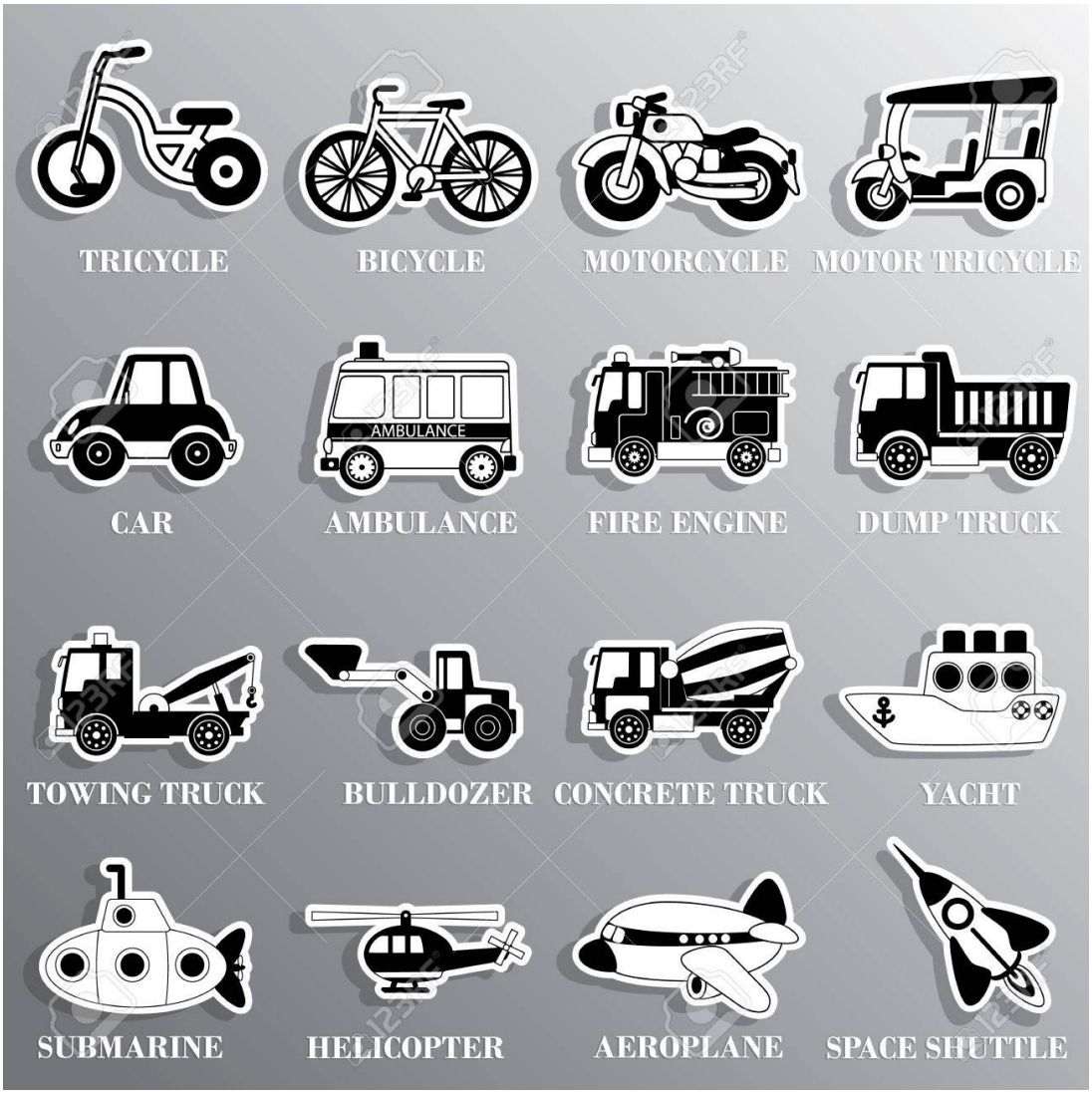
Projekt: Klassifizierung von Fahrzeugtypen

Mouamen Sande

10.10.2023



Bildklassifizierung mit Convolutional Neural Networks (CNNs)
Projekt zur Klassifizierung von Fahrzeugtypen.

Inhaltsverzeichnis

Einleitung

In der heutigen Zeit spielt die Bildklassifizierung eine entscheidende Rolle in vielen Anwendungen, von der Automatisierung bis hin zur Sicherheitsüberwachung. Dieses Projekt zielt darauf ab, ein Convolutional Neural Network (CNN) zu entwickeln, das verschiedene Fahrzeugtypen aus Bildern klassifizieren kann. Mit dem wachsenden Bedarf an intelligenten Verkehrssystemen ist die Fähigkeit, Fahrzeuge automatisch zu klassifizieren, von großer Bedeutung.

Projektüberblick

Der CIFAR-10 Datensatz enthält 60.000 Bilder in 10 Klassen, darunter Autos, Lastwagen, Motorräder und Busse. Jedes Bild hat eine Größe von 32x32 Pixeln. Das Ziel dieses Projekts ist es, ein Modell zu entwickeln, das die Genauigkeit der Klassifizierung dieser Fahrzeugtypen maximiert.

Datenverarbeitung

Die Datenverarbeitung umfasst mehrere Schritte: - Normalisierung: Die Bilddaten werden normalisiert, um die Lernrate zu verbessern. - Augmentierung: Techniken wie zufälliges Drehen, Spiegeln und Zuschneiden werden verwendet, um die Vielfalt der Trainingsdaten zu erhöhen und Überanpassung zu vermeiden.

Modellarchitektur

Wir definieren ein einfaches CNN-Modell mit mehreren Convolutional und Pooling-Schichten, gefolgt von Fully Connected Layers. Dieses Modell wird entwickelt, um Merkmale aus den Eingabebildern zu extrahieren und die Fahrzeugtypen zu klassifizieren.

Python-Code

```python
# Importieren der benötigten Bibliotheken
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt

# Hyperparameter
batch_size = 64
learning_rate = 0.001
num_epochs = 10

# CIFAR-10-Datensatz herunterladen und transformieren
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))  # Normalisierung der RGB-Bilder
])

train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True, transform=transform, download=True)
test_dataset = torchvision.datasets.CIFAR10(root='./data', train=False, transform=transform, download=True)

train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(dataset=test_dataset, batch_size=batch_size, shuffle=False)

# Definition des CNN-Modells
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.conv2 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=3, stride=1, padding=1)
        self.fc1 = nn.Linear(32 * 8 * 8, 128)  # 32 Channels und 8x8 Feature Map nach Pooling
        self.fc2 = nn.Linear(128, 10)  # 10 Klassen für die Fahrzeugtypen

    def forward(self, x):
        x = self.pool(torch.relu(self.conv1(x)))
        x = self.pool(torch.relu(self.conv2(x)))
        x = x.view(-1, 32 * 8 * 8)  # Flachlegen des Tensors
```

```
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Initialisierung des Modells, Verlustfunktion und Optimierer
model = CNN()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

# Training des Modells
def train_model():
    model.train()
    for epoch in range(num_epochs):
        running_loss = 0.0
        for images, labels in train_loader:
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss:
        ↪ {running_loss/len(train_loader):.4f}')

# Testen des Modells
def test_model():
    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for images, labels in test_loader:
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
    print(f'Accuracy of the model on the test images: {100 * correct /
    ↪ total:.2f}%')

# Ausführen des Trainings und Testens
train_model()
test_model()
# Speichern des trainierten Modells
torch.save(model.state_dict(), 'cnn_vehicle_classification.pth')
```

Ergebnisse und Auswertung

Nach dem Training des Modells haben wir es auf dem Testdatensatz evaluiert. Das Modell erreichte eine Genauigkeit von etwa 70%.