Journal of Machine Learning Research 27 (2026) 1-20 Submitted 8/25; Revised 12/25; Published 7/26

scikit-activeml: A Comprehensive and User-Friendly Active Learning Library

Marek Herde*†, Minh Tuan Pham*†, Daniel Kottke‡, Alexander Benz†, Lukas Lührs†, Pascal Mergard†, Christoph Sandrock§, Jiaying Cheng†, Atal Roghman†, Mehmet Müjde†, Lukas Rauch†, and Bernhard Sick†

Editor: Alexandre Gramfort

Abstract

scikit-activeml is a user-friendly open-source Python library for active learning on top of scikit-learn. Included are implementations of a large collection of query strategies, models, and visualization tools in pool- and stream-based active learning for classification or regression tasks with single or multiple annotators. The flexible design of the active learning cycle enables individual adaptations to a variety of learning scenarios. Our source code with comprehensive documentation is available at https://scikit-activeml.github.io.

Keywords: Active Learning, Query Strategy, Classification, Regression, Python

1. Introduction

Supervised machine learning models require sufficiently labeled data. While unlabeled data can be easily collected, the labeling process is difficult, time-consuming, or expensive in many applications. Therefore, active learning (Settles, 2009) aims to query labels for samples that maximize the model's performance. Many query strategies have been proposed, leading to broader applicability (Li et al., 2025; Cacciarelli and Kulahci, 2024). Yet, the query strategies' varying requirements resulted in divergent, mostly incompatible implementations lacking thorough testing. A unified library can address these challenges by offering query strategies in a standardized structure, accompanied by comprehensive documentation and visualizations. Such a library supports reproducible large-scale evaluation studies, enabling straightforward comparisons with baseline and state-of-the-art query strategies.

scikit-activeml (v1.0) is a **Python library for active learning** that provides:

- implementations of 60 pool- and stream-based query strategies, drawn from over 40 research articles and covering classification and regression tasks,
- adoption of the design patterns and structure from scikit-learn (Pedregosa et al., 2011), enabling the integration into existing pipelines,
- a modular and user-friendly structure allowing the development of query strategies and active learning cycles according to users' individual requirements,
- and extensive documentation with animations and tutorials covering diverse topics, such as deep active learning, interactive labeling, and evaluation studies.

* Equally contributing authors contactable via marek.herde@uni-kassel.de & tuan.pham@uni-kassel.de.
† University of Kassel, Germany. ‡ Deutsche Bahn AG, Germany. § TU Wien, Austria.

2. Structural Overview

Our active learning library `scikit-activeml` adopts the design principles, e.g., interfaces, docstrings, and nomenclature, of the popular machine learning framework `scikit-learn` (Pedregosa et al., 2011). Figure 1 details the overall structure, which includes interfaces of active learning query strategies and estimators for handling partially labeled data.

The `QueryStrategy` interface requires the implementation of a `query` method, which selects data for labeling. Since such a selection depends on the active learning paradigm, there are two subinterfaces, one for pool- and one for stream-based active learning. Query strategies for pool-based active learning (Li et al., 2025) assume the availability of a large pool of unlabeled samples from which samples can be selected. Accordingly, the `query` method takes this pool as input and outputs the selected samples. Optionally, this method can return a query strategy's estimated utility scores for querying labels of samples. In contrast, stream-based active learning (Cacciarelli and Kulahci, 2024) assumes a stream of incoming data samples. Incoming samples are assessed sequentially or in batches to decide which samples to query for labeling. All stream-based query strategies implement an `update` method to track the history of queries and may be combined with an optional budget manager to limit the number of label acquisitions over time. Another differentiation of active learning concerns the label source, typically distinguished between a single omniscient annotator and multiple error-prone annotators (Herde et al., 2021). In the latter case, annotators may provide incorrect labels, requiring the query strategy to guide both sample and annotator selection. Currently, `scikit-activeml` offers multi-annotator learning exclusively for pool-based active learning, reflecting where most research is concentrated.

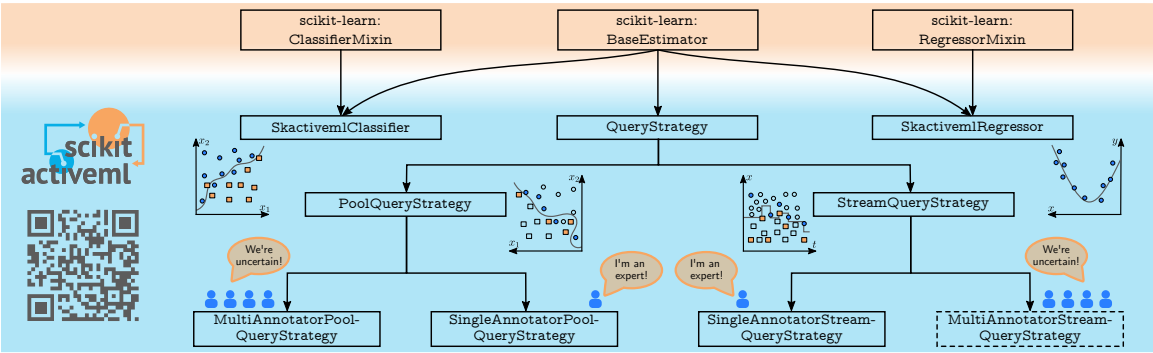

Figure 1: Nodes depict the core abstract classes of `scikit-activeml (v1.0)`, derived from `scikit-learn` abstract classes, arrows inheritance, and dashed nodes not-yet-supported features. `scikit-learn` is used here for identification only and does not imply endorsement.

Beyond query strategies, machine learning models represent another core part of `scikit-activeml`. So far, classification and regression models are supported. All of them either implement the `ClassifierMixin` or `RegressorMixin` interface to ensure compatibility with `scikit-learn`. The main difference from `scikit-learn` models is that our extensions natively accept unlabeled and labeled samples as training input. These `scikit-learn` models are integrated into our library with easy-to-use wrappers providing functionalities, such as cost-sensitive classification and probability density functions as regression outputs.

An active learning cycle is defined through a query strategy and a machine learning model. Algorithm 1 demonstrates this in a pool-based scenario with a single annotator. Samples in `X` are generated with `centers=8` classes. Current labels are stored in `y`, where unlabeled samples are indicated as `missing_label=-1`. As an example, ten random samples are labeled at the start. Using a logistic regression model wrapped with `SklearnClassifier` for filtering unlabeled samples in `X` and `y`, the cycle employs *batch active learning by diverse gradient embeddings* (BADGE, Ash et al., 2020) as query strategy returning the indices `q_ids` of the `batch_size=3` selected samples, whose labels in `y` are replaced with their true values from `y_true`. The querying continues until the maximum cycles are reached. Appendix A and our documentation provide code snippets of further (deep) active learning scenarios.

Despite `scikit-activeml`'s focus on classic `scikit-learn` machine learning models for tabular data, we also support deep learning models for image, text, and audio data together with state-of-the-art query strategies for deep active learning. For this purpose, users may implement their own architectures in `PyTorch` (Paszke et al., 2019) to be wrapped by `skorch` (Tietz et al., 2017) or they may employ pre-trained deep learning models, e.g., DINOv2 (Oquab et al., 2024), BERT (Devlin et al., 2019), and wav2vec (Baevski et al., 2020), as feature extractors. The latter approach is increasingly popular due to the ability of pre-trained deep learning models to infer rich, task-agnostic embeddings that enhance sample selection efficiency and performance in label-scarce scenarios (Hacohen et al., 2022; Yehuda et al., 2022; Huseljic et al., 2024; Gupte et al., 2024).

```python
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_blobs
from skactiveml.pool import Badge
from skactiveml.classifier import SklearnClassifier

# Generate dataset with 10 initially labeled samples.
X, y_true = make_blobs(n_samples=500, centers=8)
y = np.full_like(y_true, fill_value=-1)
rand_ids = np.random.permutation(len(X))[:10]
y[rand_ids] = y_true[rand_ids]

# Create classifier and query strategy.
clf = SklearnClassifier(
    LogisticRegression(max_iter=1000),
    classes=np.unique(y_true),
    missing_label=-1,
)
qs = Badge(missing_label=-1)

# Execute active learning cycle.
n_cycles = 20
for c in range(n_cycles):
    q_ids = qs.query(X=X, y=y, clf=clf, batch_size=3)
    y[q_ids] = y_true[q_ids]

# Fit final classifier.
clf.fit(X, y)
```

Algorithm 1: Code snippet of a basic pool-based active learning cycle for classification.

3. Related Work

Next to `scikit-activeml`, there exist several open-source Python projects for active learning. Table 1 provides an overview of the active learning scenarios supported by the individual projects, while Table 2 provides information about the organization of the projects' public repositories. Appendix D details the definitions of the tables' attributes.

A key difference between `scikit-activeml` and other active learning projects is its comprehensiveness, indicated by a large number of query strategies (cf. Appendix C) covering multiple active learning scenarios, which aligns with the objective of being a library without focusing on specific data types or query strategies. As of December 2025, the most related active learning project is `modAL`, which also supports many active learning scenarios

with `scikit-learn` as the main machine learning framework. However, `modAL` encapsulates fitting, querying, and labeling all in one class, which is straightforward but less flexible than `scikit-activeml` separating query strategies from the models. This is further detailed in Appendix D. Since 2024, only four active learning projects, including `scikit-activeml`, have published new releases. Among them, `Baal` focuses on Bayesian techniques, while `small-text` specifically targets text data. With 100 % code line coverage, `scikit-activeml` provides thoroughly tested reference implementations.

General		Query Strategies				Learning Tasks	
Name	Authors	Number	Pool	Stream	Multi-annotator	Regression	Classification
`DeepAL`	Huang	9	✓	✗	✗	✗	✓
`libact`	Yang et al.	20	✓	✗	✗	✗	✓
`ALiPy`	Tang et al.	30	✓	✗	✓	✗	✓
`modAL`	Danka and Horvath	23	✓	✓	✗	✓	✓
`ALToolbox`	Tsvigun et al.	19	✓	✗	✗	✗	✓
`Baal`	Atighehchian et al.	12	✓	✗	✗	✓	✓
`PyRelationAL`	Scherer et al.	17	✓	✗	✗	✓	✓
`small-text`	Schröder et al.	23	✓	✗	✗	✗	✓
`scikit-activeml`	Herde et al.	60	✓	✓	✓	✓	✓

Table 1: Feature comparison between active learning projects (as of December 11, 2025).

Name	Implementation					Documentation	
	Last Release	Frameworks	License	Tests	Coverage	Tutorials	Developer Guide
`DeepAL`	N/A	`PyTorch`	MIT	✗	N/A	✗	✗
`libact`	2019	`scikit-learn`	BSD-2-Clause	✓	90 %	✓	✓
`ALiPy`	2021	`scikit-learn`	BSD-3-Clause	✓	N/A	✓	✗
`modAL`	2023	`scikit-learn`, `Keras`, `skorch`	MIT	✓	97 %	✓	✓
`ALToolbox`	2022	`PyTorch`, `Transformers`	MIT	✓	N/A	✓	✓
`Baal`	2025	`PyTorch (Lightning)`, `Transformers`	Apache 2.0	✓	N/A	✓	✓
`PyRelationAL`	2024	`scikit-learn`, `PyTorch (Lightning)`	Apache 2.0	✓	94 %	✓	✓
`small-text`	2025	`scikit-learn`, `PyTorch`, `Transformers`	MIT	✓	96 %	✓	✓
`scikit-activeml`	2025	`scikit-learn`, `skorch`, `River`	BSD-3-Clause	✓	100 %	✓	✓

Table 2: Repository comparison between active learning projects (as of December 11, 2025).

4. Conclusion

We outlined core concepts of `scikit-activeml` and compared them with related Python projects for active learning. Our goal is to unify active learning processes for broader adoption. While we adhere to `scikit-learn` workflows, tasks such as object detection are not natively supported. Rather, `scikit-activeml` can serve as a reference implementation for active learning projects in these domains. Further limitations, including the current lack of principled guidance for choosing and evaluating query strategies with suitable data splits, define critical directions for future work and releases of `scikit-activeml` (see Appendix B).

Acknowledgments

We thank the reviewers and the editor for their constructive feedback. This project was funded by the state of Hesse at the University of Kassel in Germany.

Appendix A. Further Active Learning Scenarios

This appendix complements the pool-based active learning scenario for classification by providing code snippets with explanations for pool-based deep active learning, regression tasks, stream-based active learning, and active learning with multiple error-prone annotators. The corresponding examples are mostly restricted to simple (often synthetic) datasets, allowing the focus to remain on the active learning aspects rather than the data preprocessing and machine learning model design. We refer to our documentation containing more in-depth tutorials for complex real-world datasets and various types of machine learning models.

Algorithm 2 shows a code snippet of **deep pool-based active learning** (Li et al., 2025) for classification with a single annotator. This snippet demonstrates how `scikit-activeml` integrates `PyTorch` (Paszke et al., 2019) modules via `skorch` (Tietz et al., 2017) into a pool-based active learning workflow. After loading and preprocessing the `digits` dataset, a custom convolutional neural network `DigitClassificationModule` is defined using the standard `torch.nn.Module`. This network returns both class logits and a learned representation (embedding) for each input sample. The network is then wrapped by `SkorchClassifier`, which exposes it through a `scikit-learn`-compatible interface. For training via `fit` or `partial_fit`, the wrapper retains access to all relevant `PyTorch` training hyperparameters, such as the loss function, optimizer, learning rate (with schedulers), training batch size, and device configuration. For inference via `predict` and `predict_proba`, the user can describe how the network's raw outputs are to be interpreted by providing a simple mapping that assigns names (such as "probas" and "embeddings") to specific module outputs and optional post-processing steps. The first named output is treated as the source of class probabilities. At prediction time, the user can then flexibly request any subset of these named outputs to be returned in addition to the predicted class labels or class probabilities. In practice, this setup is not limited to small convolutional neural networks. Users can implement much more complex network architectures, including pre-trained models such as residual networks (He et al., 2016) and vision transformers (Oquab et al., 2024) for fine-tuning during each active learning cycle. Beyond the classifier, the query strategy `DropQuery` (Gupte et al., 2024) is instantiated with a mapping telling the classifier to provide the embeddings learned by the neural network as additional output when calling `predict` or `predict_proba`. During querying, the strategy filters the most informative samples by testing how their predictions change when dropout is applied to the features of the input samples. Only these informative samples are then clustered using their learned embeddings as a basis for selecting a diverse batch. The remainder of the code executes a pool-based active learning loop over several cycles: the query strategy selects unlabeled samples, their labels are revealed, and the classifier is updated accordingly. By calling `partial_fit`, we exemplify the use of a warm start, reusing the neural network parameters from the previous cycle as initialization (Rauch et al., 2023). In contrast, a cold start can be implemented by calling `fit`, which retrains the network from scratch in each cycle by randomly reinitializing its parameters.

```python
import numpy as np
import torch
import torch.nn.functional as F
from torch import nn
from sklearn.datasets import load_digits
from skactiveml.classifier import SkorchClassifier
from skactiveml.pool import DropQuery

# Load and preprocess the digits dataset.
digits = load_digits()
X = (digits.images[:, np.newaxis, :, :] / 16.0).astype(np.float32)
y_true = digits.target
classes = np.unique(y_true)
y = np.full_like(y_true, fill_value=np.nan, dtype=float)

# Create a convolutional neural network as a torch module.
class DigitClassificationModule(nn.Module):
    def __init__(self, n_classes=10):
        super().__init__()
        self.conv = nn.Conv2d(1, 8, 3, padding=1)
        self.fc = nn.Linear(8 * 4 * 4, n_classes)

    def forward(self, x):
        x = F.max_pool2d(F.relu(self.conv(x)), 2)
        x_embed = x.view(x.size(0), -1)
        logits = self.fc(x_embed)
        return logits, x_embed

# Wrap PyTorch module and specify its forward outputs and training hyperparameters.
clf = SkorchClassifier(
    module=DigitClassificationModule,
    # Each mapping entry is interpreted as name -> (idx in module.forward, post-hoc transform).
    forward_outputs={"probas": (0, nn.Softmax(dim=-1)), "embeddings": (1, None)},
    criterion=nn.CrossEntropyLoss,
    neural_net_param_dict={
        # Module-related parameters.
        "module__n_classes": len(classes),
        # Optimizer-related parameters.
        "max_epochs": 50,
        "optimizer": torch.optim.RAdam,
        "optimizer__lr": 0.01,
        # Data loading parameters.
        "iterator_train__shuffle": True,
        "iterator_train__batch_size": 32,
        "train_split": None,
        # Misc.
        "device": "cuda" if torch.cuda.is_available() else "cpu",
        "verbose": False,
    },
    classes=classes,
).initialize()
# Create query strategy such that: y_pred, X_embed = clf.predict(X, **clf_embedding_flag_name).
qs = DropQuery(clf_embedding_flag_name={"extra_outputs": ["embeddings"]})

# Execute active learning cycle with warm starts for retraining the classifier.
n_cycles = 5
for c in range(n_cycles):
    q_ids = qs.query(X=X, y=y, clf=clf, batch_size=32)
    y[q_ids] = y_true[q_ids]
    clf.partial_fit(X, y)
```

Algorithm 2: Code snippet of a pool-based deep active learning cycle for classification.

Algorithm 3 shows the code snippet of a simple example of **pool-based active learning for regression** (Kumar and Gupta, 2020) with a single annotator. The main difference from the code snippet for classification in Algorithm 1 is the usage of Gaussian processes as a regression model. Further, *improved greedy sampling* (iGS, Wu et al., 2019) is employed as a dedicated query strategy for regression tasks. The training labels `y` contain the target values for all observed samples `X`, where the symbol `np.nan` marks unlabeled samples, i.e., the ones where the target values have not been acquired yet. In each active learning cycle, `batch_size=10` of the unlabeled samples are selected for label acquisitions by overriding the `np.nan` symbol with the actual target values. In this example, the regressor is fitted manually outside of the `query` method by the user as an option for improved flexibility, which can be useful if the fitting process requires specialized handling. Moreover, the utility `scores` computed by the query strategy are returned for illustrative purposes.

```python
import numpy as np
from sklearn.gaussian_process import GaussianProcessRegressor
from skactiveml.pool import GreedySamplingTarget
from skactiveml.regressor import SklearnRegressor

# Generate dataset.
random_state = np.random.RandomState(0)
X = random_state.uniform(size=(200, 1), low=-1, high=1)
y_true = np.sin(X.ravel()) + random_state.randn(len(X)) * 0.1
y = np.full_like(y_true, fill_value=np.nan)

# Use 10 samples as initial training data.
y[:10] = y_true[:10]

# Create regressor and query strategy.
gp = GaussianProcessRegressor(random_state=0)
reg = SklearnRegressor(gp, random_state=0, missing_label=np.nan)
qs = GreedySamplingTarget(method="GSi", random_state=0, missing_label=np.nan)

# Execute active learning cycle.
n_cycles = 5
for c in range(n_cycles):
    reg.fit(X, y)
    q_ids, scores = qs.query(X=X, y=y, reg=reg, fit_reg=False, batch_size=10, return_utilities=True)
    y[q_ids] = y_true[q_ids]

# Fit final regressor.
reg.fit(X, y)
```

Algorithm 3: Code snippet of a pool-based active learning cycle for regression.

Algorithm 4 shows a code snippet of a simple example of **pool-based active learning with multiple annotators** (Donmez et al., 2009; Herde et al., 2021) for classification. In this scenario, we can only access error-prone annotators. Therefore, we aim to not only select the most informative samples but also annotators who provide correct labels for these samples. For this purpose, the most straightforward approach is to employ a common query strategy for selecting samples and then using multi-annotator learning models to guide the selection of annotators. These models estimate annotators' performances (Raykar et al., 2010; Herde et al., 2024) to correct their noisy labels for training accurate classifiers. In our code snippet, `y_noisy` represents the class labels of four annotators, where the first three columns correspond to three omniscient annotators and the last column to one adversarial

annotator. We train a logistic regression model via the expectation-maximization algorithm with the true labels as latent variables (Raykar et al., 2010). Samples are selected via *typical clustering* (TypiClust, Hacohen et al., 2022), while the trained model yields annotator-wise confusion matrices to estimate `A_perf` as their performances per sample for selecting the best annotators. For this purpose, different selection schemes are possible by varying the number of annotators to be queried per sample. In our example, we select `batch_size=3` samples per active learning cycle, and only a single, i.e., the estimated best, annotator for each of these three selected samples (`n_annotators_per_sample=1`).

```python
import numpy as np
from sklearn.datasets import make_blobs
from skactiveml.pool import TypiClust
from skactiveml.pool.multiannotator import SingleAnnotatorWrapper
from skactiveml.classifier.multiannotator import AnnotatorLogisticRegression

# Generate dataset with four annotators, of which the last one is always wrong.
X, y_true = make_blobs(n_samples=200, centers=5, random_state=0)
classes = np.unique(y_true)
y_noisy = np.column_stack((y_true, y_true, y_true, (y_true + 1) % 5))

# Perform active learning with zero initial labels.
y = np.full(shape=y_noisy.shape, fill_value=-1)

# Create classifier and query strategy.
clf = AnnotatorLogisticRegression(missing_label=-1, classes=classes, random_state=0)
qs = TypiClust(missing_label=-1, random_state=0)
ma_qs = SingleAnnotatorWrapper(qs, random_state=0, missing_label=-1)

# Function to be able to index via an array of indices.
idx = lambda A: (A[:, 0], A[:, 1])

# Execute active learning cycles.
n_cycles = 50
for c in range(n_cycles):
    # Fit multi-annotator classifier and compute annotators' accuracies per sample.
    clf.fit(X, y)
    _, A_perf = clf.predict(X, extra_outputs=["annotator_perf"])

    # Select samples and acquire labels.
    q_ids = ma_qs.query(X=X, y=y, batch_size=3, A_perf=A_perf, n_annotators_per_sample=1)
    y[idx(q_ids)] = y_noisy[idx(q_ids)]

# Fit final multi-annotator classifier.
clf.fit(X, y)
```

Algorithm 4: Code snippet of a pool-based active learning cycle with multiple annotators.

Algorithm 5 shows a code snippet of a simple example of **stream-based active learning** (Cacciarelli and Kulahci, 2024) with a single annotator. The data stream is given as unlabeled samples `X` and their labels `y_true`. The training data is managed with `X_train` and `y_train`. Here, we use a Parzen window classifier and Split (Žliobaitė et al., 2014) as the query strategy. The budget is set to 0.1 by default, i.e., the query strategy is allowed to query the labels of 10 % of the incoming unlabeled samples. The budget can be adjusted via the `budget` attribute for each stream-based query strategy. The classifier is retrained for each new sample. Contrary to the pool scenario, many stream-based query strategies need to keep track of the budget. Thus, after querying whether to label a sample or not, the available budget for the query strategy is updated. The separation between querying labels

and updating the available budget allows query strategies to be used within other strategies where the decision may be altered by a wrapper strategy.

```python
import numpy as np
from sklearn.datasets import load_breast_cancer
from skactiveml.classifier import ParzenWindowClassifier
from skactiveml.stream import Split
from skactiveml.utils import MISSING_LABEL

# Load dataset.
X, y_true = load_breast_cancer(return_X_y=True)
shuffle_ids = np.random.RandomState(0).permutation(len(X))
X, y_true = X[shuffle_ids], y_true[shuffle_ids]

# Initializing the training data as an empty array.
X_train, y_train = [], []

# Create classifier and query strategy.
clf = ParzenWindowClassifier(random_state=0, classes=np.unique(y_true))
qs = Split(random_state=0)

# Execute active learning cycle.
for x_t, y_t in zip(X, y_true):
    clf.fit(X_train, y_train)
    queried_indices = qs.query(candidates=x_t[None, :], clf=clf)
    qs.update(candidates=x_t[None, :], queried_indices=queried_indices)
    X_train.append(x_t)
    y_train.append(y_t if len(queried_indices) > 0 else MISSING_LABEL)

# Fit final classifier.
clf.fit(X_train, y_train)
```

Algorithm 5: Code snippet of a stream-based active learning cycle for classification.

Appendix B. Future Work

Figure 2 structures potential extensions within and around our library `scikit-activeml` into four categories.

Regarding the **methods**, we plan to extend the range of supported active learning scenarios to better reflect practical applications, for example, by adding multi-label classification (Wu et al., 2020) and multi-output regression (Li et al., 2022), which capture richer label dependencies than standard single-label settings. Since a central challenge in active learning is deciding when additional labels no longer justify their cost, we also aim to integrate stopping criteria (Vlachos, 2008) that assist users in terminating the querying process at an appropriate point.

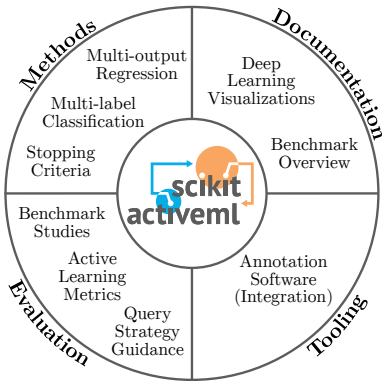

Figure 2: Potential extensions to `scikit-activeml`.

Proper **evaluation** of query strategies is essential for advancing research and deploying active learning in practical settings. We plan to provide evaluation metrics specific to active learning, such as the area under the learning curve, deficiency, and data utilization rate (Kottke et al., 2017), together with diagnostic visualizations, including the penalty matrix for pairwise strategy comparisons

across datasets (Ash et al., 2020). As there are many different query strategies available in `scikit-activeml`, researchers and practitioners may struggle to identify suitable strategies for their specific use cases. To address this, we aim to conduct a large-scale benchmark study across diverse domains, yielding empirically grounded guidance for narrowing down suitable strategies on new datasets. Nevertheless, such a benchmark does not resolve how to obtain a reliable estimate of generalization performance at low labeling cost for a specific application prior to deployment, a challenge addressed by active testing (Kossen et al., 2021).

As parts of our **documentation**, the in-depth tutorials and visualizations of query strategies already play a key role. However, while the tutorials cover scenarios of varying complexity, including deep active learning, our visualizations of the query strategies' sample selection behavior are restricted to one- and two-dimensional toy datasets. Therefore, a potential extension would be the inclusion of more realistic datasets, where sample selection behavior can be visualized in combination with dimensionality reduction techniques such as t-SNE (van der Maaten and Hinton, 2008). Furthermore, a brief overview of existing active learning benchmarks, as well as a potential benchmark based on our library, could enable users to better understand the strengths and weaknesses of individual query strategies.

While our library implements query strategies for various active learning scenarios, users are currently responsible for building their own annotation **tools** if they wish to deploy these strategies in practical labeling workflows. Currently, only a few tutorials demonstrate how to integrate our library with lightweight annotation tools. Accordingly, providing concrete integration recipes for further annotation tools and, if needed, developing a new annotation tool tailored to `scikit-activeml` would lower the barrier to real-world adoption.

Appendix C. Overview of Query Strategies

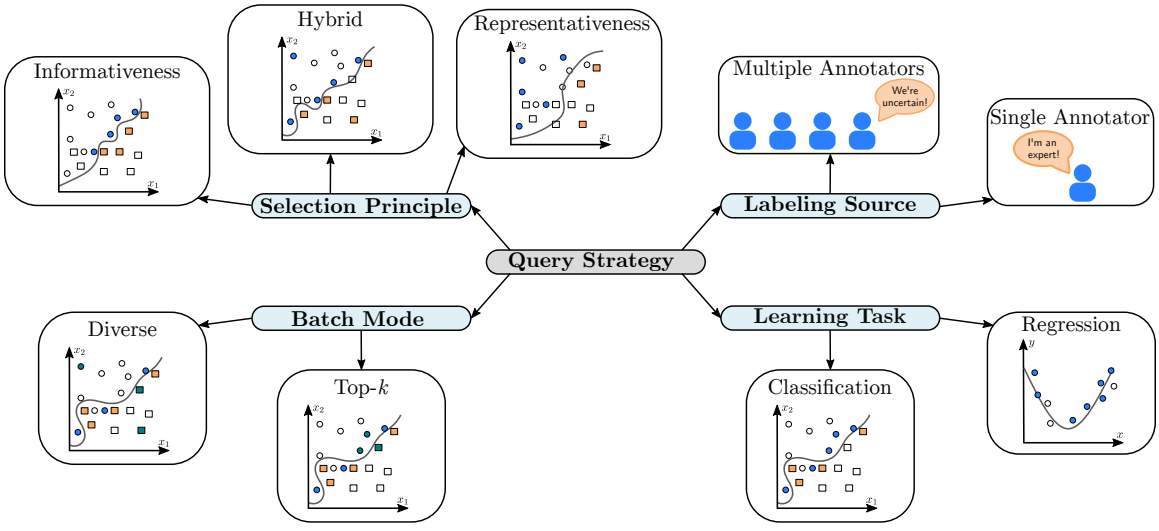

Figure 3: Mind map for categorizing the query strategies implemented by `scikit-activeml`.

Table 3 overviews the pool-based and Table 4 the stream-based query strategies implemented by `scikit-activeml`. These strategies originate from over 40 research articles,

some of which introduce multiple variants. For example, uncertainty sampling can be used with different uncertainty measures (Settles, 2009). We indicate each strategy's target learning tasks (regression and/or classification), flag multi-annotator scenarios, and mark strategies that consider diversity between samples within a selected batch. Furthermore, we categorize query strategies by their selection principles, i.e., informativeness (model uncertainty), representativeness (data-distribution coverage), and hybrid (combining both). Figure 3 provides a mind map that illustrates these different attributes of a query strategy.

Appendix D. Comparison of Active Learning Projects

Table 1 is based on a qualitative and a quantitative review of each **active learning project's query strategies**. Qualitatively, we mark pool, stream, multi-annotator, regression, and classification as "supported" whenever a built-in strategy or example code covers the scenario. Quantitatively, we count every class or function that implements a query strategy and, when one implementation offers algorithmically distinct variants, e.g., alternative uncertainty measures in the case of uncertainty sampling, loss functions in the case of expected error reduction, or disagreement measures in the case of query by committee, we count each variant once. Mere numeric hyperparameter tweaks are ignored. Counting publications instead of code would miss baselines and wrappers, such as random sampling and articles that bundle several strategies, so the code-level tally gives the most meaningful comparison.

Table 2 refers to each **active learning project's code and documentation**. We report whether tests exist and, if available, their coverage percentage. We verify the presence of a tutorial, i.e., a runnable, documented example that solves a concrete task, and a contribution guide that provides step-by-step instructions for extending the respective active learning project. We then list external machine learning frameworks, i.e., `scikit-learn` (Pedregosa et al., 2011), `PyTorch` (Paszke et al., 2019), `skorch` (Tietz et al., 2017), `PyTorch Lightning` (Falcon and Team, 2019), `Transformers` (Wolf et al., 2020), `Keras` (Chollet et al., 2015), and `River` (Montiel et al., 2021), that integrate with the active learning project, counting a framework as supported when tutorial code or other documentation lets users employ its models within the active learning workflow similar to built-in models.

Among existing active learning projects, `modAL` **is the closest to** `scikit-activeml`, as both aim to provide a unified interface for diverse active learning workflows. Beyond our library's broader range of query strategies, including more recent ones for (deep) active learning, it also differs from `modAL` in several conceptual aspects. Specifically, `scikit-activeml` promotes a separation between data, model, and query strategy, which facilitates reuse, experimentation with different components, and integration into custom pipelines. In contrast, `modAL` centers around the `ActiveLearner` class, which encapsulates sample selection via `ActiveLearner.query` as well as training and data management by internally storing the labeled dataset and updating it via `ActiveLearner.teach`. In `scikit-activeml`, the current partially labeled dataset is represented explicitly by a target array, where missing labels are encoded through a user-defined value. We provide the option to wrap `scikit-learn` models to handle partially labeled data (e.g., by ignoring unlabeled samples). By default, query strategies consider all unlabeled samples as candidates for labeling. On a code level, the differences between a typical active learning cycle in `scikit-activeml` and `modAL` are illustrated through a side-by-side comparison between Algorithm 6 and Algorithm 7.

Name	Classi-fication	Regres-sion	Multi-annotator	Diverse Batches	Variants	References
Baselines and Wrappers						
Random Sampling	✓	✓	✗	✓	1	N/A
Subsampling Wrapper	✓	✓	✗	N/A	1	N/A
Parallel Selection Wrapper[1]	✓	✓	✗	N/A	1	N/A
Single Annotator Wrapper	✓	✓	✓	N/A	1	N/A
Informativeness						
Uncertainty Sampling (US)	✓	✗	✗	✗	3	Lewis and Gale (1994) Scheffer et al. (2001) Settles (2009)
Query by Committee (QbC)	✓	✓	✗	✗	4	Seung et al. (1992) Engelson and Dagan (1996) McCallum and Nigam (1998) Burbidge et al. (2007) Beluch et al. (2018)
Expected Model Variance Reduction	✗	✓	✗	✗	1	Cohn et al. (1996)
Expected Error Reduction (EER)	✓	✗	✗	✗	2	Roy and McCallum (2001)
Value of Information (VOI)	✓	✗	✗	✗	3	Margineantu (2005) Kapoor et al. (2007) Joshi et al. (2009)
Interval Estimation Threshold (IEThresh)	✓	✗	✓	✗	1	Donmez et al. (2009)
Bayesian Active Learning by Disagreement (BALD)	✓	✗	✗	✗	1	Houlsby et al. (2011)
Expected Model Change (EMC)	✗	✓	✗	✗	1	Cai et al. (2013)
Active Learning with Cost Embedding (ALCE)	✓	✗	✗	✗	1	Huang and Lin (2016)
Uncertainty Sampling via Maximizing Average Precision (USAP)	✓	✗	✗	✗	1	Wang et al. (2018)
Expected Model Output Change (EMOC)	✗	✓	✗	✗	1	Käding et al. (2018)
Cost-Sensitive Uncertainty Sampling (CSUS)	✓	✗	✗	✗	1	Chen and Lin (2013)
Batch Bayesian Active Learning by Disagreement (BatchBALD)	✓	✗	✗	✓	1	Kirsch et al. (2019)
Epistemic Uncertainty Sampling (EpisUS)	✓	✗	✗	✗	1	Nguyen et al. (2019)
Regression-Based Kullback-Leibler Divergence Maximization	✗	✓	✗	✗	1	Elreedy et al. (2019)
Contrastive Active Learning (CAL)	✓	✗	✗	✗	1	Margatina et al. (2021)
Representativeness						
Core Set	✓	✓	✗	✓	1	Sener and Savarese (2018)
Discriminative Active Learning (DAL)	✓	✓	✗	✓	1	Gissin and Shalev-Shwartz (2019)
Greedy Sampling (GS)	✓	✓	✗	✓	3	Wu et al. (2019)
Typical Clustering (TypiClust)	✓	✓	✗	✓	1	Hacohen et al. (2022)
Probability Coverage (ProbCover)	✓	✗	✗	✓	1	Yehuda et al. (2022)
MaxHerding	✓	✓	✗	✓	1	Bae et al. (2024)

Continued on the next page.

Table 3: Pool-based query strategies implemented by `scikit-activeml` (part I).

[1] Although listed for completeness, this wrapper is not included in the reported number of query strategies, because it parallelizes utility computation for certain strategies without changing their query behavior.

Name	Classification	Regression	Multi-annotator	Diverse Batches	Variants	References
Hybrid						
Density-Weighted Uncertainty Sampling (DWUS)	✓	✗	✗	✗	1	Donmez et al. (2007)
Dual Strategy for Active Learning (DUAL)	✓	✗	✗	✗	1	Donmez et al. (2007)
Querying Informative and Representative Examples (QUIRE)	✓	✗	✗	✗	1	Huang et al. (2010) Huang et al. (2014)
Density-Diversity-Distribution -Distance Sampling (4DS)	✓	✗	✗	✓	1	Reitmaier and Sick (2013)
Multi-class Probabilistic Active Learning (McPAL)	✓	✗	✗	✗	1	Kottke et al. (2016)
Batch Active Learning by Diverse Gradient Embeddings (BADGE)	✓	✗	✗	✓	1	Ash et al. (2020)
Clustering Uncertainty-Weighted Embeddings (CLUE)	✓	✓	✗	✓	1	Prabhu et al. (2021)
Fast Active Learning by Contrastive Uncertainty (FALCUN)	✓	✗	✗	✓	1	Gilhuber et al. (2024)
Regression Tree-Based Active Learning (RT-AL)	✗	✓	✗	✓	3	Jose et al. (2024)
Dropout Query (DropQuery)	✓	✗	✗	✓	1	Gupte et al. (2024)

Table 3: Pool-based query strategies implemented by `scikit-activeml` (part II).

Name	Classification	Regression	Multi-annotator	Diverse Batches	Variants	References
Baselines and Wrappers						
Periodic Sampling	✓	✓	✗	✗	1	N/A
Stream Random Sampling	✓	✓	✗	✗	2	N/A
Informativeness						
Fixed-Uncertainty	✓	✗	✗	✗	1	Žliobaitė et al. (2014)
Randomized-Variable-Uncertainty	✓	✗	✗	✗	1	Žliobaitė et al. (2014)
Variable-Uncertainty	✓	✗	✗	✗	1	Žliobaitė et al. (2014)
Hybrid						
Density-Based Active Learning for Data Streams (DBALStream)	✓	✗	✗	✗	1	Ienco et al. (2014)
Split	✓	✗	✗	✗	1	Žliobaitė et al. (2014)
Probabilistic Active Learning (PAL) in Datastreams	✓	✗	✗	✗	1	Kottke et al. (2015)
Cognitive Dual-Query Strategy (CogDQS)	✓	✗	✗	✗	4	Liu et al. (2023)

Table 4: Stream-based query strategies implemented by `scikit-activeml`.

We demonstrate the **flexibility of our design** using two examples that adapt the basic active learning cycle to more advanced settings. In Algorithm 8, we integrate semi-supervised learning into the cycle by employing `LabelSpreading` from `scikit-learn`, which propagates labels based on similarity in the feature space. This way, we leverage the partially labeled dataset representation in `scikit-activeml`, whereas in `modAL` one would need to work around the `ActiveLearner.teach` method to accommodate partially labeled data. In Algorithm 9, we demonstrate how our library supports a flexible restriction of candidate sets for query strategies, enabling the chaining of multiple strategies. For this purpose, we first apply `UncertaintySampling` to focus on informative samples, and finally `CoreSet` to select

a diverse subset of these informative candidates. Implementing such a pipeline in `modAL` would require a custom meta query strategy that manages these selection stages and nested indexing across different candidate pools.

```python
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.linear_model import LogisticRegression
from skactiveml.classifier import SklearnClassifier
from skactiveml.pool import UncertaintySampling


# Generate the dataset.
X, y_true = make_blobs(n_samples=500, centers=8)

# Acquire 10 initially labeled samples.
rand_ids = np.random.permutation(len(X))[:10]
y = np.full_like(y_true, fill_value=-1)
y[rand_ids] = y_true[rand_ids]



# Create classifier and query strategy.
clf = SklearnClassifier(
  LogisticRegression(max_iter=1000),
  classes=np.unique(y_true),
  missing_label=-1,
)
qs = UncertaintySampling(
  method="margin_sampling",
  missing_label=-1,
)

# Execute active learning cycle.
n_cycles = 20
for c in range(n_cycles):
  q_ids = qs.query(X=X, y=y, clf=clf, batch_size=3)
  y[q_ids] = y_true[q_ids]


```

Algorithm 6: Basic pool-based active learning cycle in `scikit-activeml`.

```python
import numpy as np
from functools import partial
from sklearn.datasets import make_blobs
from sklearn.linear_model import LogisticRegression
from modAL.models import ActiveLearner
from modAL.uncertainty import margin_sampling

# Generate the dataset.
X, y_true = make_blobs(n_samples=500, centers=8)

# Acquire 10 initially labeled samples.
rand_ids = np.random.permutation(len(X))[:10]
X_initial = X[rand_ids]
y_initial = y_true[rand_ids]
X_pool = np.delete(X, rand_ids, axis=0)
y_pool = np.delete(y_true, rand_ids, axis=0)

# Create the active learner.
learner = ActiveLearner(
  query_strategy=partial(
    margin_sampling, n_instances=3
  ),
  estimator=LogisticRegression(max_iter=1000),
  X_training=X_initial,
  y_training=y_initial,
)


# Execute active learning cycle.
n_cycles = 20
for c in range(n_cycles):
  q_ids, _ = learner.query(X_pool)
  X_q, y_q = X_pool[q_ids], y_pool[q_ids]
  learner.teach(X=X_q, y=y_q)
  X_pool = np.delete(X_pool, q_ids, axis=0)
  y_pool = np.delete(y_pool, q_ids)
```

Algorithm 7: Basic pool-based active learning cycle in `modAL`.

```python
# Insert imports, classifier, and data.
...

# Create query strategy.
qs = UncertaintySampling(missing_label=-1)

# Label spreading leverages unlabeled samples.
prop = LabelSpreading()

# Execute active learning cycle.
n_cycles = 20
for c in range(n_cycles):
  # Fit label propagation as a semi-supervised model
  # on partially labeled data.
  prop.fit(X, y)

  # Retrain the classifier with class labels inferred
  # via label propagation.
  clf.fit(X, prop.predict(X))

  q_ids = qs.query(X=X, y=y, clf=clf, fit_clf=False)
  y[q_ids] = y_true[q_ids]
```

Algorithm 8: Pool-based active learning combined with semi-supervised learning.

```python
# Insert imports, classifier, and data.
...

# Create query strategies.
qs_1 = UncertaintySampling(missing_label=-1)
qs_2 = CoreSet(missing_label=-1)

# Execute active learning cycle.
n_cycles = 20
for c in range(n_cycles):
  # Keep most uncertain instances as candidates.
  c_ids = qs_1.query(
    X=X, y=y, clf=clf, batch_size=100
  )

  # Select diverse samples amongst the most
  # uncertain candidates.
  q_ids = qs_2.query(
    X=X, y=y, candidates=c_ids, batch_size=3
  )

  y[q_ids] = y_true[q_ids]
```

Algorithm 9: Pool-based active learning by chaining query strategies.

References

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In *Int. Conf. Learn. Represent.*, 2020.

Parmida Atighehchian, Frédéric Branchaud-Charron, and Alexandre Lacoste. Bayesian Active Learning for Production, a Systematic Study and a Reusable Library. *arXiv:2006.09916*, 2020.

Wonho Bae, Junhyug Noh, and Danica J. Sutherland. Generalized Coverage for More Robust Low-Budget Active Learning. In *Eur. Conf. Comput. Vis.*, pages 318–334, 2024.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In *Adv. Neural Inf. Process. Syst.*, pages 12449–12460, 2020.

William H. Beluch, Tim Genewein, Andreas Nürnberger, and Jan M. Köhler. The Power of Ensembles for Active Learning in Image Classification. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pages 9368–9377, 2018.

Robert Burbidge, Jem J. Rowland, and Ross D. King. Active Learning for Regression Based on Query by Committee. In *Int. Conf. Intell. Data Eng. Autom. Learn.*, pages 209–218, 2007.

Davide Cacciarelli and Murat Kulahci. Active Learning for Data Streams: A Survey. *Mach. Learn.*, 113(1):185–239, 2024.

Wenbin Cai, Ya Zhang, and Jun Zhou. Maximizing Expected Model Change for Active Learning in Regression. In *IEEE Int. Conf. Data Min.*, pages 51–60, 2013.

Po-Lung Chen and Hsuan-Tien Lin. Active Learning for Multiclass Cost-Sensitive Classification Using Probabilistic Models. In *Conf. Technol. Appl. Artif. Intell.*, pages 13–18, 2013.

François Chollet et al. Keras, 2015. URL `https://keras.io`. Accessed 05/2026.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active Learning with Statistical Models. *J. Artif. Intell. Res.*, 4:129–145, 1996.

Tivadar Danka and Peter Horvath. modAL: A Modular Active Learning Framework for Python. *arXiv:1805.00979*, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conf. North Am. Chapter Assoc. Comput. Linguist.: Hum. Lang. Technol.*, pages 4171–4186, 2019.

Pinar Donmez, Jaime G. Carbonell, and Paul N. Bennett. Dual Strategy Active Learning. In *Eur. Conf. Mach. Learn.*, pages 116–127, 2007.

Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. Efficiently Learning the Accuracy of Labeling Sources for Selective Sampling. In *ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 259–267, 2009.

Dina Elreedy, Amir F. Atiya, and Samir I. Shaheen. A Novel Active Learning Regression Framework for Balancing the Exploration-Exploitation Trade-Off. *Entropy*, 21(7):651, 2019.

Sean P. Engelson and Ido Dagan. Minimizing Manual Annotation Cost in Supervised Training from Corpora. In *Annu. Meet. Assoc. Comput. Linguist.*, pages 319–326, 1996.

William Falcon and The PyTorch Lightning Team. PyTorch Lightning, 2019. URL `https://github.com/Lightning-AI/pytorch-lightning`. Accessed 05/2026.

Sandra Gilhuber, Anna Beer, Yunpu Ma, and Thomas Seidl. FALCUN: A Simple and Efficient Deep Active Learning Strategy. In *Eur. Conf. Mach. Learn. Princ. Pract. Knowl. Discov. Databases*, pages 421–439, 2024.

Daniel Gissin and Shai Shalev-Shwartz. Discriminative Active Learning. *arXiv:1907.06347*, 2019.

Sanket R. Gupte, Josiah Aklilu, Jeffrey J. Nirschl, and Serena Yeung-Levy. Revisiting Active Learning in the Era of Vision Foundation Models. *Trans. Mach. Learn. Res.*, 2024.

Guy Hacohen, Avihu Dekel, and Daphna Weinshall. Active Learning on a Budget: Opposite Strategies Suit High and Low Budgets. In *Int. Conf. Mach. Learn.*, pages 8175–8195, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 770–778, 2016.

Marek Herde, Denis Huseljic, Bernhard Sick, and Adrian Calma. A Survey on Cost Types, Interaction Schemes, and Annotator Performance Models in Selection Algorithms for Active Learning in Classification. *IEEE Access*, 9:166970–166989, 2021.

Marek Herde, Lukas Lührs, Denis Huseljic, and Bernhard Sick. Annot-Mix: Learning with Noisy Class Labels from Multiple Annotators via a Mixup Extension. In *Eur. Conf. Artif. Intell.*, pages 2910–2918, 2024.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian Active Learning for Classification and Preference Learning. *arXiv:1112.5745*, 2011.

Kuan-Hao Huang. DeepAL: Deep Active Learning in Python. *arXiv:2111.15258*, 2021.

Kuan-Hao Huang and Hsuan-Tien Lin. A Novel Uncertainty Sampling Algorithm for Cost-Sensitive Multiclass Active Learning. In *IEEE Int. Conf. Data Min.*, pages 925–930, 2016.

Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active Learning by Querying Informative and Representative Examples. In *Adv. Neural Inf. Process. Syst.*, pages 892–900, 2010.

Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active Learning by Querying Informative and Representative Examples. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(10):1936–1949, 2014.

Denis Huseljic, Paul Hahn, Marek Herde, Lukas Rauch, and Bernhard Sick. Fast Fishing: Approximating BAIT for Efficient and Scalable Deep Active Image Classification. In *Eur. Conf. Mach. Learn. Princ. Pract. Knowl. Discov. Databases*, pages 280–296, 2024.

Dino Ienco, Bernhard Pfahringer, and Indrė Žliobaitė. High Density-Focused Uncertainty Sampling for Active Learning over Evolving Stream Data. In *Int. Workshop Big Data Streams Heterog. Source Min.: Algorithms Syst. Program. Models Appl.*, pages 133–148, 2014.

Ashna Jose, João P. A. de Mendonça, Emilie Devijver, Noël Jakse, Valérie Monbet, and Roberta Poloni. Regression Tree-Based Active Learning. *Data Min. Knowl. Discov.*, 38 (2):420–460, 2024.

Ajay J. Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class Active Learning for Image Classification. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2372–2379, 2009.

Christoph Käding, Erik Rodner, Alexander Freytag, Oliver Mothes, Björn Barz, and Joachim Denzler. Active Learning for Regression Tasks with Expected Model Output Changes. In *Br. Mach. Vis. Conf.*, 2018.

Ashish Kapoor, Eric Horvitz, and Sumit Basu. Selective Supervision: Guiding Supervised Learning with Decision-Theoretic Active Learning. In *Int. Jt. Conf. Artif. Intell.*, pages 877–882, 2007.

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. In *Adv. Neural Inf. Process. Syst.*, pages 7026–7037, 2019.

Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. Active Testing: Sample-Efficient Model Evaluation. In *Int. Conf. Mach. Learn.*, pages 5753–5763, 2021.

Daniel Kottke, Georg Krempl, and Myra Spiliopoulou. Probabilistic Active Learning in Datastreams. In *Int. Symp. Adv. Intell. Data Anal.*, pages 145–157, 2015.

Daniel Kottke, Georg Krempl, Dominik Lang, Johannes Teschner, and Myra Spiliopoulou. Multi-class Probabilistic Active Learning. In *Eur. Conf. Artif. Intell.*, pages 586–594, 2016.

Daniel Kottke, Adrian Calma, Denis Huseljic, Georg Krempl, and Bernhard Sick. Challenges of Reliable, Realistic and Comparable Active Learning Evaluation. *Interact. Adapt. Learn. Workshop @ Eur. Conf. Mach. Learn. Princ. Pract. Knowl. Discov. Databases*, pages 2–14, 2017.

Punit Kumar and Atul Gupta. Active Learning Query Strategies for Classification, Regression, and Clustering: A Survey. *J. Comput. Sci. Technol.*, 35(4):913–945, 2020.

David D. Lewis and William A. Gale. A Sequential Algorithm for Training Text Classifiers. In *Annu. Int. ACM-SIGIR Conf. Res. Dev. Inf. Retr.*, pages 3–12, 1994.

Cen-You Li, Barbara Rakitsch, and Christoph Zimmer. Safe Active Learning for Multi-output Gaussian Processes. In *Int. Conf. Artif. Intell. Stat.*, pages 4512–4551, 2022.

Dongyuan Li, Zhen Wang, Yankai Chen, Renhe Jiang, Weiping Ding, and Manabu Okumura. A Survey on Deep Active Learning: Recent Advances and New Frontiers. *IEEE Trans. Neural Netw. Learn. Syst.*, 36(4):5879–5899, 2025.

Sanmin Liu, Shan Xue, Jia Wu, Chuan Zhou, Jian Yang, Zhao Li, and Jie Cao. Online Active Learning for Drifting Data Streams. *IEEE Trans. Neural Netw. Learn. Syst.*, 34 (1):186–200, 2023.

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. Active Learning by Acquiring Contrastive Examples. In *Conf. Empir. Methods Nat. Lang. Process.*, pages 650–663, 2021.

Dragos D. Margineantu. Active Cost-Sensitive Learning. In *Int. Jt. Conf. Artif. Intell.*, pages 1622–1623, 2005.

Andrew K. McCallum and Kamal Nigam. Employing EM and Pool-Based Active Learning for Text Classification. In *Int. Conf. Mach. Learn.*, pages 350–358, 1998.

Jacob Montiel, Max Halford, Saulo M. Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor M. Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River: Machine Learning for Streaming Data in Python. *J. Mach. Learn. Res.*, 22(110): 1–8, 2021.

Vu-Linh Nguyen, Sébastien Destercke, and Eyke Hüllermeier. Epistemic Uncertainty Sampling. In *Int. Conf. Discov. Sci.*, pages 72–86, 2019.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. *Trans. Mach. Learn. Res.*, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Adv. Neural Inf. Process. Syst.*, pages 8026–8037, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot,

and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12(85):2825–2830, 2011.

Viraj Prabhu, Arjun Chandrasekaran, Kate Saenko, and Judy Hoffman. Active Domain Adaptation via Clustering Uncertainty-Weighted Embeddings. In *IEEE/CVF Int. Conf. Comput. Vis.*, pages 8485–8494, 2021.

Lukas Rauch, Matthias Aßenmacher, Denis Huseljic, Moritz Wirth, Bernd Bischl, and Bernhard Sick. ActiveGLAE: A Benchmark for Deep Active Learning with Transformers. In *Eur. Conf. Mach. Learn. Princ. Pract. Knowl. Discov. Databases*, pages 55–74, 2023.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from Crowds. *J. Mach. Learn. Res.*, 11(43): 1297–1322, 2010.

Tobias Reitmaier and Bernhard Sick. Let Us Know Your Decision: Pool-Based Active Training of a Generative Classifier with the Selection Strategy 4DS. *Inf. Sci.*, 230:106–131, 2013.

Nicholas Roy and Andrew McCallum. Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In *Int. Conf. Mach. Learn.*, pages 441–448, 2001.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active Hidden Markov Models for Information Extraction. In *Int. Symp. Intell. Data Anal.*, pages 309–318, 2001.

Paul Scherer, Alison Pouplin, Alice Del Vecchio, Suraj M S, Oliver Bolton, Jyothish Soman, Jake P. Taylor-King, Lindsay Edwards, and Thomas Gaudelet. PyRelationAL: A Python Library for Active Learning Research and Development. *arXiv:2205.11117*, 2022.

Christopher Schröder, Lydia Müller, Andreas Niekler, and Martin Potthast. Small-Text: Active Learning for Text Classification in Python. In *Conf. Eur. Chapter Assoc. Comput. Linguist.: Syst. Demonstr.*, pages 84–95, 2023.

Ozan Sener and Silvio Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *Int. Conf. Learn. Represent.*, 2018.

Burr Settles. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

H. S. Seung, Manfred Opper, and Haim Sompolinsky. Query by Committee. In *Annu. Workshop Comput. Learn. Theory.*, pages 287–294, 1992.

Ying-Peng Tang, Guo-Xiang Li, and Sheng-Jun Huang. ALiPy: Active Learning in Python. *arXiv:1901.03802*, 2019.

Marian Tietz, Thomas J. Fan, Daniel Nouri, Benjamin Bossan, and skorch Developers. *skorch: A scikit-learn Compatible Neural Network Library That Wraps PyTorch*, 2017. URL `https://skorch.readthedocs.io/en/stable/`. Accessed 05/2026.

Akim Tsvigun, Leonid Sanochkin, Daniil Larionov, Gleb Kuzmin, Artem Vazhentsev, Ivan Lazichny, Nikita Khromov, Danil Kireev, Aleksandr Rubashevskii, Olga Shahmatova, Dmitry V. Dylov, Igor Galitskiy, and Artem Shelmanov. ALToolbox: A Set of Tools for Active Learning Annotation of Natural Language Texts. In *Conf. Empir. Methods Nat. Lang. Process.: Syst. Demonstr.*, pages 406–434, 2022.

Laurens van der Maaten and Geoffrey Hinton. Visualizing Data Using t-SNE. *J. Mach. Learn. Res.*, 9(86):2579–2605, 2008.

Andreas Vlachos. A Stopping Criterion for Active Learning. *Comput. Speech Lang.*, 22(3): 295–312, 2008.

Hanmo Wang, Xiaojun Chang, Lei Shi, Yi Yang, and Yi-Dong Shen. Uncertainty Sampling for Action Recognition via Maximizing Expected Average Precision. In *Int. Jt. Conf. Artif. Intell.*, pages 964–970, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. In *Conf. Empir. Methods Nat. Lang. Process.: Syst. Demonstr.*, pages 38–45, 2020.

Dongrui Wu, Chin-Teng Lin, and Jian Huang. Active Learning for Regression Using Greedy Sampling. *Inf. Sci.*, 474:90–105, 2019.

Jian Wu, Victor S. Sheng, Jing Zhang, Hua Li, Tetiana Dadakova, Christine L. Swisher, Zhiming Cui, and Pengpeng Zhao. Multi-label Active Learning Algorithms for Image Classification: Overview and Future Promise. *ACM Comput. Surv.*, 53(2):1–35, 2020.

Yao-Yuan Yang, Shao-Chuan Lee, Yu-An Chung, Tung-En Wu, Si-An Chen, and Hsuan-Tien Lin. libact: Pool-Based Active Learning in Python. *arXiv:1710.00379*, 2017.

Ofer Yehuda, Avihu Dekel, Guy Hacohen, and Daphna Weinshall. Active Learning through a Covering Lens. In *Adv. Neural Inf. Process. Syst.*, pages 22354–22367, 2022.

Indrė Žliobaitė, Albert Bifet, Bernhard Pfahringer, and Geoffrey Holmes. Active Learning with Drifting Streaming Data. *IEEE Trans. Neural Netw. Learn. Syst.*, 25(1):27–39, 2014.