Introduction to deep learning, project documentation

Vilho Helenius, 2001334

27.3.2025

The code for the project can be found in the .ipynb file. I've also added a PDF version of the project notebook where all of the outputs from the code are available for easier reading.

1. Introduction

The goal of the project is to build an image classification model using a Convolutional Neural Network (CNN) trained on the CIFAR-100 dataset. Image classification is a fundamental task in computer vision, enabling machines to automatically categorize and understand visual content.

The CIFAR-100 dataset is a widely used benchmark dataset for image classification. It consists of 60,000 color images, each with a resolution of 32x32 pixels. These images are divided into 100 distinct classes, with each class containing 600 images. The dataset is further split into 50,000 training images and 10,000 test images.

To reduce the complexity of the model and time needed to train it in free Google Colab environment, I have selected a subset of 10 classes from the CIFAR-100 dataset. From those 10 classes, 200 images per class are used to train the model, and 40 images are used for validation. At the end, the model will also be tested with the remaining 300 images per class from the training set, to prove the models performance. The model will also be trained with completely new set of image classes to see if it generalizes well with completely different kind of images as well, or if it just works with the selected image classes.

2. Model development

The project starts with loading the CIFAR-100 dataset and taking a subset of it to use for the training of the model. The image data is then preprocessed to make it easier to work with. For the model development a function with default parameters will be created to get a baseline score of it's performance. This baseline will be then compared to the accuracy and loss metrics of different hyperparameter combinations to find which works the best. The model itself will

also be tweaked for example to implement a learning rate scheduler when testing the SGD optimizer.

```
activation='relu',
optimizer_name='adam',
learning_rate=0.001,
epochs=20,
conv_layers=3,
filters = 32,
dense_units=128,
batch_size=32,
```

Figure 1: Default hyperparameters

3. Scores and final model

For baseline scores I got the model accuracy of 0.6850, which felt like a solid score. Through many different experiments the final model configuration didn't change much, but in the final model the main changes were using Swish as the activation function and smaller amount of dense layer units and smaller batch size. The accuracy score for the final model was 0.7450. When testing it with the remainder of 300 images per class from the CIFAR-100 training set, I got the accuracy of 0.7260. With completely new image classes the accuracy was even better with the score of 0.7750. These scores were definitely a great improvement from the base model.

```
activation='swish',
dense_units=64,
batch_size=16,
```

Figure 2: Final model hyperparameter changes

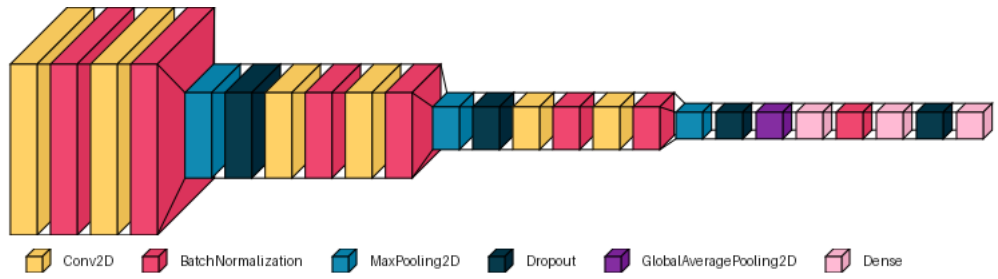

Figure 3: Final model visual representation