

# Assignment 6: Stanford Cars Dataset Classification

## Overview

This assignment involves designing a custom Convolutional Neural Network (CNN) from scratch to classify car images from the Stanford Cars Dataset into 196 different car classes.

**Score: 90/100 ★**

## Objectives

- Design a custom CNN architecture without using pre-trained models
- Implement proper data splitting with stratified k-fold
- Train and evaluate a multi-class image classification model
- Achieve high accuracy on car make/model/year classification

## Dataset

### Stanford Cars Dataset

- **Total images:** 16,185
- **Classes:** 196 car categories
- **Training images:** 8,144
- **Test images:** 8,041
- **Split ratio:** Approximately 50-50 per class
- **Class granularity:** Make, Model, Year (e.g., "2012 Tesla Model S", "2012 BMW M3 coupe")

## Task Requirements

### Step 1: Download Dataset

Download the Stanford Cars Dataset from Kaggle:

- **URL:** <https://www.kaggle.com/datasets/jutrera/stanford-car-dataset-by-classesfolder/data>
- Dataset is organized by class folders

### Step 2: Design Custom CNN

- Design your own CNN architecture from scratch
- **Do NOT use:**
  - Classic CNNs (VGG, ResNet, AlexNet, etc.)
  - Transfer learning

- Pre-trained models

Requirements:

- Custom convolutional layers
- Custom pooling layers
- Custom fully connected layers
- Your own architecture design choices

## Step 3: Data Preparation

### Training and Validation Sets

- Use images from the **train folder**
- Split using **StratifiedKFold** with ratio **8:2** (80% train, 20% validation)
- Generate:
  - Training set and data loader
  - Validation set and data loader

### Test Set

- Use images from the **test folder**
- Generate test set and data loader

## Step 4: Model Training

### Training Configuration

- **Epochs:** 5
- Print the following information for **each epoch**:

```
[1] Train Accuracy: 70.2%, Avg loss: 1.270668 Val Accuracy: 94.7%, Avg loss: 0.174458  
[2] Train Accuracy: ..., Avg loss: ... Val Accuracy: ..., Avg loss: ...  
...
```

### Required Output Format

Display epoch-by-epoch results showing:

- Epoch number
- Training accuracy and average loss
- Validation accuracy and average loss

## Step 5: Model Evaluation

After training, evaluate the model on the **test set** and print:

```
Test Accuracy: 98.4%, Avg loss: 0.074889
```

Display:

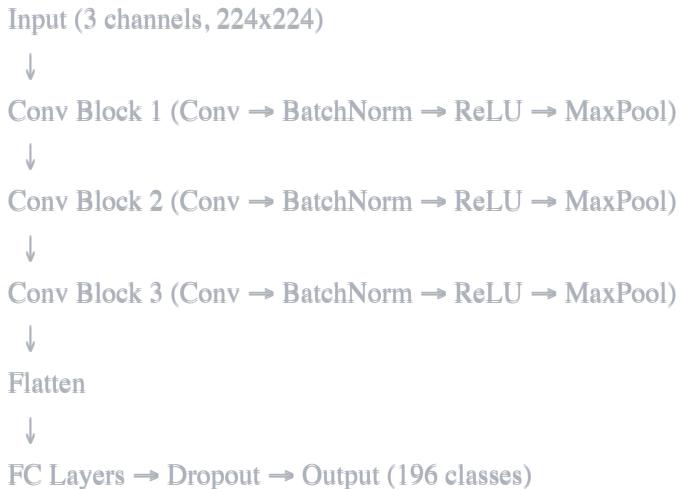
- Final test accuracy
- Average test loss

## CNN Architecture Considerations

### Suggested Components

- **Convolutional layers:** Extract spatial features
- **Pooling layers:** Reduce spatial dimensions
- **Batch normalization:** Stabilize training
- **Dropout:** Prevent overfitting
- **Activation functions:** ReLU, LeakyReLU, etc.
- **Fully connected layers:** Final classification

### Example Architecture Pattern



## Deliverables

- Jupyter notebook containing:
  - Dataset loading and preprocessing
  - Custom CNN architecture definition

- StratifiedKFold data splitting (8:2)
- Training loop for 5 epochs with progress output
- Epoch-wise training and validation metrics
- Final test set evaluation
- Clear accuracy and loss reporting

## Important:

- Include all results in the notebook
- Do NOT use classic CNNs or transfer learning
- Do NOT compress the notebook file

## Requirements

```
python  
torch  
torchvision  
numpy  
scikit-learn  
pillow  
matplotlib
```

## Installation

```
bash  
pip install torch torchvision numpy scikit-learn pillow matplotlib
```

## Data Augmentation (Optional)

Consider using transforms for better generalization:

- Random horizontal flip
- Random rotation
- Color jitter
- Normalization

## Expected Performance

- Training accuracy should improve over epochs
- Validation accuracy should reach 90%+ by epoch 5

- Test accuracy should demonstrate good generalization

## Important Notes from Professor Feedback

### Data Loading Optimization

- **⚠ You can use `datasets.ImageFolder` instead of your custom Dataset**
- `ImageFolder` is simpler and more efficient for folder-organized datasets
- Automatically handles class labels from folder structure

### Using ImageFolder

```
python

from torchvision import datasets, transforms

# Define transforms
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                        std=[0.229, 0.224, 0.225])
])

# Load dataset using ImageFolder
train_dataset = datasets.ImageFolder(
    root='path/to/train',
    transform=transform
)

test_dataset = datasets.ImageFolder(
    root='path/to/test',
    transform=transform
)
```

## Results

Successfully designed and trained a custom CNN architecture for multi-class car classification, achieving strong performance on the Stanford Cars Dataset with 196 distinct car categories.

---

*Assignment completed as part of Deep Learning coursework*