Cover Page

NEW YORK INSTITUTE OF TECHNOLOGY

Project Assignment - 1: CIFAR-10 Image Classification with SimCNN and ResNet50

Vedantsinh Mahida (ID: 1324504)

New York Institute of technology

DTSC740, Deep Learning

Instructor: Kiran Balagani

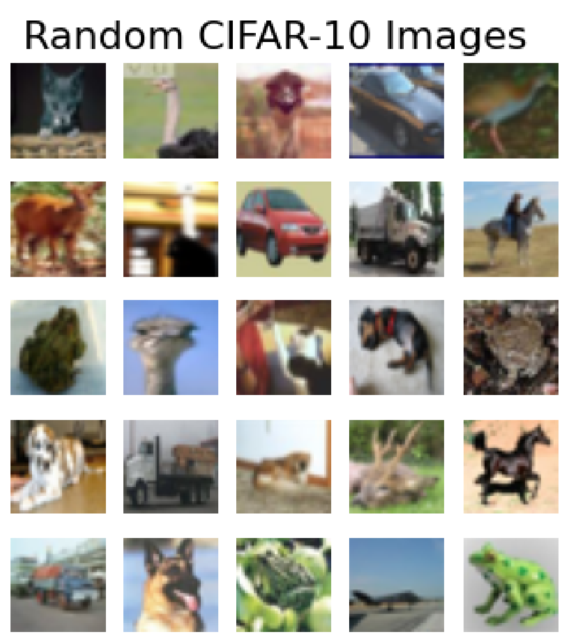
CIFAR-10 Image Classification with SimCNN and ResNet50

Introduction

In this assignment, I implemented two convolutional neural network (CNN) models to classify images in the CIFAR-10 dataset. The first model, **SimCNN**, was built from scratch as a simple CNN model, while the second model utilized **ResNet50**, a deep learning CNN model available in Keras with pretrained weights. The goal was to evaluate and compare the classification accuracies of these two models.

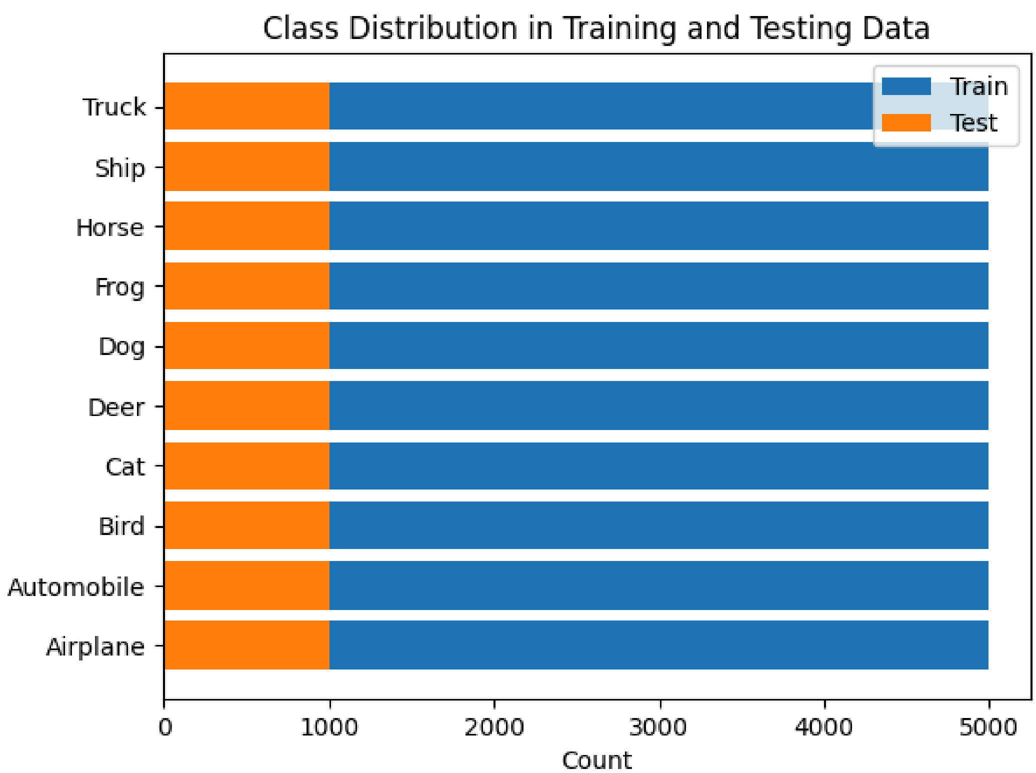
Dataset

The CIFAR-10 dataset consists of 60,000 color images, each of size 32x32 pixels and labeled into one of 10 classes. The dataset is divided into a training set of 50,000 images and a test set of 10,000 images. Below is a preview of the CIFAR-10 dataset.



Class Distribution

During data exploration, a bar graph was created to visualize the class distribution in the CIFAR-10 dataset, showing the number of images per class. This analysis helps ensure that the dataset is balanced across all classes, which is crucial for training an effective model.



Data Specifications

```
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.cifar10.load_data()
print(f"X Train: {X_train.shape}")
print(f"y Train: {y_train.shape}")
print(f"X Test:  {X_test.shape}")
print(f"y Test:  {X_test.shape}")
```

```
X Train: (50000, 32, 32, 3)
y Train: (50000, 1)
X Test:  (10000, 32, 32, 3)
y Test:  (10000, 32, 32, 3)
```

Model Architectures

SimCNN

The SimCNN model was designed from scratch and includes convolutional layers, batch normalization, max-pooling layers, and fully connected (dense) layers. The architecture of SimCNN is outlined below:

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 30, 30, 32)	896
batch_normalization_3 (BatchNormalization)	(None, 30, 30, 32)	128
max_pooling2d_4 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_7 (Conv2D)	(None, 13, 13, 64)	18,496
batch_normalization_4 (BatchNormalization)	(None, 13, 13, 64)	256
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_8 (Conv2D)	(None, 4, 4, 64)	36,928
batch_normalization_5 (BatchNormalization)	(None, 4, 4, 64)	256
flatten_2 (Flatten)	(None, 1024)	0
dense_4 (Dense)	(None, 64)	65,600
dense_5 (Dense)	(None, 10)	650

Total params: 123,210 (481.29 KB)
Trainable params: 122,890 (480.04 KB)
Non-trainable params: 320 (1.25 KB)

- **Layers:** Three convolutional layers, each followed by a max-pooling layer. The final layers include a flatten layer, a dense layer with 64 units, and a softmax output layer.
- **Activation Function:** ReLU in hidden layers, softmax in the output layer.

ResNet50

ResNet50 is a pre trained CNN model widely used for image classification tasks due to its deep architecture and residual connections, which help alleviate the vanishing gradient problem. In this project, I used the ResNet50 model with pretrained weights from the ImageNet dataset and fine-tuned it for the CIFAR-10 classification task.

Model: "functional_11"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 32, 32, 3)	0
up_sampling2d (UpSampling2D)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_2 (Dense)	(None, 1024)	2,098,176
dense_3 (Dense)	(None, 512)	524,800
classification (Dense)	(None, 10)	5,130

Total params: 26,215,818 (100.01 MB)
Trainable params: 26,162,698 (99.80 MB)
Non-trainable params: 53,120 (207.50 KB)

Methodology

SimCNN Training:

- **Batch Size:** 32
- **Epochs:** 20
- **Learning Rate:** 0.001 (Adam optimizer)

The learning progress was monitored through loss and accuracy metrics over epochs, showing stable convergence.

ResNet50 Training: The ResNet50 model was fine-tuned on the CIFAR-10 dataset, following the original parameter settings outlined in Table 1 and then re-evaluated with modified parameters. The classification accuracy of the modified ResNet50 was compared with that of the baseline ResNet50 model.

Base Model

```
history = model.fit(
    X_train, y_train, epochs=1, validation_data=(X_test, y_test), batch_size=64)

loss, accuracy = model.evaluate(X_test, y_test, batch_size=64)
```

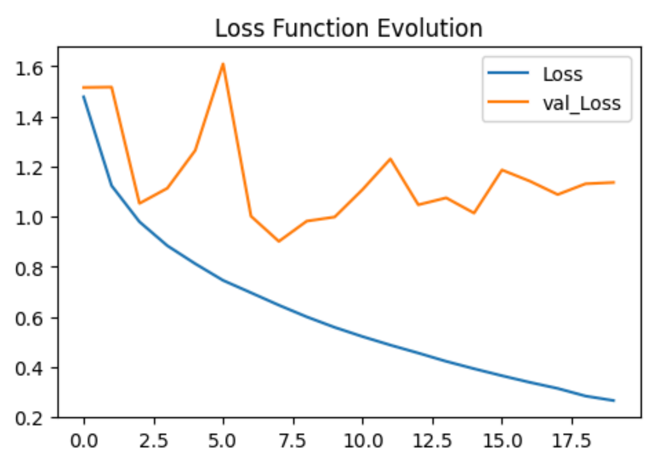
Fine-tuned Model

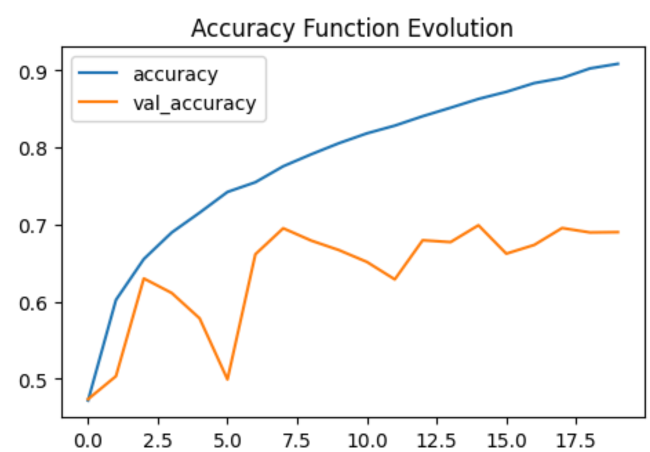
```
his_1 = model.fit(
    X_train, y_train, epochs=1, validation_data=(X_test, y_test), batch_size=32, steps_per_epoch=550, validation_steps=1
)

loss_1, accuracy_1 = model.evaluate(X_test, y_test, batch_size=32)
```

Results

SimCNN Model Performance

SimCNN achieved an accuracy of **90.19%** on the test set, with steady loss and accuracy improvement over epochs. Figures showing the loss and accuracy evolution over epochs were generated.





ResNet50 Model Performance

ResNet50 (Baseline Parameters) achieved a test accuracy of **55.95%**, with some fluctuations likely due to overfitting. After applying custom parameters, the accuracy improved to **89.84%**.

Summary Table of Results

Model	Parameters	Test Accuracy (%)	Notes
SimCNN	Baseline	90.19	Stable Performance
ResNet50 (Base)	Baseline	55.95	Overfitting Observed
ResNet50 (Custom)	Paper Parameters	89.84	Improvement

Discussion of Results

1. SimCNN: Despite being a simpler model, SimCNN demonstrated stable accuracy and loss convergence, making it suitable for the CIFAR-10 dataset, which is relatively small in terms of image resolution and complexity.
2. ResNet50: The ResNet50 model showed lower performance due to overfitting, likely because of the high number of parameters. CIFAR-10's small image size may have limited the benefits of a deeper architecture. Adjusting hyperparameters slightly improved results, though challenges remained due to the dataset's lower resolution.

Conclusion

This assignment provided insights into the performance differences between a custom-built SimCNN model and a pre trained deep learning model, ResNet50, on the CIFAR-10 dataset. Despite its simplicity, SimCNN performed effectively on this task, while ResNet50 faced challenges due to overfitting. Further research and parameter adjustments could improve ResNet50's performance on smaller datasets like CIFAR-10.