Deep learning classification using a Convolutional Neural Network (CNN)

Data analysis and pre-processing

Saamie Vincken
FHNW Internship S5

Version history

Version	Date	Changes	Author
0.1	25.10.2024	Initial setup of document	Saamie Vincken
0.2	26.10.2024	Exploration of CIFAR-10 dataset	Saamie Vincken
0.3	26.10.2024	Visualization techniques	Saamie Vincken
0.4	27.10.2024	Grad-CAM analysis	Saamie Vincken
0.5	27.10.2024	Image preprocessing and transformations	Saamie Vincken
0.9	31.10.2024	Data processing architecture overview	Saamie Vincken
1.0	31.10.2024	Final adjustments document	Saamie Vincken

Distribution

Version	Date	Distributed to	Status
1.0	31.10.2024	Canvas	None

Table of Contents

1 Introduction

This document is about the data analysis and preprocessing done before classifying images from the CIFAR-10 dataset using a Convolutional Neural Network (CNN). The goal is to prepare the CIFAR-10 images for training, ensuring data quality and compatibility for a CNN, which includes preprocessing steps like data normalization, augmentation, and visualization.

1.1 Context

This analysis and preprocessing is part of a classification task on relatively ‘simple’ data, in preparation of a more complex classification task for the Euclid Space Telescope mission. The CIFAR-10 dataset, containing 60,000 labeled images across ten categories, provides a basis for learning about- and testing CNNs. In previous documentation, research is done towards the structure and use of CNNs, while this document will cover the steps taken for pre-processing and analyzation of the dataset.

2 CIFAR-10 dataset

The images used for this classification are from the CIFAR-10 dataset, which contains 60.000 32x32 color images (RGB), split into 50.000 training images and 10.000 test images with 10 classes.

2.1 Loading the data

PyTorch offers support of the CIFAR-10 dataset through the torchvision library, which can then be used for downloading and loading the train- and test dataset.

The following parameters are used to correctly load the dataset:

- **train (bool):** Defines if the training or testing dataset should be loaded (True for training data, False for test data)
- **shuffle (bool):** Defines if the data should be shuffled before use (True for train data, False for test data)
- **batch-size (int):** Batches multiple data samples processed at once, improving memory usage and stability of the training process.
- **num_workers (int):** Paralyzes data loading allowing for multiple subprocesses to load in parallel and reduce overall training time.

2.2 Classes and frequency

The defined classes used for classification in the CIFAR-10 dataset are defined by the following labels:

- Airplanes, Cars, Birds, Cats, Deer, Dogs, Frogs, Horses, Ships, and Trucks.

To define if the classes are well balanced, the Counter module from the package "collections" is used. After implementing the module and analyzing the distribution, it showed that all 10 classes in the CIFAR-10 dataset have 5000 images, ensuring a perfect class distribution.

Equation 1 displays the formula for class (frequency) distribution.

Equation 1: Frequency distribution

$$N_i = \frac{N}{C}$$

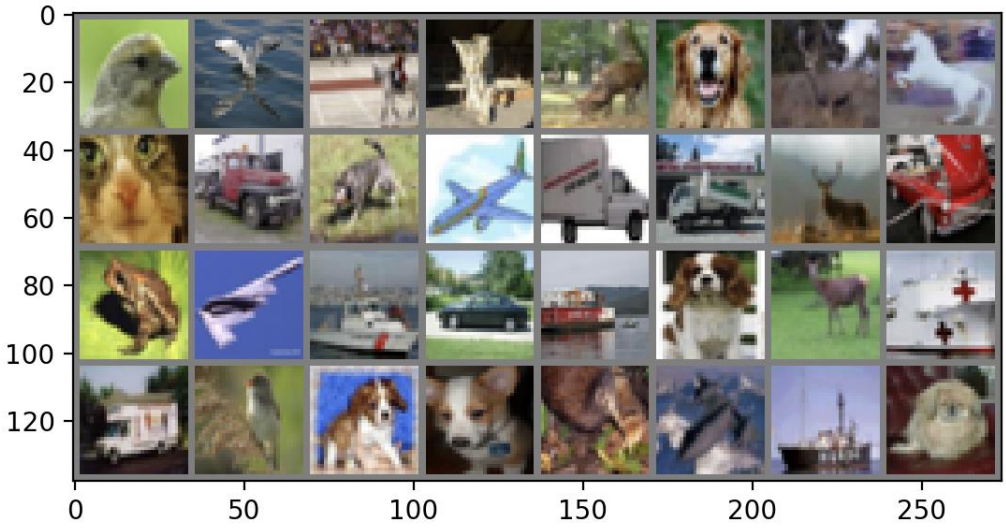
Where:

- N_i is the number of images in each class i.
- N is the total number of images in the dataset.
- C is the number of classes in the dataset.

2.3 Visualization

To understand what the images look like, a function is written to display the images in a grid. The grid is created using the Torchvision and Matplotlib packages and has the default behavior to arrange images in a grid with 8 columns. The batch size is set to 32, which means the images are displayed in a grid with 4 rows and 8 columns. This generated grid is displayed in figure 1.

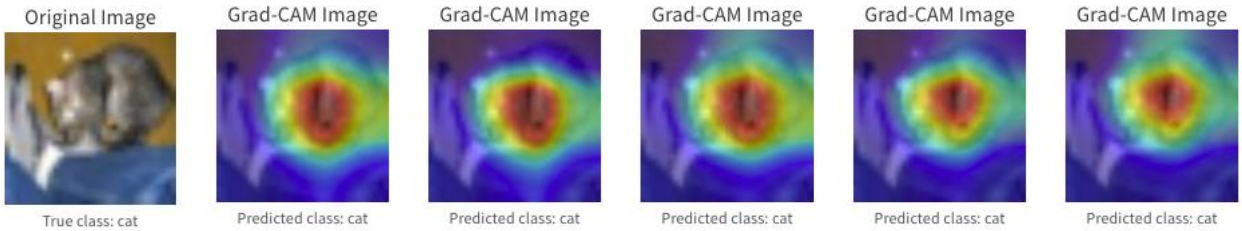
Figure 1: Image visualization grid



2.3.1 Grad-CAM Analysis

Grad-CAM (Gradient-weighted Class Activation Mapping)[1] is a tool for visualization of regions of an image that are important in the model's decisions. It highlights what areas are used for the final classification.
For the analyzation of CIFAR-10, Grad-CAM was used to compute a heatmap over multiple steps in the training process, showing the relevance of areas in an image. This result is displayed in figure 2. The red to yellow colors represent areas of high relevance, while the green to blue areas represent areas of less relevance. It shows that in all steps the model has a consistent focus around the head of the cat in the image.

Figure 2: Grad-CAM results



[1] (Helmholtz, 2023)

3 Pre-processing

To prepare the data for the classification tasks, the images are normalized and transformed to tensors, making sure that the input values are on a similar scale for faster and more stable training.

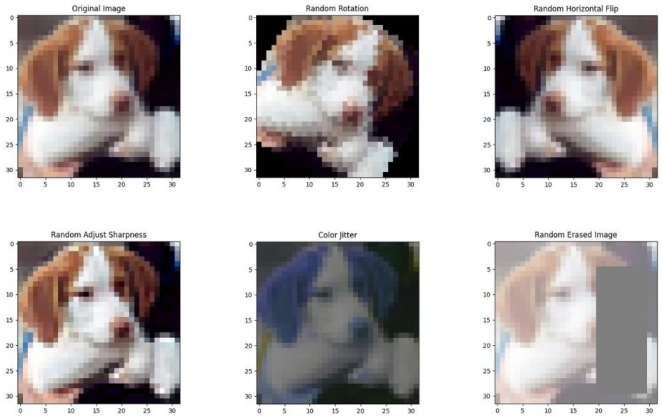
3.1 Transformations

Transformations are applied on the dataset to prepare the images for input into the model. The transformations are used to transform the input to a structure that is compatible for computational processing. Some transformations, like data augmentation, help the model to generalize better.

3.1.1 Data augmentation

Data augmentation is used to increase the diversity of the training process. There are various transformations that can be applied which help the model generalize better and reduce overfitting*. For this classification of CIFAR-10 images, various image augmentation techniques are used. Figure 3 displays the data augmentation that gave the best results for this specific task, where a random image is taken from the dataset and displayed using the augmentation techniques.

Figure 3: Data augmentation on random image from Cifar-10



These transformations will then all be applied during the data retrieval when training and testing the final model. Composing the image transformations is done using the torchvision package (chapter 6: Software requirements).

3.1.2 Tensors

A tensor is a table of numerical values indexed along several dimensions.[2] The images in CIFAR-10 are 32x32 pixels with 3 colors channels (RGB), which means that one image would be transformed to a 3D tensor. One dimension for its height, one for its width and one for its color channel.

[2] (Fleuret & Genève, Deep learning 1.4 Tensor basics and linear regression)

* *Overfitting:* When the predictions are accurate for training data but not for new data.

Each color channel is represented by 8 bits, which can have different combinations of 0s and 1s, meaning there are **$2^8 = 256$ combinations**, with a range for each color channel is **[0, 255]**.

Figure 7[3] displays the dimensions of a 3D tensor. When training larger datasets, like CIFAR-10, it is common to use batches for training. When batches are used for the images in a CNN, the images are transformed to a sequence of RGB images, called a 4D tensor. Figure 2[4] displays the dimensions of a 4D tensor.

Figure 4: 3D tensor

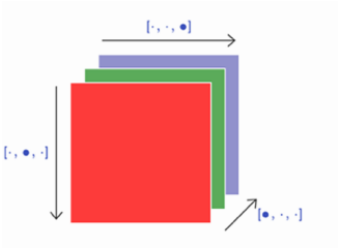
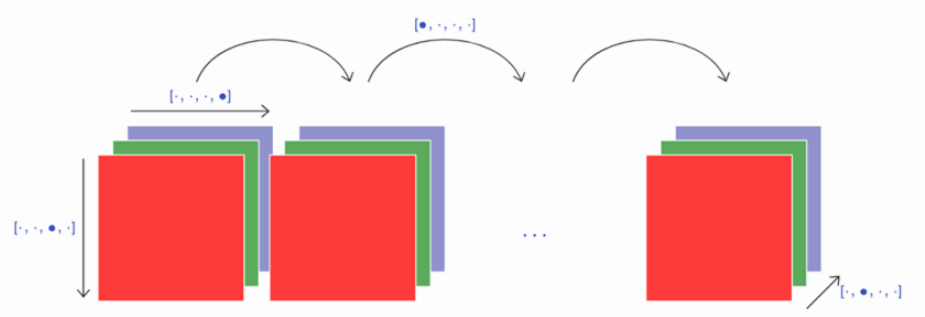


Figure 5: 4D tensor



The transformation to a tensor is calculated using Torchvision. Equation 2 displays the formula for the transformation to a tensor.

Equation 2: Transformation to tensor

$$\text{tensor}_{\text{value}} = \frac{pixel_{value}}{255}$$

Where:

- **Tensor_value** is the result range [0,1]
- **Pixel_value** is the original RGB color value [0,255]

[3] (Fleuret & Genève, Deep learning 1.4 Tensor basics and linear regression)
[4] (Fleuret & Université de Genève, Deep learning 1.5 High dimention tensors)

3.1.3 Normalization

Normalization is used to ensure all input values have a similar scale. This helps to improve the speeds and stability of training the model. During normalization for RGB images, pixel values of each image are scaled from the original range of **[0, 255]** to a standardized range centered around zero **[-1, 1]**. For more specific normalization, the mean and standard deviation of each color channel (RGB) are computed based on the training set statistics.

Fir CIFAR-10 the most commonly used normalization is:

- **Mean**: [0.4914, 0.4822, 0.4465]
- **Standard deviation**: [0.2023, 0.1994, 0.2010]

Similar to the tensor transformation, normalization is applied using tools like Torchvision. Equation 3 displays the formula for the normalization.

Equation 3: Normalization

$$X_{norm} = \frac{X - \mu}{\sigma}$$

Where:

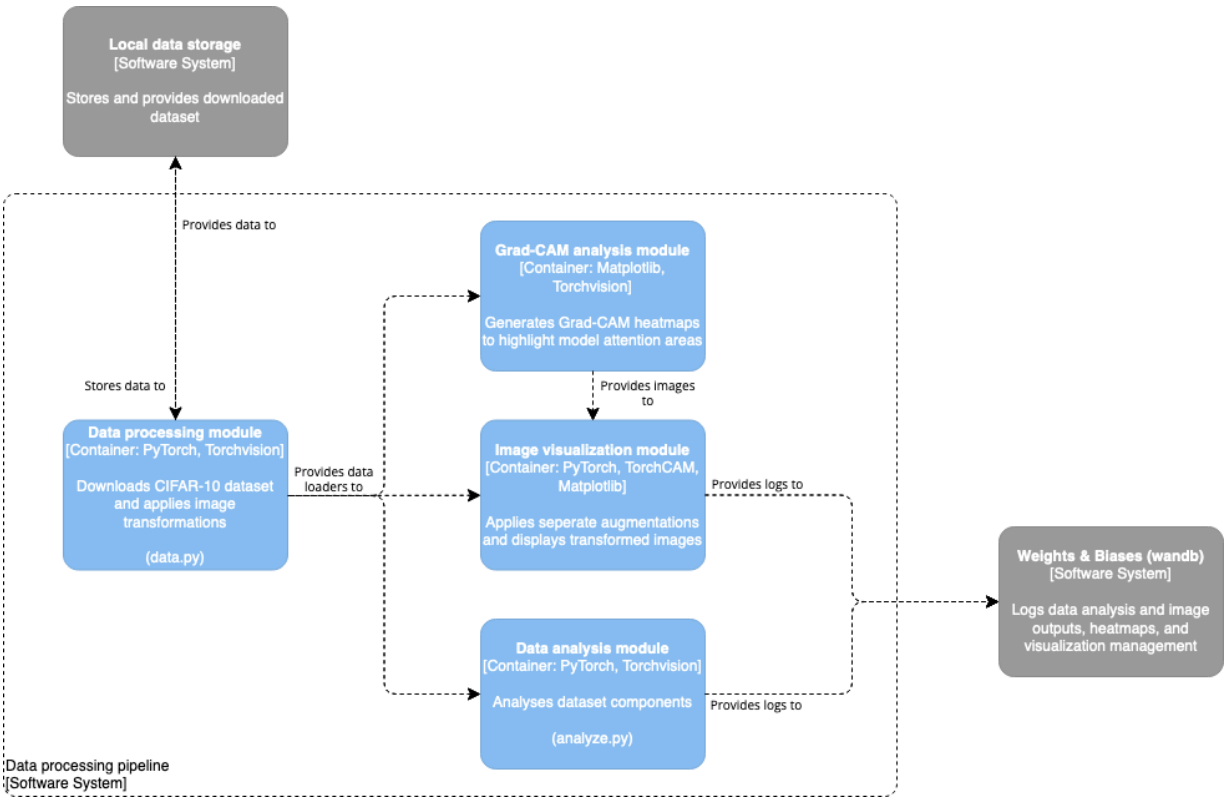
- X_{norm} is the normalized pixel value.
- X is the original pixel value.
- μ is the mean pixel value for the color channel.
- σ is the standard deviation for the color channel.

4 Data processing architecture

Finally, to provide an architecture overview on how the data is used and processed before it can be used for a deep learning classification task, a container diagram (level 2) from the C4 model is used. This diagram provides a high-level overview on the components of the data processing architecture, and contains the following items:

- **Local data storage**: This module stores the CIFAR-10 dataset and provides data access for the data processing pipeline used during analysis, training and evaluation.
- **Data processing module**: This module downloads the CIFAR-10 dataset using the Torchvision package (chapter 6: Software requirements) and applies transformations, including normalization and data augmentation.
- **Data analysis module**: This module analyzes the dataset's structure, verifying class distribution and balance to confirm the state of the data for model training.
- **Grad-CAM analysis module**: This module visualizes the areas in images that the model focuses on for classification using the Grad-CAM package (chapter 6: Software requirements).
- **Image visualization module**: This module applies augmentations and displays transformed images in a grid format, used for the analysis of preprocessing effects.
- **Weights & Biases (wandb)**: This module logs data analysis results, Grad-CAM heatmaps, and image visualizations, used for experiments and reproducibility of the model and analysis.

Figure 6: Container diagram



5 Conclusion

The preprocessing and analysis of the CIFAR-10 dataset has prepared it for the classification task using a CNN. Steps such as normalization, data augmentation, and visualization are applied to improve data consistency and cleanliness. This approach to preparing CIFAR-10 data will also be as a useful reference for future classification tasks on different datasets. Finally, the preprocessing makes sure that the data is ready for training and reduces the chance of issues that could impact the model's accuracy and reliability.

6 Software requirements

PyTorch and related libraries:

- *torch==2.4.1*: Core PyTorch library.
- *torchvision==0.19.0:* Contains datasets, model architectures, and common image transformations for PyTorch.
- *torchcam==0.4.0:* Used for class activation mapping (Grad-CAM).
- *torchmetrics==1.4.1:* For evaluation metrics.

WandB (Weights & Biases):

- *wandb==0.17.9:* For experiment tracking and visualization.

Data Handling and processing:

- *numpy==1.26.4:* For numerical operations and array manipulations.
- *matplotlib==3.9.2:* For plotting and visualization.
- *scikit-learn==1.5.2*: K-Fold cross-validation.