^{(i)}) + \lambda \times C(W, W_0) + \gamma \times \sum_{i,j} A_{i,j} \times \left\|M^{(i)} - M^{(j)}\right\|_F^2 \tag{30}$$

Here, the first term is the loss corrected by the sample-wise transition matrices. The second term confirms alignment with the prior model. The third term is the manifold regularizer for smoothness across the noise models. λ and γ are hyperparameters controlling the balance between these objectives. The optimization now involves learning both the model weights W and the sample-wise matrices $\{M^i\}$. Each update step is computed efficiently. For the transition matrices, a closed-form or projected gradient update is applied under the simplex constraint. Ambiguous text data leads to varying labels. Sensor readings near thresholds are misclassified more frequently. Such flexibility helps the model become more robust and improves generalization under difficult and dynamic conditions. It is clear that ALDN-D is computationally cheaper and easier to implement, but ALDN-ID provides better adaptability to environments. Here, the relationship between inputs and outputs shifts. Practitioners choose the appropriate variant depending on the level of change and noise in the data. The comparative analysis between ALDN-D and ALDN-ID helps in choosing the implementation that best fits a given application or dataset. These resources are mainly valuable for implementation, research replication, or algorithmic extension. This makes it easier to communicate ALDN's core concepts in presentations or applied machine learning contexts.

4. Theoretical Analysis

Theoretical Analysis: The purpose is to understand in what way a model trained on noisy data manages to generalize. Upper bounds are given on the expected error of the model trained in the current stage. The analysis focuses on the learning objective with the noise-corrected loss and a regularization term. Two types of regularization are studied: one that directly constrains the weights (ALDN-D) and another that constrains the predictions (ALDN-ID). The purpose is to show that even when labels are noisy and features are changed. The model trained using the ALDN framework still performs well if it is close to the clean prior model and the noise is not too bad. Before proving the main theorems, define important concepts used in the analysis. Let X $\subset$ R^d be the input space and Y = {1,2,,,,,C} be the label space. Let S = $\{(X_i, X_i)\}_{i=1}^{n}$ be the training set with noisy labels. D be the underlying true distribution generating clean samples (x, y). Let W ϵ $R^{d\times c}$ be a linear model. Let f(W , x) ϵ R^c be the predicted scores for all classes. The predicted class is:

$$\hat{y} = argmax_{j\in[c]} f(W, X)_j \tag{31}$$

The clean risk of a model W is defined as the expected loss on clean data:

$$R(W) = E_{(X,Y)\sim D}[\ell(f(W,X),Y)] \quad (32)$$

The empirical surrogate risk with corrected loss is:

$$R(W) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(W,X_i),\sim y_i;M) \quad (33)$$

Here, ℓ is the noise-corrected loss using transition matrix M and y_i is the noisy label. The purpose is to bound $R(\widehat{W}) - R(W^*)$, the excess risk. Here, $\widehat{W}$ is the model learned from noisy data using ALDN. W^* is the optimal model trained with clean labels. In ALDN, train a model $\widehat{W}$, by minimizing a noise-corrected loss along with a regularization term that links it to the previous model:

$$\widehat{W} = argmin_W[\frac{1}{n}\sum_{i=1}^{n} \ell(f(W,x_i)^{\sim},y_i;M) + \lambda \times Consistency(W,W_0)] \quad (34)$$

The reliability term encourages W to stay close to the prior modelW_0. It is mapped from the clean P-stage model. The assumption is that W_0 contains useful knowledge and helps guide learning when the labels in the current stage are noisy. To analyze the generalization error of a learning model, use a mathematical tool named Rademacher complexity. This helps us measure the extent to which a model trained on a sample will perform on unseen data. In this lemma, we introduce the vector contraction inequality. It is used when the loss function outputs a vector instead of a scalar. Let Φ: $R^C \longrightarrow R$ is a Lipschitz function, as it does not change too fast. Suppose Φ has Lipschitz constant L, then for any class of vector-valued functions $F \subseteq R^C$:

$$E[sup_{f\in F}\sum_{i=1}^{n} \sigma_i\Phi(f(x_i))] \leq \sqrt{2}L \times E[sup_{f\in F}\sum_{i=1}^{n}\sum_{j=1}^{c}\sigma_{i,j}f_j(x_i)] \quad (35)$$

Here, σ_i, $\sigma_{i,j}$ are Rademacher variables with equal probability. The right-hand side involves summing over the individual components of the vector. This inequality helps us convert the complexity of a vector-valued function into the sum of scalar components, which is easier to analyze. This lemma gives a bound on the possible difference between the model's empirical loss and its expected loss across all models in the hypothesis space. Let w be a class of weight matrices such that. $\|W - W_0\|_F \leq \rho$. All models in this class are within distance ρ from the prior model W_0 in Frobenius norm. Suppose the loss function $\ell(f(W,X),Y)$ is L-Lipschitz with respect to f(W, x), and predictions are computed using softmax. Then, with high probability for all W ϵ W:

$$R(W) \leq R(W) + 2L\rho\sqrt{\frac{2Clog(2d)}{n}} + 3\sqrt{\frac{\log(^2/_\delta)}{2n}} \quad (36)$$

The theorem gives an upper bound on the expected clean risk of the model learned by ALDN-D, which uses a direct constraint on the model weights. Let $\widehat{W}$ be the model trained by minimizing:

$$\widehat{W} = argmin_W[R(W) + \lambda\|W - W_0\|_F^2 \quad (37)$$

Let W^* be the ideal model that minimizes clean risk:

$$W^* = argmin_W R(W) \quad (38)$$

Then, with high probability, bound the clean risk of $\widehat{W}$ as:

$$(\widehat{W}) \leq R(W^*) + 4L\rho\sqrt{\frac{2Clog(2d)}{n}} + 6\sqrt{\frac{\log(^2/_\delta)}{2n}} + 2\lambda\rho^2 \quad (39)$$

The first term $R(W^*)$ is the best possible clean error. The second term depends on how close the model is to W_0 and on model complexity. The third term reflects uncertainty due to finite data. The fourth term is a bias introduced by the regularization. This theorem shows that as long as the model $\widehat{W}$ stays close to the prior W_0. It has enough training data; the clean risk is tightly controlled even when it is training on noisy labels. In the ALDN-ID method, the model does not directly constrain the weights. Instead, it focuses on constraining the predictions made by the model. This is done to provide more flexibility, especially when the feature space has changed a lot from the P-stage to the C-stage. The key concept is to keep the new model's predictions close to the prior model's predictions. The constant term used in this method is $\|XW - XW_0\|_{2,1}$. It measures the difference in output scores across all training samples. Here, X is the input data matrix, and W is the model being learned. W_0 is the prior model derived from the P-stage. The objective function used in ALDN-ID is:

$$\widehat{W} = argmin_W[R(W) + \lambda\|XW - XW_0\|_{2,1}] \quad (40)$$

This function combines the corrected empirical risk with a regularization term that penalizes prediction differences from the prior model. To analyze the model's generalization, define a constraint set with the prediction difference from W_0 is at most ρ'. This set is written as:

$$B = \{W \in R^{d\times c}: \|XW - XW_0\|_{2,1} \leq \rho' \quad (41)$$

Assuming the loss function is smooth, and predictions are bounded. Derive a risk bound for the learned model $\widehat{W}$. With high probability, the clean risk is bounded as follows:

$$R(\widehat{W}) \leq R(W^*) + 4L\rho'\sqrt{\frac{2\log(2d)}{n}} + 6\sqrt{\frac{\log(^2/_\delta)}{2n}} + 2\lambda\rho' \quad (42)$$

Here, W^* is the optimal model that minimizes the clean risk. L is the Lipschitz constant of the loss function. The term ρ' measures the extent to which the current model's predictions deviate from the prior model. The flexibility of ALDN-ID makes it expressly useful when the feature space has changed, and it is harder to directly reuse weights from the prior model. ALDN-D enforces a direct constraint on the

weights. It tries to keep the learned model W very close to the prior model W_0 in the parameter space. This approach is simple and effective when the feature spaces between the P-stage and C-stage are very similar.

It results in lower computational costs because it uses a standard Frobenius norm. On the other hand, ALDN-ID uses an indirect constraint by matching the model outputs instead of the weights. The regularization term $\|XW - XW_0\|_{2,1}$ allows for more flexibility because it only requires the predictions to be close, not the parameters. This is expressly helpful when new features are introduced or when some P-stage features are missing. Both methods follow a general structure in the theoretical risk bounds:

$$\text{Clean Risk} \leq \text{Optimal Risk} + \text{Complexity Term} + \text{Regularization Bias} \quad (43)$$

In this structure, the optimal risk is the best possible performance on clean data. The confidence term decreases as more data is collected. The regularization bias increases when it pushes the model too close to the prior. ALDN-D is better when the data distribution remains stable, and the features are mostly unchanged. It is simpler and faster. ALDN-ID is more appropriate when large changes are expected in the feature space or prediction behavior. It is more adaptive but requires more computation and tuning. Both methods provide solid theoretical guarantees.

5. Experiments

This section evaluates the performance of the proposed methods, ALDN-D and ALDN-ID, through a wide range of experiments. Both synthetic datasets and real-world datasets are used to demonstrate the method's robustness and adaptability. Sixteen datasets are collected in total to prepare the data for experiments. Fifteen datasets come from public sources, for instance, UCI and LIBSVM, and one real dataset, named RealDisp, is used for a practical task. The features of each dataset are split into three types: vanished features make up 45%, survived features account for 10% and new features take the remaining 45%. As a result, only a small portion of the features is shared between the two stages. This makes the problem more challenging and realistic.

Compare this approach to a deep neural network-based method named FCR. This model requires access to both P-stage and C-stage data and assumes enough training samples. Due to the small size of the C-stage data and large feature changes, the deep model does not perform well. ALDN-D and ALDN-ID maintain strong performance, demonstrating robustness in low-data environments. A feature-dependent noise version of the model, termed FDALDN-D and FDALDN-ID, is introduced. These are tested against an identified method named MEIDTM and against the original ALDN models.

The results show that FDALDN-D and FDALDN-ID outperform MEIDTM in all six test datasets. Improvements are made upon the base ALDN methods in most cases. When the C-stage data is extremely limited, the extended models sometimes underperform due to the extra complexity.

Finally, test the models in a real-world activity recognition task using the RealDisp dataset. This task involves sensors placed on different body parts. In practice, sensors move slightly or become unavailable, which changes the feature space. Label noise is common due to errors in human action labeling. Simulate this situation by using four sensors as vanished and new features, and one sensor as the surviving feature set.

Table 1. Comparison of model capabilities in experiments

Method	Uses P-stage?	Handles Feature Change?	Handles Noise?
DLCEL-C	No	No	Basic
LSL-C	No	No	Yes
Cleanlab	No	No	Yes
SURF-ID / ID	No	Partially (only survived)	No
MF-SL	Yes	Yes (via completion)	Yes
CDSPP	Yes	Yes	Yes
ALDN-ID / ID	No	Yes(model reuse)	Yes
FDALDN-ID / ID	No	Yes	Yes

In this task, ALDN-D and ALDN-ID once again outperform other methods. Higher accuracy is achieved. The label noise rate is estimated close to the true noise. This shows the methods are accurate. Noisy data is handled well. Adaptation to changing data in dynamic settings takes place. Table 1 summarizes the properties of each model tested in the experiments. It shows whether the model uses P-stage data, whether it handles dynamic features, and whether it accounts for label noise.

From this table, ALDN-D and ALDN-ID are unique in handling both feature changes and label noise, without needing access to P-stage data during testing. FDALDN-D and FDALDN-ID improve this by modelling more complex, feature-dependent noise. Methods like SURF and Cleanlab lack either dynamic feature handling or model reuse. It explains the lower performance. Table 2 presents accuracy results from the real-world task using the RealDisp dataset. It includes the way each method estimated the actual noise level in the labels.

Table 2. Real-World activity recognition results under 30% label noise

Method	Accuracy (Mean)	Estimated Noise Rate
ALDN-D	83.2%	24%
ALDN-ID	84.5%	29%
SURF-D	74.6%	--
Cleanlab	75.1%	--
MEIDTM	72.3%	--
FDALDN-D	85.1%	27%
FDALDN-ID	85.3%	28%

From this table, it is clear that the FDALDN versions of the model achieve the highest accuracy. Estimate the true noise rate (30%) very closely, demonstrating the effectiveness of modelling feature-dependent noise. The basic models like SURF-D and Cleanlab are not able to estimate noise and perform worse in accuracy. Figure 1 compares the accuracy of five different models across five datasets. For the WISDM dataset, ALDN-ID reaches the highest accuracy of around 87%. At the same time, OPID records the lowest at approximately 74.5%.

ALDN-D and Co-teaching perform well with values close to 85.5% and 80.5%, respectively. In the PAMAP2 dataset, both ALDN-D and ALDN-ID score nearly 90% with ALDN-ID slightly ahead. Co-teaching performs moderately at around 85.5%. While FESL and OPID trail at 83% and 81.5%, respectively. For the HAR dataset, ALDN-ID achieves the highest performance at nearly 93%. ALDN-D performs strongly at about 91%. Co-teaching, FESL, and OPID follow with approximate values of 88%, 86% and 85%. In the UCI1 dataset, the accuracy drops across all models. ALDN-ID again leads with about 81%, followed by ALDN-D at 79%. Co-teaching, FESL, and OPID lag behind with accuracies of approximately 75%, 73% and 71.5%, respectively. The UCI2 dataset shows ALDN-ID as the best performer with an accuracy of around 85% followed closely by ALDN-D at 84%. Co-teaching maintains a moderate level at 80%. FESL and OPID record lower values at 78.5% and 76%. ALDN-ID reliably outperforms the other methods across all datasets. OPID shows the lowest performance. The gap in accuracy between the best and worst models is most noticeable in the WISDM and UCI1 datasets.

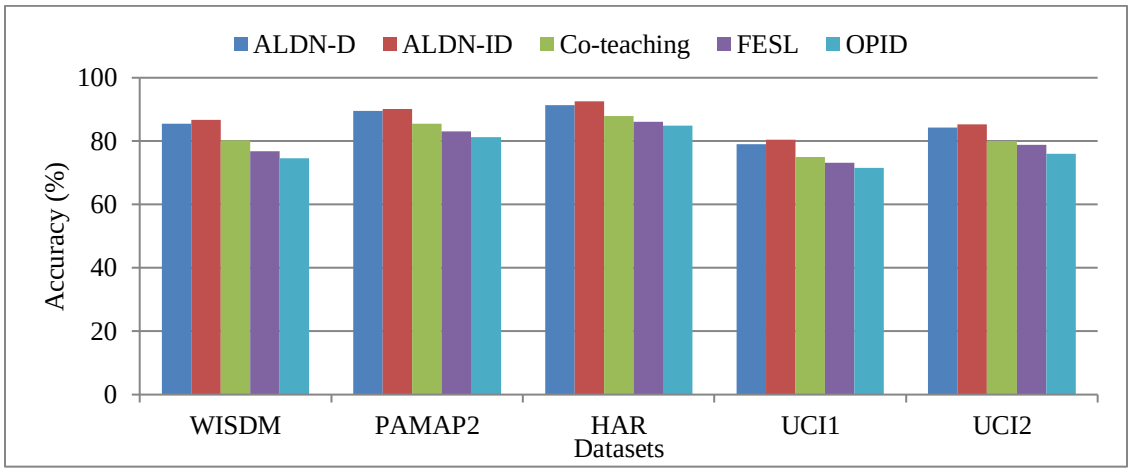

Fig. 1 Accuracy comparison across datasets

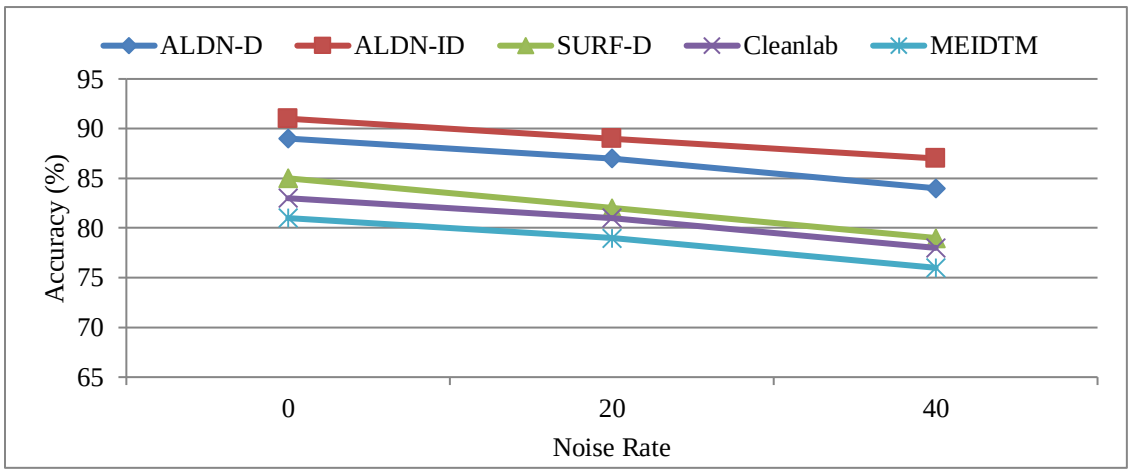

Fig. 2 Accuracy versus noise rate

Figure 2 presents the decrease in model accuracy as the noise rate increases from 0.2 to 0.5. The graph shows that higher noise rates lead to lower accuracy across all models. At a 0.2 noise rate, ALDN-ID reaches the highest accuracy at about 92% closely followed by ALDN-D at around 91%. Co-teaching performs moderately with approximately 87%. FESL and OPID show lower values at 83% and 81%. When the noise rate increases to 0.3, ALDN-ID maintains a strong position with 89% accuracy. ALDN-D drops slightly to 88%. Co-teaching stands at 84%, FESL drops to 80% and OPID follows with 78%. As the noise level continues to rise, all models experience additional drops in accuracy. At a 0.4 noise rate, ALDN-ID and ALDN-D still lead with about 86% and 85%. Co-teaching decreases to around 80%. At the same time, FESL and OPID go down to 76% and 73%. At the highest noise rate of 0.5, ALDN-ID records about 83%, and ALDN-D reaches 82%. Co-teaching drops to 75%, FESL to 70% and OPID falls to 68%. Despite the increasing noise, ALDN-ID and ALDN-D stay more robust than the other models.

Figure 3 shows the accuracy of the two models changes as the regularization parameter λ increases from 0 to 10. At λ = 0, ALDN-ID reaches about 88% accuracy. At the same time, ALDN-D starts slightly lower at around 83%. As λ increases to 0.1, both models see a large jump. ALDN-ID peaks around 92% and ALDN-D increases sharply to about 90%. When λ reaches 1, the accuracy of both models drops a bit. ALDN-ID falls to about 91% while ALDN-D comes down to 89%. As the regularization parameter continues to increase, both models show a gradual decline in accuracy. At λ = 5, ALDN-ID stays ahead at around 88.5%. ALDN-D follows at about 87%. Finally, at the highest value of λ = 10, ALDN-ID still leads with approximately 86.5%. ALDN-D drops to around 85%. ALDN-ID constantly performs better than ALDN-D at every point. Both models are sensitive to changes in the regularization parameter, especially at the lower end. The best accuracy for both is observed between λ = 0.1 and λ = 1, after which performance steadily decreases.

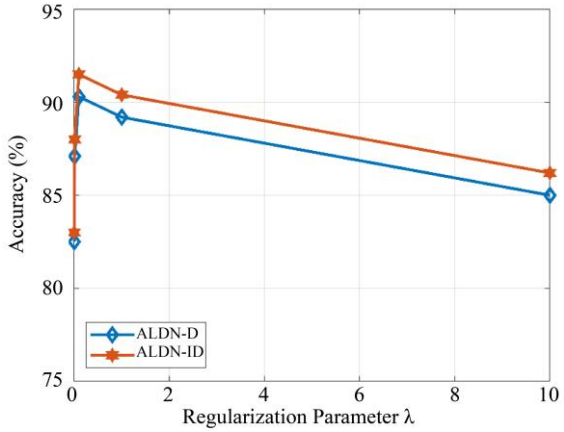

Fig. 3 Accuracy versus regularization parameter

Figure 4 illustrates the progression of model accuracy over 10 iterations. At iteration 1, FDAALDN-ID performs best with nearly 74% accuracy. FDAALDN-D and ALDN-ID follow closely at around 72%. At the same time, ALDN-D starts just behind them. Cleanlab begins near 69% and both SURF-D and MEIDTM show the lowest initial accuracy, close to 65%. As training continues, all models gradually improve. By iteration 5, FDAALDN-ID and FDAALDN-D both rise above 85%. ALDN-ID and ALDN-D make strong progress, reaching just below 85%. The other three models show smaller gains, with Cleanlab approaching 78% and the rest staying just under that. By the final iteration, FDAALDN-ID peaks above 91% accuracy, maintaining the lead throughout. FDAALDN-D follows closely with around 90%. ALDN-ID and ALDN-D stay competitive, both nearing 89%. Cleanlab finishes with nearly 80% accuracy. SURF-D and MEIDTM end a bit lower. Models using the FDAALDN architecture show the greatest and most steady improvement. ALDN-based methods show strong upward trends. Cleanlab, SURF-D, and MEIDTM show moderate improvement but remain significantly behind the others by the end.

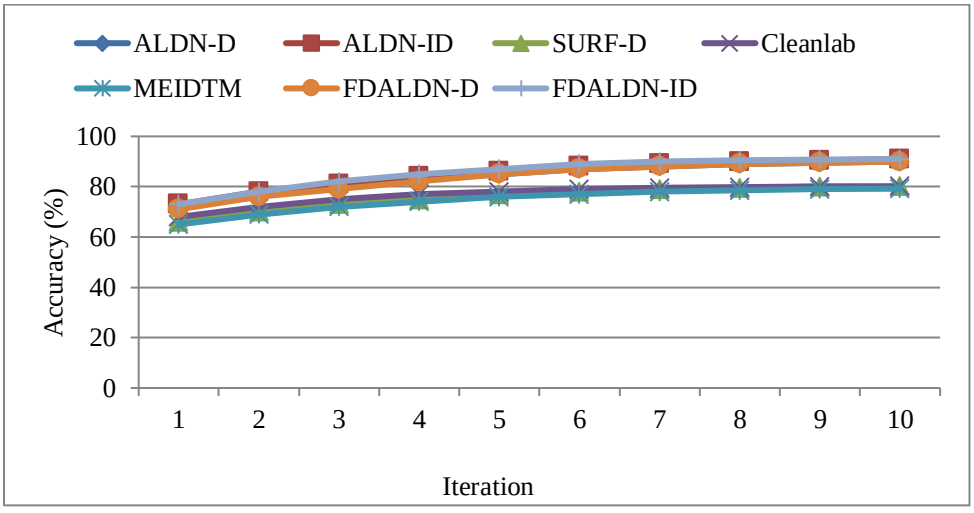

Fig. 4 Convergence curve: Iteration versus accuracy

Figure 5 compares the accuracy of different methods under three types of noise: class-dependent, instance-dependent, and feature-dependent. There are seven methods shown: ALDN-D, ALDN-ID, Cleanlab, SURF-D, MEIDTM, FDALDN-D, and FDALDN-ID. Under class-dependent noise, FDALDN-ID achieves the highest accuracy at around 93% followed closely by FDALDN-D at about 91%. ALDN-ID and ALDN-D perform well, reaching around 90% and 89% respectively. Cleanlab, SURF-D, and MEIDTM show lower performance with accuracies of about 84%, 81% and 79% respectively. In the instance-dependent setting, FDALDN-ID again shows the best performance at about 91%. It is followed by FDALDN-D with close to 89%. ALDN-ID and ALDN-D show slightly lower accuracies of around 89% and 87%. Cleanlab drops again to about 82%. SURF-D and MEIDTM fall to 78% and 76%. For feature-dependent noise, FDALDN-ID remains the top performer with roughly 90% accuracy. FDALDN-D and ALDN-ID follow scoring around 87% and 86%. ALDN-D drops to about 85%. Cleanlab goes down to 79%. SURF-D and MEIDTM perform the worst in this case. It reaches only around 75% and 72%. FDALDN-ID shows the most reliable and highest accuracy across all noise types.

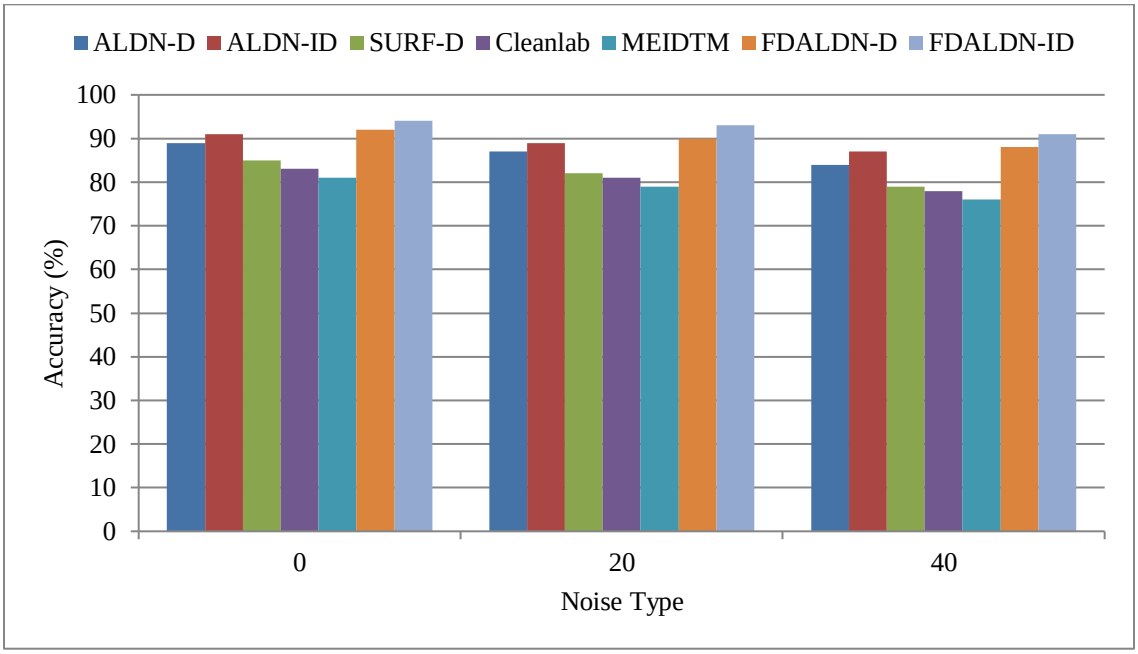

Fig. 5 Accuracy comparison under different noise types

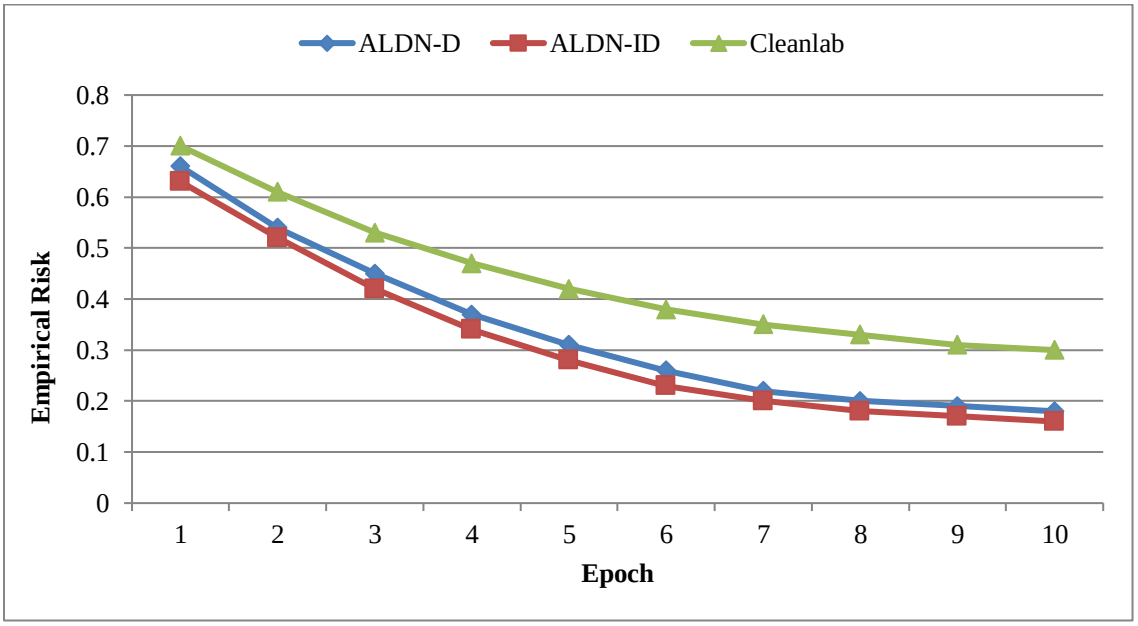

Fig. 6 Empirical risk versus epoch

Figure 6 presents the change in empirical risk across 10 training epochs for three models. The graph shows a steady reduction in loss as training progresses. At epoch 1, Cleanlab begins with the highest empirical risk around 0.71. ALDN-D starts at about 0.66. ALDN-ID is slightly better at 0.64. As training continues, all models reduce the risk values. By epoch 5, ALDN-ID has dropped to around 0.28. ALDN-D is close at 0.31, and Cleanlab is higher at approximately 0.42. This shows that both ALDN models learn faster and manage risk better than Cleanlab in the early stages. From epochs 6 to 10, ALDN-ID continues to improve. It ends with the lowest empirical risk of around 0.17. ALDN-D finishes at about 0.19. Cleanlab reduces its risk more slowly and finishes at around 0.31. The performance gap between Cleanlab and the other two models becomes wider as training progresses. These results suggest that ALDN-ID reduces empirical risk the most. ALDN-D comes next in performance. Cleanlab shows the least efficiency among the three models.

Figure 7 shows the relationship between the generalization bound and sample size for five methods. As the sample size increases from 100 to 2000, the generalization bound decreases for all methods. At the smallest sample size, MEIDTM has the highest bound near 2.9. ALDN-ID and ALDN-D are much lower, around 1.9 and 2.0. SURF-D and Cleanlab fall in between. ALDN-ID and ALDN-D generalize better even with fewer samples. As the sample size grows beyond 1000, the gap between the methods becomes clearer. ALDN-ID continues to maintain the lowest generalization bound, dropping close to 0.45. ALDN-D performs well and stays slightly above ALDN-ID. Cleanlab and SURF-D show higher values around 0.7 and 0.8, respectively. MEIDTM remains the highest, around 0.9, even with the largest sample size. The graph shows that ALDN-ID has the best generalization performance across all sample sizes, followed by ALDN-D. Cleanlab, SURF-D, and MEIDTM generalize less effectively, especially when data is limited.

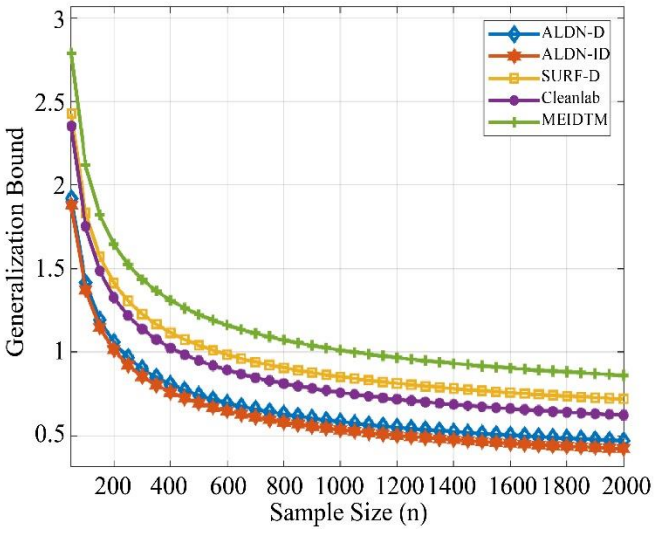

Fig. 7 Generalization bound versus sample size

Figure 8 compares the transferability scores of seven models. FDALDN-ID achieves the highest score at 0.25, followed closely by FDALDN-D at 0.22. These two models clearly show better ability to transfer learned knowledge to new tasks. ALDN-ID comes next with a score of 0.18. While ALDN-D follows with 0.15, this shows that the ALDN versions handle transfer learning reasonably well, though not as strongly as the FDALDN models. On the lower side, MEIDTM reaches a score of only 0.09. Cleanlab performs slightly worse with a score near 0.07. SURF-D has the lowest transferability score at 0.05.

These three models show weaker transfer performance. Models with the FDALDN structure are most effective at transferring knowledge. While SURF-D, Cleanlab, and MEIDTM lag behind. There is a large difference between the FDALDN models and the others. This shows that feature-dependent adaptation is useful. It helps the model generalize better to new tasks. The FDALDN models probably capture more transferable patterns by focusing on both features and noise types. ALDN models, while simpler, still outperform traditional reference methods. Lower scores in SURF-D and Cleanlab reflect limited flexibility.

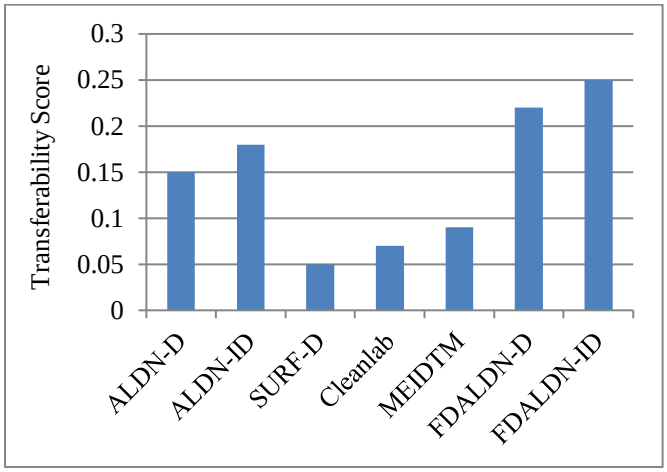

Fig. 8 Model transferability across methods

Figure 9 shows the accuracy of different models when feature dropout rates increase from 0% to 90%. As expected, accuracy decreases for all models as dropout increases. At 0% dropout, FDALDN-ID performs the best with an accuracy of nearly 94%. FDALDN-D follows closely, maintaining around 91%. ALDN-ID and ALDN-D start near 91% and 89% respectively. At the same time, SURF-D starts at about 85%. Cleanlab and MEIDTM begin at lower values, around 82% and 81%, respectively. This initial comparison shows FDALDN-ID is the most stable and accurate under full feature conditions. As dropout increases to 90%, FDALDN-ID still holds the highest accuracy at around 85% showing strong robustness. FDALDN-D drops slightly more but still stays above 80%. ALDN-ID and ALDN-D decrease steadily and end near 79% and 76%. SURF-D, Cleanlab, and MEIDTM

show more sensitivity, ending at 70%, 68% and 67% respectively. This proves that FDALDN models are more resistant to feature loss. Traditional models lose more accuracy as features are dropped. FDALDN-ID maintains the best performance throughout, showing its ability to handle missing or noisy data well.

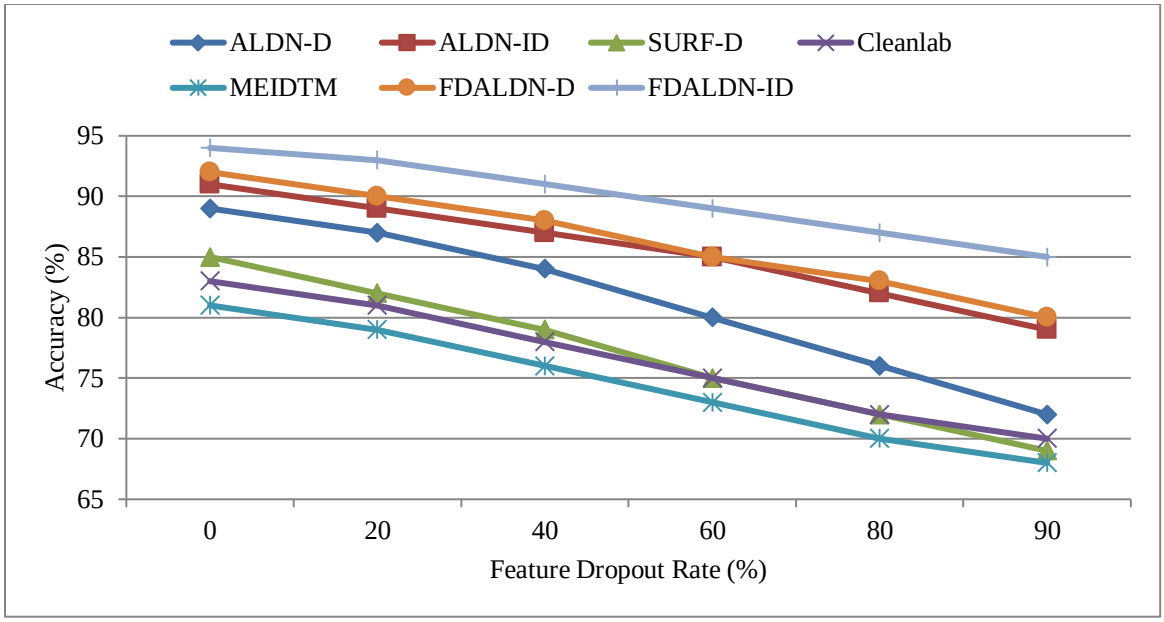

Fig. 9 Accuracy versus feature dropout rate

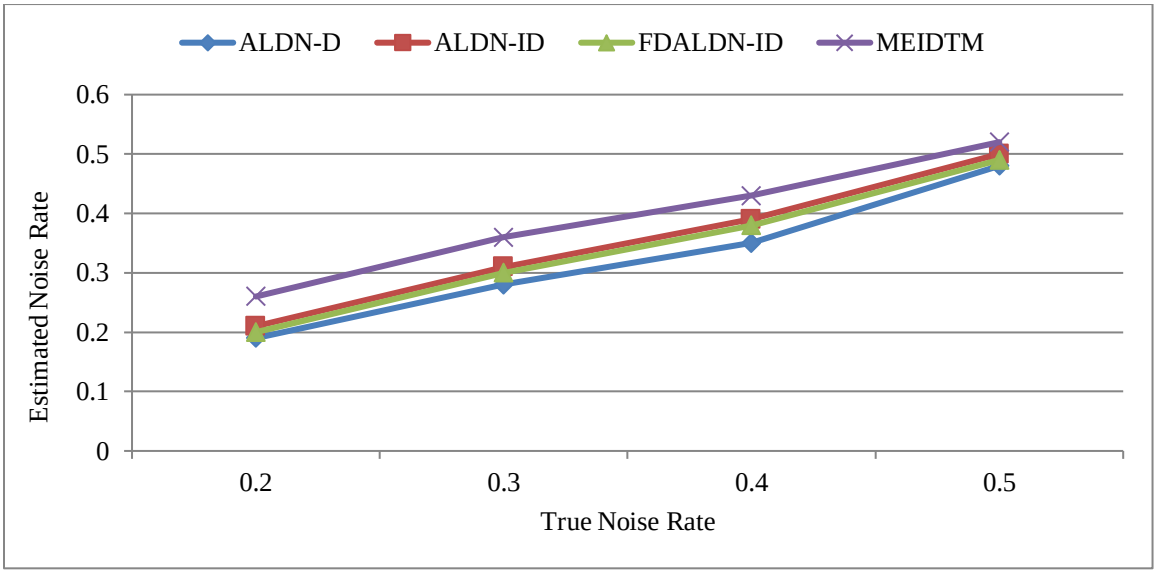

Fig. 10 Noise estimation accuracy

Figure 10 presents a comparison of four models based on the accuracy of noise rate estimation in the data. The graph shows the relationship between true noise rates and the corresponding estimated values. All the models display a clear upward leaning. This shows the estimated noise increases as the actual noise increases. ALDN-D starts at around 0.19 and goes up to about 0.48. ALDN-ID and FDALDN-ID both show a steady and close alignment with the actual noise ending near 0.50. MEIDTM constantly estimates higher noise values starting at 0.26 and rising to about 0.52. This suggests that MEIDTM inclines to overestimate noise compared to the others. FDALDN-ID shows strong performance by maintaining a close match with the true noise rate, especially at higher noise levels. ALDN-ID performs similarly well, making both models suitable for real-world applications involving moderate to high noise. ALDN-D remains close but slightly underestimates noise. The steady gap seen in MEIDTM's curve indicates less accuracy in fine-tuning to real noise levels. FDALDN-ID demonstrates the most stable and precise noise estimation behaviour in this comparison.

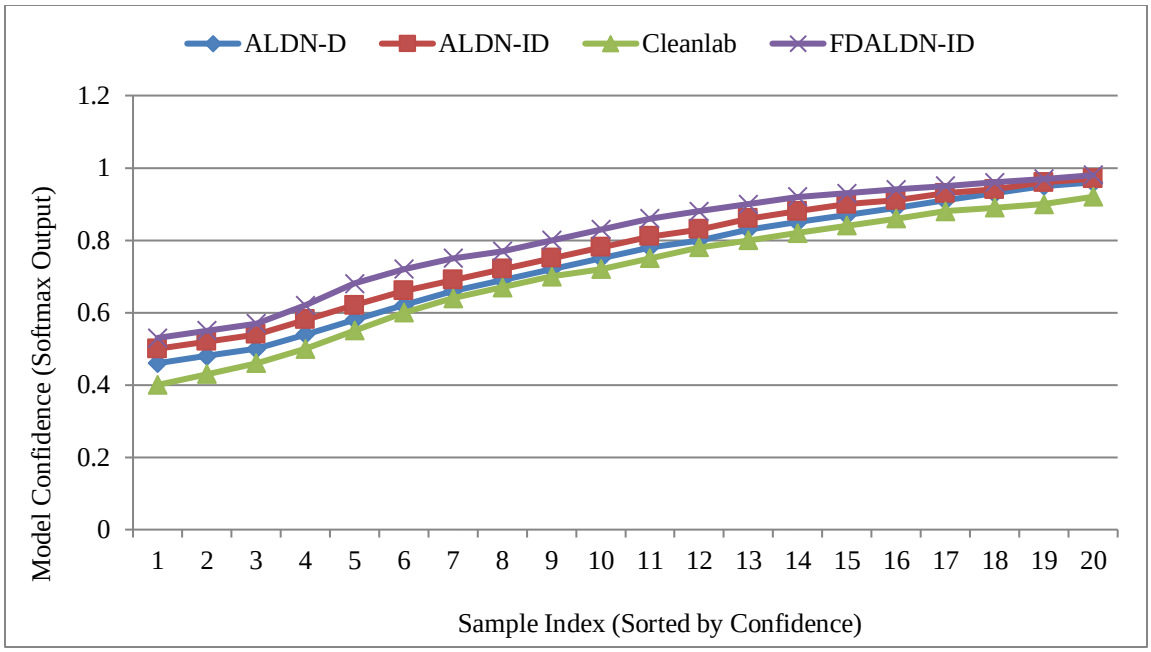

Fig. 11 Sample-Wise confidence curves

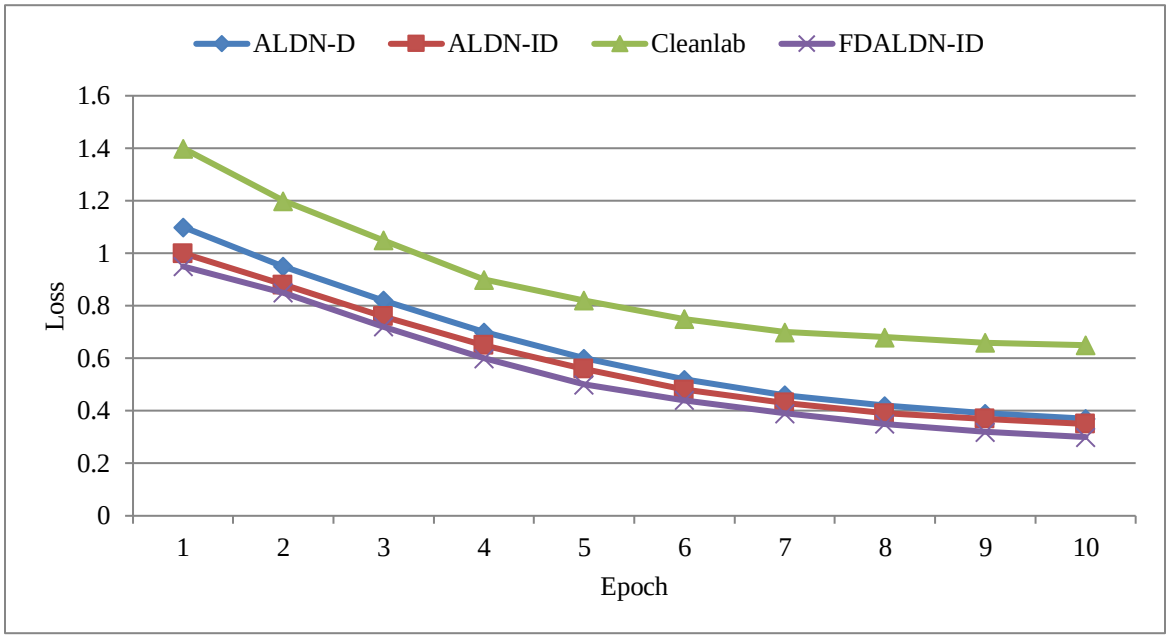

Fig. 12 Training loss curve

Figure 11 shows the confidence levels of different models in predictions. The graph displays softmax scores across sorted sample indices. Four models are compared: ALDN-D, ALDN-ID, Cleanlab, and FDALDN-ID. Across all samples, the FDALDN-ID model shows the highest confidence values starting around 0.52 and reaching close to 0.99. ALDN-ID follows closely behind with slightly lower values. ALDN-D stays a little below ALDN-ID across most of the range. Cleanlab shows the lowest confidence throughout, starting near 0.40 and ending just below 0.90. The gap between the models is most visible at the start, with FDALDN-ID having a clear lead. This shows that FDALDN-ID is more confident even in its lower-ranked predictions. The difference in confidence narrows as the sample index increases. It suggests all models agree more on higher-confidence predictions. Cleanlab lags behind the others, showing that it might be more uncertain in its outputs. ALDN-ID and ALDN-D perform similarly, but FDALDN-ID is slightly better. The pattern stays constant for FDALDN-ID. It shows higher confidence overall. It is more stable across all samples. This makes it more reliable when model certainty is important.

The Figure 12 displays the training loss across 10 epochs for four models. From the first epoch, FDALDN-ID begins with the lowest loss around 1.0. It steadily decreases to approximately 0.32 by the 10th epoch. ALDN-ID starts just above 1.1 and follows a similar downward trend. It finishes slightly above FDALDN-ID. ALDN-D starts around 1.2 and ends near 0.37, showing steady improvement but slightly behind the ID version. Cleanlab starts at around 1.4 and finishes at the highest loss level too, ending around 0.64. FDALDN-ID shows better optimization during training. It reduces loss more quickly. The reduction is more stable than the other models. By the fifth epoch, FDALDN-ID has already crossed below the 0.5 loss mark. ALDN-D and ALDN-ID hover just above it. Cleanlab stays much higher, reflecting slower convergence and potentially less effective learning. The pattern across all epochs highlights FDALDN-ID's advantage in managing loss during training. This performance gap suggests that FDALDN-ID uses more advanced learning methods. These methods lead to better training stability. The approach improves efficiency compared to the other models.

6. Conclusion

In this paper, a new problem in machine learning is studied. The problem is to learn from data when features change, and labels are noisy. This is a very real situation in many practical tasks. For example, in sensor-based recognition, some sensors stop working or are replaced. This causes the features to change. At the same time, less reliable data leads to wrong labels. To solve this, a new algorithm, ALDN, is proposed. The key impression is to reuse a model trained on earlier, clean data. This older model is mapped into the current feature space using a method of optimal transport. This mapped model is termed the prior model.

The prior model is not trained again but reused. It helps with two things. First, it helps estimate the noise in the labels. Second, it helps guide the training of the new model. A regularizer is used to make sure the new model stays close to the prior model. This gives more stability and better results. There are two versions of the method. One is ALDN-D, which uses direct constraints. The other is ALDN-ID that uses indirect constraints. Both provide strong theoretical guarantees. The method works well in practice. In all experiments, the proposed method outperformed other baseline methods. It was specifically useful when the training data was small and the features were not stable. ALDN is extended to handle more complex noise. This version is termed as FDALDN. It deals with feature-dependent noise, which is harder to manage.

References

[1] Yang Tang et al., "Introduction to Focus Issue: When Machine Learning Meets Complex Systems: Networks, Chaos, and Nonlinear Dynamics," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 6, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis, "Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers," *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 614-622, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[3] Tony Jinks, *Disappearing Object Phenomenon: An Investigation*, McFarland, pp. 1-192, 2016. [Google Scholar] [Publisher Link]

[4] Guozhu Dong et al., "Online Mining of Changes From Data Streams: Research Problems and Preliminary Results," *Proceedings of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams*, pp. 739-747, 2003. [Google Scholar] [Publisher Link]

[5] Włodzisław Duch, and Geerd H.F. Diercksen, "Feature Space Mapping as a Universal Adaptive System," *Computer Physics Communications*, vol. 87, no. 3, pp. 341-371, 1995. [CrossRef] [Google Scholar] [Publisher Link]

[6] Subutai Ahmad, and Volker Tresp, "Some Solutions to the Missing Feature Problem in Vision," *Advances in Neural Information Processing Systems*, vol. 5, pp. 393-400, 1992. [Google Scholar] [Publisher Link]

[7] Shilin Gu et al., "Adaptive Learning for Dynamic Features and Noisy Labels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 2, pp. 1219-1237, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[8] Yang Liu, and Jialu Wang, "Can Less be More? When Increasing-to-Balancing Label Noise Rates Considered Beneficial," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17467-17479, 2021. [Google Scholar] [Publisher Link]

[9] K.J.M. Janssen et al., "Updating Methods Improved the Performance of a Clinical Prediction Model in New Patients," *Journal of Clinical Epidemiology*, vol. 61, no. 1, pp. 76-86, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[10] Tony Xiao et al., "Learning from Massive Noisy Labeled Data for Image Classification," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, pp. 2691-2699, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[11] Subhas Chandra Mukhopadhyay, "Wearable Sensors for Human Activity Monitoring: A Review," *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1321-1330, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[12] Zhenning Kong, Salah A. Aly, and Emina Soljanin, "Decentralized Coding Algorithms for Distributed Storage in Wireless Sensor Networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 261-267, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[13] Hans Jonas, "The Practical Uses of Theory [with Comments]," *Social Research*, vol. 26, no. 2, pp. 127-166, 1959. [Google Scholar] [Publisher Link]

[14] Xindong Wu et al., "Online Feature Selection with Streaming Features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 5, pp. 1178-1192, 2012. [CrossRef] [Google Scholar] [Publisher Link]
[15] Quanmao Lu, Xuelong Li, and Yongsheng Dong, "Structure Preserving Unsupervised Feature Selection," *Neurocomputing*, vol. 301, pp. 36-45, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[16] Qin Zhang et al., "Online Learning from Trapezoidal Data Stream," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 10, pp. 2709-2723, 2016. [CrossRef] [Google Scholar] [Publisher Link]
[17] Chenping Hou, and Zhi-Hua Zhou, "One-Pass Learning with Incremental and Decremental Features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 11, pp. 2776-2792, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[18] Qin Zhang et al., "Online Learning from Trapezoidal Data Stream," *Asian Conference on Machine Learning (ACML)*, vol. 28, no. 10, pp. 2709-2723, 2016. [CrossRef] [Google Scholar] [Publisher Link]
[19] Chenping Hou, and Zhi-Hua Zhou, "One-Pass Learning with Incremental and Decremental Features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 11, pp. 2776-2792, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[20] Bo-Jian Hou, Lijun Zhang, and Zhi-Hua Zhou, "Prediction with Unpredictable Feature Evolution," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 10, pp. 5706-5715, 2021. [CrossRef] [Google Scholar] [Publisher Link]
[21] Chenping Hou et al., "Incremental Learning for Simultaneous Augmentation of Feature and Class," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 12, pp. 14789-14806, 2023. [CrossRef] [Google Scholar] [Publisher Link]
[22] Koby Crammer et al., "Online Passive-Aggressive Algorithms," *Journal of Machine Learning Research*, vol. 7, no. 19, pp. 551-585, 2006. [Google Scholar] [Publisher Link]
[23] Tongliang Liu, and Dacheng Tao, "Classification with Noisy Labels by Importance Reweighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 447-461, 2015. [CrossRef] [Google Scholar] [Publisher Link]
[24] Benoit Frenay, and Michel Verleysen, "Classification in the Presence of Label Noise: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845-869, 2014. [CrossRef] [Google Scholar] [Publisher Link]
[25] Jing Zhou et al., "Streamwise Feature Selection," *Journal of Machine Learning Research*, vol. 7, no. 67, pp. 1861-1885, 2006. [CrossRef] [Google Scholar] [Publisher Link]
[26] B. Han et al., "Co-Teaching: Robust Training of Deep Neural Networks withExtremely Noisy Labels," NeurIPS, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[27] Jiaheng Wei et al., "Learning with Noisy Labels Revisited: A Study Using Real-World Human Annotations," *arXiv Preprint*, pp. 1-23, 2021. [CrossRef] [Google Scholar] [Publisher Link]
[28] Bo Han et al., "Co-Teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels," *Advances in Neural Information Processing Systems*, vol. 31, 2018. [Google Scholar] [Publisher Link]
[29] B. Frenay, and M. Verleysen, "Classification in the Presence of Label Noise: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845-869, 2013. [CrossRef] [Google Scholar] [Publisher Link]
[30] Giorgio Patrini et al., "Making Deep Neural Networks Robust to Label Noise: A Loss Correction Approach," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1944-1952, 2017. [Google Scholar] [Publisher Link]
[31] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang, "Confident Learning: Estimating Uncertainty in Dataset Labels," *Journal of Artificial Intelligence Research*, vol. 70, pp. 1373-1411, 2021. [CrossRef] [Google Scholar] [Publisher Link]
[32] Nagarajan Natarajan et al., "Learning with Noisy Labels," *Advances in Neural Information Processing Systems*, vol. 26, pp. 1-9, 2013. [Google Scholar] [Publisher Link]
[33] Hao Chen et al., "A General Framework for Learning from Weak Supervision," *arXiv Preprint*, 2024. [CrossRef] [Google Scholar] [Publisher Link]
[34] Xiaobo Xia et al., "Part-Dependent Label Noise: Towards Instance-Dependent Label Noise," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1-14, 2020. [Google Scholar] [Publisher Link]
[35] Antonin Berthon et al., "Confidence Scores Make Instance-dependent Label-noise Learning Possible," *Proceedings of the 38th International Conference on Machine Learning*, vol. 139, pp. 825-836, 2021. [Google Scholar] [Publisher Link]
[36] Bo Han et al., "Co-Teaching: Robust Training of Deep Neural Networks withExtremely Noisy Labels," *NeurIPS Proceedings*, 2018. [Google Scholar] [Publisher Link]
[37] Jacob Goldberger, and Ehud Ben-Reuven, "Training Deep Neural Networks Using a Noise Adaptation Layer," *International Conference on Learning Representations*, pp. 1-9, 2017. [Google Scholar] [Publisher Link]
[38] Yuncheng Li et al., "Learning from Noisy Labels with Distillation," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1910-1918, 2017. [Google Scholar] [Publisher Link]