1 Zielsetzung der Arbeit

Ziel dieser Bachelorarbeit ist es, Methoden des passiven und aktiven Lernens in der Bildklassifikation systematisch zu untersuchen und zu vergleichen, um den manuellen Aufwand bei der Generierung von Trainingsdaten signifikant zu reduzieren. Als erster Schritt wird ein balancierter Datensatz (MNIST) verwendet, um ein passives Lernverfahren zu trainieren und dessen erzielte Klassifikationsleistung als Referenzwert (Baseline) festzulegen. Darauf aufbauend werden sämtliche Auswahlstrategien (Query-Strategien) für aktives Lernen, welche durch die Python-Bibliothek modAL bereitgestellt werden, implementiert und evaluiert. Ziel ist es, mittels dieser Strategien die Referenzleistung des passiven Lernens hinsichtlich Genauigkeit (Accuracy) zu erreichen oder idealerweise zu übertreffen.

Nach erfolgreicher Evaluation auf dem balancierten Datensatz erfolgt eine Übertragung der Methodik auf einen unbalancierten, realitätsnahen Datensatz, der Dachmaterialien aus Luftbildern umfasst. Hierbei soll analysiert werden, wie sich unterschiedliche Strategien des aktiven Lernens bei ungleicher Datenverteilung bewähren und welche Ansätze sich besonders effektiv für die Optimierung der Datenaufbereitung eignen. Die Leistungsfähigkeit der Modelle wird erneut anhand ihrer Genauigkeit miteinander verglichen.

Durch die Identifikation und Implementierung besonders effizienter aktiver Lernmethoden soll langfristig der Aufwand zur manuellen Annotation und Generierung von Trainingsdaten reduziert werden. Dies ist besonders relevant für kritische Anwendungen wie die automatisierte Erkennung von Dachmaterialien in Luftbildern, da hierdurch sowohl Kosten gespart als auch eine zügigere Entwicklung leistungsfähiger und zuverlässiger Klassifikationsmodelle ermöglicht wird.

2 Theoretische Grundlagen

2.1 Wahrscheinlichkeitsberechnung bei Klassifikatoren

Support Vector Machines (SVM) SVMs sind margin-basierte Klassifikationsmodelle, die eine trennende Hyper-Ebene mit maximalem Abstand (Margin) zu den Datenpunkten suchen [**cortes1995support**]. Klassische SVMs liefern Entscheidungswerte $f(x)$, jedoch keine Wahrscheinlichkeiten.

Wahrscheinlichkeitsberechnung: Zur Erzeugung probabilistischer Vorhersagen wird oft *Platt Scaling* eingesetzt. Dabei wird eine logistische Funktion $\sigma(f(x)) = \frac{1}{1+\exp(Af(x)+B)}$ auf die SVM-Ausgabe angepasst. Die Parameter A und B werden durch Maximum-Likelihood auf einem Validierungsdatensatz bestimmt[19].

Random Forests Ein Random Forest ist ein Ensemble von Entscheidungsbäumen [1]. Jeder Baum klassifiziert unabhängig, die finale Entscheidung erfolgt durch Mehrheitsvotum.

Wahrscheinlichkeitsberechnung: Die Wahrscheinlichkeit einer Klasse k ergibt sich als Anteil der Bäume, die k vorhersagen:

$$\hat{P}(Y = k \mid x) = \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}\{\text{Baum}_t(x) = k\}$$

Diese Wahrscheinlichkeiten können kalibriert werden, z. B. über isotone Regression.

Neuronale Netze Neuronale Netze bestehen aus mehreren Schichten mit gewichteten Verbindungen [17]. Für Klassifikation wird in der Ausgabeschicht meist die *Softmax*-Funktion verwendet:

$$\sigma(z)_i = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}$$

Wahrscheinlichkeitsberechnung: Softmax erzeugt eine Wahrscheinlichkeitsverteilung über K Klassen [2]. Das Netz wird mit Kreuzentropie-Loss trainiert, was einer Maximum-Likelihood-Schätzung entspricht. Für binäre Klassifikation reduziert sich dies zur Sigmoid-Funktion.

Logistische Regression Die logistische Regression ist ein lineares Modell zur Wahrscheinlichkeitsvorhersage [6].

Wahrscheinlichkeitsberechnung: Die Wahrscheinlichkeit ergibt sich aus der Sigmoid-Funktion:

$$P(Y = 1 \mid x) = \frac{1}{1 + \exp(-w^T x - b)}$$

Für Mehrklassenprobleme wird die Softmax-Variante verwendet. Die Parameter werden über Maximum-Likelihood geschätzt.

Naive Bayes Naive Bayes basiert auf dem Bayes-Theorem mit der Annahme bedingter Unabhängigkeit der Merkmale [**zhang2004naive**].

Wahrscheinlichkeitsberechnung: Für jede Klasse wird berechnet:

$$P(C = k \mid x) = \frac{P(C = k) \prod_j P(x_j \mid C = k)}{\sum_l P(C = l) \prod_j P(x_j \mid C = l)}$$

Diese Formel ergibt sich direkt aus dem Bayes-Theorem. Wahrscheinlichkeiten sind kalibriert, aber sensitiv gegenüber Abhängigkeitsverletzungen[12].

2.2 Unterstützte Klassifikationsmodelle in modAL

Logistische Regression:

- **Klassifikatortyp:** Linearer Klassifikator, der eine logistische Funktion (Sigmoid/Softmax) verwendet, um Wahrscheinlichkeiten für Klassen zu schätzen. Dieses statistische Regressionsmodell eignet sich für binäre und multiklassige Klassifikationsaufgaben und liefert direkt kalibrierte Klassen-Wahrscheinlichkeiten.
- **Kompatibilität mit modAL:** Vollständig kompatibel mit Unsicherheits-basierten Abfragestrategien (*uncertainty sampling*), da `predict_proba` vorhanden ist und somit Metriken wie geringste Sicherheit, kleinste Margin oder maximale Entropie direkt berechnet werden können. Ebenso problemlos in *Query-by-Committee*-Szenarien einsetzbar, da mehrere logistische Regressionsmodelle als Komitee fungieren können.
- **Einschränkungen:** Keine besonderen Einschränkungen – die logistische Regression erfordert lediglich linear trennbare Merkmale für optimale Leistung. Da sie probabilistische Ausgaben liefert, erfüllt sie die Voraussetzungen aller modAL-Strategien (für Multiklassen klassisch via One-vs-Rest oder Softmax-Regression erweitert) [8].

Support Vector Machine (SVM):

- **Klassifikatortyp:** Margin-basierter Klassifikator (kernbasiert möglich), der optimale Trennhyperflächen zwischen Klassen findet. Klassisch für binäre Klassifikation entwickelt, kann aber mittels One-vs-One oder One-vs-Rest auf Mehrklassenprobleme erweitert werden.
- **Kompatibilität mit modAL:** Grundsätzlich kompatibel mit allen Abfragestrategien, sofern ein probabilistisches Ausgabeformat verfügbar ist. Für Unsicherheits-Abfragen sollte die SVM so konfiguriert sein, dass sie Wahrscheinlichkeiten liefert (in `scikit-learn` mittels `probability=True`), da Strategien wie Least Confidence oder Entropy eine `predict_proba`-Methode erfordern. Ohne probabilistische Ausgabe kann eine SVM dennoch in *Query-by-Committee* verwendet werden, indem verschiedene SVM-Modelle als Komitee auf Basis ihrer Diskriminanzfunktion/Labels divergieren.
- **Einschränkungen:** SVMs liefern von Haus aus keine Wahrscheinlichkeiten, sondern Entscheidungswerte; die Nutzung mit Unsicherheitsmaßen erfordert daher eine nachträgliche Platt-Kalibrierung für `predict_proba`. Dies kann zusätzlichen Rechenaufwand bedeuten [4].

Random-Forest-Klassifikator:

- **Klassifikatortyp:** Ensemble-Verfahren auf Basis vieler Entscheidungsbäume im Bagging-Verfahren. Der Random Forest ist ein nicht-linearer Klassifikator, der mittels Mehrheitsabstimmung der Bäume entscheidet und dabei interne Schätzungen der Klassenwahrscheinlichkeiten über die Baum-Anteile liefert.

- **Kompatibilität mit modAL:** Sehr gut kompatibel mit Unsicherheitsstrategien, da `predict_proba` standardmäßig implementiert ist (die Wahrscheinlichkeit ergibt sich aus dem Anteil der Bäume, die eine Klasse vorhersagen). Damit können Least-Confidence, Margin und Entropie direkt angewendet werden. Ebenfalls kann ein Random Forest Teil eines Komitees sein oder selbst als Komitee betrachtet werden.
- **Einschränkungen:** Keine speziellen Einschränkungen hinsichtlich modAL – der Random Forest erfüllt die Anforderungen (probabilistische Ausgabe) und ist austauschbar mit anderen Klassifikatoren. Lediglich die Trainingszeit kann mit steigender Baumzahl wachsen [1].

k-nächste-Nachbarn (kNN):

- **Klassifikatortyp:** Instanzbasierter Klassifikator, der einen neuen Datenpunkt basierend auf den Mehrheitsklassen seiner k nächsten Nachbarn im Merkmalsraum klassifiziert. Er ist ein nicht-parametrisches Verfahren und erzeugt implizit Wahrscheinlichkeiten durch relative Häufigkeiten der Nachbarn pro Klasse.
- **Kompatibilität mit modAL:** Kompatibel mit Unsicherheits-Abfragestrategien, da der kNN-Klassifikator in `scikit-learn` `predict_proba` bietet. Somit lassen sich Uncertainty-Metriken direkt berechnen. Auch in *Query-by-Committee* einsetzbar, z.B. durch Variation von k oder Gewichtungsstrategien.
- **Einschränkungen:** Keine besonderen Einschränkungen – Wahrscheinlichkeiten sind diskret (Vielfache von $1/k$), was bei kleinen k zu groben Abstufungen führen kann. Bei großen Datensätzen ggf. langsamer [5].

Neuronales Netzwerk:

- **Klassifikatortyp:** Mehrschichtige künstliche neuronale Netze (z.B. Perzeptron-Netze oder tiefe Netze) sind leistungsfähige nicht-lineare Klassifikatoren. Sie lernen komplexe Entscheidungsgrenzen durch Aktivierungsfunktionen (ReLU, Tanh) und geben mittels Softmax Wahrscheinlichkeitsverteilungen über Klassen aus.
- **Kompatibilität mit modAL:** Über `sklearn`-ähnliche Wrapper wie `KerasClassifier` oder `skorch.NeuralNetClassifier` integrierbar. Damit sind alle modAL-Abfragestrategien direkt nutzbar. Auch *Query-by-Committee* ist durch Variation der Initialisierung oder Architektur möglich.
- **Einschränkungen:** Modell muss in modAL-Struktur eingebunden sein (`fit`, `predict`, `predict_proba`). Höherer Trainingsaufwand pro Iteration [17].

2.3 Passives Lernen

(engl. *passive learning*) bezeichnet im Kontext der Bildverarbeitung das klassische Vorgehen, bei dem ein Modell auf einem fest vorgegebenen, in der Regel vorab vollständig annotierten Datensatz trainiert wird, ohne dass der Lernalgorithmus die Auswahl oder Zusammensetzung der Trainingsdaten beeinflusst. Das System lernt also „passiv“ aus den bereitgestellten Bilddaten und muss alle relevanten Merkmale eigenständig aus dieser Datenmenge extrahieren. Im Gegensatz dazu kann ein **Active-Learning**-Verfahren (aktives Lernen) aktiv entscheiden, welche neuen Datenpunkte (etwa unbeschriftete Bilder) als Nächstes zu annotieren sind, um möglichst informative Beispiele zu erhalten [9]. Passive Lernverfahren sind folglich darauf angewiesen, dass der verfügbare Datensatz hinreichend groß und repräsentativ für das zugrunde liegende Problem ist, da das Modell seine Hypothesen nicht durch gezieltes Nachfragen oder experimentelles Sammeln weiterer Daten überprüfen kann – die Güte der gelernten Zusammenhänge wird stattdessen ausschließlich mittels Auswertung der vorhandenen (bzw. separat bereitgestellten) Validierungs- und Testdaten beurteilt. In der Praxis erfordert passives Lernen daher oft einen höheren Annotierungsaufwand: Ein passiv lernendes Modell benötigt meist deutlich mehr gelabelte Trainingsbilder, um eine vergleichbare Modellgüte zu erreichen, da es auch viele redundante oder wenig informative Instanzen mitlernen muss – wohingegen ein aktiver Lernansatz durch selektive Datenwahl den Label-Aufwand reduzieren kann [24].

2.4 Active Learning im Kontext der Bildverarbeitung

Active Learning (deutsch: *aktives Lernen*) ist ein Ansatz des überwachten maschinellen Lernens, bei dem das Lernsystem selbst aktiv auswählt, welche der unbeschrifteten Daten als Nächstes von einem Experten (dem *Orakel*) annotiert werden sollen. Der Grundgedanke besteht darin, dass ein Modell mit deutlich weniger Trainingsdaten eine vergleichbare Genauigkeit erreichen kann, wenn es strategisch bestimmt, aus welchen Daten es lernt.

Durch die iterativ vom Modell angeforderten *Queries* (typischerweise einzelne noch ungelabelte Datenpunkte) soll die Informationsausbeute maximiert werden, sodass die manuellen Labeling-Kosten minimiert werden, ohne die Vorhersagegüte wesentlich zu beeinträchtigen.

Insbesondere in der Bildverarbeitung ergeben sich hier große Effizienzgewinne, da in modernen Anwendungen meist sehr viele Bilddaten verfügbar sind, deren manuelle Annotation jedoch aufwändig und teuer ist. Active Learning erlaubt es, gezielt diejenigen Bilder aus einem großen Pool unannotierter Daten auszuwählen, die für das Modell am informativsten sind – etwa weil das aktuelle Modell bei ihnen besonders unsicher ist oder weil sie repräsentative neue Muster enthalten.

Nach jeder Annotierungsrunde wird das Modell mit den neu gelabelten Bildern erneut trainiert, sodass sich sein Leistungsvermögen sukzessive verbessert. Auf diese Weise kann ein lernendes Bildverarbeitungssystem bereits mit einem

Bruchteil der sonst nötigen Trainingsbeispiele eine hohe Genauigkeit erreichen, da es sich vorrangig auf die wichtigsten und schwierigsten Fälle konzentriert.

Dies führt insgesamt zu einer deutlichen Reduzierung des Beschriftungsaufwands bei nur geringem Verlust an Modellgüte [21].

2.5 Auswahlstrategien bei Active Machine Learning

Im Laufe der Zeit wurden verschiedene strategien entwickelt um die Informationsgüte unbeschrifteter Beispiele abzuschätzen [18]. Nur die Beispiele mit der höchsten Informationsgüte werden für das Orakel zur Annotierung ausgewählt um das Machine Learning Modell effizient zu trainieren. Nachfolgend werden verschiedene Auswahlstrategien erörtert.

2.5.1 Uncertainty Sampling

Beim Uncertainty Sampling wählt das Modell Datenpunkte aus, bei denen sich das Modell am unsichersten über die korrekte Klasse ist. Uncertainty Sampling ist einfach zu implementieren. Typische Quantifizierungen der Unsicherheit sind Least Confidence, Margin Sampling und Entropy Sampling [18].

Least Confidence Hierbei wählt das Modell den Datenpunkt zur Annotierung, dessen höchstwahrscheinliche Klasse die niedrigste Wahrscheinlichkeit besitzt [23].

Beispiel: Betrachte ein Modell zur Erkennung von Dachmaterialien, welches für 100 ungelabelte Bilder die Klassenwahrscheinlichkeiten schätzen soll. Für Bild A berechnet das Modell folgende Wahrscheinlichkeiten:

$$P(\text{Ziegel}) = 0{,}33, \quad P(\text{Beton}) = 0{,}33, \quad P(\text{Schiefer}) = 0{,}34 \tag{1}$$

Hier liegt die Top-Wahrscheinlichkeit bei 34 %. Wenn das Modell für alle anderen Bilder eine höhere Top-Wahrscheinlichkeit generiert, erfüllt Bild A das *Least Confidence*-Kriterium und wird zur Annotierung durch das Orakel ausgewählt.

Die Klassenwahrscheinlichkeit kann über folgende Methoden berechnet werden:

- Softmax-Funktion
- logistische Funktionen
- Häufigkeitsstatistiken
- Kalibrierungsmethoden

Margin Sampling Bei *Margin Sampling* wird die Unsicherheit eines Modells daran gemessen, wie knapp das Modell zwischen den beiden wahrscheinlichsten Klassen unterscheidet [22].

Das Modell berechnet hierfür zunächst für jede Klasse eine Wahrscheinlichkeit (zum Beispiel über eine Softmax-Funktion oder andere Wahrscheinlichkeitsmodelle), sortiert diese anschließend nach absteigender Wahrscheinlichkeit und berechnet die *Margin* zwischen der erstplatzierten und der zweitplatzierten Klasse:

$$\text{Margin} = P(\text{Klasse}_1) - P(\text{Klasse}_2) \tag{2}$$

Die Datenpunkte mit der kleinsten Margin werden für das Orakel zur Annotation ausgewählt, weil das Modell hier kaum einen eindeutigen Favoriten hat.

Entropy Sampling Diese Strategie misst die Unsicherheit einer Vorhersage mit der Entropie der vorhergesagten Klassenverteilung [15]. Je gleichmäßiger die Verteilung der Wahrscheinlichkeiten über die Klassen, desto höher ist die Entropie. Das bedeutet, das Modell kann sich nicht auf eine Klasse festlegen und ist somit sehr unsicher. Dort lernt das Modell am meisten hinzu, weil es die Entscheidung offensichtlich kaum getroffen hat.

Beispiel: Betrachte ein Klassifikationsproblem mit drei möglichen Klassen A, B, C. Ein Modell gibt für eine Instanz folgende Wahrscheinlichkeiten aus:

$$P(A) = 0{,}33, \quad P(B) = 0{,}34, \quad P(C) = 0{,}33 \tag{3}$$

Die Wahrscheinlichkeiten sind nahezu gleichmäßig verteilt. Dies führt zu einer hohen Entropie, was darauf hindeutet, dass das Modell sehr unsicher bezüglich der richtigen Klasse ist. Ein Datenpunkt mit solch hoher Entropie wäre für Entropy Sampling besonders interessant, um durch Hinzufügen dieses unsicheren Datenpunktes zum Trainingssatz die Entscheidungskompetenz des Modells zu verbessern.

Die Entropie H berechnet sich dabei wie folgt:

$$H = -\sum_i P(i) \cdot \log\left(P(i)\right) \tag{4}$$

Konkret ergibt sich für das obige Beispiel:

$$H = -\left[0{,}33 \cdot \log(0{,}33) + 0{,}34 \cdot \log(0{,}34) + 0{,}33 \cdot \log(0{,}33)\right] \approx 1{,}0986 \tag{5}$$

Dieser Wert liegt nahe am Maximum (bei drei Klassen maximal $\log(3) \approx 1{,}0986$), was die hohe Unsicherheit verdeutlicht.

2.5.2 Committee-basiertes Sampling im Active Learning für Bildklassifikation

Ein Komitee aus mehreren Modellen wird gebildet, um informative, unbeschriftete Beispiele auszuwählen. Zunächst werden alle Modelle des Komitees auf dem aktuell verfügbaren, beschrifteten Datensatz trainiert. Anschließend wird für jedes unbeschriftete Bild die Vorhersage jedes Komitee-Modells betrachtet.

Die zentrale Idee besteht darin, diejenigen Datenpunkte auszuwählen, bei denen sich die Modelle am stärksten uneinig sind, da Uneinigkeit auf eine hohe Unsicherheit hindeutet [10].

Die Uneinigkeit kann beispielsweise anhand der Varianz oder Entropie der Vorhersagen gemessen werden:

- **Varianz-basierte Uneinigkeit:**

$$\text{Uneinigkeit}_{\text{Var}}(x) = \frac{1}{M} \sum_{m=1}^{M} (P_m(x) - \bar{P}(x))^2 \tag{6}$$

  wobei M die Anzahl der Modelle im Komitee, $P_m(x)$ die Vorhersagewahrscheinlichkeit von Modell m für den Datenpunkt x und $\bar{P}(x)$ die durchschnittliche Vorhersagewahrscheinlichkeit über alle Modelle ist.

- **Entropie-basierte Uneinigkeit:**

$$\text{Uneinigkeit}_{\text{Entropie}}(x) = -\sum_{i} \bar{P}(y_i|x) \cdot \log \bar{P}(y_i|x) \tag{7}$$

  wobei $\bar{P}(y_i|x)$ der Mittelwert der vorhergesagten Wahrscheinlichkeiten für Klasse y_i über alle Modelle ist.

2.5.3 Expected Error Reduction

Bei der *Expected Error Reduction*-Methode im Active Learning wird dasjenige unbeschriftete Beispiel aus dem Pool ausgewählt, dessen Hinzunahme – nachdem es vom Orakel korrekt beschriftet und dem Training hinzugefügt wurde – den erwarteten Fehler des Klassifikationsmodells am stärksten verringert.

Die Auswahl erfolgt formal nach folgendem Kriterium:

$$x^* = \arg\max_{x \in \text{Pool}} \mathbb{E}\left[\text{Error}(\mathcal{M}_{\text{neu}}) - \text{Error}(\mathcal{M}_{\text{alt}})\right] \tag{8}$$

Dabei bezeichnet $\mathcal{M}_{\text{neu}}$ das Modell nach Hinzufügen des neuen Punktes x und $\mathcal{M}_{\text{alt}}$ das ursprüngliche Modell.

Diese Strategie ist theoretisch gut fundiert, da sie direkt auf die Reduktion des generalisierten Fehlermaßes abzielt. Allerdings ist sie oft rechenaufwendig, weil für jeden Kandidaten das Modell neu trainiert und die Fehleränderung erneut abgeschätzt werden muss [20].

2.5.4 Bayesian Optimization

Bayesian Optimization hilft beim Active Learning, indem gezielt Bilder ausgewählt werden, die für das Modell besonders nützlich sind. Dazu nutzt man ein spezielles Hilfsmodell (ein Bayessches Surrogatmodell, z. B. einen Gaussian Process oder ein neuronales Netz), das für jedes unbeschriftete Bild abschätzt, wie unsicher sich das Modell bei der Vorhersage ist oder wie viel neue Information das Label liefern könnte.

Formal betrachtet nutzt man dafür eine Akquisitionsfunktion $\alpha(x)$, welche für jedes Bild x bewertet, wie groß der erwartete Informationsgewinn oder die Unsicherheit ist:

$$x^* = \arg\max_{x \in \text{Pool}} \alpha(x) \tag{9}$$

Typische Akquisitionsfunktionen sind beispielsweise:

- **Expected Improvement (EI)**:

$$\text{EI}(x) = \mathbb{E}\left[\max\left(0, f(x) - f(x^+)\right)\right] \tag{10}$$

  wobei $f(x^+)$ die beste bisher beobachtete Performance beschreibt.

- **Upper Confidence Bound (UCB)**:

$$\text{UCB}(x) = \mu(x) + \kappa \cdot \sigma(x) \tag{11}$$

  wobei $\mu(x)$ die geschätzte mittlere Performance und $\sigma(x)$ die Unsicherheit der Schätzung ist.

Das ausgewählte Bild wird anschließend gelabelt, wodurch das Modell schneller und mit weniger Labels lernt, gute Vorhersagen zu treffen [11].

2.5.5 Ranked Batch-Mode Sampling

Ranked Batch-Mode Sampling ist eine Active-Learning-Methode, die unbeschriftete Daten (z. B. Bilder) nach ihrem geschätzten Nutzen ordnet, um effizient mehrere gleichzeitig zur Beschriftung auszuwählen. Dabei kombiniert sie zwei Kriterien:

1. Unsicherheit des Modells bei der Vorhersage.
2. Diversität (Vielfalt) der ausgewählten Bilder.

Formal ergibt sich der Auswahlprozess durch das Sortieren aller Kandidatenbilder x nach einer Bewertungsfunktion $U(x)$, die Unsicherheit und Diversität kombiniert:

$$U(x) = \alpha \cdot \text{Unsicherheit}(x) + (1 - \alpha) \cdot \text{Diversität}(x) \tag{12}$$

Dabei ist $\alpha \in [0, 1]$ ein Gewichtungsparameter, der die Balance zwischen Unsicherheit und Diversität steuert.

Anschließend werden die top-N Bilder auf Basis dieser Rangliste ausgewählt. Dadurch spart man Zeit und Aufwand, da das Modell nicht ständig neu trainiert werden muss, und gleichzeitig wird verhindert, dass ähnliche Bilder mehrfach ausgewählt werden [3].

2.5.6 Information Density

Die *Information Density*-Methode ist ein Active-Learning-Ansatz, der bei der Auswahl unbeschrifteter Bilder zwei Kriterien kombiniert:

1. Die Unsicherheit des Modells bei der Klassifizierung eines Bildes.
2. Die Repräsentativität bzw. Dichte des Bildes in Bezug auf alle anderen unbeschrifteten Daten.

Konkret erhält jedes unklassifizierte Bild einen Wert für die Vorhersageunsicherheit (z. B. basierend auf Entropie oder Konfidenz) und einen Wert dafür, wie ähnlich dieses Bild den übrigen unbezeichneten Bildern ist (etwa gemessen als durchschnittliche Ähnlichkeit im Merkmalsraum zu den anderen Datenpunkten). Multiplikativ kombiniert man diese beiden Werte zu einer Gesamtwertung, der *Informationsdichte*:

$$\text{InformationDensity}(x) = \text{Unsicherheit}(x) \times \text{Dichte}(x) \tag{13}$$

Dadurch werden bevorzugt solche Bilder ausgewählt, die nicht nur schwer für das aktuelle Modell zu klassifizieren sind, sondern auch stellvertretend für viele andere noch unbezeichnete Bilder stehen.

Nutzen gegenüber rein unsicherheitsbasierten Verfahren:

Im Unterschied zu einfachen Unsicherheits-Strategien, die oft Ausreißer als nächste Beispiele vorschlagen, fokussiert die Informationsdichte-Methode auf informative und repräsentative Instanzen. Das bedeutet, sie vermeidet es, ihr Labeling-Budget an seltene Sonderfälle zu verschwenden. Stattdessen wählt sie Bilder aus dichten Regionen des Datenraums, was dem Klassifizierungsmodell mehr allgemeingültige Informationen liefert. In der Praxis führt dies meist zu schnellerer Verbesserung der Modellgenauigkeit mit weniger gelabelten Daten, da die ausgewählten Beispiele die Verteilung der Daten besser abdecken und somit die Generalisierungsleistung erhöhen [13].

2.6 Herausforderungen durch Klassenungleichgewicht beim Active Learning

2.6.1 Modellverzerrung

Bei stark unbalancierter Klassenverteilung lernt das Modell überwiegend die Merkmale der dominanten Klasse. Unterrepräsentierte Klassen werden vernachlässigt, wodurch das Modell verzerrt wird und neue Beispiele tendenziell der

Mehrheitsklasse zugeordnet werden. In der automatischen Erkennung von Dachmaterialien bedeutet dies beispielsweise, dass seltene Dachtypen, wie Reetdächer oder Solaranlagen, häufig fälschlicherweise als der häufigste Materialtyp klassifiziert werden[14].

2.6.2 Selektionsbias im aktiven Lernen

Viele Active-Learning-Strategien, insbesondere unsicherheitsbasiertes Sampling, berücksichtigen die Klassenverteilung nicht ausreichend. Sie neigen dazu, bevorzugt Bilder der häufig vorkommenden Klassen für Annotationen auszuwählen und seltene Klassen zu vernachlässigen. Dies führt dazu, dass der annotierte Datensatz im Active-Learning-Zyklus entweder unausgewogen bleibt oder sich sogar weiter verschlechtert, was die Lernfähigkeit und somit die Verbesserung der Erkennung von Minderheitsklassen beeinträchtigt[14].

2.6.3 Verminderte Klassifikationseffektivität

Aufgrund der Verzerrung des Modells und der unausgewogenen Stichprobenauswahl verschlechtert sich die Klassifikationsleistung für Minderheitsklassen. Während für die Mehrheitklasse oft hohe Gesamtgenauigkeiten erreicht werden, versagt das Modell regelmäßig bei der korrekten Identifikation seltener Klassen, was sich in geringen Recall- und Präzisionswerten niederschlägt. Eine wissenschaftliche Studie bestätigt, dass traditionelle Active-Learning-Methoden bei Bildklassifikationsaufgaben mit unbalancierten Datensätzen aufgrund ignorierter Verteilungsungleichgewichte deutlich an Effektivität einbüßen [14].

3 Merkmale zur Klassifikation von Dachmaterialien in Luftbildern

Typischerweise werden in Luftbildaufnahmen die folgenden Merkmalsarten herangezogen, um unterschiedliche Dachmaterialien zu erkennen:

3.1 Farbmerkmale

Charakteristische Farbwerte oder -verteilungen des Daches (z. B. rote Ziegeldächer vs. graue Metall- oder Betondächer) liefern wichtige Hinweise auf das Material[16].

3.2 Texturmerkmale

Die Oberflächenstruktur eines Daches (rau vs. glatt) und repetitive Muster (etwa Ziegel- oder Schindelmuster) werden über Textur-Features erfasst. Solche Merkmale können beispielsweise durch statistische Kennzahlen (Grauwert-Kooccurence-Matrizen, lokale Binärmuster etc.) beschrieben werden[16].

3.3 Spektrale Merkmale

Multispektrale oder hyperspektrale Luftbilder erlauben die Nutzung zusätzlicher Wellenlängenbänder (etwa Nahinfrarot), um materialspezifische Reflexionseigenschaften zu erkennen. Spektrale Signaturen und Indizes (z. B. Vegetationsindex NDVI zur Erkennung von bewachsenen Gründächern) helfen, Materialien durch ihre charakteristischen Absorptions- und Reflexionseigenschaften zu unterscheiden[16].

3.4 Geometrisch-strukturelle Merkmale

Form und Struktur der Dachflächen (z. B. Flachdach, Satteldach), Kanten und Konturen sowie das Muster von Dachaufbauten gehören ebenfalls zu den nützlichen Features. Auch räumlicher Kontext (Größe des Gebäudes, Umgebung) kann indirekt auf das Material hinweisen (etwa Industriehalle mit Blechdach vs. Wohnhaus mit Ziegeln)[16].

3.5 Merkmalsextraktion mittels CNN

Moderne Bildklassifikationsverfahren wie Convolutional Neural Networks (CNNs) können diese Merkmale direkt aus den Bildern automatisch lernen und kombinieren. Ein CNN extrahiert in frühen Schichten zumeist einfache Kanten- und Texturmerkmale, in mittleren Schichten komplexere Muster und Farbzusammenhänge und in späten Schichten hochabstrakte Merkmale, die mit Dachmaterial-Kategorien korrelieren. Auf diese Weise entfällt die manuelle Merkmalsdefinition weitgehend, und das Netzwerk lernt aus RGB- und ggf. zusätzlichen Spektralbändern selbstständig die optimalen Merkmalskombinationen für die Klassifikation [16].

Im Kontext von aktivem Lernen und automatischer Annotation werden diese Verfahren weiter verbessert: Beim aktiven Lernen wählt das Modell gezielt die informativsten oder unsicher klassifizierten Dachflächen aus, die dann vom Menschen nachannotiert werden. Dadurch erhält das CNN effizient zusätzliche Trainingsdaten gerade für schwierige Fälle, was die Erkennungsleistung steigert. Eine automatische Annotation großer Luftbilddatensätze kann z. B. durch vorhandene Geodaten (etwa Gebäude-Kataster mit bekannten Dachmaterialien) oder durch vortrainierte Modelle erfolgen, um dem Netzwerk initial Trainingsbeispiele zu liefern. Diese Kombination aus reichhaltigen Merkmalen, CNN-basiertem Merkmalslernen sowie aktivem Lernen für die iterative Datenannotation führt zu immer präziseren Ergebnissen in der Dachmaterialklassifikation [16].

4 Einfaches Active-Learning-Beispiel mit modAL

Der folgende Python-Code zeigt, wie man mit der Bibliothek `modAL` [7] ganz einfach aktives Lernen durchführt. Wir nutzen dafür den bekannten Iris-Datensatz und den Random-Forest-Klassifikator. Die verwendete *Query-Strategie* bestimmt, welche Daten als nächstes beschriftet werden sollen. In diesem Beispiel nutzen

wir `uncertainty_sampling`, was bedeutet, dass jeweils die Daten ausgewählt werden, bei denen sich das Modell am unsichersten ist. Diese Strategie lässt sich leicht durch eine andere Funktion ersetzen.

```
# Notwendige Bibliotheken importieren
import numpy as np
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from modAL.models import ActiveLearner
from modAL.uncertainty import uncertainty_sampling

# Iris-Daten laden und Start-Datensatz festlegen
X, y = load_iris(return_X_y=True)
initial_idx = np.random.choice(range(len(X)), size=5,
    replace=False)
X_initial, y_initial = X[initial_idx], y[initial_idx]

# Restlicher Pool von unlabeled Daten
X_pool = np.delete(X, initial_idx, axis=0)
y_pool = np.delete(y, initial_idx, axis=0)

\# ActiveLearner mit RandomForest und verwendeter Query-
    Strategie
learner = ActiveLearner(
estimator=RandomForestClassifier(),
query_strategy=uncertainty_sampling,
X_training=X_initial,
y_training=y_initial
)

\# 3 unsicherste Datenpunkte aus dem unlabeled Pool
    filtern
query_idx, query_instances = learner.query(X_pool,
    n_instances=3)

# (Simulierte) Annotation (hier mit bekannten Labels)
learner.teach(X=X_pool[query_idx], y=y_pool[query_idx])

# Annotierte Daten aus dem unlabeled Pool entfernen
X_pool = np.delete(X_pool, query_idx, axis=0)
y_pool = np.delete(y_pool, query_idx, axis=0)
```

Austausch der Query-Strategie: Um eine andere Query-Strategie zu verwenden, tausche einfach `uncertainty_sampling` gegen eine andere Strategie-Funktion aus, zum Beispiel `random_sampling` oder `entropy_sampling`.

Literatur

[1] Leo Breiman. In: *Machine Learning* 45.1 (2001), S. 5–32. ISSN: 0885-6125. DOI: 10.1023/a:1010933404324.

[2] John S. Bridle. „Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition“. In: *Neurocomputing*. Springer Berlin Heidelberg, 1990, S. 227–236. ISBN: 9783642761539. DOI: 10.1007/978-3-642-76153-9_28.

[3] Thiago N.C. Cardoso u. a. „Ranked batch-mode active learning“. In: *Information Sciences* 379 (Feb. 2017), S. 313–337. ISSN: 0020-0255. DOI: 10.1016/j.ins.2016.10.037.

[4] Corinna Cortes und Vladimir Vapnik. „Support-vector networks“. In: *Machine Learning* 20.3 (Sep. 1995), S. 273–297. ISSN: 1573-0565. DOI: 10.1007/bf00994018.

[5] T. Cover und P. Hart. „Nearest neighbor pattern classification“. In: *IEEE Transactions on Information Theory* 13.1 (Jan. 1967), S. 21–27. ISSN: 1557-9654. DOI: 10.1109/tit.1967.1053964.

[6] D. R. Cox. „The Regression Analysis of Binary Sequences“. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 20.2 (Juli 1958), S. 215–232. ISSN: 1467-9868. DOI: 10.1111/j.2517-6161.1958.tb00292.x.

[7] Tivadar Danka und Peter Horvath. *modAL: A modular active learning framework for Python*. 2018. DOI: 10.48550/ARXIV.1805.00979.

[8] Stephan Dreiseitl und Lucila Ohno-Machado. „Logistic regression and artificial neural network classification models: a methodology review“. In: *Journal of Biomedical Informatics* 35.5–6 (Okt. 2002), S. 352–359. ISSN: 1532-0464. DOI: 10.1016/s1532-0464(03)00034-0.

[9] Paul Fischer. *Algorithmisches Lernen*. Vieweg+Teubner Verlag, 1999. ISBN: 9783663119562. DOI: 10.1007/978-3-663-11956-2.

[10] Yoav Freund u. a. In: *Machine Learning* 28.2/3 (1997), S. 133–168. ISSN: 0885-6125. DOI: 10.1023/a:1007330508534.

[11] Yarin Gal, Riashat Islam und Zoubin Ghahramani. *Deep Bayesian Active Learning with Image Data*. 2017. DOI: 10.48550/ARXIV.1703.02910.

[12] Kimiya Gohari u. a. „A Bayesian latent class extension of naive Bayesian classifier and its application to the classification of gastric cancer patients“. In: *BMC Medical Research Methodology* 23.1 (Aug. 2023). ISSN: 1471-2288. DOI: 10.1186/s12874-023-02013-4.

[13] Yingjie Gu, Zhong Jin und Steve C. Chiu. „Active learning combining uncertainty and diversity for multi-class image classification“. In: *IET Computer Vision* 9.3 (Juni 2015), S. 400–407. ISSN: 1751-9640. DOI: 10.1049/iet-cvi.2014.0140.

[14] Qiuye Jin u. a. „Deep active learning models for imbalanced image classification“. In: *Knowledge-Based Systems* 257 (Dez. 2022), S. 109817. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2022.109817.

[15] Ajay J. Joshi, Fatih Porikli und Nikolaos Papanikolopoulos. „Multi-class active learning for image classification“. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Juni 2009. DOI: 10.1109/cvpr.2009.5206627.

[16] Jonguk Kim u. a. „CNN Algorithm for Roof Detection and Material Classification in Satellite Images“. In: *Electronics* 10.13 (Juli 2021), S. 1592. ISSN: 2079-9292. DOI: 10.3390/electronics10131592.

[17] Yann LeCun, Yoshua Bengio und Geoffrey Hinton. „Deep learning“. In: *Nature* 521.7553 (Mai 2015), S. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539.

[18] David D. Lewis und William A. Gale. „A Sequential Algorithm for Training Text Classifiers“. In: *SIGIR '94*. Springer London, 1994, S. 3–12. ISBN: 9781447120995. DOI: 10.1007/978-1-4471-2099-5_1.

[19] Hsuan-Tien Lin, Chih-Jen Lin und Ruby C. Weng. „A note on Platt's probabilistic outputs for support vector machines“. In: *Machine Learning* 68.3 (Aug. 2007), S. 267–276. ISSN: 1573-0565. DOI: 10.1007/s10994-007-5018-6.

[20] Stephen Mussmann u. a. *Active Learning with Expected Error Reduction*. 2022. DOI: 10.48550/ARXIV.2211.09283.

[21] Jens Röder. „Active Learning: New Approaches, and Industrial Applications“. In: (2013). DOI: 10.11588/HEIDOK.00014379.

[22] Tobias Scheffer, Christian Decomain und Stefan Wrobel. „Active Hidden Markov Models for Information Extraction“. In: *Advances in Intelligent Data Analysis*. Springer Berlin Heidelberg, 2001, S. 309–318. ISBN: 9783540448167. DOI: 10.1007/3-540-44816-0_31.

[23] Burr Settles und Mark Craven. „An analysis of active learning strategies for sequence labeling tasks“. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08*. EMNLP '08. Association for Computational Linguistics, 2008, S. 1070. DOI: 10.3115/1613715.1613855.

[24] Alaa Tharwat und Wolfram Schenck. „A Survey on Active Learning: State-of-the-Art, Practical Challenges and Research Directions“. In: *Mathematics* 11.4 (Feb. 2023), S. 820. ISSN: 2227-7390. DOI: 10.3390/math11040820.