In [ ]:
```
"""
============================
Project Report
============================

Authors: Gössl Marcel, Marek Simon, Schrenk Dominik, Unger Miriam
Date: 06.11.2025
Course: Computer Vision
Github Repo: https://github.com/Dom4i/CV_GarbageProject

----------------------------------------
1. Dataset Description
----------------------------------------
• What dataset did you use?
    Bei dem Datensatz handelt es sich um Bilder von verschiedenen Abfalltypen. D
    Die Bilder des Datensatzes haben die Dimension 512x384 (pixel)
• How many images/classes does it contain?
    Gesamtanzahl der Bilder: 2527
    Klassen:
        Cardboard: 403
        Glass: 501
        Metal: 410
        Paper: 594
        Plastic: 482
        Trash: 137

• Was the dataset balanced or imbalanced?
    Mit Ausnahme der Klasse "Trash" beeinhaltet jede Klasse zwischen 403 und 594
    Die Klasse "Trash" hat allerdings nur 137 Bilder und damit deutlich weniger
• What was the imaging source or environment?
    Die Bilder wurden mit einer optischen Kamera aufgenommen
    Es handelt sich um Abfall, der auf einem einfärbigen Hintergrund aufgenommen
        Daher wird vermutet, dass (mit Ausnahme der "Cardboard" und der "Paper"
• How did you preprocess or split the data?
    Für das Training wurden die Bilder auf 224×224 Pixel skaliert (aus Performan

    Datenaugmentation: Zur Robustheit gegenüber leichten Lage- und Helligkeitsän
----------------------------------------
2. Relation to Real Applications
----------------------------------------
• How does this project relate to real-world use cases?
• In what kind of system could your model be applied?
• What practical benefits could it have?

    Die Mülltrennung ist essentiell für die Verarbeitung von Abfall. Da große Ab
    Eine Beispielanwendung für ein Klassifizierungsmodell von Müllarten kann die
        Z.B. Überprüfung von schon getrenntem Müll (der von Haushalten bereits i
        Z.B. Trennung von Mischungen aus verschiedenen Müllarten (z.B. öffentlic

----------------------------------------
3. Problems and Solutions
----------------------------------------
• Which problems or challenges did you encounter?
    Während der Entwicklung traten mehrere Herausforderungen auf. Besonders prob
    von kaputten Gegenständen bis zu Essensresten. Diese fehlende visuelle Einhe

    Darüber hinaus kam es regelmäßig zu Verwechslungen zwischen den Klassen „Met
    die Oberflächen und Formen dieser Materialien stark ähneln und teilweise ref
```

```
• How did you solve or work around them?
    Um diese Probleme abzumildern, wurde eine Reihe von Optimierungen und Experi
    Durch Datenaugmentation (z. B. Rotation, Flip, leichte Farbvariationen) wurd
    Mit Klassengewichten wurde der Einfluss der seltenen Trash-Bilder im Trainin
    Zusätzlich wurden verschiedene Hyperparameter wie Lernrate, Batchgröße, Epoc

    In einem weiteren Versuch wurde die Klasse „Trash“ vollständig entfernt, wod
    Dennoch blieb die Klasse im finalen Modell enthalten, um dem ursprünglichen

----------------------------------------
4. Implementation Details
----------------------------------------
    Das Projekt wurde vollständig in Python umgesetzt, unter Verwendung der Bibl
    Zusätzlich kamen torchvision (für Datenvorverarbeitung und Augmentation) sow

    Zunächst wurden die Bilder mit passenden Transforms (Resize, Normalisierung,
    Anschließend wurde ein eigenes Convolutional Neural Network (SmallCNN) entwo
    Dieses Modell wurde bewusst kompakt gehalten, um auch auf Geräten ohne High-

    Für das Training wurde der AdamW-Optimizer mit einer lernratenbasierten Anpa

----------------------------------------
5. Results and Evaluation
----------------------------------------
    Das Modell wurde in mehreren Trainingsdurchläufen mit unterschiedlichen Para
    – mit moderater Datenaugmentation und optimaler Epochenwahl – eine Accuracy

    In einer Zwischenversion, in der die Klasse „Trash“ vollständig entfernt wur
    Dieses Ergebnis bestätigte, dass die stark variierende und unterrepräsentier

    Zur Auswertung wurden eine Confusion Matrix sowie ein Classification Report
    Zusätzlich wurden mehrere Beispielbilder und Grad-CAM-Visualisierungen erste
    Diese Visualisierungen werden auch in der Präsentation verwendet, um das Ver

----------------------------------------
6. Discussion and Learnings
----------------------------------------
    Insgesamt zeigte das Projekt sehr deutlich, wie stark die Ergebnisse bei Dee
    Schon kleine Änderungen an Lernrate, Batchgröße oder Augmentation führten zu
    sondern dass viel Ausprobieren, Beobachten und Anpassen (und Warten) notwend

    Ein weiteres wichtiges Learning war der Einfluss der Hardware-Performance. W
    mussten andere Teammitglieder ohne dedizierte GPU oft 10–15 Minuten pro Trai
    die Bildgröße auf 224×224 Pixel zu reduzieren, um die Rechenzeit zu verkürze

    Trotz dieser Hürden war das Projekt lehrreich und zeigte, dass praktische Er
    Wir konnten ein grundsätzlich funktionierendes Modell entwickeln, verstehen,
"""
```

In [5]:

```
# Projekt-Pfad setzen und prüfen
import sys
from pathlib import Path
proj_root = Path(r"C:\Users\Dominik\Desktop\FH\5. SEMESTER\Computer Vision und N
if str(proj_root) not in sys.path:
    sys.path.insert(0, str(proj_root))
print('Project root:', proj_root)
data_dir = proj_root / 'data' / 'TrashType_Image_Dataset'
models_dir = proj_root / 'models'
print('Expected data directory:', data_dir)
```

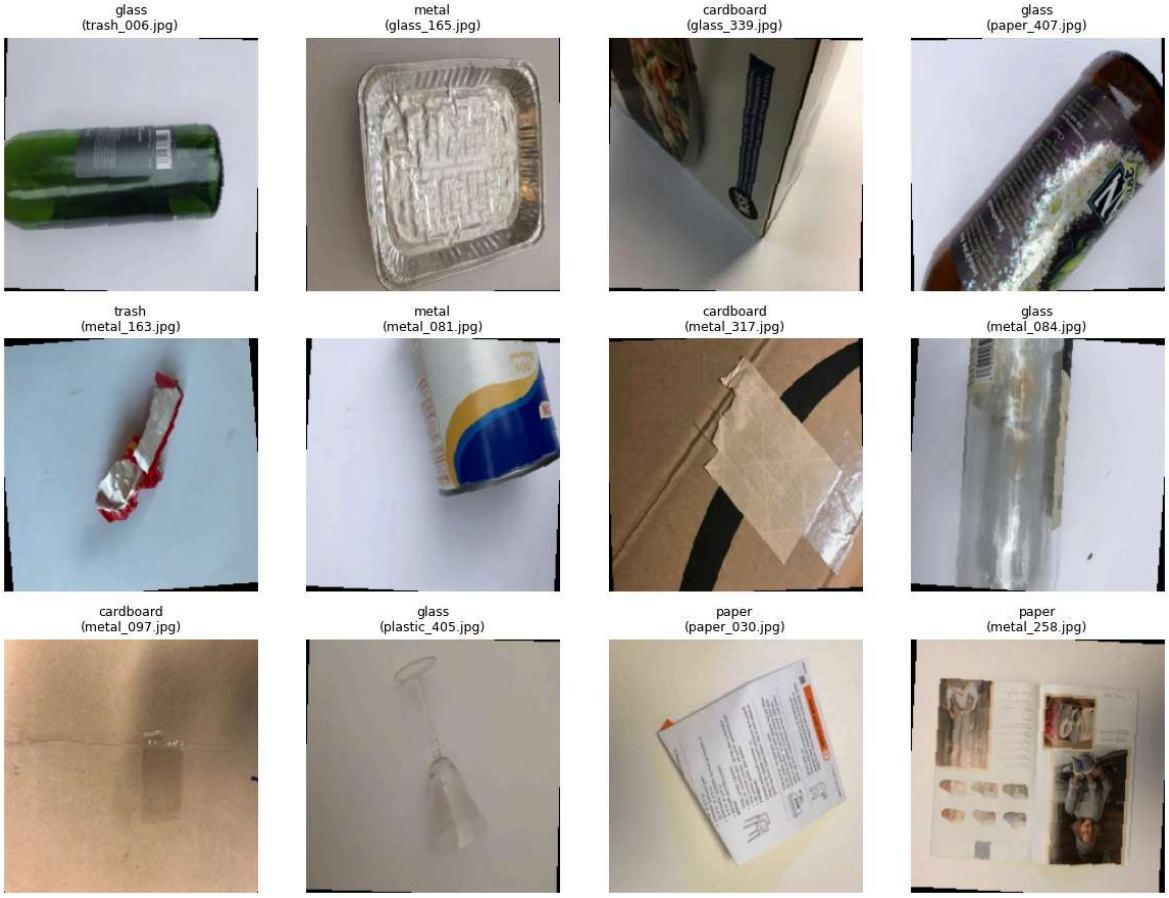
```
print('Models directory:', models_dir)
if not data_dir.exists():
    print('\nHinweis: Datensatz nicht gefunden. Bitte lade den Kaggle-Datensatz
    print(data_dir)
else:
    print('Datensatz gefunden, Notebook kann fortfahren.')
```

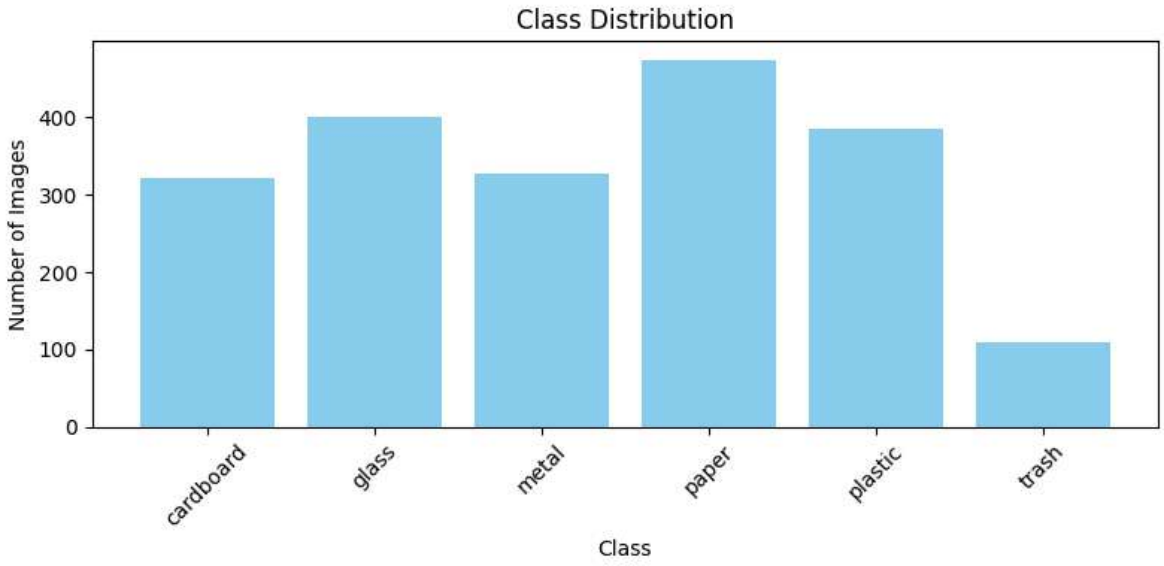
```
Project root: C:\Users\Dominik\Desktop\FH\5. SEMESTER\Computer Vision und Natural
Language Processing\CV_GarbageProject
Expected data directory: C:\Users\Dominik\Desktop\FH\5. SEMESTER\Computer Vision
und Natural Language Processing\CV_GarbageProject\data\TrashType_Image_Dataset
Models directory: C:\Users\Dominik\Desktop\FH\5. SEMESTER\Computer Vision und Nat
ural Language Processing\CV_GarbageProject\models
Datensatz gefunden, Notebook kann fortfahren.
```

In [6]:

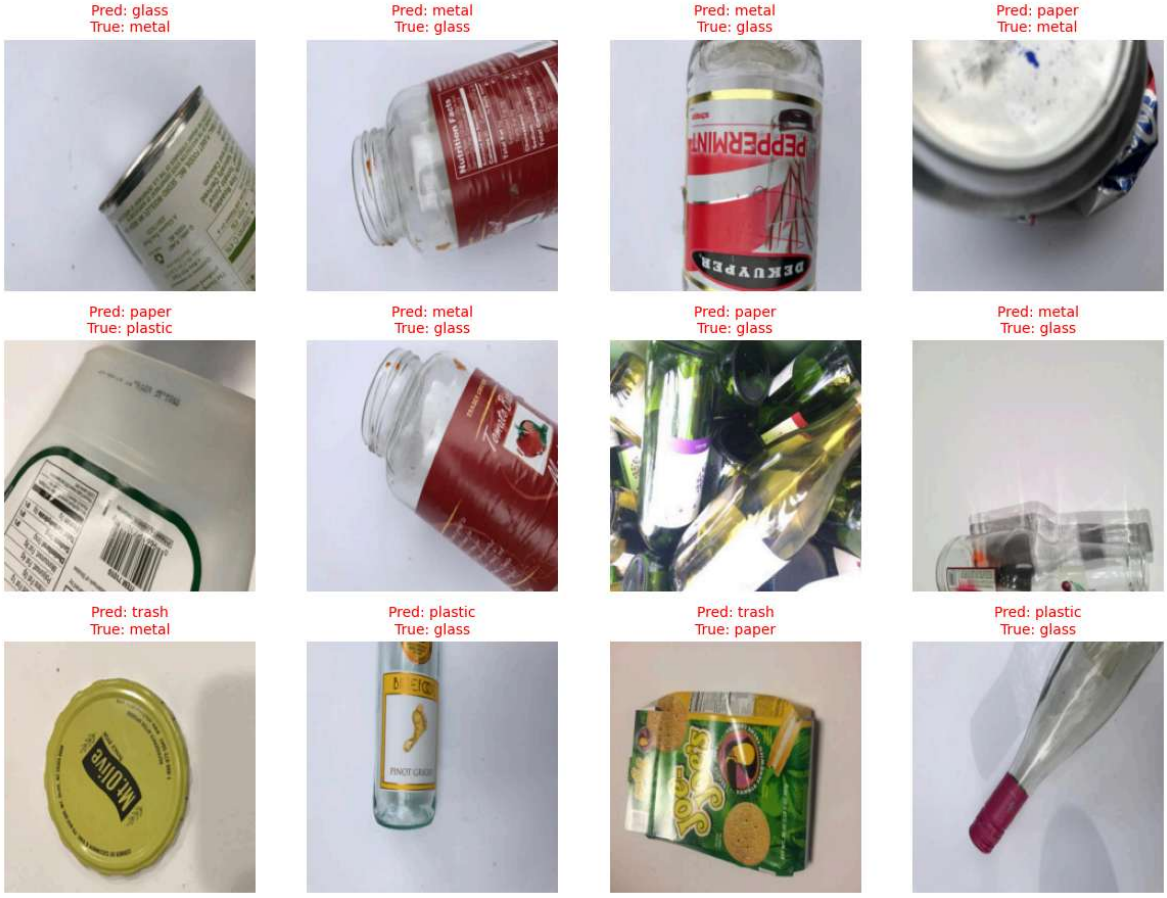
```
# Demo: fit(...) mit 35 Epochen (vorsichtig: langer Lauf!)
from src.train import fit
import traceback

if not data_dir.exists():
    print('Datensatz fehlt – überspringe fit(...)')
else:
    try:
        fit(
            data_dir=str(data_dir),
            models_dir=str(models_dir),
            img_size=224,
            batch_size=96,
            epochs=35,
            lr=1e-3,
            val_split=0.2,
            use_class_weights=True,
            seed=42,
            preview=True
        )
    except Exception:
        print('Fehler beim Ausführen von fit():')
        traceback.print_exc()
```

```
Device: cuda
Dataloaders ready | GPU=Yes | Workers=14 | Batch=96 | Augment=On
```


glass
(trash_006.jpg)
metal
(glass_165.jpg)
cardboard
(glass_339.jpg)
glass
(paper_407.jpg)
trash
(metal_163.jpg)
metal
(metal_081.jpg)
cardboard
(metal_317.jpg)
glass
(metal_084.jpg)
cardboard
(metal_097.jpg)
glass
(plastic_405.jpg)
paper
(paper_030.jpg)
paper
(metal_258.jpg)

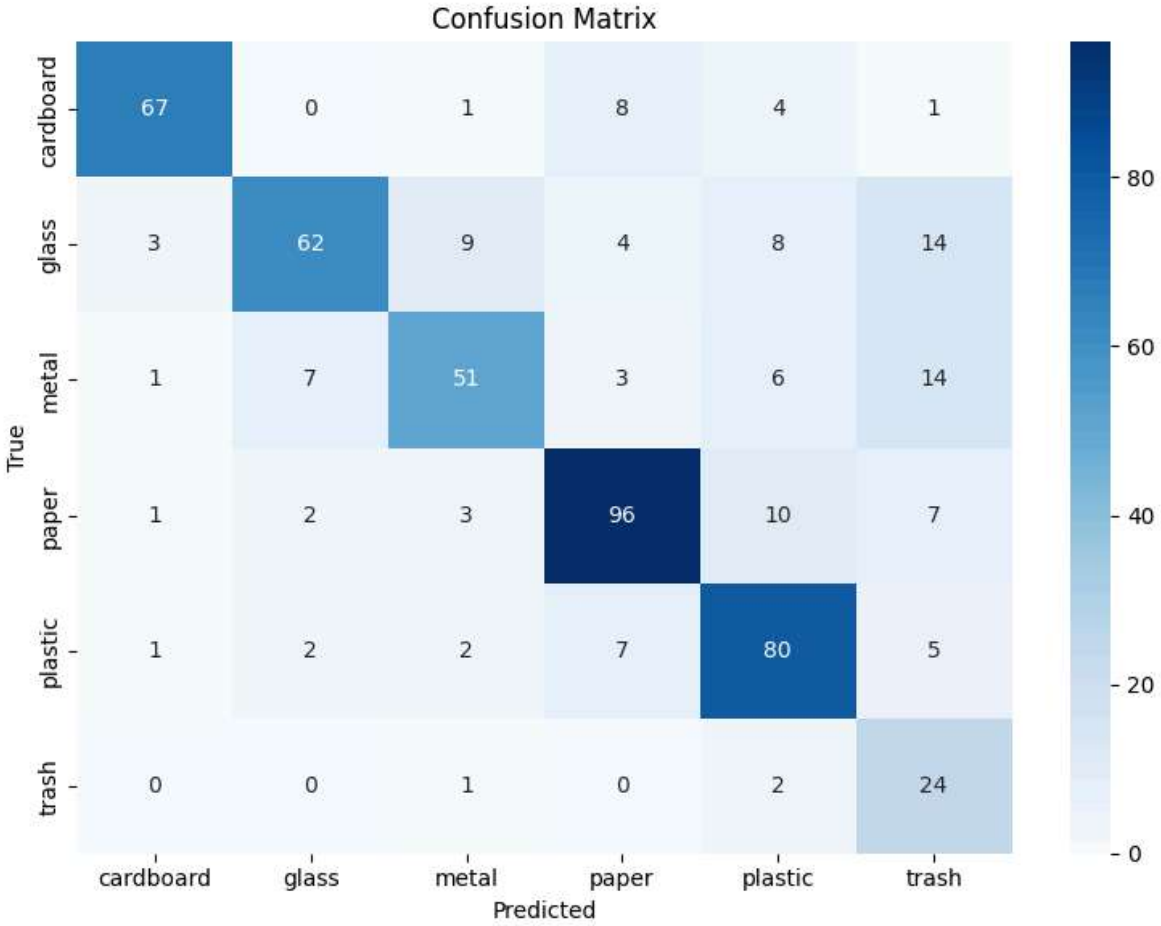

Class Distribution
Number of Images
0
100
200
300
400
cardboard
glass
metal
paper
plastic
trash
Class

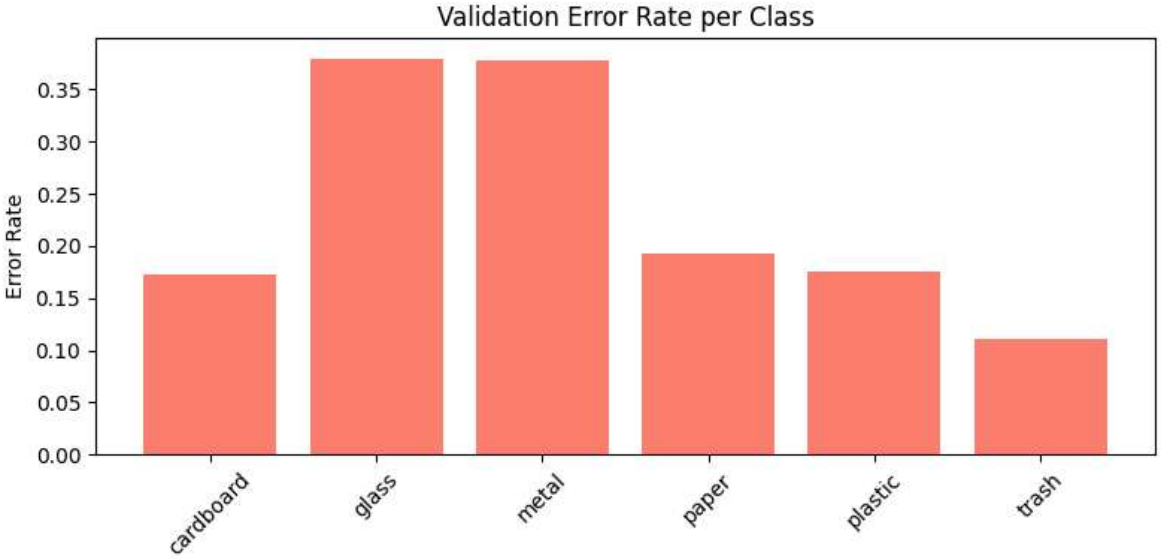
```
Epoch 01: train_loss=1.5527 acc=0.412 | val_loss=1.5779 acc=0.395 | 25.6s
Epoch 02: train_loss=1.4272 acc=0.478 | val_loss=1.3590 acc=0.516 | 2.0s
Epoch 03: train_loss=1.3736 acc=0.510 | val_loss=1.3467 acc=0.536 | 2.0s
Epoch 04: train_loss=1.3751 acc=0.529 | val_loss=1.2473 acc=0.632 | 2.0s
Epoch 05: train_loss=1.3633 acc=0.522 | val_loss=1.3483 acc=0.490 | 1.9s
Epoch 06: train_loss=1.3108 acc=0.556 | val_loss=1.3228 acc=0.532 | 1.9s
Epoch 07: train_loss=1.3305 acc=0.553 | val_loss=1.2408 acc=0.605 | 1.9s
Epoch 08: train_loss=1.3421 acc=0.521 | val_loss=1.3067 acc=0.585 | 1.9s
Epoch 09: train_loss=1.2882 acc=0.573 | val_loss=1.2921 acc=0.542 | 1.9s
Epoch 10: train_loss=1.2677 acc=0.573 | val_loss=1.3064 acc=0.605 | 1.9s
Epoch 11: train_loss=1.2277 acc=0.611 | val_loss=1.2622 acc=0.597 | 1.9s
Epoch 12: train_loss=1.2510 acc=0.599 | val_loss=1.1783 acc=0.676 | 1.9s
Epoch 13: train_loss=1.2069 acc=0.634 | val_loss=1.1377 acc=0.662 | 2.0s
Epoch 14: train_loss=1.1849 acc=0.636 | val_loss=1.1833 acc=0.656 | 1.8s
Epoch 15: train_loss=1.1976 acc=0.640 | val_loss=1.1742 acc=0.672 | 1.9s
Epoch 16: train_loss=1.2005 acc=0.616 | val_loss=1.1932 acc=0.664 | 1.9s
Epoch 17: train_loss=1.1671 acc=0.646 | val_loss=1.1490 acc=0.676 | 2.0s
Epoch 18: train_loss=1.1734 acc=0.648 | val_loss=1.1273 acc=0.666 | 2.0s
Epoch 19: train_loss=1.1580 acc=0.667 | val_loss=1.1120 acc=0.706 | 1.9s
Epoch 20: train_loss=1.1500 acc=0.661 | val_loss=1.1233 acc=0.682 | 2.0s
Epoch 21: train_loss=1.1519 acc=0.679 | val_loss=1.0963 acc=0.733 | 2.0s
Epoch 22: train_loss=1.1426 acc=0.680 | val_loss=1.1321 acc=0.702 | 1.9s
Epoch 23: train_loss=1.1305 acc=0.690 | val_loss=1.0732 acc=0.715 | 2.0s
Epoch 24: train_loss=1.1232 acc=0.687 | val_loss=1.0965 acc=0.708 | 1.9s
Epoch 25: train_loss=1.1315 acc=0.674 | val_loss=1.0723 acc=0.719 | 2.0s
Epoch 26: train_loss=1.1212 acc=0.683 | val_loss=1.0964 acc=0.721 | 1.9s
Epoch 27: train_loss=1.1248 acc=0.670 | val_loss=1.0739 acc=0.719 | 2.0s
Epoch 28: train_loss=1.1214 acc=0.694 | val_loss=1.1093 acc=0.660 | 2.0s
Epoch 29: train_loss=1.1029 acc=0.694 | val_loss=1.0596 acc=0.721 | 2.0s
Epoch 30: train_loss=1.1007 acc=0.692 | val_loss=1.0751 acc=0.747 | 2.0s
Epoch 31: train_loss=1.0956 acc=0.701 | val_loss=1.0419 acc=0.733 | 1.9s
Epoch 32: train_loss=1.1096 acc=0.690 | val_loss=1.0512 acc=0.715 | 1.9s
Epoch 33: train_loss=1.0897 acc=0.703 | val_loss=1.0476 acc=0.753 | 1.9s
Epoch 34: train_loss=1.0793 acc=0.716 | val_loss=1.0453 acc=0.733 | 1.9s
Epoch 35: train_loss=1.0830 acc=0.722 | val_loss=1.0353 acc=0.751 | 1.9s

Classification report (val):
              precision    recall  f1-score   support

   cardboard       0.92      0.83      0.87        81
       glass       0.85      0.62      0.72       100
       metal       0.76      0.62      0.68        82
       paper       0.81      0.81      0.81       119
     plastic       0.73      0.82      0.77        97
       trash       0.37      0.89      0.52        27

    accuracy                           0.75       506
   macro avg       0.74      0.76      0.73       506
weighted avg       0.79      0.75      0.76       506

Confusion matrix (val):
[[67  0  1  8  4  1]
 [ 3 62  9  4  8 14]
 [ 1  7 51  3  6 14]
 [ 1  2  3 96 10  7]
 [ 1  2  2  7 80  5]
 [ 0  0  1  0  2 24]]

Best checkpoint saved to: C:\Users\Dominik\Desktop\FH\5. SEMESTER\Computer Vision
und Natural Language Processing\CV_GarbageProject\models\best_weights.pt
```

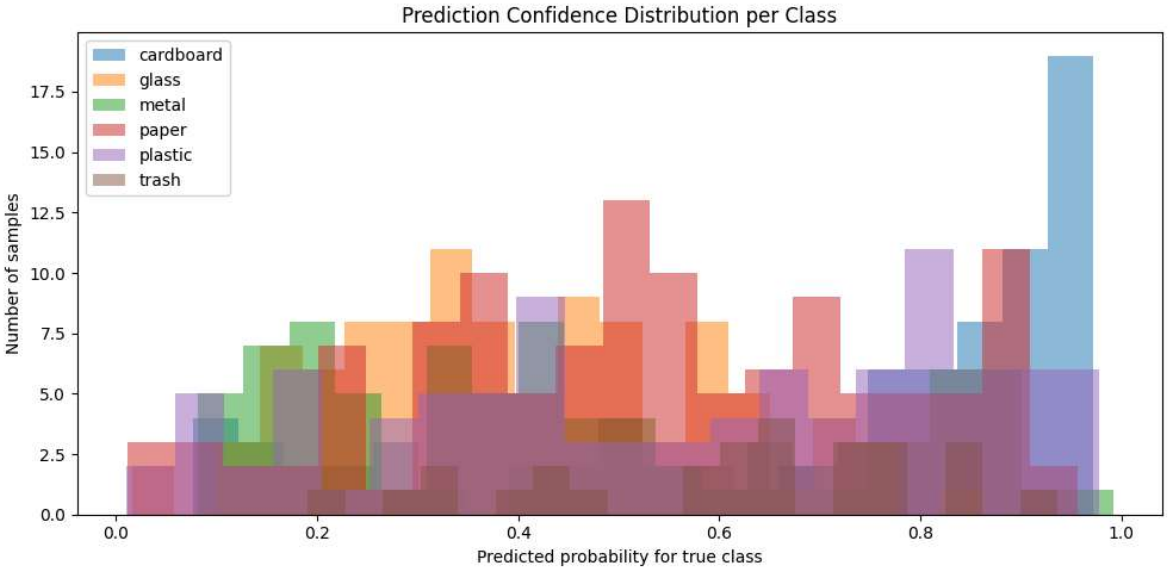

Pred: glass
True: metal
Pred: metal
True: glass
Pred: metal
True: glass
Pred: paper
True: metal
Pred: paper
True: plastic
Pred: metal
True: glass
Pred: paper
True: glass
Pred: metal
True: glass
Pred: trash
True: metal
Pred: plastic
True: glass
Pred: trash
True: paper
Pred: plastic
True: glass


Loss per Epoch
Train Loss
Val Loss
1.6
1.5
1.4
1.3
1.2
1.1
0
5
10
15
20
25
30
35

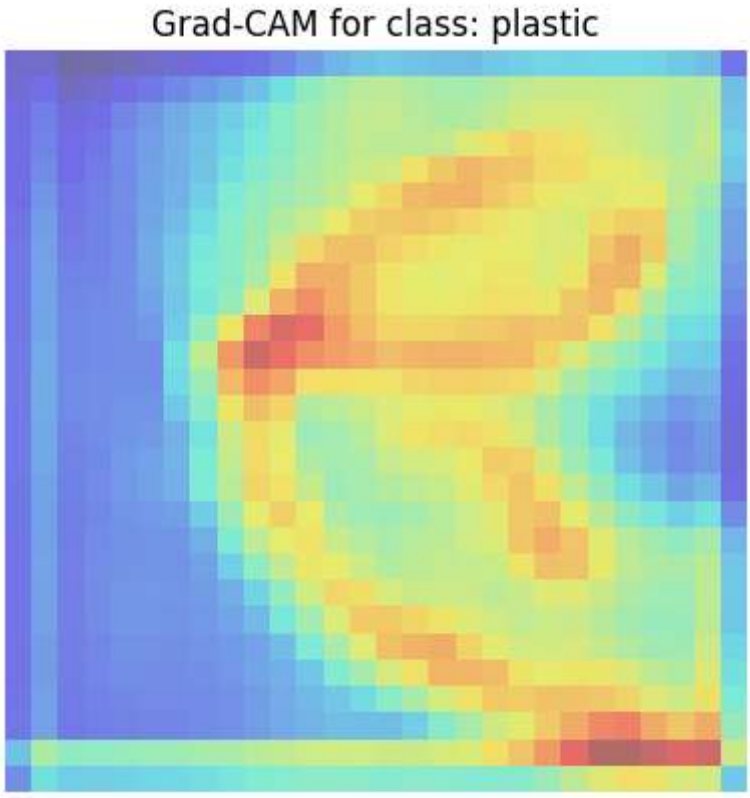
Accuracy per Epoch
Train Acc
Val Acc
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
30
35

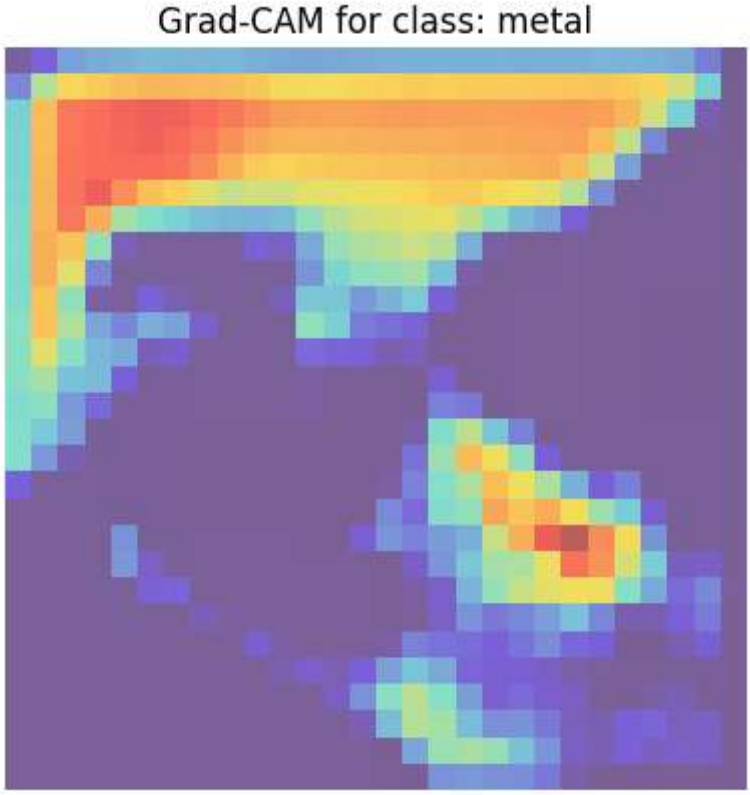

Confusion Matrix
cardboard
glass
metal
paper
plastic
trash
True
Predicted
67 0 1 8 4 1
3 62 9 4 8 14
1 7 51 3 6 14
1 2 3 96 10 7
1 2 2 7 80 5
0 0 1 0 2 24
80
60
40
20
0

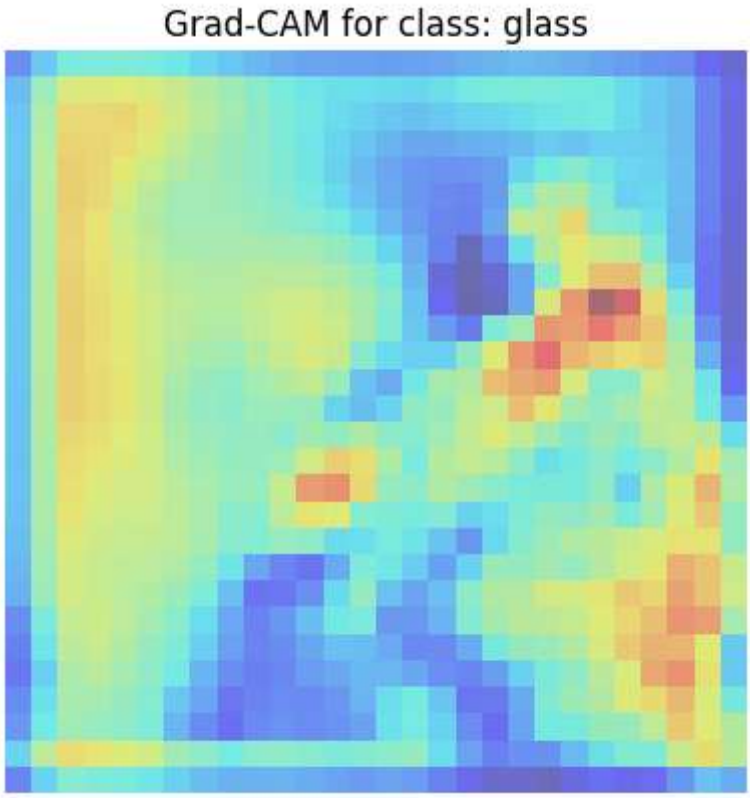

Validation Error Rate per Class
Error Rate
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
cardboard
glass
metal
paper
plastic
trash

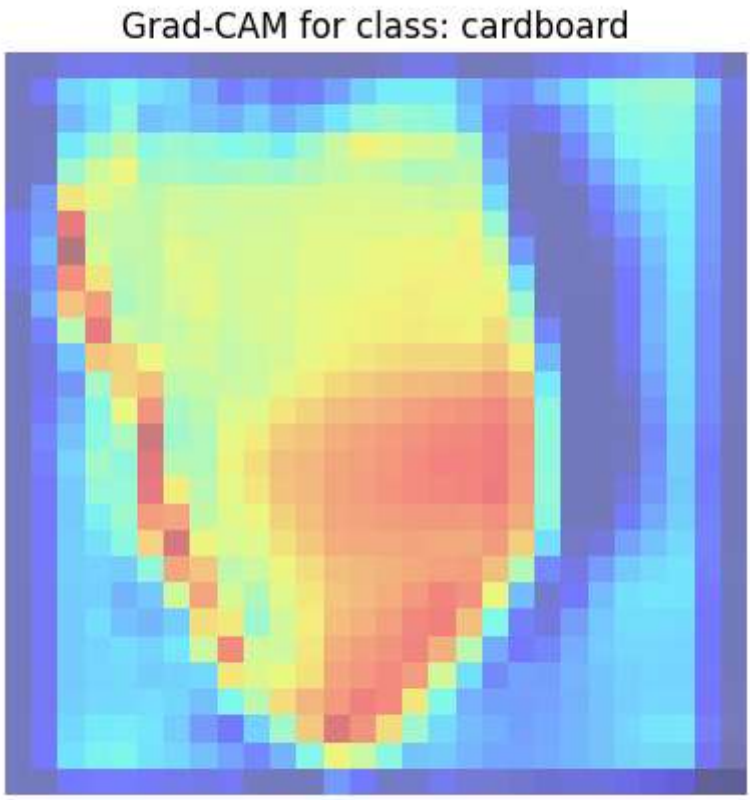


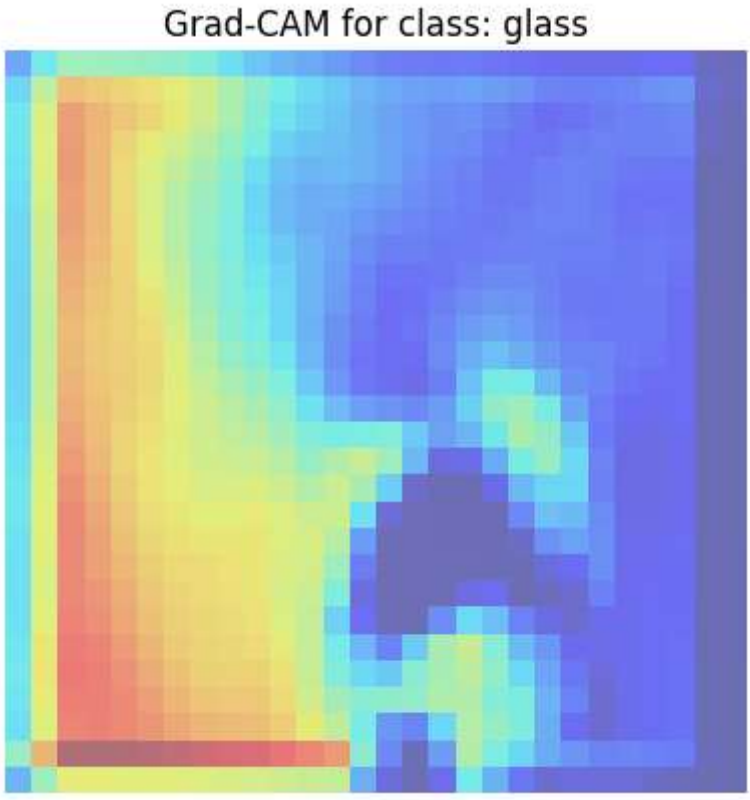
Grad-CAM Visualisierung für Beispielbilder:




Grad-CAM for class: metal


Grad-CAM for class: glass


Grad-CAM for class: cardboard


Grad-CAM for class: glass