Step 2- Data

Data is images of different foods; I classify if food is healthy or unhealthy. I have a dataset that include 16 206 images of different foods.

In this point my task is just classify if food is healthy or unhealthy so I decide which files are healthy and which unhealthy:

**Healthy food:**

1. Bread
2. Apple
3. Banana
4. Dairy Product
5. Edamame
6. Egg
7. Lettuce
8. Meat
9. Tomato
10. Vegetable-Fruit

**Unhealthy food:**

1. Cake
2. Dessert
3. Fried food
4. Hamburger
5. Noodles-Pasta
6. Pizza
7. Rice
8. French fries
9. Hotdog
10. Tacos

Later in my own time or if I have time in this assignment, I make my model also classify how many calories food have using different CSV files that I have found.

Images are from different database and also some that I have collected, There is images from <http://foodcam.mobi/dataset256.html> , <https://www.kaggle.com/datasets/dansbecker/food-101> and <https://www.kaggle.com/datasets/trolukovich/food11-image-dataset> . Data that I collected by myself is some Apple images, Banana images, Lettuce images, Tomato images, these support vegetable-fruit file that is in one of these datasets I downloaded.

## Making images of the same size

When I have collected my images and decide which are healthy food and which unhealthy I need make images of the same quality and size, for this I use image preprocessing. I use OpenCV and PIL (Python Imaging Library) for this. Code is reading images from folders and converting all images to RGB color space, Images are resized while maintaining aspect ration by using a center crop to ensure consistent dimensions. After this I save images in a new folder and I also change images that are non-JPEG to JPEG format, I also keep the original filenames. Because my dataset is + 16 000 images I also add parallel processing to speed up my image preprocessing. My code also show bar that I can see progress and how many files is preprocessed so I know that all files are processed.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Tekoälyn generoima sisältö voi olla virheellistä.

Figure 1 Preprocessing code in progress

File structure before preprocessing was like in figure 2 below and after preprocessing it is like in figure 3 below.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, viiva

Tekoälyn generoima sisältö voi olla virheellistä.

Figure 2 Data structure before preprocessing

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, viiva

Tekoälyn generoima sisältö voi olla virheellistä.

Figure 3 Data structure after preprocessing

Preprocessing is labeling images for healthy and unhealthy folders and is giving images class depending on their original folder like Apple and Pizza. In healthy folder I have now 8571 image and unhealthy 7631 images. Labeling does not use any metadata (CSV with class labels) but this labeling should work for this assignment.

Scaling for images is not pixel-value scaling but it’s resizing and converting image format, also images get same quality, and they keep aspect ratio.

So, after preprocessing I have now two files that have healthy foods and unhealthy foods that are same quality and same size, after this I can move to the next part and that is splitting training, validation and test sets.

***The code I use for this part is named: image\_preprocessing.py***

## Splitting data

My training set size is 70%, validation set is 15% and test set is 15%, I also maintain class balance (healthy/unhealthy). My code have TEST\_SIZE and VAL\_SIZE variables so I can easily adjust them if needed, Train set is then calculated how much is left from test and validation set. I use random seed (SEED = 42) to ensure the same splits every run. Also, my splitting code is doing augmentation like rotation and flipping in my training set images to make sure there are different kinds of images that model can train as good as possible and with various images.

After running the code, I print how many images I have in each set (figure 4), also I have now a new folder and it have a new data structure. This structure is shown in figure 5.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Tekoälyn generoima sisältö voi olla virheellistä.

Figure 4 Each set size

Kuva, joka sisältää kohteen teksti, diagrammi, kuvakaappaus, Samansuuntainen

Tekoälyn generoima sisältö voi olla virheellistä.

Figure 5 Data structure after augmented splitting

So now my set for model are:

* Training set:
  + Healthy images: 17 997
  + Unhealthy images: 16 023
* Validation set:
  + Healthy images: 1 286
  + Unhealthy images: 1 145
* Test set:
  + Healthy images: 1 286
  + Unhealthy images: 1 145

Kuva, joka sisältää kohteen Luonnonmukainen ruoka, hedelmä, tuote, dieettiruoka

Tekoälyn generoima sisältö voi olla virheellistä.

Figure 6 Example of how training set healthy food looks

I didn’t use augmentation to validation or test sets because they don’t need those images, only training sets needs to be augmented. Validation set is used for tune hyperparameters without bias and test set is final unbiased evaluation of real-world performance so if validation set or test set have augmented images model can give false high accuracy when it could recognize “seen” variations, and it also defeats the purpose of having unseen evaluation data.

***Code for data splitting is named: Augmented\_splitting\_dataset.py***

### Tools and methods for step 2

Below is tools and methods listed what I used for this step, I also use anaconda environment named AppliedML that I have done just for these assignments and that include all possible libraries I need to run machine learning codes. I also use GitHub to save my steps and their branch named data for step 2 documentary and codes.

**1. Technical Specifications**

* **Programming Language:** Python 3.10
* **IDE:** Spyder

**2. Libraries Used**

| **Library** | **Purpose** | **Key Functions/Notes** |
| --- | --- | --- |
| os | File/directory operations | os.path.join(), os.makedirs(), os.listdir() |
| shutil | File copying/moving | shutil.copy2() (preserves metadata) |
| numpy | Numerical operations | Array handling (np.array(), astype(np.float32)) |
| Pillow (PIL) | Image processing | Image.open(), Image.rotate(), Image.transpose() |
| sklearn.model\_selection | Data splitting | train\_test\_split() (stratified splits) |
| multiprocessing | Parallel processing | Pool() for multi-core usage |
| tqdm | Progress tracking | tqdm() for progress bars |

**3. Image Preprocessing Steps**

1. **Resizing**:
   * Target size: 224x224 pixels (standard for CNNs like ResNet)
   * Aspect ratio preserved → Center cropping applied.
2. **Color Space**:
   * Converted to **RGB** (3 channels) for model compatibility.
   * Non-RGB images (e.g., grayscale) are automatically converted.
3. **Format Standardization**:
   * All images saved as **JPEG** (quality=85) for consistency.
   * Original formats (PNG/BMP) are converted.

**4. Augmentation Techniques** *(Training Set Only)*

* **Horizontal Flip**: Image.FLIP\_LEFT\_RIGHT (50% chance)
* **Rotation**: 10-degree rotation with border padding.
* **Augmentation Factor**: 2x (each image generates 2 variants).

**5. Dataset Splitting**

* **Split Ratio**: 70% Train / 15% Validation / 15% Test
* **Stratification**: Maintains class balance (healthy/unhealthy).
* **No Augmentation** for Val/Test sets (ensures unbiased evaluation).