Image Classification

using Cifar-10 and Cifar-100 Datasets

Elisa Du

Smart Technologies

December 15, 2023

Table of Contents

[1. Introduction 1](#_Toc153646943)

[2. Dataset 1](#_Toc153646944)

[2.1 CIFAR-10 1](#_Toc153646945)

[2.2 CIFAR-100 2](#_Toc153646946)

[2.3. Combining CIFAR-10 with CIFAR-100 2](#_Toc153646947)

[3. Data Preprocessing 3](#_Toc153646948)

[4. Training 3](#_Toc153646949)

[4.1 Model Selection 3](#_Toc153646950)

[4.2 Batch Generator 4](#_Toc153646951)

[4.3 Hyperparameter Tweaking 4](#_Toc153646952)

[References 5](#_Toc153646953)

1. Introduction

The primary objective of the project is to build a convolutional neural network model that can classify between twenty-four different classes of the CIFAR-10 and CIFAR-100 datasets. The classes are car, bird, cat, deer, dog, horse, truck, cattle, fox, baby, boy, girl, man, woman, rabbit, squirrel, tree, bicycle, bus, motorcycle, pickup truck, train, lawn mower and tractor. This involves obtaining, preparing and exploring the data sets. Different models will then be built and the hyperparameters will be adjusted based on the accuracy of the model. At the end of the project, the model with the best test accuracy will be presented.

# 2. Dataset

The underlying datasets can be downloaded from 'https://cs.toronto.edu/~kriz/cifar.html'. They consist of different classes, separated into training and test images with corresponding labels. Each training and test image is also divided into several batches.

### 2.1 CIFAR-10

The CIFAR-10 dataset consists of 50000 training and 10000 test 32x32 colour images, 10 classes with 5 training and 1 test batch. Each class contains 6000 images. Each batch contains randomly selected images from each class, while the training batches contain 10000 images each and the test batches contain 1000 images each. When the training batches are added together, there are exactly 5000 images from each class.

The following 10 classes are in the dataset:

* Airplane
* Automobile
* Bird
* Cat
* Deer
* Dog
* Frog
* Horse
* Ship
* Truck

### 2.2 CIFAR-100

Similar to the CIFAR-10 dataset, this dataset also contains classes of images. There are 100 classes of 600 images each, divided into a five-to-one split, with 500 training images and 100 test images per class. Each class is grouped into a superclass. The data set contains 20 superclasses. The images are labelled with a 'fine' label for the exact class and a 'coarse' label for the superclass.

The classes contained are shown below:

|  |  |
| --- | --- |
| Superclass | Classes |
| aquatic mammals | beaver, dolphin, otter, seal, whale |
| fish | aquarium fish, flatfish, ray, shark, trout |
| flowers | orchids, poppies, roses, sunflowers, tulips |
| food containers | bottles, bowls, cans, cups, plates |
| fruit and vegetables | apples, mushrooms, oranges, pears, sweet peppers |
| household electrical devices | clock, computer keyboard, lamp, telephone, television |
| household furniture | bed, chair, couch, table, wardrobe |
| insects | bee, beetle, butterfly, caterpillar, cockroach |
| large carnivores | bear, leopard, lion, tiger, wolf |
| large man-made outdoor things | bridge, castle, house, road, skyscraper |
| large natural outdoor scenes | cloud, forest, mountain, plain, sea |
| large omnivores and herbivores | camel, cattle, chimpanzee, elephant, kangaroo |
| medium-sized mammals | fox, porcupine, possum, raccoon, skunk |
| non-insect invertebrates | crab, lobster, snail, spider, worm |
| people | baby, boy, girl, man, woman |
| reptiles | crocodile, dinosaur, lizard, snake, turtle |
| small mammals | hamster, mouse, rabbit, shrew, squirrel |
| trees | maple, oak, palm, pine, willow |
| vehicles 1 | bicycle, bus, motorcycle, pickup truck, train |
| vehicles 2 | lawn-mower, rocket, streetcar, tank, tractor |

### 2.3. Combining CIFAR-10 with CIFAR-100

To extract the necessary classes for the classification problem, we first need to find the labels associated with the searched label names. For CIFAR-10, this information is stored in a separate file 'batches.meta' as a dictionary. For CIFAR-100, the meta file contains only the label names stored as lists, separated between the fine labels and the coarse label names as a dictionary. So we define number labels by the index of the names.

We concentrate first on getting all the images with the associated labels stored in separate lists. This is achieved by iterating through the batches and extracting the necessary information. For CIFAR-100, the fine and coarse labels are also included. The required classes also include a superclass ‘tree’, so we replace all fine labels that have the corresponding class with a new identical label.

We then filter the lists by the required labels and combine the two datasets by first remapping the labels and appending one to the end of the other dataset. The remapping is necessary to avoid duplication of labels when combining the datasets.

The resulting dataset consists of 45,500 images. 7 classes (from CIFAR-10) contain 5000 images each, the required superclass from CIFAR-100 has 2500 images and all other classes have 500 images each. All images are implemented as 3072-pixel bytes.

# 3. Data Preprocessing

Looking at the distribution of the underlying dataset, the classes are highly unbalanced. Therefore, we downsample the classes with highly significant images. The limit is set at 3000 images. In addition, the images are transformed into 32x32 RGB images. These images are displayed at a 90-degree angle. This introduces a compensating rotation to rotate them back.

Smaller classes are handled by data augmentation applied by the batch generator in the training.

The images are converted to greyscales. This allows the shape of the images to be highlighted.

# 4. Training

The following section describes how the model architecture is selected and what measurements are taken to optimize the best model and minimize overfitting.

### 4.1 Model Selection

For the model, we recreated different model architectures that showed good results on the CIFAR-10 dataset and adapted them to the existing code. Thus, each of the models uses the Adam optimizer with a learning rate of 0.001 a. It is applied to grayscale images with an input shape of 32x32x1. All models run with an epoch number of 40 and a batch size of 64.

The best model is selected by the accuracy of the validation and test data. The table below shows the results.

|  |  |  |
| --- | --- | --- |
| Model | Validation Accuracy | Test Accuracy |
| LeNet model (Loane, 2023) | 56.22% | 60.41% |
| Modified LeNet model (Loane, 2023) | 48.78% | 52.64% |
| 4-layer CNN (Płotka, 2018) | 60,78% | 64,20% |
| 6-layer CNN (Płotka, 2018) | 62.67% | 65.63% |
| 8-layer CNN | 59.25% | 62.91% |
| 6-layer CNN with Batch Normalization (Brownlee, 2020) | 65.75% | 68.91% |
| 4-layer CNN with Batch Normalization (Chansung, 2018) | 62.24% | 65.98% |

The 6-layer CNN with batch normalization shows the most promising results for our dataset based on these accuracy values. It is built on the architecture of the VGG models introduced by the Visual Geometry Group at the University of Oxford (Zisserman, 2015). It is designed for image classification tasks and consists of stacking multiple convolutional layers with small 3x3 filters followed by max-pooling layers. The model already uses dropout regularization and batch normalization to stabilize and speed up the learning process (Brownlee, 2020).

### 4.2 Batch Generator

To handle the unbalanced data set, we use a batch generator with data augmentation. Two different methods are tested. The first, which is also used in the model selection, is the ImageDataGenerator provided by TensorFlow (Tensorflow, 2023). It randomly adds horizontal and vertical shifts of up to 10% of the total width and height of the images for each batch. The images are also randomly zoomed by up to 20%, sheared and rotated by up to 10 degrees.

datagen = ImageDataGenerator(width\_shift\_range=0.1, height\_shift\_range = 0.1, zoom\_range = 0.2, shear\_range = 0.1, rotation\_range=10)

The second generator is a self-defined batch generator. It iterates through the training dataset up to the batch size and randomly applies a data augmentation to the images. With a probability of 50%, the image is randomly cropped after being resized to 38x38. The image is then randomly flipped vertically, brightened and contrast adjusted.

def random\_augment(image):

  if np.random.rand() < 0.5:

    image = tf.image.resize(image, [38, 38], method=tf.image.ResizeMethod.NEAREST\_NEIGHBOR)

    image = tf.image.random\_crop(image, size=(32, 32, 3))

  image = tf.image.random\_flip\_left\_right(image)

  image = tf.image.random\_brightness(image, 0.1, 111)

  image = tf.image.random\_contrast(image, 1, 1.5,111)

  return image

In the test of both generators, the first has slightly better results with a difference of 0.82%. Furthermore, the epoch times are six times better than with the other generator. For further optimization of the hyperparameters, we therefore use the ImageDataGenerator provided.

### 4.3 Hyperparameter Tweaking

While we used previously grayscaled images, it has shown that using RGB images improves the accuracy by almost 2%. The test accuracy is 70,88%.

The learning-rate is set by 0.001. Compared to setting the learning-rate to 0.002 and 0.0015. The model seems to converge faster with the initial learning-rate. Although, while the accuracy between lr=0.001 and lr=0.002 are almost the same, lr=0.0015 improved the test accuracy to 71.99%.

# References

Brownlee, J. (2020, August 28). *Machine Learning Mastery*. Retrieved 12 16, 2023, from https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/

Chansung, P. (2018, April 17). *Towards Data Science*. Retrieved 15 12, 2023, from https://towardsdatascience.com/cifar-10-image-classification-in-tensorflow-5b501f7dc77c

Loane, J. (2023, November). *Github*. Retrieved 12 15, 2023, from https://github.com/johnloane/st\_23\_sd4a/blob/master/GTS\_23\_sd4a.ipynb

Płotka, S. (2018, August 27). *Ermlab*. Retrieved 12 15, 2023, from https://ermlab.com/en/blog/nlp/cifar-10-classification-using-keras-tutorial/

*Tensorflow*. (2023, December 16). Retrieved from https://www.tensorflow.org/api\_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

Zisserman, K. S. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *https://arxiv.org/pdf/1409.1556.pdf*, 14.

Different Learning rate impact, Taken Lr=0.0015

A graph of loss and validate

Description automatically generated

Figure Lr=0.002

A graph of loss and validate

Description automatically generated

Figure Lr=0.001

A graph of a line graph

Description automatically generated

Figure Lr=0.0015

Impact on different epochs

A graph of loss and validate

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure Epoch=100A graph of loss and validate

Description automatically generated

Figure Epoch=150

Own Data Generator

A graph with blue lines and numbers

Description automatically generatedA graph of loss and validate

Description automatically generated

A screen shot of a computer program

Description automatically generatedA graph of loss and validate

Description automatically generated