

Sneezy

September 7, 2024

1 Sneezy defeating Google Recaptcha

1.1 Task

Thema: Eine interessante und lehrreiche Datenanalyse auf einem von Ihnen wählbaren Datenset

Einschränkung: Keines der “klassischen” Datensets aus scikit-learn oder Keras. Die Arbeit soll Ihre Kompetenzen im Bereich Maschinelles Lernen demonstrieren. Die Arbeit soll die Bereiche “Domain-verständnis”, “Datenvorverarbeitung”, “Analyse” und “Visualisierung” abdecken. Sie können sich an anderen Arbeiten orientieren, müssen das Gelernte dann aber auf Ihren gewählten Analysegegenstand übertragen. Verwendete Quellen müssen im Notebook angegeben werden. Format: Ein vollständiges und in sich abgeschlossenes Jupyter Notebook

Das vollständig ausgeführte Jupyter Notebook ist zusätzlich auch als PDF-Datei einzureichen. Falls die analysierten Daten zu umfangreich sind um sie mitabzugeben, reicht ein Link auf das Datenset.
Gruppengröße: 4 Personen (in Sonderfällen 3 Personen)

Bearbeitungszeitraum: **16.08 - 08.09.2023**

23.08: Einreichung einer Projektskizze (ca. eine DIN A4-Seite): untersuchte Daten, gewählte Fragestellung, geplantes Vorgehen, Aufgabenverteilung in der Gruppe

30.08: Abgabe eines Zwischenstands (lauffähiges Jupyter Notebook) und eines Zwischenberichts (ca. eine DIN A4-Seite): erreichter Stand, aufgetretene Herausforderungen, begründete Abweichungen von der Projektskizze

08.09: Abgabe der finalen Version (vollständiges Jupyter Notebook + Erklärung) Erklärung: Unterschriebene Eigenständigkeitserklärung + Aufschlüsselung der Arbeitsaufteilung innerhalb der Gruppe (Hauptverantwortlichkeiten für Bestandteile + individueller Beitrag in Prozent der Gesamtleistung)

Arbeitsumfang: 40 - 50 Arbeitsstunden pro Person

Bewertungskriterien laut Masterhausarbeitsvorlage:

Gliederung der Arbeit / Aufbau und Darstellung der Problemstellung / Systematik / Struktur (“roter Faden”) Wissenschaftlichkeit / Inhaltliche Vollständigkeit und Richtigkeit / Themenrelevanz / Quellenarbeit / Eigenleistung Klarheit der Darstellung & Stringenz der Argumentationskette / formale Korrektheit / Rechtschreibung / Schreibstil Zielpublikum: Studierende Ihres Studiengangs

Fokus: Demonstration Ihrer Kompetenzen + Wissensvermittlung (das konkrete Analyseergebnis ist nachrangig)

1.2 Projektskizze

1.2.1 Google reCAPTCHA V2

Hintergrund und Kontext Google reCAPTCHA ist ein Sicherheitswerkzeug, das entwickelt wurde, um zwischen menschlichen Benutzern und Bots zu unterscheiden und so Missbrauch und Cyberangriffe zu verhindern. Die Version, reCAPTCHA V2, stellt Benutzern Herausforderungen wie die Identifizierung von Objekten in Bildern. Das Umgehen dieser Sicherheitsmaßnahme mittels maschinellen Lernens und tiefen neuronalen Netzen ist sowohl technisch als auch wissenschaftlich interessant, da es die Leistungsfähigkeit dieser Modelle testet und gleichzeitig zur Verbesserung der Sicherheit von CAPTCHA-Systemen beitragen kann.

Fragestellungen und Ziele Zu welcher Genauigkeit können derzeit Modelle mittels Verfahren des maschinellen Lernens optimiert werden, um in der Anwendung Google's Recaptcha V2 Bilder korrekt zu klassifizieren?

- Welche Veränderung der Modellgüte kann mit aktuellen Methoden der Vorverarbeitung und der Datenaugmentierung aus Forschung und Praxis erzielt werden?
- Was sind die neuesten Entwicklungen (State-of-the-Art) in der Bildverarbeitung mit maschinellem Lernen, insbesondere bei der Verwendung von tiefen neuronalen Netzen wie Inceptionv3?
- Welche in der Forschung bestehenden Metriken zur Klassifikation eignen sich zur Lösung des oben beschriebenen Anwendungsfalls?

Geplantes Vorgehen und Aufgabenverteilung

- Explorative Datenanalyse (EDA): Untersuchung der Daten durch Visualisierungen und deskriptive Statistiken, um erste Einblicke zu gewinnen. (Hauptverantwortlich: Rares, Niklas)
- Datenvorbereitung: Erstellen eines geeigneten Datensatzes durch Resizing, Resampling und extrahieren von Labels aus der Ordnerstruktur (Paarprogrammierung)
- Modellierung: Auswahl und Anwendung geeigneter statistischer Modelle oder Algorithmen zur Beantwortung der Fragestellung. (Hauptverantwortlich: Leon)
- Ergebnisse und Interpretation: Analyse der Ergebnisse der Modelle, Interpretation der Befunde und Vergleich mit bestehenden Theorien. (Paarprogrammierung)
- Berichterstattung: Erstellung eines detaillierten Berichts, der die Ergebnisse zusammenfasst und Empfehlungen basierend auf den Befunden gibt. (Paarprogrammierung)

Das Projekt wird im Google Colab1 entwickelt.

Literatur und Quellen Dataset: <https://www.kaggle.com/datasets/cry2003/google-recaptcha-v2-images>

Notebook – InceptionV3: <https://www.kaggle.com/code/ahmedhossam666/google-recaptchaca>

ResNet Paper: <https://arxiv.org/abs/1512.03385>

Google RecaptchaV2: <https://developers.google.com/recaptcha/docs/display>

1.3 Experimentparameter

Datenparameter (2 Ausprägungen) - Methoden zur Qualitätsverbesserung von Bilddatensätzen - Data Augmentation (Ohne, Mit (0.05, 0.05, True)) - Zoom (Begründen dass dadurch relevante Inhalte fehlen können (Ampel)) - Kontrast - 0.05 / 0.025 (Mountain) - Helligkeit - 0.05 / 0.025 (Mountain) - horizontale Spiegelung - horizontale Spiegelung

- Datenbalancierung (Ohne, Mit)
 - Erzeugung eines balancierten Datensatzes mittels Datenaugmentierung
- Art der Bereitstellung
 - Stapelgröße (batch_size)
 - * 64

Trainingsparameter (3 Ausprägungen) - Modellhyperparameter - Optimierungsalgorithmus - adam - Verlustmetrik - kategorische Kreuzentropie (categorical_crossentropy) - Lernrate - 0.001

- Verwendete neuronale Netzarchitekturen
 - ResNet50V2
 - InceptionV3
 - LeNet 5 Architecture (Eigenentwicklung anhand der Literatur)
- Untersuchte Metriken (werden immer alle betrachtet)
 - Accuracy
 - f1-score
 - Precision
 - Recall

Insgesamt ergeben sich $((2 * 2) - 1) * 3 = 9$ zu untersuchende Kombinationen

1.4 Setup

In der Entwicklung dieser Arbeit wird die Programmiersprache Python in ihrer Version 3.10.14 verwendet.

Alle benötigten zusätzlichen Bibliotheken können den je nach Betriebssystem unterschiedlichen Textdateien im Ordner `setup` entnommen werden. Die Installation der Bibliotheken kann z.B. in einer virtuellen Umgebung durch `conda` oder `venv` mittels dem Befehl `pip install setup/windows/requirements.txt`, bzw. `pip install setup/mac/requirements.txt` durchgeführt werden.

1.5 0. Unterstützende Funktionen

Dieses Kapitel dient der Definition von Hilfsfunktionen.

Dabei wird ein Objekt des Logging Modules genutzt, um zusätzliche Informationen auszugeben und unterschiedliche Detailgrade (info, debug) zu ermöglichen.

```
[1]: """ This module defines the logging component."""
import logging

def create_logger(log_level: str, logger_name: str = "custom_logger"):
    """Create a logging based on logger.
```

```

Args:
    log_level (str): Kind of logging
    logger_name (str, optional): Name of logger

Returns:
    logger: returns logger
"""

logger = logging.getLogger(logger_name)
logger.setLevel(logging.DEBUG)

if logger.hasHandlers():
    logger.handlers.clear()

console_handler = logging.StreamHandler()
if log_level == "DEBUG":
    console_handler.setLevel(logging.DEBUG)
elif log_level == "INFO":
    console_handler.setLevel(logging.INFO)
elif log_level == "WARNING":
    console_handler.setLevel(logging.WARNING)
elif log_level == "ERROR":
    console_handler.setLevel(logging.ERROR)
else:
    raise ValueError("Invalid log level provided")

formatter = logging.Formatter(
    "%(asctime)s - %(levelname)s - %(message)s",
    datefmt="%Y-%m-%d %H-%M-%S",
)
console_handler.setFormatter(formatter)

logger.addHandler(console_handler)

return logger

```

```
[2]: logger = create_logger(
    log_level="INFO",
    logger_name=__name__,
)
```

1.6 1. Laden der Daten

In der Phase Datenbereitstellung werden aus den Ursprungsdatenquelle die Daten ausgewählt, die für den Analyseprozess notwendig sind.

Wichtige Aufgaben sind dabei die Datenaufbereitung und als begleitende Aufgabe das Datenmanagement sowie die explorative Datenanalyse (vgl. Schulz et al. 2022, S. 36).

Google Drive in Google Colab-Notebook einbinden

```
[3]: # from google.colab import drive  
# drive.mount('/content/drive')
```

Erstelle einen Ordner auf deinem lokalen Laufwerk und navigiere hinein

```
[4]: # import os  
# file_path = '/content/drive/MyDrive/MADS2400'  
# if os.path.isdir(file_path):  
#     %cd /content/drive/MyDrive/MADS2400  
# else:  
#     %mkdir /content/drive/MyDrive/MADS2400  
#     %cd /content/drive/MyDrive/MADS2400
```

Durch die Nutzung eines Tokens wird der Datensatz lokal, oder auf dem Google Drive kopiert.

Dabei ist es bewährte Praxis, Tokens und andere sensitive Informationen in einer environment datei (.env) zu speichern.

Folgend wird demnach eine .env Datei erwartet, in welcher der Token als Variable git_fine_grained_token gespeichert ist.

```
[5]: %%capture  
!pip install python-dotenv
```

```
[6]: import os  
from dotenv import load_dotenv  
load_dotenv()  
  
load_path = 'Google-Recaptcha-V2-Images'  
# TODO Do not forget about the token!!!  
if not os.path.isdir('/content/drive/MyDrive/MADS2400/  
    ↪Google-Recaptcha-V2-Images') or not os.path.isdir(load_path):  
    git_fine_grained_token = os.environ["git_fine_grained_token"]  
    username = 'RaresMihai11'  
    repository = 'Google-Recaptcha-V2-Images'  
    !git clone https://{}@github.com/{}/{}  
else:  
    logger.info("Datenrepository wurde bereits geladen")
```

fatal: destination path 'Google-Recaptcha-V2-Images' already exists and is not an empty directory.

1.7 2. Explorative Datenanalyse

Nachdem die Daten nun innerhalb dieses Notebooks verfügbar gemacht wurden, empfehlen Schulz et al. eine explorative Datenanalyse (vgl. Schulz et al. 2022, S. 43). Dabei gilt es, die Menge und

Qualität der vorliegenden Daten zu prüfen sowie eine Vielzahl von Hypothesen zu untersuchen. Diese müssen allerdings nicht mit einem hohen Detaillierungsgrad dokumentiert werden (vgl. Schulz et al. 2022, S. 43). Am Ende dieses Kapitels wird ein Zwischenfazit gezogen, um zu klären ob weitere Datenaufbereitungsschritte notwendig sind.

1.7.1 2.1 Verzeichnisstruktur und Dateianzahl

Im ersten Schritt wird die Verzeichnisstruktur und die Anzahl der Bilder genauer untersucht, um eine Übersicht über die vorliegenden Bilder in den verschiedenen Klassen zu erhalten. Die Verwendung von os ermöglicht eine systematische Durchsuchung der Verzeichnisse, während PIL die Verarbeitung der Bilddaten sicherstellt, um sowohl die Anzahl als auch die Bildformate zu prüfen. Diese Herangehensweise wird häufig empfohlen, um eine vollständige Datenanalyse durchzuführen und potenzielle Datenqualitätsprobleme frühzeitig zu identifizieren (vgl. Müller und Guido 2017; vgl. Provost und Fawcett 2013).

```
[7]: import os
from PIL import Image

main_dir = './Google-Recaptcha-V2-Images/'
folders = ["Bicycle", "Bridge", "Bus", "Car", "Chimney", "Crosswalk",
           "Hydrant", "Motorcycle", "Mountain", "Other", "Palm", "Stair", ↴
           "TLight"]

folder_image_data = {}

for folder in folders:

    folder_path = os.path.join(main_dir, folder)

    if os.path.isdir(folder_path):
        image_files = []

        for file_name in os.listdir(folder_path):
            if file_name.lower().endswith(('png', 'jpg', 'jpeg')):
                file_path = os.path.join(folder_path, file_name)

                with Image.open(file_path) as img:
                    img_format = img.format
                    image_files.append({
                        'image': img.copy(),
                        'format': img_format
                    })

    folder_image_data[folder] = {
        "count": len(image_files),
        "images": image_files
    }
```

```
logger.debug(f'{folder}: {len(image_files)} Bilder')
```

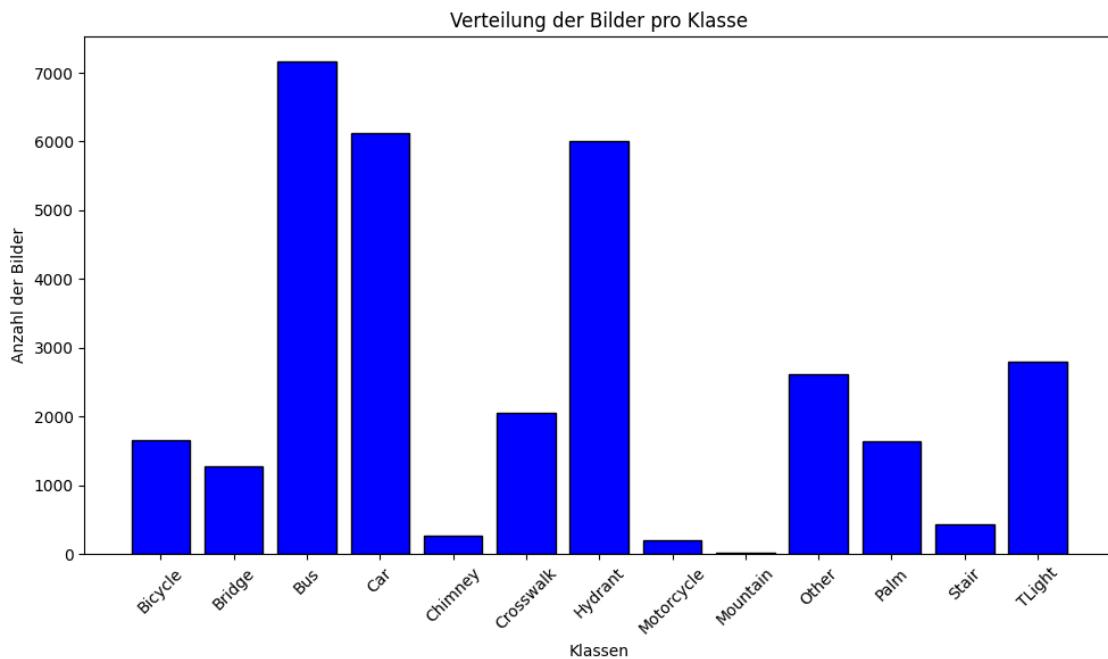
1.7.2 2.2 Klassenverteilung

Das Prüfen der Klassenverteilung ist eine wichtige Maßnahme, um sicherzustellen, dass ein Klassifikationsmodell sowohl die Mehrheits- als auch die Minderheitsklassen gut vorhersagen kann und keine Klasse vernachlässigt wird. Daher wird die Anzahl der Bilder in jeder Klasse wurde überprüft und in einem Balkendiagramm dargestellt. Es ist zu erkennen, dass der Datensatz nicht ausgeglichen ist. Das kann zu einem Bias im Modell führen und die Modellleistung beeinträchtigen (vgl. Géron 2019).

```
[8]: import matplotlib.pyplot as plt

folders = folder_image_data.keys()
values = [folder['count'] for folder in folder_image_data.values()]

plt.figure(figsize=(10, 6))
plt.bar(folders, values, color='blue', edgecolor='black')
plt.xlabel('Klassen')
plt.ylabel('Anzahl der Bilder')
plt.title('Verteilung der Bilder pro Klasse')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



1.7.3 2.3 Dateiendungen prüfen und Konvertierung in JPG

Die Verteilung der Bildformate (z. B. PNG, JPG) wurde analysiert, um sicherzustellen, dass alle relevanten Formate berücksichtigt werden. Zudem wurde eine zusätzliche Kategorie für Bilddateien mit anderen Dateiendungen eingerichtet. Die Bildformate JPG und PNG weisen unterschiedliche Eigenschaften auf, insbesondere hinsichtlich der verwendeten Farbdatenkanäle (Channels). Aufgrund dieser Unterschiede wurde beschlossen, alle Bilder in das JPG-Format zu konvertieren, um eine einheitliche Datenbasis zu gewährleisten.

```
[9]: import matplotlib.pyplot as plt

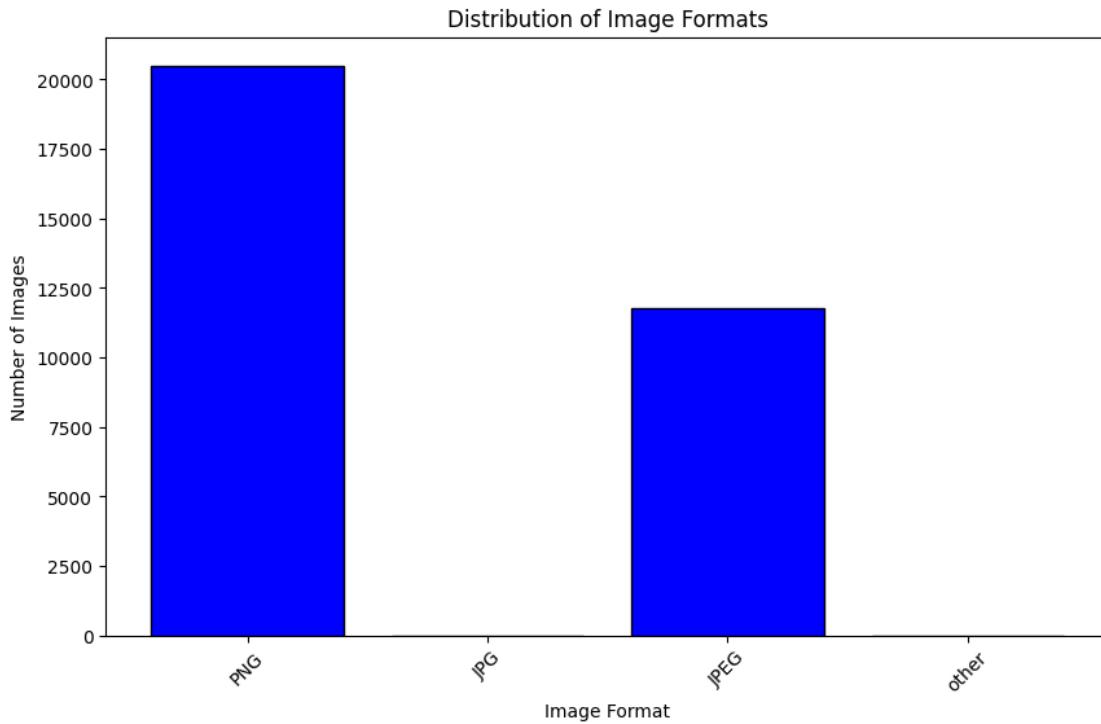
def check_file_types():
    file_format_counts = {'PNG': 0, 'JPG': 0, 'JPEG': 0, 'other': 0}

    for folder, data in folder_image_data.items():
        for img in data['images']:
            img_format = img['format'] if img['format'] else 'other'

            if img_format in file_format_counts:
                file_format_counts[img_format] += 1
            else:
                file_format_counts['other'] += 1

    plt.figure(figsize=(10, 6))
    plt.bar(file_format_counts.keys(), file_format_counts.values(), color='blue', edgecolor='black')
    plt.xlabel('Image Format')
    plt.ylabel('Number of Images')
    plt.title('Distribution of Image Formats')
    plt.xticks(rotation=45)
    plt.show()

check_file_types()
```



```
[10]: import matplotlib.pyplot as plt

count = 0
for folder, data in folder_image_data.items():
    converted_images = []

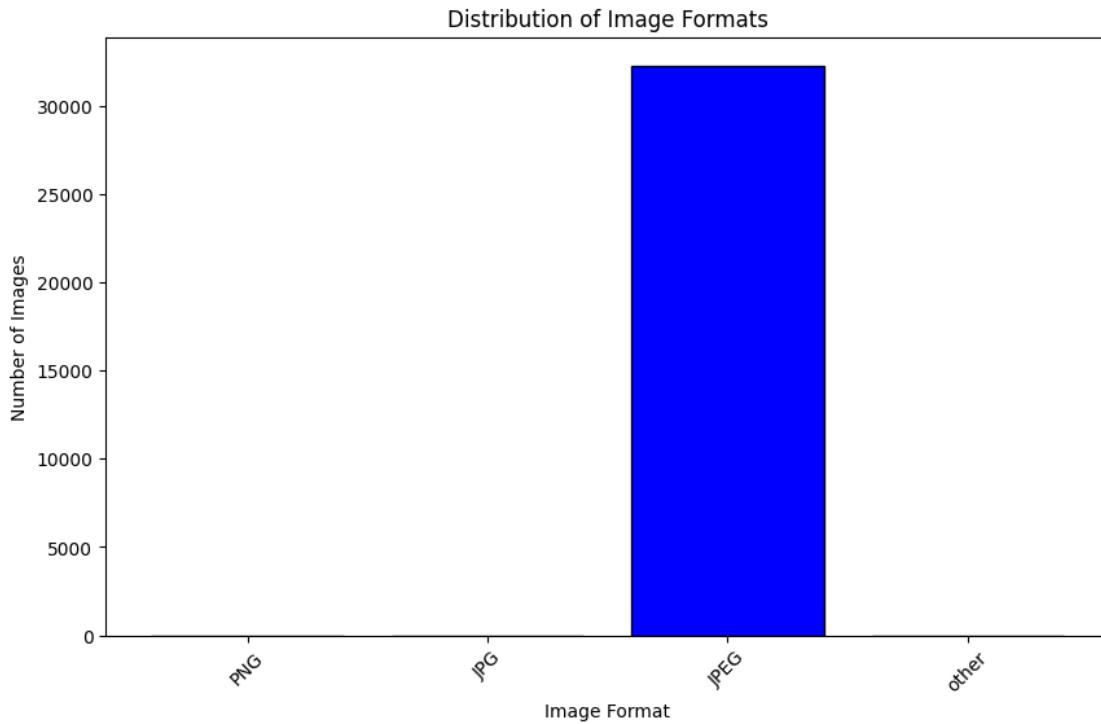
    for img in data['images']:
        if img['format'] == 'PNG':
            count += 1
            img_rgb = img['image'].convert('RGB')
            converted_images.append({'image': img_rgb, 'format': 'JPEG'})
        else:
            converted_images.append(img)

    folder_image_data[folder]['images'] = converted_images

logger.info(f"Insgesamt {count} Bilder konvertiert")
```

2024-09-07 20-09-39 - INFO - Insgesamt 20503 Bilder konvertiert

```
[11]: check_file_types()
```



1.7.4 2.4 Bildgrößen und Auflösungen

Die Verteilung der Bildbreiten und -höhen wurde visualisiert, um ein besseres Verständnis der Größenverteilung innerhalb des Datensatzes zu erlangen. Diese Information ist relevant für die Entscheidung über die Bildskalierung und -normalisierung in späteren Schritten der Datenvorverarbeitung.

```
[12]: image_sizes = set()

for folder, data in folder_image_data.items():
    for img_data in data['images']:
        img = img_data['image']
        size = img.size
        image_sizes.add(size)

unique_sizes = list(image_sizes)

logger.info(f"Menge an unterschiedlichen Bildgrößen: {unique_sizes}")
```

2024-09-07 20-09-39 - INFO - Menge an unterschiedlichen Bildgrößen: [(100, 100), (120, 120)]

1.7.5 2.5 Plotte ein Bild aus jedem Ordner

Ein visueller Eindruck des Datensatzes wurde durch das Anzeigen von Beispielbildern aus jedem Ordner gewonnen.

```
[13]: import matplotlib.pyplot as plt

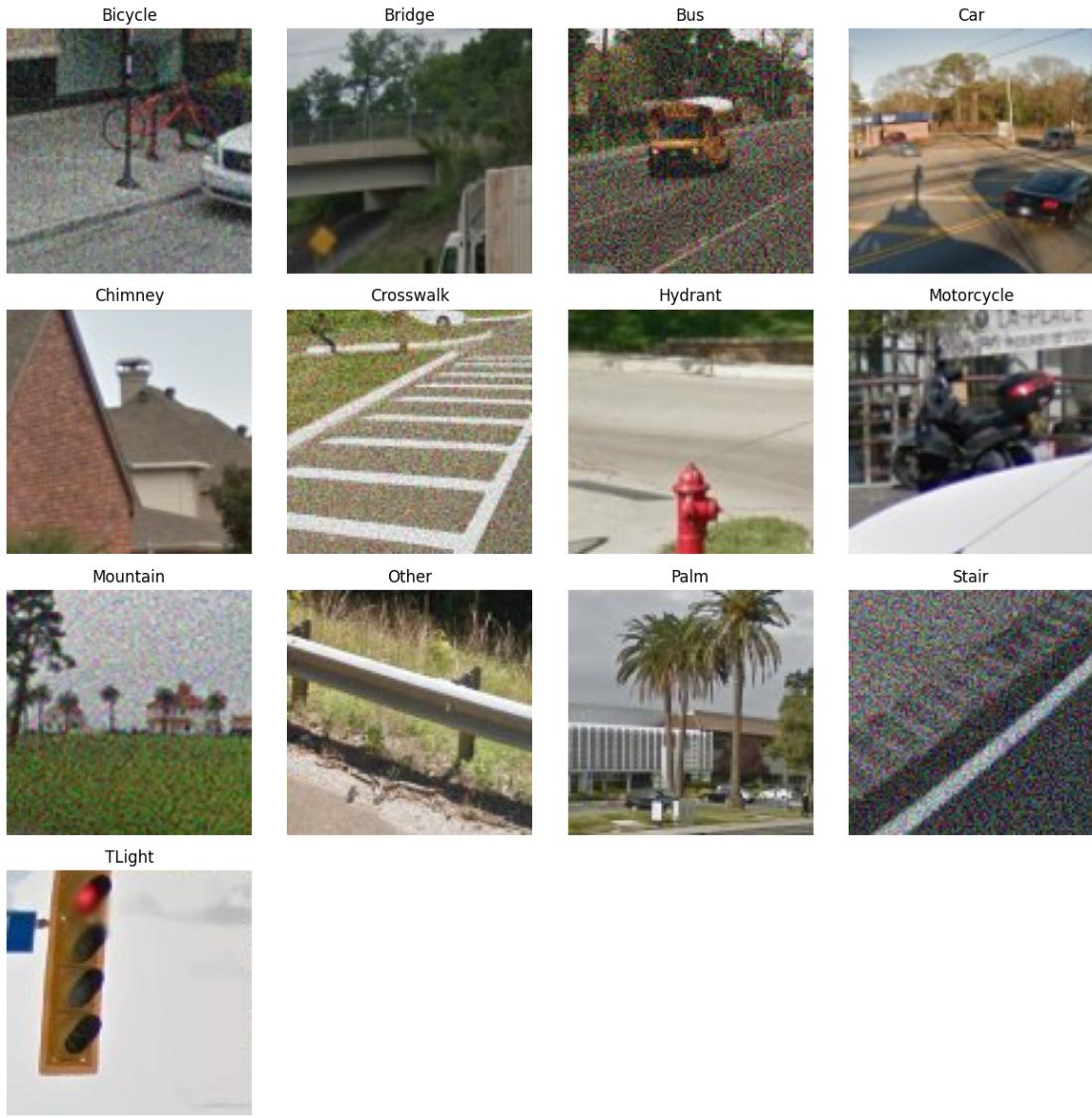
fig, axes = plt.subplots(4, 4, figsize=(12, 12))
count = 0
axes = axes.flatten()

for folder, data in folder_image_data.items():
    if count >= 16:
        break

    images = data['images']
    if images:
        img_dict = images[0]
        img = img_dict['image']
        axes[count].imshow(img)
        axes[count].set_title(folder)
        axes[count].axis('off')
        count += 1

for i in range(count, 16):
    axes[i].axis('off')

plt.tight_layout()
plt.show()
```



1.7.6 2.6 Zusammenfassung und Schlussfolgerungen der EDA

Die explorative Datenanalyse (EDA) hat gezeigt, dass die Klassen im Datensatz unterschiedlich verteilt sind. Beispielsweise enthält die Klasse „Car“ 6.123 Bilder, während die Klasse „Mountain“ lediglich 19 Bilder umfasst, was zu einer signifikanten Ungleichheit in der Datenverteilung führen kann. Diese Ungleichheit könnte potenzielle Anomalien darstellen, insbesondere in den weniger vertretenen Kategorien wie „Mountain“, die möglicherweise zusätzliche Aufmerksamkeit und weitere Analyse erfordern. Um eine ausgewogene Modellierung sicherzustellen, könnte eine Datenbalancierung erforderlich sein. Eine gleichmäßige Verteilung der Daten über alle Klassen hinweg ist entscheidend, um eine Verzerrung zugunsten der überrepräsentierten Klassen zu vermeiden. Darüber hinaus könnte die Sammlung zusätzlicher Daten für unterrepräsentierte Kategorien, wie z. B. „Mountain“, notwendig sein, um die Modellleistung zu verbessern und eine gerechtere Klassifikation zu ermöglichen (vgl. Müller und Guido 2017).

1.8 3. Erstellen eines Trainingsdatensatzes

Das Ziel dieses Abschnitts ist die Erzeugung eines Datensatzes, welcher für die maschinelle Verarbeitung geeignet ist.

1.8.1 3.1 Konfiguration der Experimentparameter

Zur besseren Lesbarkeit und Wartbarkeit wurden Datenvorbereitung-, Training- und Modellauswahl-Parameter generalisiert und an einer zentralen Stelle definiert.

```
[14]: from tensorboard.plugins.hparams import api as hp

HP_AUGMENT_DATA = hp.HParam('augment_data', hp.Discrete([True, False]))
HP_DATA_BALANCE = hp.HParam('data_balance', hp.Discrete([True, False]))

HP_DATA_ZOOM = hp.HParam('data_zoom', hp.RealInterval(-1.0, 1.0))
HP_DATA_CONTRAST = hp.HParam('data_contrast', hp.RealInterval(-1.0, 1.0))
HP_DATA_BRIGHTNESS_LOW = hp.HParam('data_brightness_low', hp.RealInterval(-1.0, ↴1.0))
HP_DATA_BRIGHTNESS_UP = hp.HParam('data_brightness_up', hp.RealInterval(-1.0, 1. ↴0))
HP_DATA_FLIP = hp.HParam('data_flip', hp.Discrete([True, False]))

HP_TRAINING_LOSS = hp.HParam('loss', hp.Discrete(['categorical_crossentropy']))
HP_TRAINING_OPTIMIZER = hp.HParam('optimizer', hp.Discrete(['adam', 'sgd']))
HP_TRAINING_EPOCHS = hp.HParam('epochs', hp.IntInterval(1, 10))

HP_MODEL_SELECTION = hp.HParam('model', hp.Discrete(['resnet50v2', ↴'inceptionv3', 'leNet5']))
```

2024-09-07 20:09:40.518877: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.

2024-09-07 20:09:40.526873: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:485] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered

2024-09-07 20:09:40.536524: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered

2024-09-07 20:09:40.538887: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered

2024-09-07 20:09:40.545689: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.

```
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations,  
rebuild TensorFlow with the appropriate compiler flags.
```

```
2024-09-07 20:09:41.322892: W
```

```
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not  
find TensorRT
```

Folgend sind 3 Code-Blöcke mit unterschiedlichen Parametereinstellungen definiert. Für die Auswertung der Forschungsfragen wurden diese Einstellungen der Parameter untersucht.

Beim Durchführen des Notebooks gilt es eine der Parametereinstellung als *hparams* zu speichern. Alle später definierten Modelle werden dann auf der jeweiligen Weisen trainiert:

- Keine Augmentation und keine Datenbalancierung
- Augmentation und keine Datenbalancierung
- Augmentation und Datenbalancierung

```
[15]: no_augment_unbalanced = {  
    HP_AUGMENT_DATA: False,  
    HP_DATA_BALANCE: False,  
  
    HP_DATA_ZOOM: 0.05,  
    HP_DATA_CONTRAST: 0.05,  
    HP_DATA_BRIGHTNESS_LOW: -0.05,  
    HP_DATA_BRIGHTNESS_UP: 0.05,  
    HP_DATA_FLIP: True,  
  
    HP_TRAINING_LOSS: 'categorical_crossentropy',  
    HP_TRAINING_OPTIMIZER: 'adam',  
    HP_TRAINING_EPOCHS: 10,  
}  
  
augment_unbalanced = {  
    HP_AUGMENT_DATA: True,  
    HP_DATA_BALANCE: False,  
  
    HP_DATA_ZOOM: 0.05,  
    HP_DATA_CONTRAST: 0.05,  
    HP_DATA_BRIGHTNESS_LOW: -0.05,  
    HP_DATA_BRIGHTNESS_UP: 0.05,  
    HP_DATA_FLIP: True,  
  
    HP_TRAINING_LOSS: 'categorical_crossentropy',  
    HP_TRAINING_OPTIMIZER: 'adam',  
    HP_TRAINING_EPOCHS: 10,  
}  
  
augment_balanced = {  
    HP_AUGMENT_DATA: True,  
    HP_DATA_BALANCE: True,
```

```

    HP_DATA_ZOOM: 0.05,
    HP_DATA_CONTRAST: 0.05,
    HP_DATA_BRIGHTNESS_LOW: -0.05,
    HP_DATA_BRIGHTNESS_UP: 0.05,
    HP_DATA_FLIP: True,

    HP_TRAINING_LOSS: 'categorical_crossentropy',
    HP_TRAINING_OPTIMIZER: 'adam',
    HP_TRAINING_EPOCHS: 10,
}

```

```
[16]: # hparams = no_augment_unbalanced
# hparams = augment_unbalanced
hparams = augment_balanced
```

1.8.2 3.2 Daten augmentieren

Die Frage, wie stark man Daten durch Augmentierung erhöhen sollte, ist ein wichtiger Forschungsbereich im maschinellen Lernen. Die Praxis, Datenklassen auf ähnliche Größenordnungen zu bringen, basiert auf dem Bedürfnis, das Modell vor Überanpassung an eine bestimmte Klasse zu schützen und sicherzustellen, dass alle Klassen ausreichend repräsentiert sind. Es gibt jedoch verschiedene Studien und Richtlinien, die aufzeigen, dass eine signifikante Augmentierung sinnvoll sein kann:

Studien zur Verbesserung der Klassifikationsleistung durch Datenaugmentierung:

Wong et al. (2016) in ihrem Paper “Understanding Data Augmentation for Classification” zeigen, dass Augmentierung insbesondere bei kleinen Datensätzen die Generalisierungsfähigkeit eines Modells erheblich verbessern kann. Sie heben hervor, dass selbst drastische Erhöhungen der Datensetze durch Augmentierung (um das 10- bis 100-fache) bei unterrepräsentierten Klassen zu einer verbesserten Leistung führen können. Perez und Wang (2017) in “The Effectiveness of Data Augmentation in Image Classification using Deep Learning” analysieren den Effekt von Datenaugmentierung in verschiedenen Szenarien und finden heraus, dass eine drastische Erhöhung der Anzahl von Trainingsbeispielen durch Augmentierung zu einer signifikant besseren Leistung führen kann, insbesondere wenn die Augmentierungen realistische Varianten der Bilder erzeugen. Balance von Klassen:

Buda, Maki, und Mazurowski (2018) in ihrem Paper “A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks” untersuchen die Auswirkungen von Klassenungleichgewichten und zeigen, dass ein Ausgleich der Klassenverteilung durch Datenaugmentierung oder andere Techniken entscheidend ist, um die Performance eines Modells zu verbessern. Sie betonen, dass eine zu große Diskrepanz in der Klassenverteilung zu einer schlechteren Modellleistung führt. Praktische Leitlinien:

Die Praxis der “Over-Sampling”-Technik durch Augmentierung (bei der unterrepräsentierte Klassen stark augmentiert werden) ist eine weit verbreitete Methode, um Klassifikationsmodelle robuster zu machen. Es gibt keine festgelegte Grenze für das “Wie viel”, aber es ist üblich, die kleineren Klassen auf eine ähnliche Größe wie die größeren zu bringen, um ein ausgewogenes Training zu ermöglichen.

3.2.1 Augmentierung der Daten durch Zoom, Kontrast, Helligkeit und vertikale Spiegelung

3.2.1.1 Definition und Ausführung der Augmentierungspipeline `augmentation_pipeline()` erstellt eine Augmentierungspipeline basierend auf übergebenen Parametern wie Zoom, Kontrast, Helligkeit und Spiegelung. Die Funktion `augment_class_images()` nimmt Bilddaten eines bestimmten Ordners und führt Augmentierungen durch, um die Anzahl der Bilder auf eine gewünschte Zielmenge zu erhöhen. Augmentierte Bilder werden der Liste hinzugefügt, bis die gewünschte Bildanzahl erreicht ist.

```
[17]: import numpy as np
import tensorflow as tf

def augmentation_pipeline(zoom:float = hparams[HP_DATA_ZOOM],
                           contrast:float = hparams[HP_DATA_CONTRAST],
                           brightness:list = [hparams[HP_DATA_BRIGHTNESS_LOW], hparams[HP_DATA_BRIGHTNESS_UP]],
                           flip: bool = hparams[HP_DATA_FLIP]):

    if hparams[HP_DATA_FLIP]:
        logger.debug(f"Augmentierungs Parameter: Zoom - +/- {zoom} %, Kontrast +/- +/- {contrast} %, Helligkeit - -{brightness[0]}% bis +{brightness[0]}%, Spiegelung - {'Trifft zu' if flip == True else 'Trifft nicht zu'}")
        data_augmentation = tf.keras.Sequential([
            tf.keras.layers.RandomZoom(height_factor=zoom, seed=42),
            tf.keras.layers.RandomContrast(factor=contrast, seed=42),
            tf.keras.layers.RandomBrightness(factor=brightness, seed=42),
            tf.keras.layers.RandomFlip(mode='horizontal', seed=42),
        ])
    else:
        logger.debug(f"Augmentierungs Parameter: Zoom - +/- {zoom} %, Kontrast +/- +/- {contrast} %, Helligkeit - -{brightness[0]}% bis +{brightness[0]}%, Spiegelung - {'Trifft zu' if flip == True else 'Trifft nicht zu'}")
        data_augmentation = tf.keras.Sequential([
            tf.keras.layers.RandomZoom(height_factor=zoom, seed=42),
            tf.keras.layers.RandomContrast(factor=contrast, seed=42),
            tf.keras.layers.RandomBrightness(factor=brightness, seed=42),
        ])
    return data_augmentation

def augment_class_images(folder:str, folder_data, target_count):

    current_count = folder_data['count']
    images = folder_data['images']
    if folder == "Mountain":
        data_augmentation = augmentation_pipeline(contrast=0.025, brightness=[-0.025,0.025])
```

```

else:
    data_augmentation = augmentation_pipeline()

while current_count < target_count:
    for img_dict in images:
        img = img_dict['image']
        img_array = np.array(img.convert('RGB'))

        img_augmented = data_augmentation(tf.expand_dims(img_array, 0))
        img_augmented = tf.squeeze(img_augmented).numpy().astype("uint8")

        img_augmented_pil = Image.fromarray(img_augmented)

        images.append({'image': img_augmented_pil})
        current_count += 1

    if current_count >= target_count:
        break

folder_data['count'] = current_count
folder_data['images'] = images

```

```

[18]: if hparams[HP_AUGMENT_DATA] == True:

    for folder, data in folder_image_data.items():
        logger.info(f"Augmentiere Bilder der Klasse: {folder}")
        target_count = 2000
        if data['count'] < target_count:
            augment_class_images(folder, data, target_count)
        elif hparams[HP_DATA_BALANCE] == True and data['count'] > target_count:
            logger.info(f"Reduziere Bilder der Klasse: {folder} auf"
                        f"{target_count}.")
            data['images'] = data['images'][:target_count]
            data['count'] = target_count
            logger.info(f"Klasse {folder} enthält nun {data['count']} Bilder.")
        for folder, data in folder_image_data.items():
            logger.info(f"Klasse {folder} enthält nun {data['count']} Bilder.")

    else:
        logger.debug(f"HP_AUGMENT_DATA = {hparams[HP_AUGMENT_DATA]}")
        logger.info(f"Die Daten wurden NICHT augmentiert.")

```

2024-09-07 20-09-42 - INFO - Augmentiere Bilder der Klasse: Bicycle
 WARNING: All log messages before absl::InitializeLog() is called are written to
 STDERR
 I0000 00:00:1725732582.219058 551274 cuda_executor.cc:1001] could not open file
 to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
 Your kernel may have been built without NUMA support.
 I0000 00:00:1725732582.246612 551274 cuda_executor.cc:1001] could not open file

```
to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
I0000 00:00:1725732582.246648 551274 cuda_executor.cc:1001] could not open file
to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
I0000 00:00:1725732582.249670 551274 cuda_executor.cc:1001] could not open file
to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
I0000 00:00:1725732582.249704 551274 cuda_executor.cc:1001] could not open file
to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
I0000 00:00:1725732582.249715 551274 cuda_executor.cc:1001] could not open file
to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
I0000 00:00:1725732582.373118 551274 cuda_executor.cc:1001] could not open file
to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
I0000 00:00:1725732582.373172 551274 cuda_executor.cc:1001] could not open file
to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2024-09-07 20:09:42.373183: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:2112] Could not identify NUMA
node of platform GPU id 0, defaulting to 0. Your kernel may not have been built
with NUMA support.
I0000 00:00:1725732582.373212 551274 cuda_executor.cc:1001] could not open file
to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2024-09-07 20:09:42.373230: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:2021] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 9711 MB memory: -> device: 0,
name: NVIDIA GeForce RTX 3060, pci bus id: 0000:01:00.0, compute capability: 8.6
2024-09-07 20-09-47 - INFO - Klasse Bicycle enthält nun 2000 Bilder.
2024-09-07 20-09-47 - INFO - Augmentiere Bilder der Klasse: Bridge
2024-09-07 20-09-57 - INFO - Klasse Bridge enthält nun 2000 Bilder.
2024-09-07 20-09-57 - INFO - Augmentiere Bilder der Klasse: Bus
2024-09-07 20-09-57 - INFO - Reduziere Bilder der Klasse: Bus auf 2000.
2024-09-07 20-09-57 - INFO - Klasse Bus enthält nun 2000 Bilder.
2024-09-07 20-09-57 - INFO - Augmentiere Bilder der Klasse: Car
2024-09-07 20-09-57 - INFO - Reduziere Bilder der Klasse: Car auf 2000.
2024-09-07 20-09-57 - INFO - Klasse Car enthält nun 2000 Bilder.
2024-09-07 20-09-57 - INFO - Augmentiere Bilder der Klasse: Chimney
2024-09-07 20-10-23 - INFO - Klasse Chimney enthält nun 2000 Bilder.
2024-09-07 20-10-23 - INFO - Augmentiere Bilder der Klasse: Crosswalk
2024-09-07 20-10-23 - INFO - Reduziere Bilder der Klasse: Crosswalk auf 2000.
2024-09-07 20-10-23 - INFO - Klasse Crosswalk enthält nun 2000 Bilder.
2024-09-07 20-10-23 - INFO - Augmentiere Bilder der Klasse: Hydrant
2024-09-07 20-10-23 - INFO - Reduziere Bilder der Klasse: Hydrant auf 2000.
2024-09-07 20-10-23 - INFO - Klasse Hydrant enthält nun 2000 Bilder.
```

```
2024-09-07 20-10-23 - INFO - Augmentiere Bilder der Klasse: Motorcycle
2024-09-07 20-10-49 - INFO - Klasse Motorcycle enthält nun 2000 Bilder.
2024-09-07 20-10-49 - INFO - Augmentiere Bilder der Klasse: Mountain
2024-09-07 20-11-18 - INFO - Klasse Mountain enthält nun 2000 Bilder.
2024-09-07 20-11-18 - INFO - Augmentiere Bilder der Klasse: Other
2024-09-07 20-11-18 - INFO - Reduziere Bilder der Klasse: Other auf 2000.
2024-09-07 20-11-18 - INFO - Klasse Other enthält nun 2000 Bilder.
2024-09-07 20-11-18 - INFO - Augmentiere Bilder der Klasse: Palm
2024-09-07 20-11-23 - INFO - Klasse Palm enthält nun 2000 Bilder.
2024-09-07 20-11-23 - INFO - Augmentiere Bilder der Klasse: Stair
2024-09-07 20-11-47 - INFO - Klasse Stair enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Augmentiere Bilder der Klasse: TLight
2024-09-07 20-11-47 - INFO - Reduziere Bilder der Klasse: TLight auf 2000.
2024-09-07 20-11-47 - INFO - Klasse TLight enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Bicycle enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Bridge enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Bus enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Car enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Chimney enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Crosswalk enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Hydrant enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Motorcycle enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Mountain enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Other enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Palm enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse Stair enthält nun 2000 Bilder.
2024-09-07 20-11-47 - INFO - Klasse TLight enthält nun 2000 Bilder.
```

3.2.1.2 Demonstration der Augmentierung

```
[19]: import numpy as np
import matplotlib.pyplot as plt

fig, axes = plt.subplots(4, 4, figsize=(12, 12))
count = 0
axes = axes.flatten()

data_augmentation = augmentation_pipeline()

for folder, data in folder_image_data.items():
    if count >= 16:
        break
    images = data['images']

    if images:
        img_dict = images[0]
        img = img_dict['image']
        img_array = np.array(img.convert('RGB'))
```

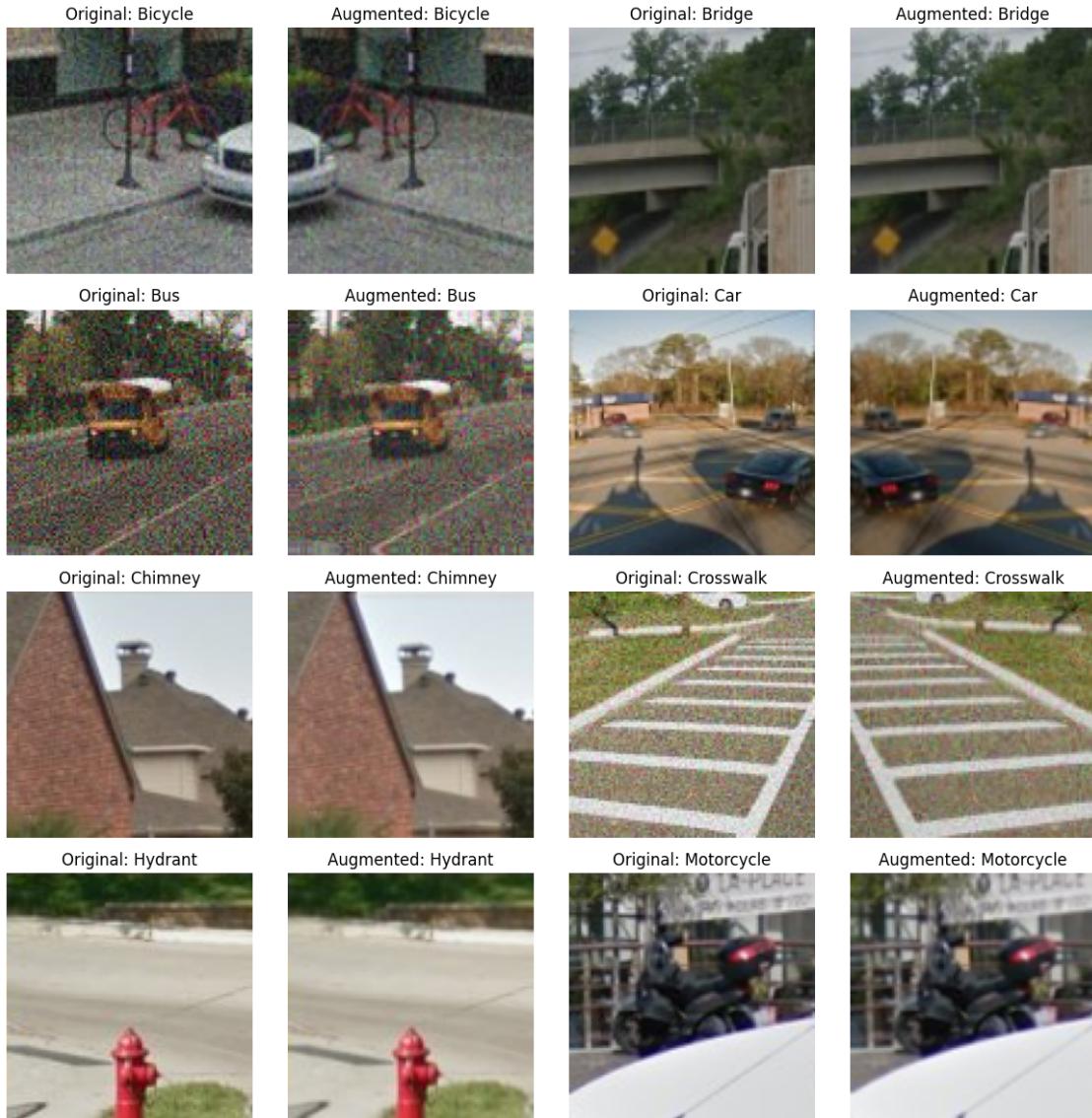
```
axes[count].imshow(img_array)
axes[count].set_title(f"Original: {folder}")
axes[count].axis('off')
count += 1

img_augmented = data_augmentation(tf.expand_dims(img_array, 0))
img_augmented = tf.squeeze(img_augmented).numpy().astype("uint8")

axes[count].imshow(img_augmented)
axes[count].set_title(f"Augmented: {folder}")
axes[count].axis('off')
count += 1

for i in range(count, 16):
    axes[i].axis('off')

plt.tight_layout()
plt.show()
```



Besondere Untersuchung der Klasse *Mountain* um die Auswirkungen der Augmentierung anhand der mengenmäßig kleinsten Klasse visuell zu beurteilen

```
[20]: import random
import numpy as np
import matplotlib.pyplot as plt

mountain_images = folder_image_data['Mountain']['images']
random.shuffle(mountain_images)
selected_images = mountain_images[:81]

if hparams[HP_AUGMENT_DATA]:
```

```
    fig, axes = plt.subplots(9, 9, figsize=(20, 20))
else:
    fig, axes = plt.subplots(5, 5, figsize=(10, 10))
axes = axes.flatten()

for i, img_dict in enumerate(selected_images):
    img = img_dict['image']
    img_array = np.array(img.convert('RGB'))
    axes[i].imshow(img_array)
    axes[i].axis('off')

for i in range(len(selected_images), len(axes)):
    axes[i].axis('off')

plt.tight_layout()
plt.show()
```



3.2.2 Datensatz in einem temporären Ordner speichern Keras kann mit Dictionaries nicht arbeiten, aus diesem Grund werden die Daten in einem temporären Ordner gespeichert.

```
[21]: import os, sys
import traceback
import tempfile
from PIL import Image
import matplotlib.pyplot as plt

temp_dir = tempfile.mkdtemp()
try:
    for folder, data in folder_image_data.items():
        folder_path = os.path.join(temp_dir, folder)
```

```

os.makedirs(folder_path, exist_ok=True)

for i, img_dict in enumerate(data['images']):
    img = img_dict['image']
    img_rgb = img.convert('RGB')

    img_path = os.path.join(folder_path, f'image_{i}.jpg')
    img_rgb.save(img_path, format='JPEG')

for root, dirs, files in os.walk(temp_dir):
    level = root.replace(temp_dir, '').count(os.sep)
    indent = ' ' * 4 * level

    subindent = ' ' * 4 * (level + 1)
    for f in files:
        img_path = os.path.join(root, f)
        with Image.open(img_path) as img:
            width, height = img.size

fig, axes = plt.subplots(4, 4, figsize=(12, 12))

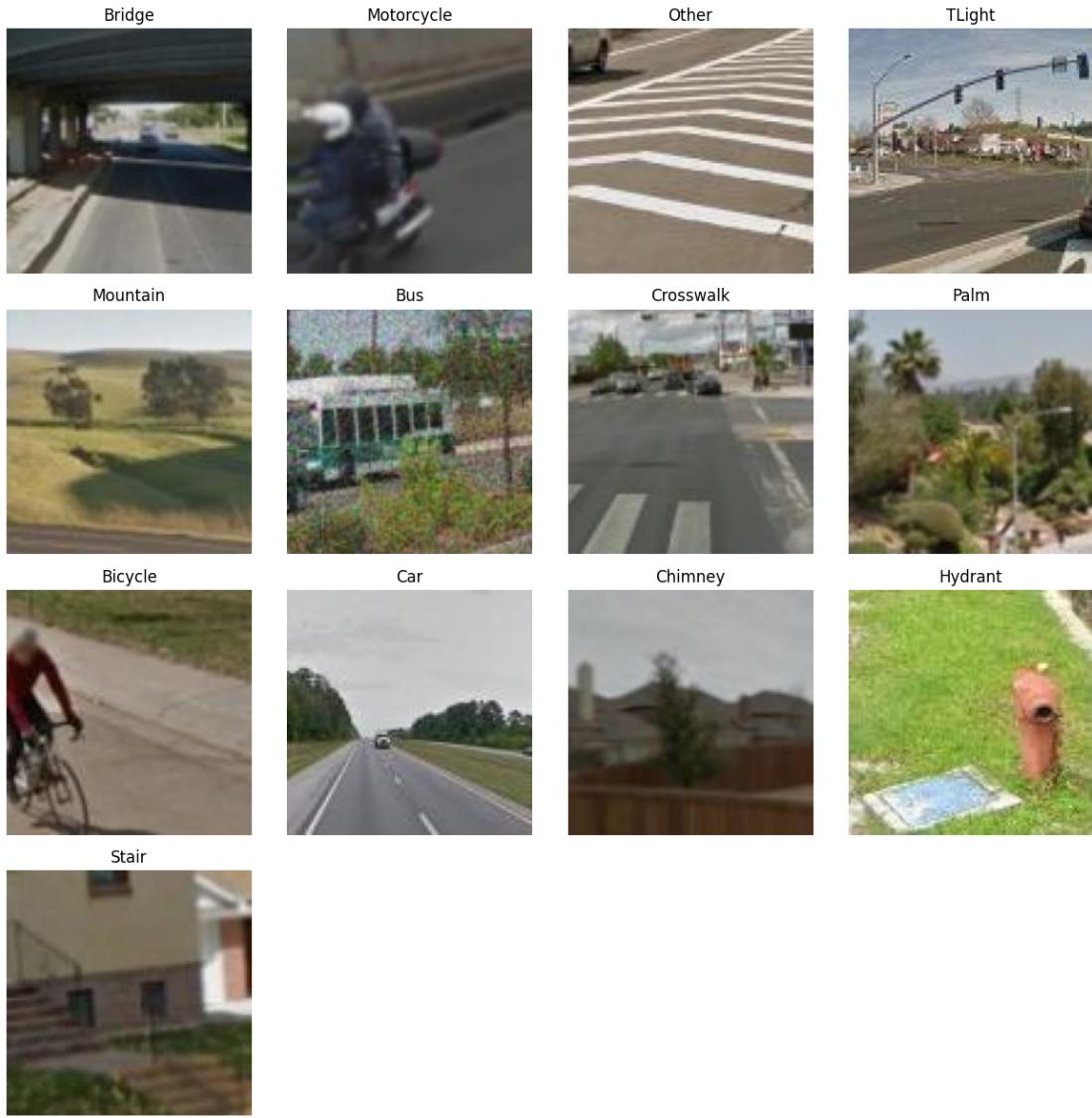
count = 0
axes = axes.flatten()
for folder in os.listdir(temp_dir):
    if count >= 16:
        break
    folder_path = os.path.join(temp_dir, folder)
    if os.path.isdir(folder_path):
        img_files = os.listdir(folder_path)
        if img_files:
            img_path = os.path.join(folder_path, img_files[0])
            img = Image.open(img_path)
            axes[count].imshow(img)
            axes[count].set_title(folder)
            axes[count].axis('off')
            count += 1

for i in range(count, 16):
    axes[i].axis('off')

plt.tight_layout()
plt.show()

except Exception as e:
    logger.error("Fehler in der Erstellung des temporären Ordners")
    traceback.print_exception(*sys.exc_info())

```



1.8.3 3.3 Datensatz erzeugen mithilfe Keras Bibliotheken

Anhand der zur Verfügung stehenden Keras function `image_dataset_from_directory` ist es möglich, einfach aus dem bestehenden vorverarbeiteten Bilderverzeichnis zwei in Zahlen kodierte Datensätze zu erzeugen.

Mittels dessen Funktionsparameter kann unmittelbar eine Größenanpassung auf 120x120 Pixel und eine Bündelung in Stapel (Batches) je 64 Bildtensoren durchgeführt werden.

Die Erkennung des Klasse eines Bildes erfolgt aus der Verortung in ihrem zugehörigen Klassenordner.

Eine Klassenbezeichnung im Datensatz wird kategorisch, also als 13-dimensionaler Tensor mit binärer Ausprägung an entsprechender Stelle definiert.

Die Sortierung der Klassen im Tensor wird alphanumerisch vorgenommen.

1. Bicycle	8. Motorcycle
2. Bridge	9. Mountain
3. Bus	10. Other
4. Car	11. Palm
5. Chimney	12. Stair
6. Crosswalk	13. TLight
7. Hydrant	

```
[22]: import tensorflow as tf

try:
    batch_size = 64

    train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
        temp_dir,
        labels="inferred",
        label_mode="categorical",
        class_names=None,
        color_mode="rgb",
        batch_size=batch_size,
        image_size=(120, 120),
        shuffle=True,
        seed=42,
        validation_split=0.2,
        subset='training',
        interpolation="bilinear",
        follow_links=False,
        crop_to_aspect_ratio=False,
        pad_to_aspect_ratio=False,
        data_format='channels_last',
        verbose=True
    )

    validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
        temp_dir,
        labels="inferred",
        label_mode="categorical",
        class_names=None,
        color_mode="rgb",
        batch_size=batch_size,
        image_size=(120, 120),
        shuffle=True,
        seed=42,
        validation_split=0.2,
        subset='validation',
```

```

        interpolation="bilinear",
        follow_links=False,
        crop_to_aspect_ratio=False,
        pad_to_aspect_ratio=False,
        data_format='channels_last',
        verbose=True
    )

    logger.info(f"Trainingsdatensatz enthält {len(train_dataset)} Stapel je {batch_size} Bilder")
    logger.info(f"Validierungsdatensatz enthält {len(validation_dataset)} Stapel je {batch_size} Bilder")

    for element in train_dataset:
        logger.debug(f"shape X_train: {element[0].shape}")
        logger.debug(f"shape Y_train: {element[1].shape}")
        break

finally:
    logger.info("Erfolgreich Trainings- und Validierungsdatensatz erstellt")

```

Found 26000 files belonging to 13 classes.

Using 20800 files for training.

Found 26000 files belonging to 13 classes.

Using 5200 files for validation.

2024-09-07 20-11-55 - INFO - Trainingsdatensatz enthält 325 Stapel je 64 Bilder

2024-09-07 20-11-55 - INFO - Validierungsdatensatz enthält 82 Stapel je 64

Bilder

2024-09-07 20-11-55 - INFO - Erfolgreich Trainings- und Validierungsdatensatz erstellt

Folgend werden die erstellten Datensätze visuell überprüft.

```
[23]: import numpy as np
import matplotlib.pyplot as plt

def show_class_images(dataset):
    category_images = {}
    class_names = dataset.class_names
    for images, labels in dataset:
        for img, label in zip(images, labels):
            category = class_names[np.argmax(label)]
            if category not in category_images:
                category_images[category] = img.numpy()
            if len(category_images) == len(class_names):
                break
    if len(category_images) == len(class_names):
```

```

        break

num_categories = len(category_images)
fig, axes = plt.subplots(1, num_categories, figsize=(15, 5))
for ax, (category, img) in zip(axes, category_images.items()):
    ax.imshow(img.astype("uint8"))
    ax.set_title(category)
    ax.axis("off")

plt.tight_layout()
plt.show()

show_class_images(validation_dataset)

```



[24]: show_class_images(train_dataset)



1.9 4. Entwicklung und Optimierung von neuronalen Netzarchitekturen

Zur Beantwortung der Forschungsfragen werden zum Einen moderne, vortrainierte Modelle aus dem [Keras Applications-Modul](#) untersucht. Diese Modelle wurden auf großen Datensätzen wie ImageNet vortrainiert, was den Vorteil bietet, dass sie für eine Vielzahl von Anwendungen direkt genutzt oder als Basis für das sogenannte Transfer Learning verwendet werden können. Die in dieser Arbeit gestellte Aufgabe zur Klassifizierung von Google Recaptcha Bildern kann dadurch mit minimalem zusätzlichen Training bewältigt werden. Aus der großen Auswahl an vortrainierten Modellen wurden ResNet50V2 und InceptionV3 ausgewählt, da diese unter verhältnismäßig geringer Modellgröße eine gute Leistung beim Training mit dem ImageNet Datensatz aufweisen.

Zum Anderen wird die historische Architektur LeNet-5 betrachtet, welche wegweisend für die spätere Entwicklung von Convolutional Neural Networks (CNNs) war.¹ Ein Vergleich zwischen LeNet-5 und modernen Architekturen wie ResNet oder Inception zeigt die enorme Entwicklung der Komplexität und Leistung von neuronalen Netzen in den letzten Jahrzehnten.

Nach der Erstellung der neuronalen Netze beschreibt dieses Notebook die durchgeführten Optimierungsprozesse. Zur Überwachung des Modelltrainings werden verschiedene Callback-Funktionen

¹LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

definiert, um Daten zur Modelleistung sowie das Modell selbst zu speichern. Dazu bietet sich bereits durch die Nutzung der Tensorflow und Keras Bibliotheken die Integration mit Tensorboard an.

Die durch Tensorboard exportierten Daten im Ordner *logs* können so später mittels eines interaktiven Dashboard eingesehen werden. Die Metadaten zum Training werden entsprechend der Experimentparameter Art des Datensatzes, Modell und Zeitpunkt sortiert. (i. e. *logs/augment-unbalanced/leNet5/captcha-05.09.2024 23-14-05*) Aus der Notebook Python-Umgebung heraus kann mittels `tensorboard --logdir ./logs` ein [lokaler Webserver](#) gestartet werden, welcher das Dashboard anzeigt.

Final kann in Kapitel 4.4 nach Angabe des Speicherpfads, eins der optimierten Modelle geladen werden, um die Klassen einer Menge von Bildern vorherzusagen.

Notiz: Die gewählte Menge von Bildern soll dabei lediglich dazu dienen die Funktionsweise eines DeepLearning-Modells darzustellen und entspricht keinem zuvor vom Training exkludiertem Testdatensatz.

1.9.1 4.1 Überprüfung der GPU Unterstützung

Diese Zelle überprüft die Anbindung der Softwareumgebung an eine GPU.

```
[25]: import sys
import tensorflow as tf
from tensorflow import keras as keras
import platform

print(f"Python Platform: {platform.platform()}")
print(f"Python {sys.version}")
print()
print(f"Tensor Flow Version: {tf.__version__}")
print(f"Keras Version: {keras.__version__}")
print()
gpu = len(tf.config.list_physical_devices('GPU'))>0
print("GPU is", "available" if gpu else "NOT AVAILABLE")
```

```
Python Platform: Linux-5.15.153.1-microsoft-standard-WSL2-x86_64-with-glibc2.35
Python 3.10.14 | packaged by conda-forge | (main, Mar 20 2024, 12:45:18) [GCC
12.3.0]
```

```
Tensor Flow Version: 2.17.0
```

```
Keras Version: 3.5.0
```

```
GPU is available
```

1.9.2 4.2 Erstellung verschiedener neuronaler Netzarchitekturen

4.2.1 ResNet50V2 Zur Anpassung des ResNet50V2 Modells auf die betrachtete Aufgabenstellung, werden darauf aufbauend Schichten dem Modell hinzugefügt.

```
[26]: import tensorflow.keras as keras
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout

base = keras.applications.ResNet50V2(
    include_top=False,
    weights='imagenet',
    input_shape=(120, 120, 3),
    name='resnet50v2')

resnet50v2 = Sequential()
resnet50v2.add(base)
resnet50v2.add(Dropout(0.2))
resnet50v2.add(Flatten())
resnet50v2.add(Dense(13, activation='softmax'))

resnet50v2.summary()
```

Model: "sequential_8"

Layer (type)	Output Shape	Param #
resnet50v2 (Functional)	(None, 4, 4, 2048)	23,564,800
dropout (Dropout)	(None, 4, 4, 2048)	0
flatten (Flatten)	(None, 32768)	0
dense (Dense)	(None, 13)	425,997

Total params: 23,990,797 (91.52 MB)

Trainable params: 23,945,357 (91.34 MB)

Non-trainable params: 45,440 (177.50 KB)

4.2.2 InceptionV3 Zur Anpassung des InceptionV3 Modells auf die betrachtete Aufgabenstellung, werden darauf aufbauend Schichten dem Modell hinzugefügt.

```
[27]: import tensorflow.keras as keras
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout
```

```

base = keras.applications.InceptionV3(
    include_top=False,
    weights="imagenet",
    input_shape=(120, 120, 3),
    name="inception_v3",
)

inceptionv3 = Sequential()
inceptionv3.add(base)
inceptionv3.add(Dropout(0.2))
inceptionv3.add(Flatten())
inceptionv3.add(Dense(13, activation='softmax'))

inceptionv3.summary()

```

Model: "sequential_9"

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 2, 2, 2048)	21,802,784
dropout_1 (Dropout)	(None, 2, 2, 2048)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_1 (Dense)	(None, 13)	106,509

Total params: 21,909,293 (83.58 MB)

Trainable params: 21,874,861 (83.45 MB)

Non-trainable params: 34,432 (134.50 KB)

4.2.3 LeNet5

```
[28]: from tensorflow.keras.layers import Conv2D, Dense, Flatten, AveragePooling2D
leNet5 = Sequential()
leNet5.add(Conv2D(filters=6, strides=1, kernel_size=(5, 5), activation='relu',
      input_shape=(120, 120, 3)))
leNet5.add(AveragePooling2D(strides=2, pool_size=(2, 2)))
```

```

leNet5.add(Conv2D(filters=16, strides=1, kernel_size=(5, 5),
                  activation='relu'))
leNet5.add(AveragePooling2D(strides=2, pool_size=(2, 2)))
leNet5.add(Flatten())
leNet5.add(Dense(13, activation='softmax'))
leNet5.summary()

```

```

/home/leonhenne/miniforge3/envs/nak_ml/lib/python3.10/site-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
    super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential_10"

```

Layer (type)	Output Shape	Param #
conv2d_94 (Conv2D)	(None, 116, 116, 6)	456
average_pooling2d_9 (AveragePooling2D)	(None, 58, 58, 6)	0
conv2d_95 (Conv2D)	(None, 54, 54, 16)	2,416
average_pooling2d_10 (AveragePooling2D)	(None, 27, 27, 16)	0
flatten_2 (Flatten)	(None, 11664)	0
dense_2 (Dense)	(None, 13)	151,645

Total params: 154,517 (603.58 KB)

Trainable params: 154,517 (603.58 KB)

Non-trainable params: 0 (0.00 B)

1.9.3 4.3 Trainingsprozess

4.3.1 Überwachung der Trainingsprozesse

```
[29]: import time
if hparams[HP_AUGMENT_DATA]:
```

```

        dataset_type = 'augment'
else:
    dataset_type = 'no_augment'

if hparams[HP_DATA_BALANCE]:
    dataset_type = dataset_type + '-balanced'
else:
    dataset_type = dataset_type + '-unbalanced'

```

```
[30]: import tensorflow.keras as keras
from keras.callbacks import TensorBoard, ModelCheckpoint, ProgbarLogger, CSVLogger

def create_callbacks(log_dir: str, model_dir: str) -> list:
    tensorboard_callback = TensorBoard(
        log_dir=log_dir,
        histogram_freq=1,
        update_freq='epoch',
        write_graph=True)

    checkpoint_callback = ModelCheckpoint(
        filepath=model_dir,
        save_weights_only=False,
        monitor='val_accuracy',
        mode='max',
        save_best_only=True,
        verbose=1)

    progressbar_callback = ProgbarLogger()

    csv_callback = CSVLogger(log_dir + 'logs.csv')

    callbacks= [tensorboard_callback,
               checkpoint_callback,
               progressbar_callback,
               csv_callback]

    return callbacks

```

4.3.2 Optimierung von ResNet50V2

```
[31]: hparams[HP_MODEL_SELECTION] = 'resnet50v2'
logger.info(f"Ausgewähltes Modell: {hparams[HP_MODEL_SELECTION]}")

log_dir = f'./logs/{dataset_type}/{hparams[HP_MODEL_SELECTION]}/captcha-{time.strftime("%d.%m.%Y %H-%M-%S", time.localtime())}/'
model_dir = f'{log_dir}/models/' + 'model-epoch.{epoch:02d}-val_loss.{val_loss:.2f}-val_acc.{val_accuracy:.2f}.keras'
```

```

logger.info(f"Speichere LOGS in: '{log_dir}'")
logger.info(f"Speichere optimierte Modelle in: '{model_dir}'")

with tf.summary.create_file_writer(log_dir).as_default():
    hp.hparams(hparams)

callbacks = create_callbacks(log_dir, model_dir)

```

```

2024-09-07 20-11-59 - INFO - Ausgewähltes Modell: resnet50v2
2024-09-07 20-11-59 - INFO - Speichere LOGS in: './logs/augment-
balanced/resnet50v2/captcha-07.09.2024 20-11-59/'
2024-09-07 20-11-59 - INFO - Speichere optimierte Modelle in: './logs/augment-
balanced/resnet50v2/captcha-07.09.2024 20-11-59//models/model-
epoch.{epoch:02d}-val_loss.{val_loss:.2f}-val_acc.{val_accuracy:.2f}.keras'

```

[32]: `from tensorflow.keras import metrics`

```

resnet50v2.compile(optimizer=hparams[HP_TRAINING_OPTIMIZER],
                    loss=hparams[HP_TRAINING_LOSS],
                    metrics=[
                        metrics.CategoricalAccuracy(name = 'accuracy'),
                        metrics.Precision(name = 'precision'),
                        metrics.Recall(name = 'recall'),
                        metrics.AUC(name = 'auc')
                    ])

```

[33]: `resnet50v2.fit(train_dataset, epochs=hparams[HP_TRAINING_EPOCHS], validation_data=validation_dataset, callbacks=callbacks)`

Epoch 1/10

```

WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1725732731.124644 553130 service.cc:146] XLA service 0x55ce9853e7a0
initialized for platform CUDA (this does not guarantee that XLA will be used).
Devices:
I0000 00:00:1725732731.124693 553130 service.cc:154] StreamExecutor device
(0): NVIDIA GeForce RTX 3060, Compute Capability 8.6
2024-09-07 20:12:11.531523: I
tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR
crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
2024-09-07 20:12:13.237743: I
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:531] Loaded cuDNN
version 8907
I0000 00:00:1725732755.629525 553130 device_compiler.h:188] Compiled cluster
using XLA! This line is logged at most once for the lifetime of the process.

325/325          0s 100ms/step -
accuracy: 0.5315 - auc: 0.8875 - loss: 1.7582 - precision: 0.6894 - recall:

```

```

0.3734
Epoch 1: val_accuracy improved from -inf to 0.20942, saving model to
./logs/augment-balanced/resnet50v2/captcha-07.09.2024 20-11-59//models/model-
epoch.01-val_loss.109.44-val_acc.0.21.keras
325/325          80s 131ms/step -
accuracy: 0.5316 - auc: 0.8876 - loss: 1.7579 - precision: 0.6895 - recall:
0.3735 - val_accuracy: 0.2094 - val_auc: 0.6133 - val_loss: 109.4358 -
val_precision: 0.2042 - val_recall: 0.1794
Epoch 2/10
325/325          0s 99ms/step -
accuracy: 0.6982 - auc: 0.9585 - loss: 0.9682 - precision: 0.8071 - recall:
0.6014
Epoch 2: val_accuracy improved from 0.20942 to 0.68788, saving model to
./logs/augment-balanced/resnet50v2/captcha-07.09.2024 20-11-59//models/model-
epoch.02-val_loss.1.36-val_acc.0.69.keras
325/325          37s 112ms/step -
accuracy: 0.6983 - auc: 0.9585 - loss: 0.9678 - precision: 0.8072 - recall:
0.6015 - val_accuracy: 0.6879 - val_auc: 0.9493 - val_loss: 1.3572 -
val_precision: 0.7719 - val_recall: 0.6138
Epoch 3/10
325/325          0s 99ms/step -
accuracy: 0.7845 - auc: 0.9784 - loss: 0.6791 - precision: 0.8552 - recall:
0.7223
Epoch 3: val_accuracy improved from 0.68788 to 0.77058, saving model to
./logs/augment-balanced/resnet50v2/captcha-07.09.2024 20-11-59//models/model-
epoch.03-val_loss.0.74-val_acc.0.77.keras
325/325          37s 113ms/step -
accuracy: 0.7845 - auc: 0.9784 - loss: 0.6790 - precision: 0.8552 - recall:
0.7224 - val_accuracy: 0.7706 - val_auc: 0.9742 - val_loss: 0.7353 -
val_precision: 0.8327 - val_recall: 0.7227
Epoch 4/10
325/325          0s 99ms/step -
accuracy: 0.8381 - auc: 0.9871 - loss: 0.4944 - precision: 0.8887 - recall:
0.7945
Epoch 4: val_accuracy did not improve from 0.77058
325/325          36s 111ms/step -
accuracy: 0.8381 - auc: 0.9871 - loss: 0.4943 - precision: 0.8887 - recall:
0.7946 - val_accuracy: 0.7287 - val_auc: 0.9591 - val_loss: 0.9613 -
val_precision: 0.7704 - val_recall: 0.6948
Epoch 5/10
325/325          0s 99ms/step -
accuracy: 0.8783 - auc: 0.9928 - loss: 0.3532 - precision: 0.9114 - recall:
0.8525
Epoch 5: val_accuracy improved from 0.77058 to 0.81058, saving model to
./logs/augment-balanced/resnet50v2/captcha-07.09.2024 20-11-59//models/model-
epoch.05-val_loss.0.71-val_acc.0.81.keras
325/325          36s 112ms/step -
accuracy: 0.8783 - auc: 0.9928 - loss: 0.3531 - precision: 0.9114 - recall:

```

```
0.8525 - val_accuracy: 0.8106 - val_auc: 0.9730 - val_loss: 0.7065 -
val_precision: 0.8425 - val_recall: 0.7881
Epoch 6/10
325/325          0s 98ms/step -
accuracy: 0.8994 - auc: 0.9940 - loss: 0.3050 - precision: 0.9219 - recall:
0.8788
Epoch 6: val_accuracy improved from 0.81058 to 0.81327, saving model to
./logs/augment-balanced/resnet50v2/captcha-07.09.2024 20-11-59//models/model-
epoch.06-val_loss.0.73-val_acc.0.81.keras
325/325          36s 111ms/step -
accuracy: 0.8994 - auc: 0.9940 - loss: 0.3050 - precision: 0.9219 - recall:
0.8788 - val_accuracy: 0.8133 - val_auc: 0.9720 - val_loss: 0.7286 -
val_precision: 0.8330 - val_recall: 0.7990
Epoch 7/10
325/325          0s 99ms/step -
accuracy: 0.9162 - auc: 0.9961 - loss: 0.2447 - precision: 0.9327 - recall:
0.9028
Epoch 7: val_accuracy did not improve from 0.81327
325/325          36s 111ms/step -
accuracy: 0.9162 - auc: 0.9961 - loss: 0.2446 - precision: 0.9327 - recall:
0.9028 - val_accuracy: 0.7831 - val_auc: 0.9635 - val_loss: 0.8698 -
val_precision: 0.8155 - val_recall: 0.7625
Epoch 8/10
325/325          0s 98ms/step -
accuracy: 0.8677 - auc: 0.9872 - loss: 0.5578 - precision: 0.9088 - recall:
0.8309
Epoch 8: val_accuracy did not improve from 0.81327
325/325          36s 109ms/step -
accuracy: 0.8675 - auc: 0.9872 - loss: 0.5588 - precision: 0.9086 - recall:
0.8306 - val_accuracy: 0.7035 - val_auc: 0.9513 - val_loss: 1.0507 -
val_precision: 0.7601 - val_recall: 0.6679
Epoch 9/10
325/325          0s 98ms/step -
accuracy: 0.8584 - auc: 0.9890 - loss: 0.4287 - precision: 0.8973 - recall:
0.8308
Epoch 9: val_accuracy did not improve from 0.81327
325/325          36s 110ms/step -
accuracy: 0.8585 - auc: 0.9890 - loss: 0.4285 - precision: 0.8973 - recall:
0.8309 - val_accuracy: 0.8102 - val_auc: 0.9736 - val_loss: 0.6774 -
val_precision: 0.8474 - val_recall: 0.7879
Epoch 10/10
325/325          0s 99ms/step -
accuracy: 0.9316 - auc: 0.9973 - loss: 0.1944 - precision: 0.9460 - recall:
0.9189
Epoch 10: val_accuracy did not improve from 0.81327
325/325          36s 111ms/step -
accuracy: 0.9316 - auc: 0.9973 - loss: 0.1944 - precision: 0.9460 - recall:
0.9189 - val_accuracy: 0.7990 - val_auc: 0.9684 - val_loss: 0.7702 -
```

```
val_precision: 0.8220 - val_recall: 0.7850
```

```
[33]: <keras.src.callbacks.history.History at 0x7f1410487820>
```

4.3.3 Optimierung von InceptionV3

```
[34]: hparams[HP_MODEL_SELECTION] = 'inceptionv3'
logger.info(f"Ausgewähltes Modell: {hparams[HP_MODEL_SELECTION]}")

log_dir = f'./logs/{dataset_type}/{hparams[HP_MODEL_SELECTION]}/captcha-{time.
    strftime("%d.%m.%Y %H-%M-%S", time.localtime())}/'
model_dir = f'{log_dir}/models/' + 'model-epoch.{epoch:02d}-val_loss.{val_loss:.
    2f}-val_acc.{val_accuracy:.2f}.keras'
logger.info(f"Speichere LOGS in: '{log_dir}'")
logger.info(f"Speichere optimierte Modelle in: '{model_dir}'")

with tf.summary.create_file_writer(log_dir).as_default():
    hp.hparams(hparams)

callbacks = create_callbacks(log_dir, model_dir)
```

```
2024-09-07 20-18-44 - INFO - Ausgewähltes Modell: inceptionv3
```

```
2024-09-07 20-18-44 - INFO - Speichere LOGS in: './logs/augment-
balanced/inceptionv3/captcha-07.09.2024 20-18-44/'
```

```
2024-09-07 20-18-44 - INFO - Speichere optimierte Modelle in: './logs/augment-
balanced/inceptionv3/captcha-07.09.2024 20-18-44//models/model-
epoch.{epoch:02d}-val_loss.{val_loss:.2f}-val_acc.{val_accuracy:.2f}.keras'
```

```
[35]: from tensorflow.keras import metrics
```

```
inceptionv3.compile(optimizer=hparams[HP_TRAINING_OPTIMIZER],
                     loss=hparams[HP_TRAINING_LOSS],
                     metrics=[
                         metrics.CategoricalAccuracy(name = 'accuracy'),
                         metrics.Precision(name = 'precision'),
                         metrics.Recall(name = 'recall'),
                         metrics.AUC(name = 'auc')
                     ])
```

```
[36]: inceptionv3.fit(train_dataset, epochs=hparams[HP_TRAINING_EPOCHS], 
    validation_data=validation_dataset, callbacks=callbacks)
```

```
Epoch 1/10
325/325          0s 72ms/step -
accuracy: 0.6710 - auc: 0.9345 - loss: 1.1195 - precision: 0.8252 - recall:
0.5479
Epoch 1: val_accuracy improved from -inf to 0.64135, saving model to
./logs/augment-balanced/inceptionv3/captcha-07.09.2024 20-18-44//models/model-
epoch.01-val_loss.25.96-val_acc.0.64.keras
```

```
325/325          89s 114ms/step -
accuracy: 0.6712 - auc: 0.9346 - loss: 1.1186 - precision: 0.8253 - recall:
0.5483 - val_accuracy: 0.6413 - val_auc: 0.9094 - val_loss: 25.9643 -
val_precision: 0.6971 - val_recall: 0.5763
Epoch 2/10
325/325          0s 72ms/step -
accuracy: 0.8194 - auc: 0.9818 - loss: 0.6087 - precision: 0.8767 - recall:
0.7718
Epoch 2: val_accuracy improved from 0.64135 to 0.75577, saving model to
./logs/augment-balanced/inceptionv3/captcha-07.09.2024 20-18-44//models/model-
epoch.02-val_loss.0.91-val_acc.0.76.keras
325/325          28s 85ms/step -
accuracy: 0.8194 - auc: 0.9818 - loss: 0.6086 - precision: 0.8768 - recall:
0.7718 - val_accuracy: 0.7558 - val_auc: 0.9590 - val_loss: 0.9144 -
val_precision: 0.7957 - val_recall: 0.7273
Epoch 3/10
325/325          0s 72ms/step -
accuracy: 0.8639 - auc: 0.9902 - loss: 0.4291 - precision: 0.9011 - recall:
0.8314
Epoch 3: val_accuracy did not improve from 0.75577
325/325          27s 83ms/step -
accuracy: 0.8639 - auc: 0.9902 - loss: 0.4291 - precision: 0.9011 - recall:
0.8314 - val_accuracy: 0.0788 - val_auc: 0.5013 - val_loss: 659.5352 -
val_precision: 0.0789 - val_recall: 0.0779
Epoch 4/10
325/325          0s 73ms/step -
accuracy: 0.7942 - auc: 0.9738 - loss: 0.7853 - precision: 0.8590 - recall:
0.7343
Epoch 4: val_accuracy did not improve from 0.75577
325/325          27s 83ms/step -
accuracy: 0.7943 - auc: 0.9738 - loss: 0.7847 - precision: 0.8590 - recall:
0.7344 - val_accuracy: 0.1010 - val_auc: 0.5340 - val_loss: 43.1347 -
val_precision: 0.1013 - val_recall: 0.0842
Epoch 5/10
325/325          0s 72ms/step -
accuracy: 0.8135 - auc: 0.9788 - loss: 0.6765 - precision: 0.8680 - recall:
0.7643
Epoch 5: val_accuracy improved from 0.75577 to 0.85538, saving model to
./logs/augment-balanced/inceptionv3/captcha-07.09.2024 20-18-44//models/model-
epoch.05-val_loss.0.48-val_acc.0.86.keras
325/325          28s 85ms/step -
accuracy: 0.8136 - auc: 0.9788 - loss: 0.6759 - precision: 0.8681 - recall:
0.7644 - val_accuracy: 0.8554 - val_auc: 0.9862 - val_loss: 0.4795 -
val_precision: 0.8863 - val_recall: 0.8319
Epoch 6/10
325/325          0s 72ms/step -
accuracy: 0.8997 - auc: 0.9944 - loss: 0.3090 - precision: 0.9224 - recall:
0.8786
```

```

Epoch 6: val_accuracy improved from 0.85538 to 0.86442, saving model to
./logs/augment-balanced/inceptionv3/captcha-07.09.2024 20-18-44//models/model-
epoch.06-val_loss.0.44-val_acc.0.86.keras
325/325          28s 85ms/step -
accuracy: 0.8997 - auc: 0.9944 - loss: 0.3089 - precision: 0.9225 - recall:
0.8787 - val_accuracy: 0.8644 - val_auc: 0.9866 - val_loss: 0.4427 -
val_precision: 0.8875 - val_recall: 0.8469
Epoch 7/10
325/325          0s 72ms/step -
accuracy: 0.9200 - auc: 0.9962 - loss: 0.2403 - precision: 0.9342 - recall:
0.9061
Epoch 7: val_accuracy did not improve from 0.86442
325/325          27s 83ms/step -
accuracy: 0.9200 - auc: 0.9962 - loss: 0.2403 - precision: 0.9342 - recall:
0.9061 - val_accuracy: 0.8154 - val_auc: 0.9737 - val_loss: 0.6870 -
val_precision: 0.8322 - val_recall: 0.8067
Epoch 8/10
325/325          0s 72ms/step -
accuracy: 0.9052 - auc: 0.9933 - loss: 0.3163 - precision: 0.9196 - recall:
0.8921
Epoch 8: val_accuracy did not improve from 0.86442
325/325          27s 82ms/step -
accuracy: 0.9052 - auc: 0.9933 - loss: 0.3162 - precision: 0.9197 - recall:
0.8921 - val_accuracy: 0.8538 - val_auc: 0.9814 - val_loss: 0.5264 -
val_precision: 0.8733 - val_recall: 0.8433
Epoch 9/10
325/325          0s 72ms/step -
accuracy: 0.9332 - auc: 0.9972 - loss: 0.1985 - precision: 0.9428 - recall:
0.9240
Epoch 9: val_accuracy did not improve from 0.86442
325/325          27s 83ms/step -
accuracy: 0.9332 - auc: 0.9972 - loss: 0.1985 - precision: 0.9428 - recall:
0.9240 - val_accuracy: 0.8590 - val_auc: 0.9831 - val_loss: 0.5118 -
val_precision: 0.8721 - val_recall: 0.8481
Epoch 10/10
325/325          0s 72ms/step -
accuracy: 0.9416 - auc: 0.9978 - loss: 0.1712 - precision: 0.9499 - recall:
0.9358
Epoch 10: val_accuracy did not improve from 0.86442
325/325          27s 83ms/step -
accuracy: 0.9416 - auc: 0.9978 - loss: 0.1711 - precision: 0.9499 - recall:
0.9358 - val_accuracy: 0.8202 - val_auc: 0.9701 - val_loss: 0.7682 -
val_precision: 0.8310 - val_recall: 0.8123

```

[36]: <keras.src.callbacks.history.History at 0x7f13943511e0>

4.3.4 Optimierung von LeNet5

```
[37]: hparams[HP_MODEL_SELECTION] = 'leNet5'
logger.info(f"Ausgewähltes Modell: {hparams[HP_MODEL_SELECTION]}")

log_dir = f'./logs/{dataset_type}/{hparams[HP_MODEL_SELECTION]}/captcha-{time.
    strftime("%d.%m.%Y %H-%M-%S", time.localtime())}/'
model_dir = f'{log_dir}/models/' + 'model-epoch.{epoch:02d}-val_loss.{val_loss:.
    2f}-val_acc.{val_accuracy:.2f}.keras'
logger.info(f"Speichere LOGS in: '{log_dir}'")
logger.info(f"Speichere optimierte Modelle in: '{model_dir}'")

with tf.summary.create_file_writer(log_dir).as_default():
    hp.hparams(hparams)

callbacks = create_callbacks(log_dir, model_dir)
```

2024-09-07 20-24-19 - INFO - Ausgewähltes Modell: leNet5
2024-09-07 20-24-19 - INFO - Speichere LOGS in: './logs/augment-
balanced/leNet5/captcha-07.09.2024 20-24-19/'
2024-09-07 20-24-19 - INFO - Speichere optimierte Modelle in: './logs/augment-
balanced/leNet5/captcha-07.09.2024 20-24-19//models/model-
epoch.{epoch:02d}-val_loss.{val_loss:.2f}-val_acc.{val_accuracy:.2f}.keras'

```
[38]: from tensorflow.keras import metrics

leNet5.compile(optimizer=hparams[HP_TRAINING_OPTIMIZER],
               loss=hparams[HP_TRAINING_LOSS],
               metrics=[
                   metrics.CategoricalAccuracy(name = 'accuracy'),
                   metrics.Precision(name = 'precision'),
                   metrics.Recall(name = 'recall'),
                   metrics.AUC(name = 'auc')
               ])
```

```
[39]: leNet5.fit(train_dataset, epochs=hparams[HP_TRAINING_EPOCHS],  

    validation_data=validation_dataset, callbacks=callbacks)
```

Epoch 1/10
20/325 2s 9ms/step -
accuracy: 0.0937 - auc: 0.5118 - loss: 196.2723 - precision: 0.0948 - recall:
0.0901
2024-09-07 20:24:23.050941: I
external/local_xla/xla/stream_executor/cuda/cuda_assembly_compiler.cc:393] ptxas
warning : Registers are spilled to local memory in function
'input_reduce_select_fusion', 8 bytes spill stores, 8 bytes spill loads

320/325 0s 9ms/step -
accuracy: 0.1758 - auc: 0.6128 - loss: 35.4630 - precision: 0.1944 - recall:

```
0.0336
Epoch 1: val_accuracy improved from -inf to 0.32750, saving model to
./logs/augment-balanced/leNet5/captcha-07.09.2024 20-24-19//models/model-
epoch.01-val_loss.2.12-val_acc.0.33.keras
325/325          8s 15ms/step -
accuracy: 0.1769 - auc: 0.6140 - loss: 34.9923 - precision: 0.1968 - recall:
0.0336 - val_accuracy: 0.3275 - val_auc: 0.7854 - val_loss: 2.1249 -
val_precision: 0.5753 - val_recall: 0.1110
Epoch 2/10
324/325          0s 9ms/step -
accuracy: 0.4064 - auc: 0.8332 - loss: 1.9009 - precision: 0.6834 - recall:
0.1549
Epoch 2: val_accuracy improved from 0.32750 to 0.41135, saving model to
./logs/augment-balanced/leNet5/captcha-07.09.2024 20-24-19//models/model-
epoch.02-val_loss.1.90-val_acc.0.41.keras
325/325          3s 10ms/step -
accuracy: 0.4066 - auc: 0.8333 - loss: 1.9004 - precision: 0.6835 - recall:
0.1551 - val_accuracy: 0.4113 - val_auc: 0.8372 - val_loss: 1.9035 -
val_precision: 0.6305 - val_recall: 0.2192
Epoch 3/10
319/325          0s 9ms/step -
accuracy: 0.5474 - auc: 0.9059 - loss: 1.4569 - precision: 0.7559 - recall:
0.3181
Epoch 3: val_accuracy improved from 0.41135 to 0.45212, saving model to
./logs/augment-balanced/leNet5/captcha-07.09.2024 20-24-19//models/model-
epoch.03-val_loss.1.87-val_acc.0.45.keras
325/325          3s 10ms/step -
accuracy: 0.5479 - auc: 0.9061 - loss: 1.4555 - precision: 0.7562 - recall:
0.3188 - val_accuracy: 0.4521 - val_auc: 0.8531 - val_loss: 1.8683 -
val_precision: 0.6199 - val_recall: 0.2963
Epoch 4/10
324/325          0s 9ms/step -
accuracy: 0.6460 - auc: 0.9454 - loss: 1.1099 - precision: 0.8186 - recall:
0.4819
Epoch 4: val_accuracy improved from 0.45212 to 0.45788, saving model to
./logs/augment-balanced/leNet5/captcha-07.09.2024 20-24-19//models/model-
epoch.04-val_loss.2.07-val_acc.0.46.keras
325/325          3s 10ms/step -
accuracy: 0.6461 - auc: 0.9454 - loss: 1.1096 - precision: 0.8187 - recall:
0.4820 - val_accuracy: 0.4579 - val_auc: 0.8494 - val_loss: 2.0716 -
val_precision: 0.5749 - val_recall: 0.3615
Epoch 5/10
323/325          0s 8ms/step -
accuracy: 0.7190 - auc: 0.9644 - loss: 0.8961 - precision: 0.8465 - recall:
0.5818
Epoch 5: val_accuracy improved from 0.45788 to 0.47135, saving model to
./logs/augment-balanced/leNet5/captcha-07.09.2024 20-24-19//models/model-
epoch.05-val_loss.2.26-val_acc.0.47.keras
```

```
325/325          3s 10ms/step -
accuracy: 0.7191 - auc: 0.9644 - loss: 0.8956 - precision: 0.8466 - recall:
0.5820 - val_accuracy: 0.4713 - val_auc: 0.8450 - val_loss: 2.2592 -
val_precision: 0.5587 - val_recall: 0.3969
Epoch 6/10
319/325          0s 8ms/step -
accuracy: 0.7758 - auc: 0.9776 - loss: 0.7015 - precision: 0.8820 - recall:
0.6752
Epoch 6: val_accuracy did not improve from 0.47135
325/325          3s 10ms/step -
accuracy: 0.7760 - auc: 0.9776 - loss: 0.7011 - precision: 0.8821 - recall:
0.6755 - val_accuracy: 0.4675 - val_auc: 0.8323 - val_loss: 2.5761 -
val_precision: 0.5436 - val_recall: 0.4037
Epoch 7/10
325/325          0s 8ms/step -
accuracy: 0.8150 - auc: 0.9845 - loss: 0.5795 - precision: 0.8964 - recall:
0.7352
Epoch 7: val_accuracy improved from 0.47135 to 0.47538, saving model to
./logs/augment-balanced/leNet5/captcha-07.09.2024 20-24-19//models/model-
epoch.07-val_loss.2.97-val_acc.0.48.keras
325/325          3s 10ms/step -
accuracy: 0.8150 - auc: 0.9845 - loss: 0.5795 - precision: 0.8964 - recall:
0.7353 - val_accuracy: 0.4754 - val_auc: 0.8255 - val_loss: 2.9653 -
val_precision: 0.5328 - val_recall: 0.4275
Epoch 8/10
323/325          0s 8ms/step -
accuracy: 0.8544 - auc: 0.9897 - loss: 0.4639 - precision: 0.9154 - recall:
0.7933
Epoch 8: val_accuracy did not improve from 0.47538
325/325          3s 10ms/step -
accuracy: 0.8544 - auc: 0.9897 - loss: 0.4639 - precision: 0.9154 - recall:
0.7933 - val_accuracy: 0.4700 - val_auc: 0.8238 - val_loss: 3.0465 -
val_precision: 0.5171 - val_recall: 0.4271
Epoch 9/10
319/325          0s 8ms/step -
accuracy: 0.8768 - auc: 0.9922 - loss: 0.3946 - precision: 0.9277 - recall:
0.8281
Epoch 9: val_accuracy improved from 0.47538 to 0.48212, saving model to
./logs/augment-balanced/leNet5/captcha-07.09.2024 20-24-19//models/model-
epoch.09-val_loss.3.90-val_acc.0.48.keras
325/325          3s 9ms/step -
accuracy: 0.8767 - auc: 0.9921 - loss: 0.3948 - precision: 0.9276 - recall:
0.8280 - val_accuracy: 0.4821 - val_auc: 0.8126 - val_loss: 3.8984 -
val_precision: 0.5151 - val_recall: 0.4600
Epoch 10/10
320/325          0s 8ms/step -
accuracy: 0.8912 - auc: 0.9932 - loss: 0.3463 - precision: 0.9304 - recall:
0.8518
```

```

Epoch 10: val_accuracy improved from 0.48212 to 0.49577, saving model to
./logs/augment-balanced/leNet5/captcha-07.09.2024 20-24-19//models/model-
epoch.10-val_loss.3.92-val_acc.0.50.keras
325/325          3s 9ms/step -
accuracy: 0.8912 - auc: 0.9932 - loss: 0.3461 - precision: 0.9305 - recall:
0.8519 - val_accuracy: 0.4958 - val_auc: 0.8115 - val_loss: 3.9227 -
val_precision: 0.5247 - val_recall: 0.4717

```

[39]: <keras.src.callbacks.history.History at 0x7f130c381d80>

1.9.4 4.4 Vorhersage anhand optimiertem Modell

```

[40]: import numpy as np
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
from tensorflow import keras

loaded_model = keras.models.load_model("logs/final/augment-unbalanced/
                                         inceptionv3/captcha-05.09.2024 23-00-32/models/model-epoch.04-val_loss.0.
                                         38-val_acc.0.87.keras")

folders = ["Bicycle", "Bridge", "Bus", "Car", "Chimney", "Crosswalk",
           "Hydrant", "Motorcycle", "Mountain", "Other", "Palm", "Stair", „
           „TLight”]

image_paths = [
    'Google-Recaptcha-V2-Images/Hydrant/0a05f251-260f-4ada-9005-5326e50e1848.
    jpg',
    'Google-Recaptcha-V2-Images/Bus/0ae6ac38-005c-4519-9e4c-8d72cc7f4d45.jpg',
    'Google-Recaptcha-V2-Images/Bicycle/0b433101-7a68-4b29-b875-ac45c9680489.
    jpg',
    'Google-Recaptcha-V2-Images/Car/0f8723fa-5ec7-409d-aac9-5e845cdba592.jpg',
    'Google-Recaptcha-V2-Images/Bridge/0a91630a-db06-4fb4-bba1-17ae331db395.
    jpg',
    'Google-Recaptcha-V2-Images/Chimney/
    Chimney$95df5fdc7a8d1f84ba73fbc13820b215.png',
    'Google-Recaptcha-V2-Images/Crosswalk/7c1cff3c-ed14-4434-a8bb-83f3c80150b8.
    jpg',
    'Google-Recaptcha-V2-Images/TLight/00b93f32-ef8b-4bf1-b80a-0c015e8d49cd.
    jpg',
    'Google-Recaptcha-V2-Images/Palm/1cc7ea4e-3204-4374-8335-c9f9d2f22dd3.jpg',
    'Google-Recaptcha-V2-Images/Stair/0ther$7a093d9078b7770b79b371ecbaf1e238.
    png',

```

```

'Google-Recaptcha-V2-Images/Motorcycle/
↳Motorcycle$2e17b45892f5061a2dca5d109f52a304.png',
'Google-Recaptcha-V2-Images/Mountain/
↳Mountain$a971ebef63e9af4221cad93dc9260199.png',
'Google-Recaptcha-V2-Images/Other/Other$0a3119044a5e6776dba11c9b06338c00.
↳png',
'Google-Recaptcha-V2-Images/Mountain/
↳Mountain$399053106902e615176bb63ce1f8b9b3.png',
'Google-Recaptcha-V2-Images/Car/3f3210b9-8116-4025-9619-1b97a1c2b008.jpg',
'Google-Recaptcha-V2-Images/Bridge/4b7b4b28-42e7-4219-9050-0c3233ab6111.jpg'
]

real_labels = ['Hydrant', 'Bus', 'Bicycle', 'Car', 'Bridge', 'Chimney', ↳
↳ 'Crosswalk', 'TLight', 'Palm', 'Stair', 'Motorcycle', ↳
↳ 'Mountain', 'Other', 'Mountain', 'Car', 'Bridge' ]

fig, axs = plt.subplots(4, 4, figsize=(10, 10))
axs = axs.ravel()

for i, img_path in enumerate(image_paths):
    img = image.load_img(img_path, target_size=(120, 120))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)

    prediction = loaded_model.predict(img_array)
    predicted_label_index = np.argmax(prediction, axis=1)[0]
    predicted_label = folders[predicted_label_index]

    confidence = prediction[0][predicted_label_index] * 100

    axs[i].imshow(image.load_img(img_path))
    axs[i].axis('off')
    axs[i].set_title(f'Real: {real_labels[i]}\nPredicted: ↳
↳ {predicted_label}\nConfidence: {confidence:.2f}%')

plt.tight_layout()
plt.show()

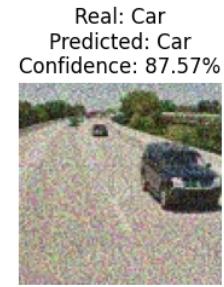
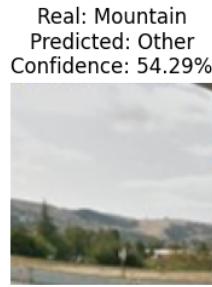
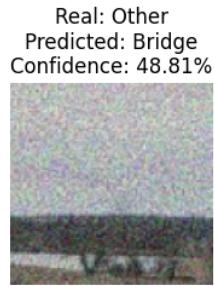
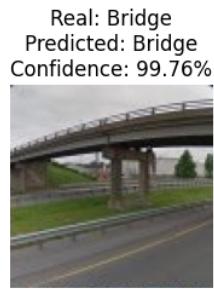
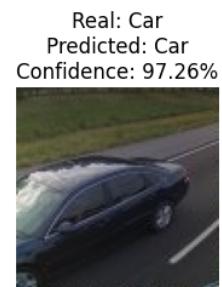
```

```

1/1          7s 7s/step
1/1          0s 20ms/step
1/1          0s 19ms/step
1/1          0s 20ms/step
1/1          0s 24ms/step
1/1          0s 19ms/step

```

1/1 0s 18ms/step
1/1 0s 17ms/step
1/1 0s 17ms/step
1/1 0s 18ms/step
1/1 0s 25ms/step
1/1 0s 18ms/step
1/1 0s 19ms/step
1/1 0s 23ms/step
1/1 0s 17ms/step
1/1 0s 17ms/step



1.10 5. Evaluation und Bewertung der Optimierungsprozesse

1.10.1 5.1 RQ 1: Welche Veränderung der Modellgüte kann mit aktuellen Methoden der Vorverarbeitung und der Datenaugmentierung aus Forschung und Praxis erzielt werden?

Mit Datenvorverarbeitung

Ohne Datenvorverarbeitung

Alternativ Nur Zoom, Nur Kontrast, Nur Helligkeit, Nur Spiegelung

1.10.2 5.2 RQ 2: Was sind die neuesten Entwicklungen (State-of-the-Art) in der Bildverarbeitung mit maschinellem Lernen, insbesondere bei der Verwendung von tiefen neuronalen Netzen wie Inceptionv3?

5.2.1 ResNet50V2

5.2.2 InceptionV3

5.2.3 LeNet5

1.10.3 RQ 3: Welche in der Forschung bestehenden Metriken zur Klassifikation eignen sich zur Lösung des oben beschriebenen Anwendungsfalls?

1.11 Aufteilung der Gruppenleistung

```
[41]: import pandas as pd

# Manuelle Erstellung des JSON-Datensatzes basierend auf den CSV-Daten
data = {
    "Kapitel": [
        "Set up der Entwicklungsumgebung",
        "Laden der Daten",
        "Explorative Datenanalyse",
        "Erstellen eines Trainingsdatensatzes",
        "Entwicklung und Optimierung von neuronalen Netzarchitekturen",
        "Evaluation und Bewertung der Optimierungsprozesse",
        "Dokumentation",
        "Literatur"
    ],
    "Leon (%)": [70, 30, 20, 30, 60, 40, 33.3, 25],
    "Niklas (%)": [15, 10, 40, 25, 20, 50, 33.3, 55],
    "Rares (%)": [15, 60, 40, 45, 20, 10, 33.3, 20],
    "Hauptverantwortlichen": [
        "Leon", "Rares", "Niklas, Rares", "Rares", "Leon", "Niklas",
        "Sneezy-Team", "Niklas"
    ]
}
```

```
# Erstellen des DataFrames  
df = pd.DataFrame(data)  
df
```

[41]:

	Kapitel	Leon (%)	Niklas (%)	\
0	Set up der Entwicklungsumgebung	70.0	15.0	
1	Laden der Daten	30.0	10.0	
2	Explorative Datenanalyse	20.0	40.0	
3	Erstellen eines Trainingsdatensatzes	30.0	25.0	
4	Entwicklung und Optimierung von neuronalen Net...	60.0	20.0	
5	Evaluation und Bewertung der Optimierungsprozesse	40.0	50.0	
6	Dokumentation	33.3	33.3	
7	Literatur	25.0	55.0	

	Rares (%)	Hauptverantwortlichen
0	15.0	Leon
1	60.0	Rares
2	40.0	Niklas, Rares
3	45.0	Rares
4	20.0	Leon
5	10.0	Niklas
6	33.3	Sneezy-Team
7	20.0	Niklas