

# Projektarbeit Computer Vision

Konrad Matusewicz; 594585

23.09.2024

# Contents

<b>1 Einleitung</b>	<b>3</b>
<b>2 Datensatz</b>	<b>3</b>
<b>3 Vorgehensweise</b>	<b>5</b>
<b>4 Modellvorstellung und Ergebnispräsentation</b>	<b>8</b>
4.1 VGG16 Architektur . . . . .	8
4.2 Xception Architektur . . . . .	9
4.3 Vortrainierte VGG16 Architektur + Transfer Learning . . . . .	11
4.4 Vergleich der Modelle . . . . .	12
<b>5 Fazit</b>	<b>13</b>
<b>6 KI-Promts-Verzeichnis ChatGPT</b>	<b>14</b>

# 1 Einleitung

In der heutigen Zeit gewinnen Convolutional Neural Networks zunehmend an Bedeutung, insbesondere im Bereich der Bildverarbeitung und -klassifikation.

Das Ziel dieser Projektarbeit besteht darin, den vollständigen Ablauf zur Erstellung eines Bildklassifikators zu erlernen und praktisch umzusetzen. Beginnend mit der Erstellung eines eigenen Bilddatensatzes über die Auswahl geeigneter CNN-Architekturen bis hin zur Anwendung verschiedener Trainingsstrategien wird der gesamte Prozess erprobt. Im Mittelpunkt steht die Entwicklung eines zuverlässigen Bildklassifikators, der auf einem individuell erstellten Datensatz trainiert wird. Dieser Datensatz umfasst mindestens 500 Bilder pro Klasse und wurde speziell für diese Arbeit erstellt, um eine realitätsnahe Anwendung sicherzustellen.

Ein weiterer Schwerpunkt der Arbeit liegt auf der Erforschung und dem Vergleich unterschiedlicher Modelle und Trainingsmethoden. Hierbei werden drei verschiedene CNN-Architekturen implementiert, darunter sowohl klassische Modelle wie VGG16 als auch Transfer-Learning-Ansätze. Zur Steigerung der Leistungsfähigkeit der Modelle werden zusätzlich verschiedene Data Augmentation-Techniken angewandt, um die Robustheit der Klassifikatoren zu verbessern.

Abschließend wird eine detaillierte Analyse der Trainingsprozesse durchgeführt, einschließlich der Dokumentation relevanter Hyperparameter wie Epochenanzahl, Batch Size und Lernrate. Die Modelle werden schließlich anhand von Metriken wie Genauigkeit und Verlust sowie mit Hilfe einer Confusion Matrix evaluiert, um die Klassifikationsleistung zu bewerten und die beste Trainingsstrategie zu identifizieren.

# 2 Datensatz

Der erstellte Datensatz umfasst zwei Klassen, wobei jede Klasse durch mindestens 500 Bilder dargestellt wird, um eine ausreichende Vielfalt der Motive zu gewährleisten. Die erste Klasse stellt städtische Umgebungen dar. Die Bilder dieser Kategorie wurden bewusst aus einer Vielzahl von Perspektiven und Entfernung aufgenommen, um verschiedene Aspekte städtischer Landschaften abzudecken. Dazu gehören Aufnahmen direkt von der Straße, die Details der städtischen Infrastruktur einfangen, sowie Bilder, die aus Fenstern heraus fotografiert wurden, um eine erhöhte Perspektive auf die städtische Umgebung zu bieten. Darüber hinaus wurden Aufnahmen aus größerer Distanz gemacht, um ganze Stadtbilder im Zusammenspiel mit der umliegenden Natur oder Landschaft zu dokumentieren.

Die zweite Klasse des Datensatzes fokussiert sich auf ländliche Umgebungen beziehungsweise auf Landschaften. Auch hier wurde bei der Zusammenstellung der Motive auf eine breite Vielfalt geachtet. Die Bilder beinhalten verschiedene Landschaftstypen, um die natürliche Vielfalt abzubilden. Hierzu gehören Aufnahmen von Aussichtspunkten auf Bergen, die felsige Strukturen zeigen, ebenso wie Bilder von bewachsenen Wiesen und Wäldern, die die vegetativen Aspekte der Natur hervorheben. Zusätzlich wurden Küstenlandschaften fotografiert.

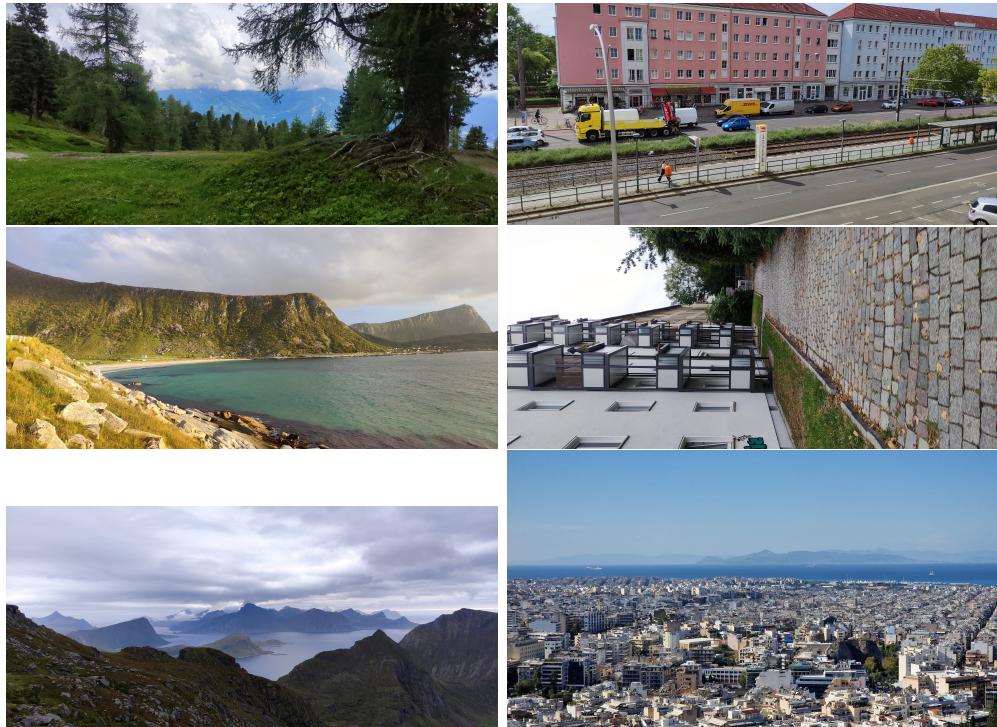


Figure 1: Ausgewählte Beispiele aus dem Datensatz

## 3 Vorgehensweise

Im Rahmen der Implementierung des Convolutional Neural Networks (CNN) werden drei verschiedene Modelle unter Anwendung unterschiedlicher Trainingsmethoden analysiert. Diese Ansätze ermöglichen eine Bewertung der Leistungsfähigkeit und Effizienz verschiedener Architekturen und Trainingsstrategien.

- VGG16 Architektur: Das erste Modell basiert auf der VGG16-Architektur, einem tiefen und bewährten CNN-Modell, das durch seine strukturierte Schichtung von Faltungs- und Pooling-Schichten bekannt ist. Diese Architektur bietet eine solide Grundlage für die Bildklassifikation und wird als Benchmark-Modell verwendet.

---

```
from tensorflow.keras.applications import VGG16

# VGG16 Architektur
modell = VGG16(weights=None, # Ohne vortrainierte Gewichte
                input_shape=(150, 150, 3), classes=1, include_top=True)
# Modell kompilieren
modell.compile(optimizer='adam', loss='binary_crossentropy',
               metrics=['accuracy'])
```

---

- Xception Architektur: Das zweite Modell verwendet die Xception-Architektur, welche auf der Inception-Architektur basiert, jedoch durch den Einsatz von Depthwise Separable Convolutions eine verbesserte Effizienz und Rechenleistung bietet. Diese Architektur erlaubt komplexere Faltungsoperationen und ist besonders für größere und anspruchsvollere Datensätze geeignet.

---

```
from tensorflow.keras.applications import Xception

# Xception Architektur
modell = Xception(weights=None, # Ohne vortrainierte Gewichte
                    input_shape=(150, 150, 3), classes=1, include_top=True)
# Modell kompilieren
modell.compile(optimizer='adam', loss='binary_crossentropy',
               metrics=['accuracy'])
```

---

- Vortrainierte VGG16 Architektur + Transfer Learning:  
Das dritte Modell nutzt ein vortrainiertes VGG16-Modell in Kombination mit Transfer Learning. Hierbei werden die initialen Schichten, die auf einem großen Datensatz trainiert wurden, beibehalten, während die letzten Schichten an den spezifischen Datensatz der vorliegenden Arbeit angepasst und weiter trainiert werden. Transfer Learning ermöglicht es, die bereits erlernten Merkmale aus den frühen Schichten zu

nutzen, was insbesondere bei kleineren Datensätzen von Vorteil sein kann.

---

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

# VGG16 mit vortrainierten Gewichten
base_model = VGG16(weights='imagenet', # Vortrainierte Gewichte verwenden
                     include_top=False,
                     input_shape=(150, 150, 3))

x = base_model.output
x = GlobalAveragePooling2D()(x) # Global Average Pooling Layer hinzufügen
x = Dense(512, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x)

# Erstelle das Modell
modell = Model(inputs=base_model.input, outputs=predictions)

# Einfrieren der Schichten des Basis-Modells
for layer in base_model.layers:
    layer.trainable = False

# Modell kompilieren
modell.compile(optimizer='adam', loss='binary_crossentropy',
               metrics=['accuracy'])
```

---

Zusätzlich wird beim Transfer Learning die Technik des Freezing Layers eingesetzt. Hierbei werden bestimmte Schichten des Modells eingefroren“, sodass deren Gewichte während des Trainings nicht verändert werden. Dies dient dazu, die in den frühen Schichten bereits erlernten allgemeinen Merkmale zu bewahren, während die Anpassungen auf den tieferen Ebenen des Modells erfolgen.

Um die Trainingsqualität zu verbessern und die Generalisierungsfähigkeit der Modelle zu steigern, wird eine Data Augmentation angewendet. Diese Techniken erweitern den Trainingsdatensatz künstlich und sorgen für eine höhere Robustheit des Modells, indem sie es auf verschiedene Bildvariationen vorbereiten. Die folgenden Data Augmentation-Methoden kommen zum Einsatz:

1. Rotation: Zufällige Drehungen der Bilder bis zu 30 Grad, um verschiedene Ausrichtungen der Motive abzudecken.
2. Breiten- und Höhenschiebung: Verschiebungen der Bilder um bis zu 20 % in horizontaler und vertikaler Richtung, um Variationen in der Position der Motive zu simulieren.

3. Scheren: Verzerrungen der Bilder um bis zu 20 %, um die Form der Objekte zu variieren und das Modell auf unterschiedliche Verzerrungen vorzubereiten.
4. Zoom: Ein Zoom von bis zu 20 %, um sowohl Nahaufnahmen als auch entfernte Perspektiven zu simulieren.
5. Horizontal Flip: Zufälliges horizontales Spiegeln, um symmetrische Varianten der Motive zu erzeugen und das Modell auf spiegelbildliche Szenarien zu trainieren.

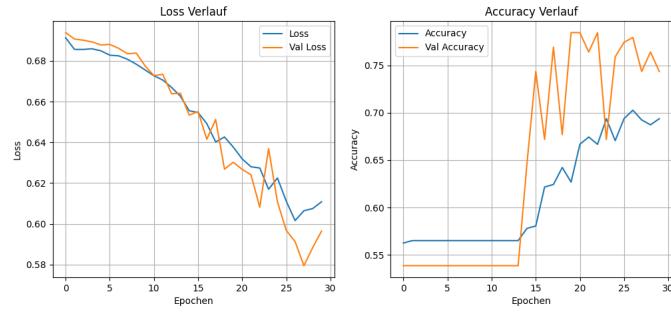
Darüber hinaus wird Early Stopping als zusätzliche Optimierungsstrategie verwendet. Diese Technik beendet das Training frühzeitig, wenn die Validierungsleistung feststeckt oder sich verschlechtert. Early Stopping hilft dabei, Überanpassung zu vermeiden, indem es verhindert, dass das Modell zu lange trainiert und dabei anfängt, sich zu stark an den Trainingsdatensatz anzupassen, ohne auf neuen Daten gut abzuleiten.

## 4 Modellvorstellung und Ergebnispräsentation

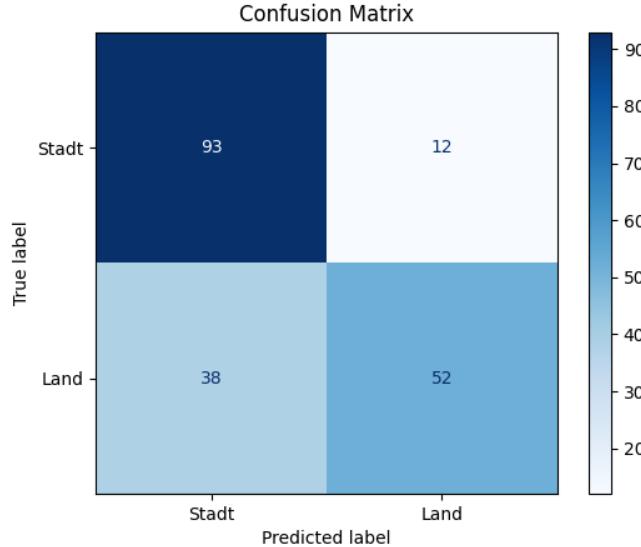
In diesem Kapitel werden die drei entwickelten Modelle detailliert beschrieben und die jeweiligen erzielten Ergebnisse präsentiert. Jedes Modell wurde mit einer spezifischen Architektur und Trainingsmethode implementiert, um die Leistung in der Bildklassifikation zu maximieren. Darüber hinaus werden die Modelle auf Grundlage ihrer Klassifikationsgenauigkeit und Validierungsleistung verglichen.

### 4.1 VGG16 Architektur

Das erste Modell basiert auf der klassischen VGG16-Architektur. Dieses tiefe CNN-Modell zeichnet sich durch eine feste Schichtenstruktur aus und bestehend aus aufeinanderfolgenden Faltungs- und Pooling-Schichten. VGG16 wurde ohne vortrainierte Gewichte implementiert, um das Modell vollständig auf dem eigenen Datensatz zu trainieren.



(a) Trainingsverlauf



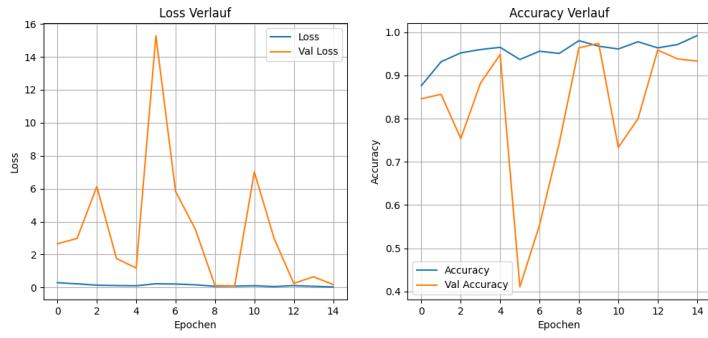
(b) Confusion-Matrix

Ergebnisse:

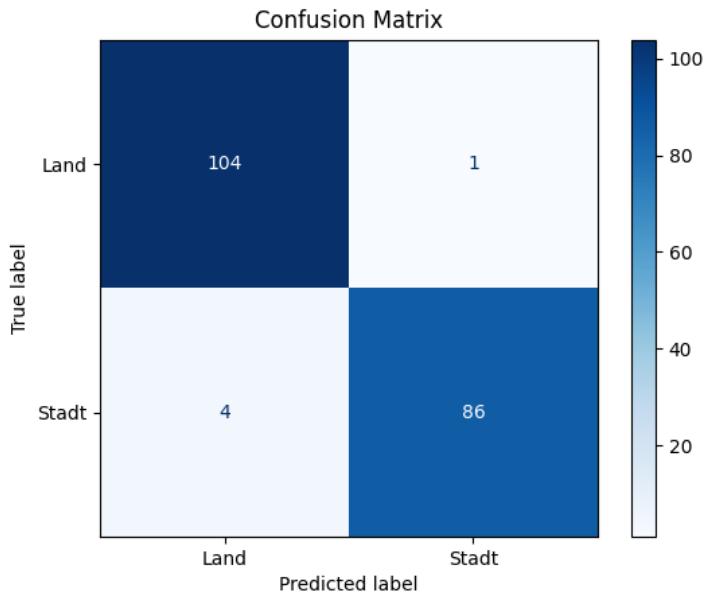
- Trainingsgenauigkeit: Das Modell erreichte nach einer bestimmten Anzahl von Epochen eine hohe Trainingsgenauigkeit, was auf eine gute Anpassung an den Trainingsdatensatz hinweist.
- Validierungsgenauigkeit: Die Validierungsgenauigkeit blieb etwas hinter der Trainingsgenauigkeit zurück, was auf Überanpassung hindeuten könnte.
- Verlustfunktion: Die Verlustkurve zeigte ein typisches Verhalten, wobei der Verlust anfangs stark abnahm und sich im weiteren Verlauf stabilisierte.
- Confusion Matrix: Die Confusion Matrix zeigte eine solide Leistung bei der Unterscheidung der beiden Klassen, jedoch war eine leichte Verwechslung zwischen ähnlichen Bildmotiven erkennbar.

## 4.2 Xception Architektur

Das zweite Modell nutzt die Xception-Architektur, die durch ihre tiefere und effizientere Struktur bekannt ist. Diese Architektur verwendet Depthwise Separable Convolutions, was sowohl die Rechenleistung als auch die Modellgenauigkeit bei Bildklassifikationen verbessert.



(a) Trainingsverlauf



(b) Confusion-Matrix

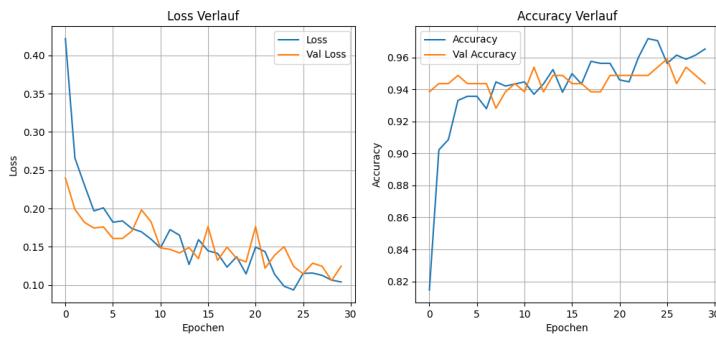
Ergebnisse:

- Trainingsgenauigkeit: Die XCception-Architektur zeigte eine schnellere Konvergenz im Vergleich zur VGG16-Architektur, wobei die Trainingsgenauigkeit nach weniger Epochen ein hohes Niveau erreichte.
- Validierungsgenauigkeit: Die Validierungsgenauigkeit dieses Modells war höher als bei VGG16, was auf eine bessere Generalisierungsfähigkeit hinweist.
- Verlustfunktion: Der Trainingsverlust bleibt über alle Epochen hinweg relativ konstant und nahe bei 0. Dies bedeutet, dass das Modell auf den Trainingsdaten gut funktioniert. Die Validierungskurve weist jedoch starke Schwankungen auf, dies lässt sich ebenfalls auf overfitting zurückführen.

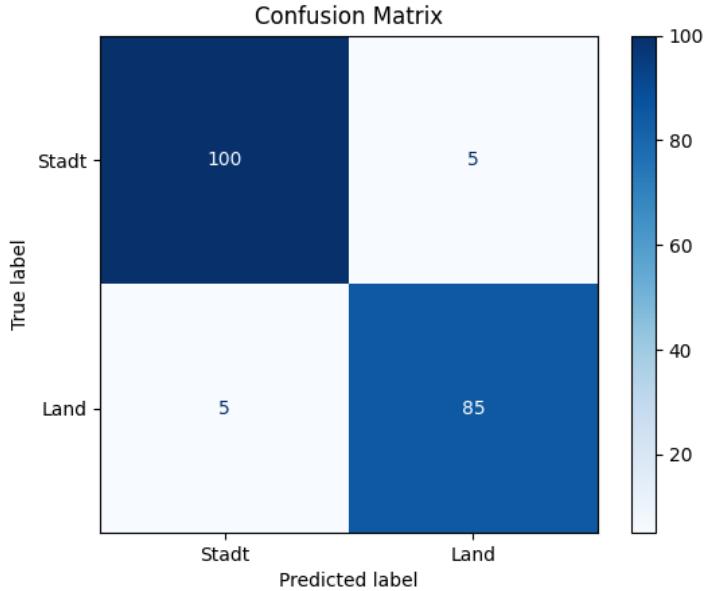
- Confusion Matrix: Die Verwechslungen zwischen den Klassen waren deutlich geringer als bei VGG16, was auf eine höhere Präzision bei der Klassifizierung hindeutet.

### 4.3 Vortrainierte VGG16 Architektur + Transfer Learning

Das dritte Modell basiert auf einem vortrainierten VGG16-Modell, das mittels Transfer Learning angepasst wurde. Hierbei wurden die frühen Schichten des Netzwerks eingefroren, um die bereits erlernten Merkmale zu bewahren, während die tieferen Schichten für die spezifische Klassifikationsaufgabe weiter trainiert wurden.



(a) Trainingsverlauf



(b) Confusion-Matrix

Ergebnisse:

- Trainingsgenauigkeit: Das vortrainierte Modell erreichte schnell eine hohe Trainingsgenauigkeit, da die frühen Schichten bereits gut auf allgemeine Bildmerkmale trainiert waren.
- Validierungsgenauigkeit: Die Validierungsgenauigkeit war im Vergleich zu den anderen beiden Modellen am höchsten. Dies deutet darauf hin, dass Transfer Learning besonders vorteilhaft für kleinere Datensätze ist, da das Modell von den bereits erlernten Merkmalen profitiert.
- Verlustfunktion: Der Verlust des Modells sank stark in den ersten Epochen und stabilisierte sich früh, was auf eine effiziente Nutzung der vortrainierten Gewichte zurückzuführen ist.
- Confusion Matrix: Die Confusion Matrix zeigte eine gute Trennung zwischen den beiden Klassen, wobei nur minimale Verwechslungen auftraten.

#### 4.4 Vergleich der Modelle

- Trainings- und Validierungsgenauigkeit: Das vortrainierte VGG16-Modell mit Transfer Learning lieferte die besten Ergebnisse hinsichtlich der Validierungsgenauigkeit, gefolgt von Xception. Das "from scratch" trainierte VGG16-Modell zeigte die niedrigste Validierungsgenauigkeit, was auf Überanpassung hindeutet.
- Verlustkurven: Alle Modelle zeigten einen typischen Verlauf der Verlustkurve, jedoch stabilisierten sich die Verlustfunktionen der vortrainierten VGG16-Modells schneller.
- Data Augmentation: Alle Modelle profitierten deutlich von den angewandten Data Augmentation-Techniken, was sich in einer erhöhten Robustheit und einer besseren Generalisierung auf den Validierungsdatensatz zeigte.

Insgesamt führte die Kombination aus fortschrittlichen Architekturen und Transfer Learning, unterstützt durch umfangreiche Data Augmentation, zu einer erkennbaren Leistungssteigerung der Modelle.

## 5 Fazit

Im Rahmen dieser Projektarbeit wurde erfolgreich der gesamte Ablauf zur Erstellung eines Bildklassifikators mittels Convolutional Neural Networks umgesetzt. Beginnend mit der Erstellung eines eigenen Datensatzes, über die Implementierung verschiedener Modellarchitekturen bis hin zur Anwendung mehrerer Trainingsstrategien, wurden wichtige Erkenntnisse über die Leistungsfähigkeit und die Herausforderungen bei der Erstellung von CNNs gewonnen.

Die durchgeführten Experimente mit den drei Modellen – VGG16, Xception und einem vortrainierten VGG16-Modell mit Transfer Learning – verdeutlichen, dass moderne Architekturansätze wie Xception sowie Transfer Learning, besonders bei kleineren Datensätzen, deutliche Vorteile bieten. Während das VGG16-Modell ohne Vortraining ordentliche Ergebnisse lieferte, zeigte sich durch den Einsatz vortrainierter Netzwerke eine erheblich gesteigerte Generalisierungsfähigkeit. Dies beweist, dass Transfer Learning eine effektive Methode ist, um vorhandene und bereits erlernte Merkmale effizient zu nutzen und die Trainingsdauer zu verkürzen.

Durch den gezielten Einsatz von Data Augmentation konnte eine künstliche Erweiterung des Datensatzes erreicht werden, was zu robusteren Modellen führte und die Überanpassung deutlich reduzierte. Ergänzend half die Early Stopping-Strategie, das Training zu optimieren und overfitting zu vermeiden, indem das Training gestoppt wurde, sobald sich die Leistung nicht mehr verbesserte.

Insgesamt konnte gezeigt werden, dass die sorgfältige Auswahl der Modellarchitektur, der Trainingsstrategien sowie der Optimierungsverfahren entscheidend für die Leistungsfähigkeit eines CNN-Modells sind. Die Kombination aus modernen Architekturen, Transfer Learning und umfassenden Data Augmentation-Techniken erwies sich als besonders effektiv, um robuste Bildklassifikatoren zu entwickeln.

## 6 KI-Promts-Verzeichnis ChatGPT

- Datenvorbereitung und Laden

Prompt: "Schreibe eine Funktion, die Bilddateien aus einem Verzeichnis lädt und ihnen basierend auf den Ordnernamen Labels zuweist."

Resultat: Funktion zum Laden und Zuweisen von Labels aus Bilddateien, wobei Labels aus den Verzeichnisnamen abgeleitet werden.

- Data Augmentation

Prompt: "Erstelle eine Data Augmentation, die verschiedene Bildtransformationen wie Rotation, Zoom und horizontales Flippen anwendet."

Resultat: Verwendung von ImageDataGenerator zur Durchführung von Augmentationen.

- VGG16 Modell mit Transfer Learning

Prompt: "Implementiere ein VGG16 Modell mit Transfer Learning, das auf einem vortrainierten Modell auf ImageNet basiert. Füge eine GlobalAveragePooling2D-Schicht und eine Dense-Schicht hinzu."

Resultat: VGG16 Modell mit eingefrorenen Schichten des vortrainierten Modells, globale Pooling-Schicht und Dense Layer zur Klassifizierung in 2 Klassen.

- Modellevaluation

Prompt: "Wie kann ich eine Confusion Matrix anzeigen, um die Vorhersagen meines Modells einzuschätzen?"

Resultat: Anzeige der Confusion Matrix mit ConfusionMatrixDisplay zur Evaluierung der Modellleistung.

- Live-Video-Test

Prompt: "Schreibe einen Code, um das trainierte Modell auf Live-Videodaten anzuwenden und die Vorhersagen auf dem Bildschirm anzulegen."

Resultat: Funktion, die die Vorhersagen des Modells auf Live-Video-Frames anzeigt.

- Erklärung von Val Loss

Prompt: "Was ist Val Loss und warum ist ein niedriger Wert besser?"

Resultat: Erklärung des Validierungsverlusts und dessen Bedeutung für die Modellbewertung.

- Overfitting beheben

Prompt: "Welche Strategien kann ich verwenden, um Overfitting zu vermeiden?"

Resultat: Strategien wie Early Stopping, Regularisierung, Dropout und Data Augmentation zur Vermeidung von Overfitting.

- Wissenschaftliche Umschreibung

Prompt: "Schreibe den folgenden Text wissenschaftlich um: "Der erstellte Datensatz umfasst 2 Klassen je mindestens 500 Bilder. Die erste Klasse ist die Stadt. Die Bilder wurden aus verschiedenen Perspektiven aufgenommen. Sowohl direkt von der Straße aus, aus dem Fenster aber auch aus größerer Entfernung, um eine Stadt innerhalb einer Landschaft zu fotografieren. Die zweite Klasse ist das Land oder besser zutreffend Landschaft. Hierbei wurde ebenfalls darauf geachtet, unterschiedliche Motive wie Strände, felsige Berglandschaften oder stark grün bewachsene Motive zu wählen."

- KI-Verzeichnis

Prompt: "Erstelle mir ein komplettes KI-Promts-Verzeichnis vom gesamten Chat"

Resultat: KI-Prompt-Verzeichnis