# Active Learning in the Wild - Building Better AI Models with Less Data

**Prashant Singh**

Senior AI Director at Quantaleap Inc. and Independent Researcher, B.Tech., Electrical Engineering, Indian Institute of Technology (IIT), Roorkee

**Abstract**
Active learning is an interactive machine learning paradigm in which the model selectively queries an oracle (e.g. a human annotator) to label the most informative unlabeled examples. This enables building accurate models using far fewer labeled data than traditional supervised learning. In this paper, we define active learning and contrast it with conventional "passive" learning. We review common active learning architectures (pool-based, stream-based, membership query synthesis), label selection strategies (uncertainty-based, diversity-based) and describe in detail how to implement an active learning loop using modern tools (e.g. scikit-learn, PyTorch, modAL). We compare the results of applying active learning to the Iris Dataset[11] and Titanic Dataset[12] for classification using various label selection strategies. We then present a real-world–inspired case study of using active learning on drone-collected power-line inspection imagery. In this scenario, a convolutional vision model (e.g. YOLOv8) is iteratively refined by selectively querying a few ambiguous frames for expert annotation, dramatically reducing labelling effort. We report on potential savings: for example, an AWS-case active learning pipeline achieved ~90% reduction in labelling cost and cut annotation turnaround from weeks to hours [1]. We also discuss limitations and pitfalls of active learning (e.g. human-in-the-loop cost, computational overhead, class imbalance issues [2][3]). In summary, active learning can greatly improve data efficiency and agility of AI model development, but it requires careful design of query strategies and system integration.

**Keywords:** Active Learning, Data Labelling Efficiency, Machine Learning Optimization, Anomaly Detection, Drone Imagery Analysis

## 1. What is Active Learning

Active learning is a machine learning strategy in which the model actively selects which data points should be labeled to improve its performance, rather than relying on a fixed, fully labeled dataset. In active learning, one typically starts with a small seed of labeled examples and a large pool of unlabeled data. The model is trained on the small labeled set, then uses a query strategy (often based on prediction uncertainty) to choose the "most informative" (or diverse) samples to be annotated by an oracle (usually a human expert). These newly labeled examples are added back to the training set and the model is retrained. This cycle repeats, allowing the model to focus its labelling effort on the most valuable data, till a pre-determined accuracy is reached or a labelling budget is exhausted. In other words, the learning algorithm "asks questions" (queries labels) about the examples it is least certain about. Active learning seeks to achieve strong performance with far fewer training samples by iteratively querying an oracle for

labels on carefully selected data points.[4]
This approach contrasts with standard supervised learning, where a large labeled dataset is prepared in advance and the model learns passively from it. In active learning, by contrast, the model plays an active role in data selection. For example, Balakrishnan describes active learning as letting the algorithm query the "most uncertain or impactful samples" to label, rather than labelling vast amounts of data at random [2]. Likewise, Li et al. note that active learning iteratively asks an oracle to label selected samples in a human-in-the-loop process [4]. The ultimate goal is to minimize labelling cost and time while achieving accuracy close to that of a model trained on a much larger fully labeled set. In practice, active learning has been shown to yield competitive performance with significantly fewer labeled examples [4].
Active learning applies broadly across domains – from image and text classification to anomaly detection. It is especially valuable when labelling is expensive or time-consuming, such as in medical imaging or autonomous driving. By focusing human effort on the data examples that the model finds most confusing, active learning maximizes the value of each labeled example. As one practitioner notes, "active learning is a strategy to optimize the human labor required to build effective ML systems" [5]. Ultimately, active learning is a form of data-centric AI: it seeks to curate the best possible training data incrementally so that the model learns rapidly and efficiently.

## 2. Difference Between Traditional and Active Learning

In traditional supervised learning (sometimes called "passive learning"), the model is trained on a fixed, pre-labeled dataset. All available training examples come with labels before training begins and the model simply learns from these examples. Once training is finished, the learning process is complete – there is no further interaction or selection of new data. This approach requires gathering and labelling potentially vast datasets upfront, which can be costly and laborious. For example, passive learning might require tens or hundreds of thousands of labeled images to reach high accuracy. In this setting, the data labelling is independent of the model: labels are collected once and for all.
In active learning, by contrast, labelling is adaptive and focused. Instead of labelling a large static set, the model starts with a small labeled seed and then *iteratively* selects which additional samples should be labeled. Key differences include:

- **Labelling Budget**: Active learning often assumes a limited labelling budget. The algorithm strategically allocates this budget to the most informative samples. In passive learning, there is no such selection strategy; all samples (or a random subset) are labeled.
- **Data Selection**: In active learning, the model uses a *query strategy* (e.g. uncertainty sampling) to pick examples to query for labels [4][2]. In passive learning, there is no query strategy – the training set is pre-defined.
- **Adaptivity**: Active learning is adaptive – each round of labelling can use information from the current model. Passive learning is static and does not adapt based on model uncertainty.
- **Cost and Efficiency**: Because active learning focuses on "hard" or ambiguous cases, it can achieve comparable performance with far fewer labeled examples. Passive learning typically needs more labeled data and time to reach the same accuracy. As one source notes, active learning "doesn't need as many labels due to the impact of informative samples; passive learning needs more data, labels and time to train a model to achieve the same results" [6].

For example, in a computer vision task, a passive approach might require tens of thousands of labeled images of every object class to train a detector. An active learning approach would start with a handful of examples and only ask an expert to label additional images where the model is uncertain (e.g. images with occluded or rare objects). This can reduce the total labelling cost significantly while achieving similar model quality. In sum, active learning "actively participates" in label acquisition to optimize learning efficiency [6], whereas traditional supervised learning is "passive" with respect to data selection.

## 3. Architectures of Active Learning

Active learning can be implemented in several scenarios or "architectures," often categorized as **pool-based**, **stream-based** and **membership query synthesis** modes [5][3]. Each mode determines how the unlabeled data is accessed and queried:

- **Pool-based Active Learning**: There is a large *pool* of unlabeled data available. In each iteration, the active learner evaluates all (or a random subset of) unlabeled examples using a query strategy and selects the most informative examples to be labeled. This is the most common setting in literature and practical systems. For instance, the model might rank all images in an unlabeled pool by prediction confidence (or another utility measure) and then request labels for the top-k lowest-confidence examples [5]. Pool-based AL is powerful but can be memory-intensive, since it often requires scoring many samples each round.
- **Stream-based (Selective) Active Learning**: Unlabeled data arrives as a stream (e.g. one example at a time). For each new example, the model decides on-the-fly whether to query its label or discard it. This could be based on a confidence threshold: if the model is uncertain about the current sample, it queries the oracle, otherwise it assumes the model's prediction is correct. Stream-based AL is often used when data cannot be stored en masse and arrives continuously. It tends to be computationally lighter, as each instance is considered in sequence. However, it can be less optimal than pool-based AL because early budget exhaustion may prevent the model from seeing later examples. In real-world systems like video streams or sensor feeds, stream-based querying can catch surprising cases (e.g. anomalies) as they occur, but one must carefully manage the query budget so as not to run out of labels too soon [3][5].
- **Membership Query Synthesis**: The model is allowed to create or synthesize new examples for labelling, rather than only selecting from existing data. In this mode, the learner may generate counterfactual or augmented samples that it believes would be informative. For example, in image tasks, one might apply transformations (scaling, cropping, brightness changes) to create new inputs that cover underrepresented conditions. This is akin to data augmentation guided by the model's needs. Membership query synthesis can be very effective when labeled data is extremely scarce, but it requires domain-specific mechanisms to ensure the generated samples are realistic. For instance, one could synthesize a new view of a power-line image by slightly perturbing an existing one, then query whether an anomaly is present. In practice, this mode is less common, but concepts like generative models or GAN-based query synthesis fall under this category.

Often active learning systems combine multiple strategies for sample selection for labelling. For example, one might use pool-based sampling with a "committee" of models to gauge disagreement (query-by-committee) or select a sample based on the model's confidence (or lack thereof) in its prediction. Architecturally, an active learning loop typically involves a training pipeline and a labelling pipeline (see Fig. 1). The training pipeline trains or fine-tunes the model on the current labeled data. The labelling pipeline scores or inspects new unlabeled data, selects samples according to a query strategy and sends them to annotators. The newly labeled data is fed back into the training set for the next iteration. (For example, AWS implemented separate labelling and training code pipelines to automate this iterative loop. [1])

Several query selection strategies are employed in active learning to determine which data points to label:

- **Uncertainty Sampling:** Selects data points where the model's predictions have the least confidence.
- **Query by Committee (QBC):** Utilizes multiple models to identify data points where there is disagreement among their predictions.
- **Diversity Sampling:** Ensures a diverse set of examples is labelled to avoid overfitting to a specific subset of data.

[2]

More advanced techniques include Bayesian active learning or expected model change. Ultimately, the architecture of an active learning system depends on the domain requirements (e.g. whether data streams in, the cost of labelling and computational resources) and the chosen query strategy.

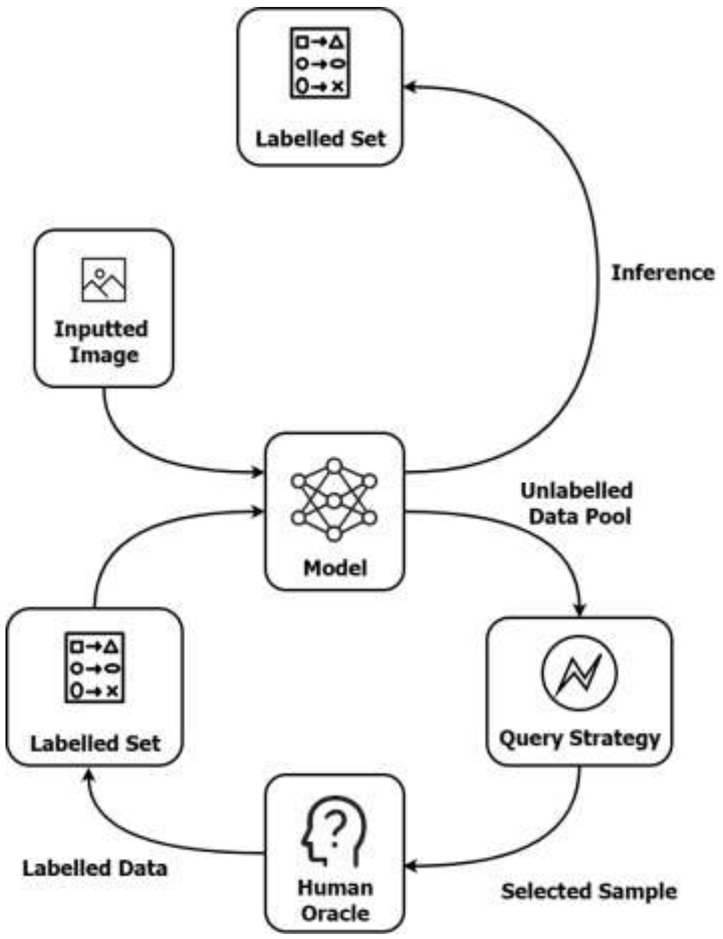

**Figure 1. General pipeline of a pool-based active learning system. Starting from a small labeled set (left), train a model (center), apply a query strategy to an unlabeled data pool (right) and send selected samples to an oracle for annotation. Labeled samples are returned to the training set and the cycle repeats.**

## 4. Technical Implementation of Active Learning

Implementing an active learning system involves several components: data handling, model training, query strategy and human annotation. Fortunately, many machine-learning libraries and frameworks support active learning workflows. In Python, for example, **modAL** is a popular open-source library that provides a modular AL framework built on scikit-learn [8]. ModAL's ActiveLearner class orchestrates the AL loop. One defines a base estimator (any scikit-learn classifier or regressor or even a neural network via wrappers) and a query strategy. A simple code snippet illustrates usage:

```
from modAL.models import ActiveLearner
from modAL.uncertainty import uncertainty_sampling
from sklearn.ensemble import RandomForestClassifier

# X_init, y_init: initial labeled features and labels
# X_pool: unlabeled feature pool
learner = ActiveLearner(estimator=RandomForestClassifier(),
query_strategy=uncertainty_sampling,
X_training=X_init, y_training=y_init)

# Query the most uncertain point
query_idx, query_instance = learner.query(X_pool)

# Suppose an oracle (human) provides its label y_new
learner.teach(X_pool[query_idx], y_new)
```

In this example, the uncertainty_sampling strategy selects the sample whose predicted class has lowest confidence. ModAL also supports other strategies (margin sampling, entropy, committee, Bayesian methods) and handles batch querying. Importantly, modAL can wrap any model. For instance, one can use a neural network by passing a Keras or PyTorch model (wrapped via an adapter) as the estimator [13]. Other active learning toolkits exist as well, such as ALiPy (Active Learning in Python) [14] which provides dozens of AL algorithms or custom scripts in research code. In practice, many teams implement their own loop using core ML frameworks.

At a lower level, the query strategies can be implemented manually. A common approach is uncertainty sampling using **model prediction probabilities**. If a classifier outputs softmax probabilities over classes, one measure of uncertainty is the highest-class probability: uncertainty = 1 – max(probabilities) [7]. The instance with the largest uncertainty is selected. Concretely:

```
probs = model.predict_proba(X_pool)        # shape (n_samples, n_classes)
uncertainty = 1.0 - np.max(probs, axis=1)  # U(x) = 1 - P(max_class|x)
query_idx = np.argmax(uncertainty)         # index of highest uncertainty
```

Other uncertainty measures include *margin sampling* (difference between top two class probabilities) and *entropy sampling* (entropy of the probability distribution) [7]. In unlabelled data pipelines, one typically computes these scores for all (or many) unlabeled points and picks the top-k.

For deep learning models (e.g. CNNs on images), one often uses frameworks like PyTorch or TensorFlow. The model can be trained on the current labeled set as usual (with standard training code). To get uncertainty estimates, common practices include using *Monte Carlo Dropout* (enable dropout at inference time to get a distribution of predictions) [15] or *ensembles of models* [16]. Libraries like modAL can also wrap PyTorch by using hybrid code. Alternatively, one could use scikit-learn estimators (e.g. SVM, random forests) inside modAL for tasks like text classification or when data is smaller.

Scaling active learning to large data is an engineering challenge. For example, scoring a large unlabeled pool on every iteration can be computationally expensive. Techniques like *batch active learning* (query a batch per round) [17], *subset selection* (score a random subset of the pool) [18] or *incremental model updates* (warm-start training) [19] are used to mitigate overhead. Some practitioners integrate active learning with cloud infrastructure: AWS, Google and others provide labelling and pipeline tools. For instance, an AWS active learning pipeline used SageMaker Ground Truth for labelling, S3 for storage and CodePipeline for orchestration [1]. In that system, new images are scored by a SageMaker model; those with low confidence trigger a Ground Truth labelling task; and a retraining job periodically incorporates the newly labeled data. [1]

In summary, a technical implementation involves: (1) training an initial model on a small labeled dataset, (2) scoring or evaluating a pool of unlabeled data according to a query criterion, (3) soliciting labels for the selected examples (via a labelling interface or tool) and (4) retraining or fine-tuning the model. Code libraries like modAL[8] and ALiPy automate much of this loop, but custom integration with data pipelines and UI/annotation tools is usually needed for production. For example, open-source platforms like Roboflow or Labelbox offer active learning workflows where new data is sent for annotation when it falls below a confidence threshold [5]. Overall, implementing active learning requires both algorithmic code and system integration to manage data flows and human-in-the-loop labelling.

## 5. Comparative Analysis of Uncertainty, Committee and Acquisition Function Strategies for Sample Selection – Iris Dataset

To better understand the practical effectiveness of different active learning strategies, we conducted a comparative analysis using the classic Iris dataset [11]. This section evaluates three widely used sample selection techniques—Uncertainty Sampling, Query-by-Committee, Acquisition Function-based Selection—by applying them to the same classification task and observing how quickly each method converges to high model accuracy with minimal labeled data. The goal is to empirically demonstrate how the choice of strategy affects learning efficiency, particularly when labeled data is scarce.

The Iris dataset is a well-known multivariate dataset introduced by Ronald A. Fisher in 1936. It consists of 150 samples, each described by four features—sepal length, sepal width, petal length and petal width—classified into one of three species of the Iris flower: Setosa, Versicolor and Virginica. Owing to its balanced structure and interpretability, the dataset is commonly used for evaluating machine learning models, particularly classification algorithms. Its small size and well-separated classes make it ideal for benchmarking active learning strategies in controlled experiments.

I initialized the ActiveLearner with 10 randomly selected rows from the dataset. Then I ran the Active Learning loops with Query Sizes of 10, 20 and 30 instances. The results are shown in Fig. 2, Fig. 3 and Fig. 4.

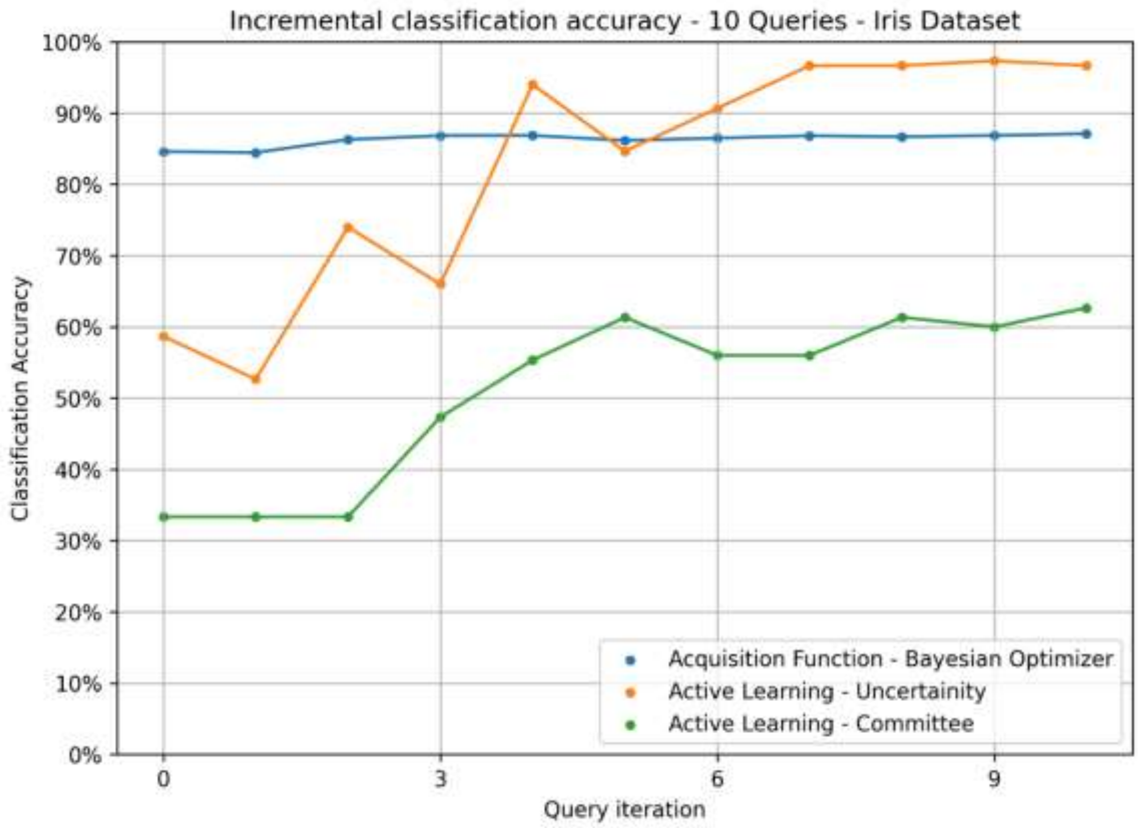

**Figure 2. Classification Accuracy changes for Acquisition Function (Blue), Uncertainity (Orange) and Committee (Green) based label selection strategies. Each point depicts the accuracy after the 'x' samples have been chosen, with a total of 10 query samples. Appendix-I**

In Figure 2, we can see that the classification accuracy improves drastically (from 58.6% to 96.2%) for Uncertainty based sampling. For the Acquisition Function based sampling, the accuracy improves only marginally (from 85% to 87.2%). Committee based sampling performs worst in this case, getting to a maximum accuracy of 63.1%. However, as we can see in Figure 3, the change in accuracy seems to follow a pattern as the number of query samples is increased.

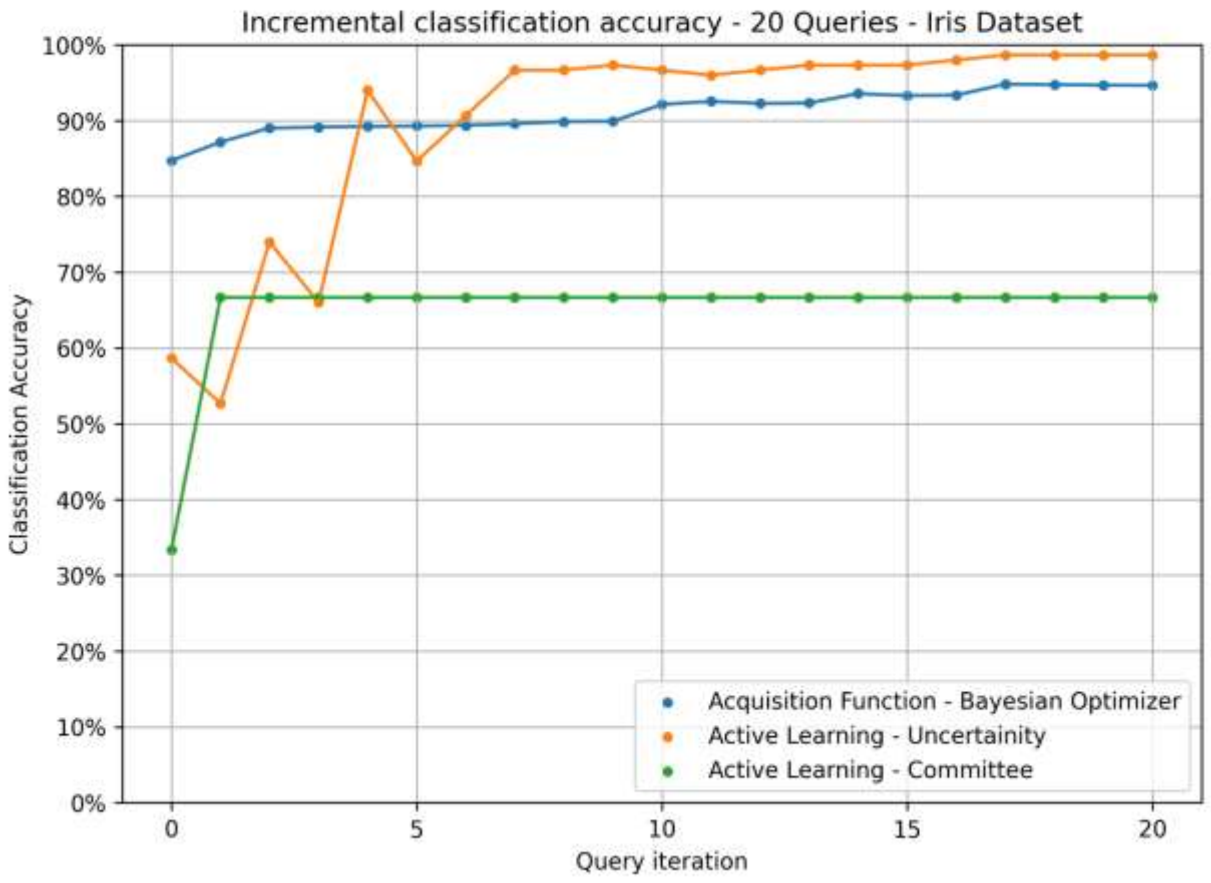

**Figure 3. Classification Accuracy changes for Acquisition Function (Blue), Uncertainity (Orange) and Committee (Green) based label selection strategies. Each point depicts the accuracy after the 'x' samples have been chosen, with a total of 20 query samples. Appendix-I**

For 20 query samples, the accuracy for Uncertainty based sampling improves slightly (97.8%) when compared to 10 query samples (96.2%). However, Acquisition Function based accuracy improves significantly (from 87.2% to 94.8%). There is no impact of increased queries on the accuracy for the Committee based sample selection.

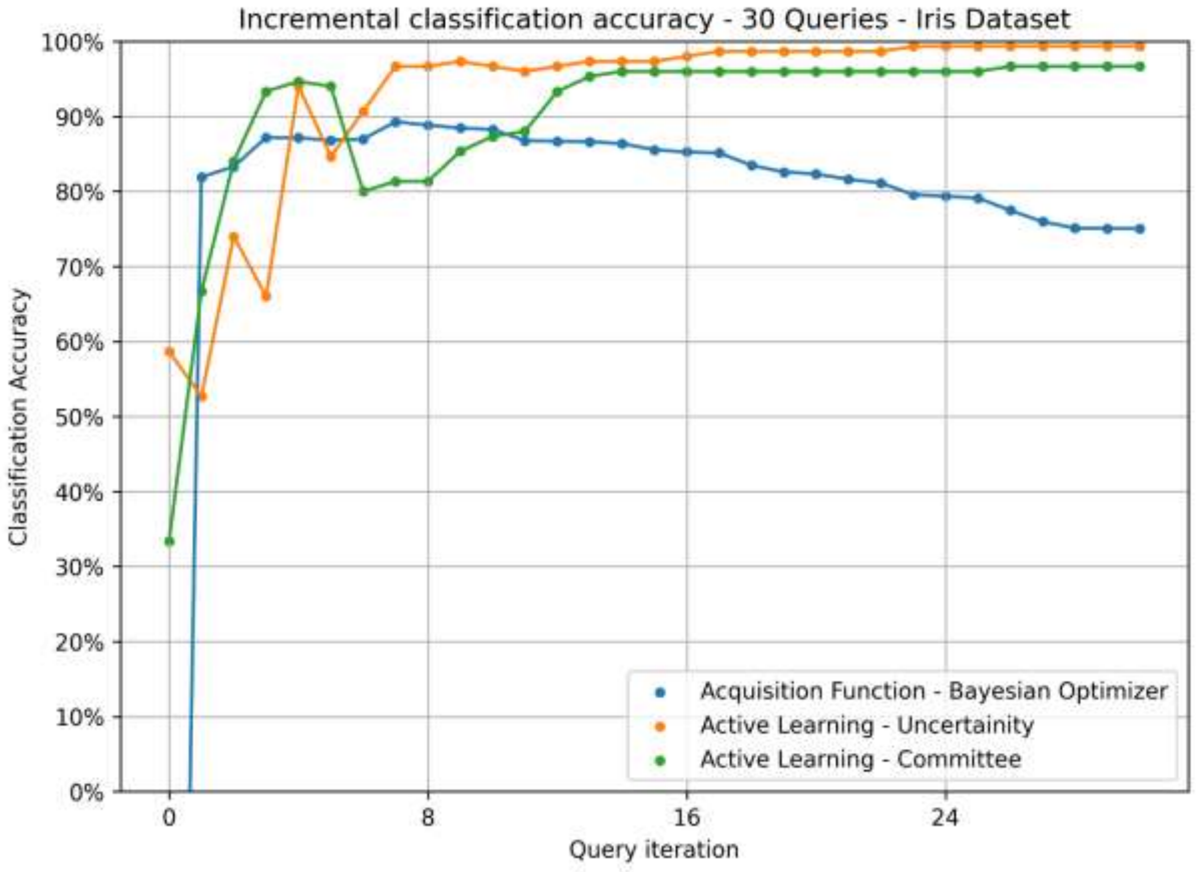

**Figure 4. Classification Accuracy changes for Acquisition Function (Blue), Uncertainity (Orange) and Committee (Green) based label selection strategies. Each point depicts the accuracy after the 'x' samples have been chosen, with a total of 30 query samples. Appendix-I**

As seen in Figure 4, for 30 sample queries, the performance of uncertainty-based selection remains highest (98.7%). However, Acquisition Function based learning performance degrades to 86%. Also, Committee based sample selection accuracy improves significantly and gets to 95.8%. This anomalous behavior of the Acquisition Function and Committee based selection can be attributed to:

1. 30 queries are a significant sample size (20% of the total size of 150)
2. The differences in the randomly selected initial sample that the model was trained on.

In conclusion, Uncertainity based sample performed best and most consistently for the classification task for the IRIS dataset. Now let us look at the result of another experiment I performed with the Titanic dataset.

## 6. Comparative Analysis of Uncertainty and Committee Strategies – Titanic Dataset

To assess the robustness of active learning strategies in real-world, noisy datasets with mixed data types, we extend our comparative analysis to the Titanic dataset. This section contrasts Uncertainty Sampling and Query-by-Committee approaches, focusing on how effectively each strategy accelerates model learning in a binary classification setting with class imbalance and feature heterogeneity. By applying these methods iteratively, we examine the rate of improvement in model accuracy relative to the number of labeled instances, providing insights into their practical utility beyond well-structured benchmark datasets. Here the committee consists of 2 learners using uncertainity based strategy and Random Forest

Classifier internally.

The Titanic dataset, made popular through Kaggle's machine learning competitions, contains passenger data from the infamous 1912 shipwreck. The task is to predict survival (binary classification: Survived or Not) based on a variety of features such as age, gender, passenger class, fare and number of family members aboard. With a mix of numerical, categorical and missing values, the Titanic dataset poses realistic data preprocessing and modeling challenges, making it a suitable candidate for testing active learning strategies in practical scenarios.

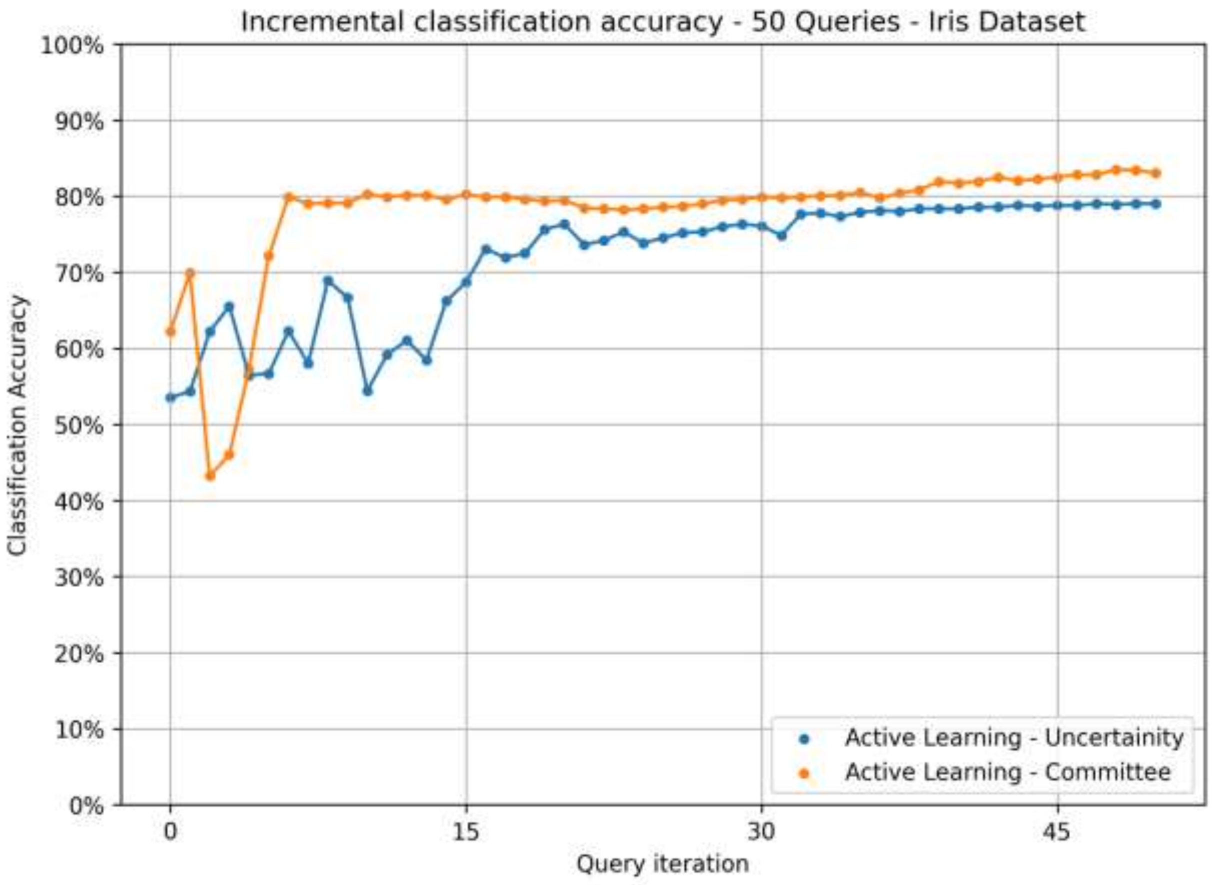

**Figure 5. Classification Accuracy changes for Uncertainity (Blue) and Committee (Orange) based label selection strategies. Each point depicts the accuracy after the 'x' samples have been chosen, with a total of 50 query samples. Appendix-II**

As seen in Figure 5, for a binary classification task with a dataset of 891 rows and query size of 50 (~5%), the accuracy of both Uncertainty and Committee based strategies is comparable. Also, from the results it is clear that the accuracy increase in the committee based strategy increased only marginally after the 11th query, as compared to the Uncertainty based sampling whose accuracy peaked at around 21st instance. The overall accuracy of Committee based model was higher (84.6%) as compared to that of Uncertainty based model (78.9%). This illustrates that for binary classification where the class sizes are skewed (549 survivors vs 342 Non-survivors), Committee based query selection works better than a single Uncertainity based model.

## 7. Example of Active Learning: Drone Footage Anomaly Detection in Power Lines

To illustrate active learning in practice, consider a (hypothetical but plausible) real-world scenario: using drone imagery to detect anomalies on electrical power lines. Utilities are increasingly using UAVs with cameras to inspect power lines for defects (e.g. broken insulators, vegetation overgrowth, corrosion). A standard approach would collect thousands of drone images and train a vision model to classify or localize faults. However, anomalous conditions are rare and varied. Labelling all possible anomalies in flight footage would be extremely laborious. This is an ideal setting for Active Learning.

**Scenario:** A power company deploys a drone along a transmission line corridor, capturing video or still frames of towers and lines under various conditions. The goal is to build a model that can detect anomalies (e.g. a broken clamp, a leaning pole, vegetation contacting the line). Suppose we start with a handful of labeled images covering a few common cases (e.g. 100 images labeled as “normal” and 50 labeled as “anomaly”). We train an initial convolutional neural network (e.g. YOLOv8, as used in recent power-line detection work [9]) on this seed set.

Now we have a large pool of unlabeled frames from subsequent drone flights. Using an active learning pipeline, we apply the current model to this pool and identify the images where the model is least certain or predicts an anomaly. For example, the model might output low-confidence detections or conflicting class probabilities on some frames. We select a batch of these uncertain frames (e.g. 100 images) and send them to expert inspectors for annotation (the “oracle”). The experts label whether each frame contains an anomaly and its type/location. These newly labeled images are added to the training set and the model is retrained or fine-tuned.

This loop repeats: each iteration the model becomes more accurate and the annotation budget is focused on the remaining difficult cases. The annotators do not waste time labelling frames that are already easy (the model is confident) or redundant; instead they concentrate on edge cases. Over several rounds, the model might require only a few hundred labeled images to reach high accuracy, whereas a passive strategy might have needed thousands.

As a concrete example, Santos *et al.* recently trained a YOLOv8 model for detecting power lines in UAV imagery, achieving over 90% mAP on visible images and over 96% on thermal images [9]. In an active learning extension of this task, one could take a pre-trained YOLO model as the base and then use active learning to specifically find and label anomalies. For instance, if the drone captures novel vegetation patterns or rare hardware (an outlier not seen in the original dataset), the model’s output probabilities would reflect its uncertainty. A simple code-level implementation might use modAL with a Keras/TensorFlow YOLO model wrapped as the estimator.

```
from modAL.models import ActiveLearner
from modAL.uncertainty import uncertainty_sampling
from ultralytics import YOLO

# Load a pre-trained YOLOv8 model (PyTorch-based)
model = YOLO('yolov8x.pt')
# pretrained weights

# Convert YOLO to an ActiveLearner using a wrapper (hypothetical)
learner = ActiveLearner(estimator=model,
                query_strategy= uncertainty_sampling,
                X_training=X_init, y_training=y_init)

# Run active learning loop on new drone frames
query_idx, query_samples = learner.query(X_unlabeled, n_instances=10)
# batch of 10

# Send these frames to human experts for annotation
labels = oracle.annotate(query_samples)
learner.teach(X_unlabeled[query_idx], labels)
```

After a few iterations, the YOLO model would have seen representative examples of different anomaly types. Uncertain-sample selection might include frames with partial occlusion, unusual angles or borderline defects. In this way, we build a robust anomaly detector with minimal labelling.

Importantly, implementing this system in practice also involves infrastructure. For example, one might store drone images in a cloud bucket and trigger the active learning loop whenever new data arrives. A typical workflow could use tools like AWS S3 (for storage), SageMaker (for training and inference) and SageMaker Ground Truth (for human labelling) as in the AWS blog example [1]. The labelling interface would present only the selected images to annotators. Each annotation round might yield only a small percentage of images needing labels (those with low confidence).

In summary, for the drone power-line use case, active learning would operate as follows:

1. **Initial Model**: Train an object detector (e.g. YOLOv8) on a small set of labeled normal/anomaly images.
2. **Inference**: Run the model on new drone footage frames. Collect model outputs and confidence scores.
3. **Query Strategy**: Use an uncertainty measure (e.g. lowest predicted probability for the detected anomaly class) to rank frames.
4. **Annotation**: Send the top-k uncertain frames to domain experts to confirm anomaly labels (and possibly localization).
5. **Update**: Add the new labels to training data and retrain the model.
6. **Repeat**: Iterate until performance plateaus or budget is exhausted.

This pipeline ensures that the human experts' time is spent labelling only the frames that most improve the model, rather than exhaustively labelling every frame of video. In practice, steps 2–4 could be automated with a script or pipeline and step 5 (retraining) could be done on a schedule or automatically triggered when new labels arrive. End-to-end, this active learning system could build a field-ready anomaly detector much faster than a purely passive approach.

## 8. Results (Time Save, Money Save, Agility)

Active learning's main benefit is efficiency: it reduces labelling time and cost while enabling quick model improvement. Quantitatively, active learning can dramatically cut the required labeled examples. In many reported cases, only a fraction of the full dataset needs annotation. For example, one industry report showed that an active learning pipeline enabled over 90% reduction in labelling cost [1]. In the AWS case study, an automotive in-cabin sensing project retrained a model with far fewer human labels

by automating the bulk of annotation and only asking humans for the hardest examples.
Time savings are also significant. Rather than waiting for a labelling project to finish before training a model (often taking weeks), active learning creates an *iterative loop*. Each active learning iteration (label a batch, retrain, redeploy) can take hours instead of days. This agility means faster time-to-deployment. In the AWS example, a job that normally took weeks of annotation was cut to hours of processing [1], thanks to automated selection of images for labelling.
Beyond raw cost and time, active learning improves model agility. If new types of anomalies appear (say a new kind of equipment anomaly on the lines), active learning can rapidly incorporate them. The model flags any unfamiliar patterns (as low confidence) and queries for labels.
Some companies have reported 20–30% improvement in accuracy for vision models after deploying active learning tools, alongside labelling savings. For example, a testimonial from a robotics company noted a 30% annotation error reduction and faster deployment using an active learning platform [10]. While such numbers vary by case, the general pattern is clear: focusing expert effort on the most informative data gets the most bang for the buck.
Overall, incorporating active learning in a production pipeline can reduce annotation needs by order-of-magnitude while still meeting model targets. Active learning often makes the difference between a stalled proof-of-concept and a deployed system.

## 9. Disadvantages and Limitations of Active Learning

Despite its advantages, active learning has limitations that while active learning reduces the number of labels needed, every queried example still must be labeled by a (often expert) annotator. In domains like medical imaging or power systems, such experts are expensive. If the task is complex, each annotation can also be time-consuming. Thus, the cost of obtaining each active learning label may be higher than that of simpler tasks. In other words, active learning shifts some burden from quantity of labels to quality of labels.
Second, each round of new labels typically requires retraining or fine-tuning the model. This can be computationally expensive, especially for large deep learning models. For example, retraining a CNN dozens of times can consume significant GPU time, offsetting some of the data savings. In practice, one can mitigate this by using incremental learning (warm-starting from the previous model) or by retraining only after a batch of new labels rather than each example.
Third, if the query strategy is poorly chosen, the model may pick uninformative or noisy examples. A classic pitfall is **sampling bias**: uncertainty sampling tends to select borderline or ambiguous examples, which may not cover the overall data distribution. Rare classes or outlier conditions might never be queried if they do not trigger high uncertainty in the current model. To mitigate this, one can use strategies like supervised contrastive active learning (SCAL) and deep feature modeling (DFM) [20].
Also, if one class (e.g. “normal” vs. “anomaly”) dominates the data, the model may become overconfident on normal samples. An uncertainty strategy may then keep querying only “hard normals” and miss anomalies. Moreover, in imbalanced settings the pool of informative samples may be skewed. Balakrishnan warns that AL can suffer poor generalization if imbalanced data is not handled carefully [2]. Real-world data is often imbalanced, so practitioners must augment uncertainty sampling with other heuristics (e.g. cluster-based sampling or always including some minority-class examples).
Another limitation is that active learning assumes a model (even if weak) exists to measure uncertainty. If the initial labeled set is extremely small or not representative, the model’s estimates may be

unreliable. In early rounds, the model might select unhelpful queries because it "doesn't know what it doesn't know." Cold-starting active learning sometimes requires seeding the model with a few examples of each class or careful initialization.

Real-world deployment issues also arise. Many active learning algorithms are tested on static datasets, but a production setting involves data pipelines, user interfaces and possibly streaming data. The Weizenbaum study notes that in practical systems, one cannot easily compare different query strategies without extra labelling and issues like unknown data distribution can complicate AL's effectiveness [3]. In other words, designing a robust active learning system for the wild is non-trivial: it must deal with concept drift, integration with annotation tooling, version control of models/data and potential feedback loops.

Finally, there is no free lunch: active learning only helps if labelling is the bottleneck. If unlabeled data is scarce or labels are cheap, traditional training may suffice. If model development time or compute is the constraint, the extra complexity of AL might not be worthwhile.

In summary, active learning's disadvantages include the ongoing need for human-in-the-loop labelling (and thus expert time) [2], increased computational overhead for repeated training, potential sampling biases (especially with imbalanced classes) [2][3] and engineering complexity in building an end-to-end loop. These limitations mean that while active learning is powerful, it must be applied judiciously and often supplemented with other strategies (semi-supervised learning, data augmentation, etc.).

## 10. Conclusion

Active learning offers a powerful paradigm for building more accurate AI models with far less labeled data. By iteratively selecting and labelling the most informative data points, active learning focuses human effort where it matters most. This yields leaner models, faster training cycles and large savings in annotation cost. As we have seen, active learning is not a single algorithm but a framework – one that incorporates strategies like uncertainty sampling, committee disagreement and diversity to query data. Architecturally, a typical system combines a machine learning model, a query engine and human annotators in a loop. Modern tools like modAL and cloud ML services can streamline implementation.

We saw that for multiclass datasets like the IRIS dataset where the data is more or less evenly distributed across the classes, uncertainity based sampling works best. However, with binary data where there is a significant skew between normal vs anomalous data, as with the Titanic Dataset, a committee of uncertainity based learners works much better and converges much faster than a single learner.

Industry examples (e.g. AWS's pipeline) suggest that annotation effort can drop by an order of magnitude, turning weeks of manual labelling into hours [1] for example like drone-based power-line inspection, when Active Learning is applied.

Nonetheless, active learning has practical limits. It hinges on human oracles, sufficient initial models and careful handling of corner cases. It introduces overhead in orchestrating the loop and can struggle with skewed data. Researchers continue to explore hybrid approaches (combining active learning with semi-supervised or self-supervised learning) to mitigate these issues.

In conclusion, as one summary notes, active learning enables "smarter models with reduced data-labelling requirements" [2]. Going forward, as AI systems increasingly operate in dynamic, data-rich environments, active learning will be a key tool for scalable, sustainable model building.

## 11. References

1. Y. Yu et al., *"Build an Active Learning Pipeline for Automatic Annotation of Images with AWS Services,"* AWS AI/ML Blog, Apr. 2024. Available: https://aws.amazon.com/blogs/machine-learning/build-an-active-learning-pipeline-for-automatic-annotation-of-images-with-aws-services/
2. S. Balakrishnan, *"Active Learning in Machine Learning: A Step Towards Smarter Models,"* IEEE Teaching Excellence Hub (blog), Dec. 2024. Available: https://teaching.ieee.org/active-learning-in-machine-learning-a-step-towards-smarter-models/
3. S. Nenno, *"Potentials and Limitations of Active Learning for the Reduction of Energy Consumption During Model Training,"* Weizenbaum Journal of the Digital Society, vol. 4, no. 1, 2024. Available: https://ojs.weizenbaum-institut.de/index.php/wjds/article/view/4_1_3/122
4. D. Li et al., *"A Survey on Deep Active Learning: Recent Advances and New Frontiers,"* in *IEEE Trans. Neural Netw. Learn. Syst.* (accepted), 2024. Available: https://ar5iv.labs.arxiv.org/html/2405.00334
5. Joseph Nelson, *"What is Active Learning? The Ultimate Guide,"* in *Roboflow Blog*, Sep. 2024. Available: https://blog.roboflow.com/what-is-active-learning/
6. Frederik Hvilshøj, *"Active Learning in Machine Learning: Guide & Strategies,"* in *Encord Blog,* Sep. 2023. Available: https://encord.com/blog/active-learning-machine-learning-guide/
7. T Danka, *"Uncertainty sampling — modAL documentation"*, in *modAL Documentation.* Available: https://modal-python.readthedocs.io/en/latest/content/query_strategies/uncertainty_sampling.html
8. T. Danka, *"modAL: A modular active learning framework for Python3,"* Doc (online), 2018. [Online]. Available: https://modal-python.readthedocs.io.
9. T. Santos et al., *"UAV Visual and Thermographic Power Line Detection Using Deep Learning,"* Sensors, vol. 24, no. 17, p. 5678, Aug. 2024. Available: https://www.mdpi.com/1424-8220/24/17/5678
10. *"Encord | Label & Curate Multimodal Data for AI,"* in *Encord Documentation*. Available at: https://docs.encord.com/platform-documentation/GettingStarted/encord-introduction
11. UCI Machine Learning et al., "Iris Species," in Kaggle. Available at: https://www.kaggle.com/datasets/uciml/iris
12. M Yasser H, "Titanic Dataset", in Kaggle. Available at: https://www.kaggle.com/datasets/yasserh/titanic-dataset
13. T. Danka, *"Keras models in modAL workflows,"* Doc (online), 2018. [Online]. Available: https://modal-python.readthedocs.io/en/latest/content/examples/Keras_integration.html
14. Tang, Ying-Peng et al., "*ALiPy: Active learning in python,*" arXiv:1901.03802 2019. Available at: https://arxiv.org/abs/1901.03802
15. H. Sun et al., "*An active learning method based on Monte Carlo dropout neural network for high-dimensional reliability analysis,",* Reliability Engineering & System Safety Vol. 262, 2025. Available at: https://www.sciencedirect.com/science/article/pii/S0951832025003709
16. Sui Q, Ghosh SK., "*Active Learning for Stacking and AdaBoost-Related Models.*" Stats. 2024; 7(1):110-137. Available at: https://doi.org/10.3390/stats7010008
17. Andreas Kirsch, "*Black-Box Batch Active Learning for Regression*", arXiv:2302.08981 [cs.LG], 2023. Available at: https://arxiv.org/abs/2302.08981

18. Andreas Kirsch, "*Advancing Deep Active Learning & Data Subset Selection: Unifying Principles with Information-Theory Intuitions*", arXiv:2401.04305 [cs.LG], 2024. Available at: https://arxiv.org/abs/2401.04305
19. Shen, M. et al., "*Step Out and Seek Around: On Warm-Start Training with Incremental Data*", arXiv:2406.04484 [cs.CV], 2024. Available at: https://arxiv.org/abs/2406.04484v1
20. Krishnan R. et al., "Mitigating Sampling Bias and Improving Robustness in Active Learning", arXiv:2109.06321 [cs.LG], 2021. Available at: https://arxiv.org/abs/2109.06321