# Systems with Machine Learning
Report 3

## What is the type of the problem? (LK,ML,LP)

The problem is a classification task. The aim is to create a system that can identify and categorize flowers into five groups (tulips, roses, dandelions, sunflowers, daisies).
This classification must account for variations in lighting, background, and flower appearance.

## What kind of training methods you can use (supervised training, unsuperviced, etc.)? (LK,ML,LP)

The labelled images in the dataset mean supervised learning is the best way to train the model. This teaches the model to predict the flower category of new images by learning the characteristics of each category during training.

# What are possible machine learning methods and models to solve you problem? Choose one of them and justify why. (LK,ML,LP)

There are several machine learning models that could be suitable for this task:

**Convolutional Neural Networks (CNNs):** Ideal for image recognition tasks because they can capture spatial hierarchies in images.

**Support Vector Machines (SVM):** Can be used for smaller datasets or as a baseline model.

**Transfer learning with pretrained models:** Use models like ResNet that have been trained on large datasets like ImageNet. You can then fine-tune these models on your specific dataset.

A Convolutional Neural Network (CNN) is a good choice for this problem. CNNs are good at image classification because they can automatically detect important features.

# What are possible loss / fitness functions for your problem? Choose one of them and justify why. (LK,ML,LP)

**Cetegorical Cross-Entropy:** Measures the performance of a classification model. The output is a probability value between 0 and 1.

**Mean Squared Error (MSE):** Used for regression.

**Hinge Loss:** Used for SVMs.

Categorical Cross-Entropy Loss is the best choice for a loss function. It's especially good for multi-class classification problems, where each prediction is a probability distribution across several categories.

This loss function compares the predicted probabilities with the actual distribution, where the true class is 1 and all others are 0. It encourages the model to make more accurate predictions.

# Observe and present visually on a XY plot changes in loss/fitness during the training. Compare results on TRAIN and VAL datasets. (LK,ML,LP)


## Loss Over Epochs

The training loss started high but quickly decreased, showing that the model made large errors but quickly learned.

The validation loss started higher than the training loss, then followed a similar downward trend, showing that the model is generalising well to new data.

After the initial epochs, both training and validation loss stabilised, with the validation loss showing some fluctuations.

## Accuracy Over Epochs

The training accuracy is increasing, which shows the model is learning from the training data.

The validation accuracy is also increasing, but with more variability. This may indicate that the validation data is challenging for the model.

The validation accuracy is similar to the training accuracy, which means the model is generalising well.

The loss and accuracy trends show that the model is learning well without overfitting. The lower initial validation loss suggests that the validation set is simple or regularisation is effective. Overall, accuracy improves over time for all datasets.

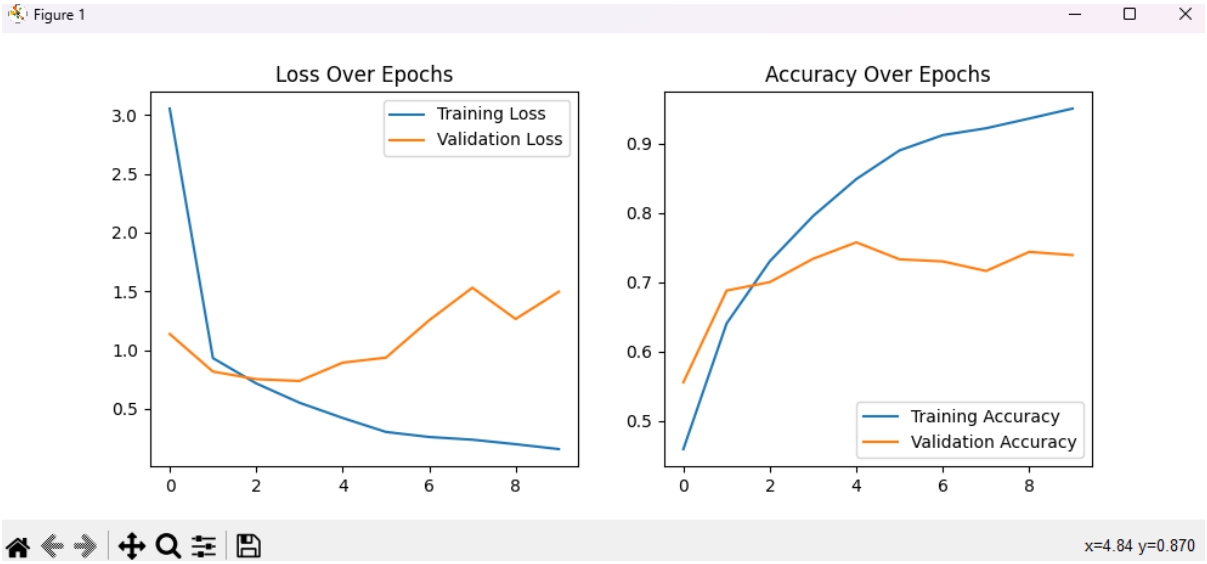
# Training on SPLIT1 datasets
# Try to overfit your model. Describe this process and results. (LK,ML,LP)

### To overfit we've removed 10 epochs, so there were 10 epochs in total.

We've also removed regularization, batch normalization and reduced dropout.

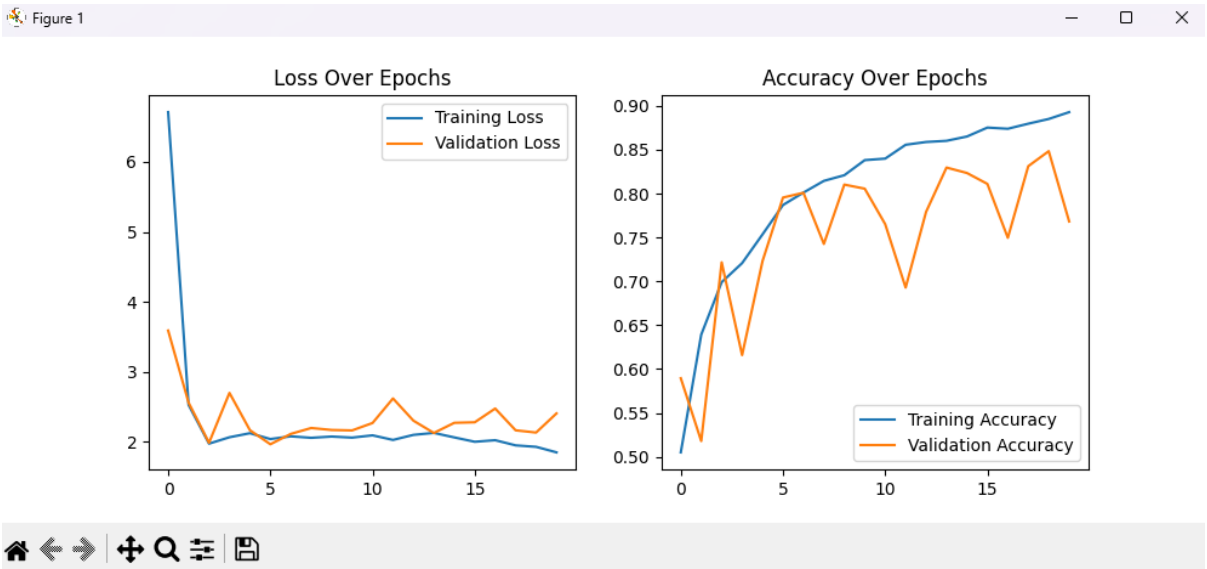
```
# Budowanie modelu CNN
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=img_shape),
    MaxPooling2D(pool_size=(2, 2)),
    #BatchNormalization(),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    #BatchNormalization(),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    #BatchNormalization(),

    Flatten(),
    Dense( units: 128, activation='relu'),
    Dropout(0.001),
    Dense( units: 5, activation='softmax')
])
```

### Results are shown below:




**Mean time of one epoch =~ 43s**

Training on SPLIT2 datasets
Compare results (time, accuracy, etc.) to the training on SPLIT1 (LK,ML,LP)




The second graph shows a more stable learning process with less fluctuation in validation metrics, suggesting better generalisation. Both models improve over time and neither shows signs of severe overfitting.

Due to the implementation of data augmentation, we achieved better results in the second split.

The dandelion dataset included images of the flower at different stages of growth. This variability introduced a level of complexity that could potentially impact negatively the model's recognition capabilities for this class.

Training on SPLIT3 datasets

Compare results (time, accuracy/loss on TRAIN dataset, accuracy/loss on VAL dataset) to the training on SPLIT2 (LK,ML,LP)

Choosen the best model is: (according to validation results), save it as MODEL3. (LK,ML,LP)

SYSTEMS WITH MACHINE LEARNING

# FLOWER RECOGNITION

# REPORT 2

Łukasz Kulesza, index 184764
Ludwik Piksa, index 184775
Michał Łomowski index 184790

## Data description

Data was downloaded from the Kaggle platform from three different datasets. Then it was grouped into five categories: Tulips, Roses, Dandelions, Sunflowers and Daisies. The images vary in size, resolution and quality providing a diverse range of data.
A total of 13,106 images have been successfully collected for the dataset. All images in the dataset are stored in JPEG format, a widely used and standardized format for digital images.

To ensure data integrity (due to use of three large datasets) and remove duplicates, a method based on computing the SHA256 hash of image files was used. This unique identifier represents the content of each file. Consequently, 3 duplicates were eliminated from the "Tulip" category, 5 from "Daisy," 1 from "Dandelion," 10 from "Sunflower," and 7 from "Rose”. Files that did not have the RGB extension were also removed.

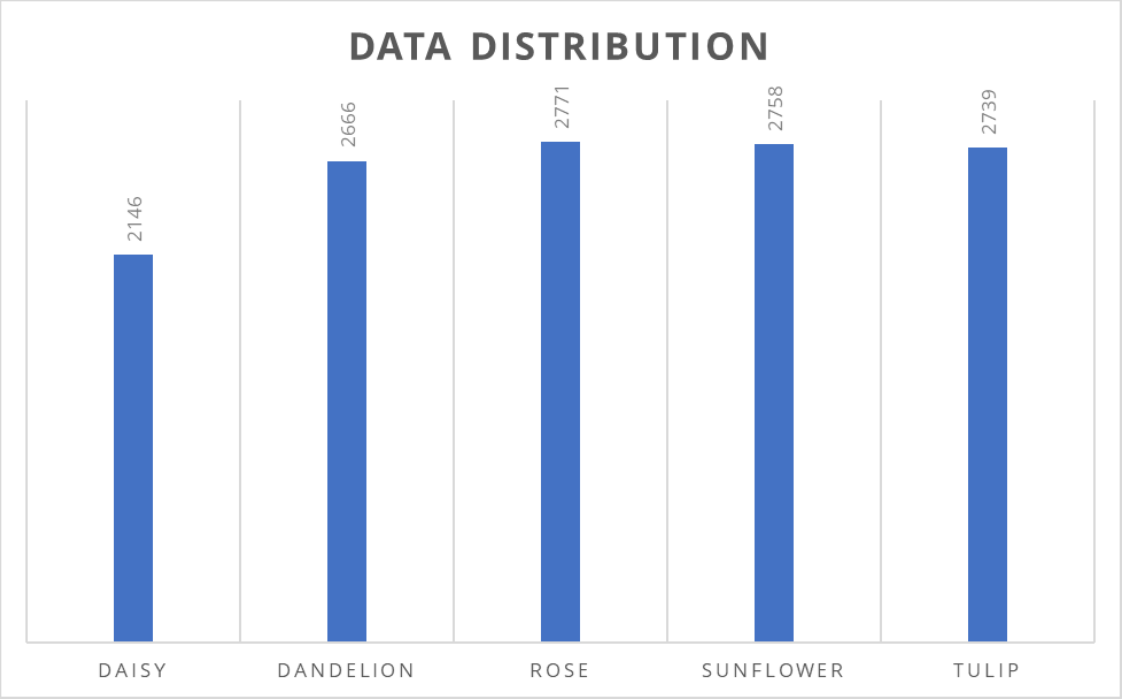
Furthermore, each image was resized to dimensions of 64x64 pixels, 128x128 pixels, and 256x256 pixels. This resizing step facilitates subsequent evaluation to identify images that yield optimal performance and efficiency for the model.

## Dataset content

After initial data processing the dataset comprises a total of 13,080 RGB images, divided to five categories. Each image is labeled with its corresponding category. The labeling follows a categorical format, indicating the specific class to which each image belongs.
The dataset comprises images in JPEG format, encompassing a variety of resolutions and sizes. Each category is represented by a varying number of images as follows:

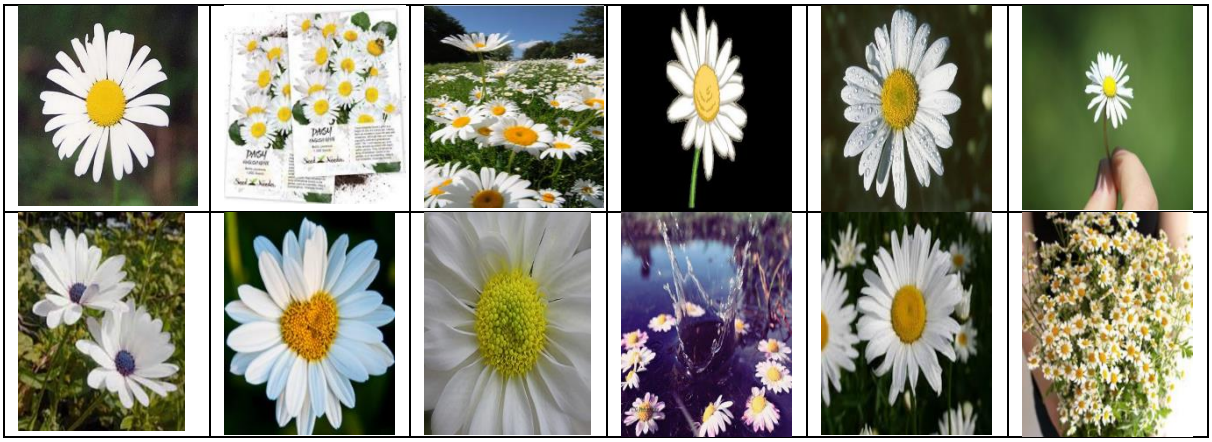
- Daisy: 2146 images (16.4%)
- Dandelion: 2666 images (20.4%)
- Sunflower: 2758 images (21.20%)
- Tulip: 2739 images (21.1%)
- Rose: 2771 images (20.9%)

Each class contributes approximately within a close range of 20% to 21.20% of the total dataset, indicating a balanced distribution and minimizing class imbalance issues during model training and evaluation.

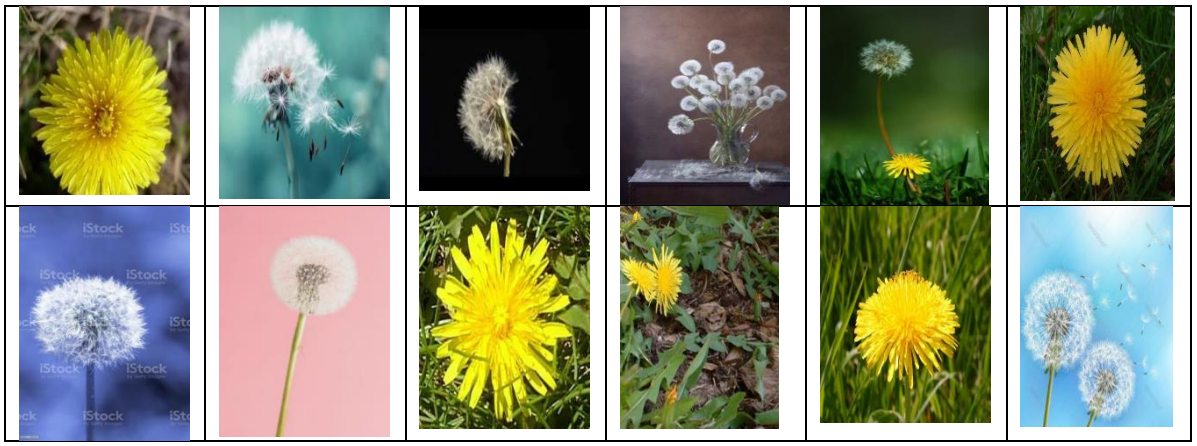


# Data relations

The first group “Daisy” has a few common themes. They occur in singles, pairs or many flowers in one photo at once. This group will probably be easy for the ML to recognize because of similar patterns and distinguishable colors.



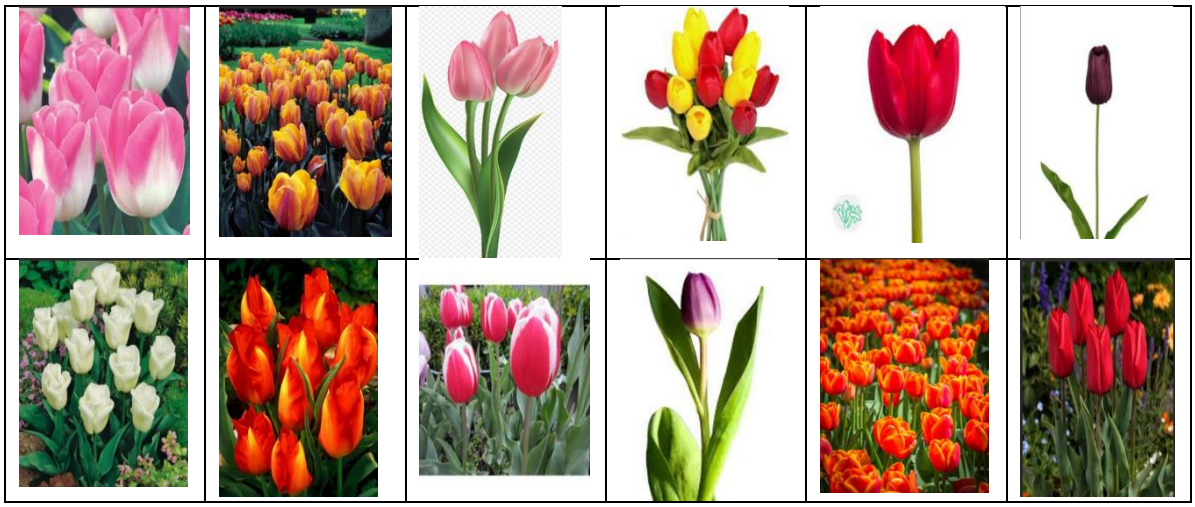
The second group is “Dandelion”. This group might be a bit harder for ML to recognize as Dandelion may appear in two different stages of development: flower or seed head. The flower has yellow colored inflorescence and seed head’s color is white.



The third group is “Rose”. Commonly met similarity of roses is the color red, and it may help our model to recognize them. Although roses can also appear in different colors like purple, pink and orange, most of them have a similar pattern of one flower being in the middle of a photo.

The fourth group is “Sunflower”. This group, as the “Dandelion” group, will probably be easy for ML model to recognize because of its distinguishable color patterns of inflorescence. Sunflowers may come in different numbers on each photo, but most of them have one flower in the middle which may help ML model to recognize them.

The last group is “Tulip”. Tulips may differ in color, but almost all tulips are photographed from one side, with their leaves and inflorescence visible. This pattern will be vital for our model to distinguish them.



We made sure to check if the dataset contains any duplicate photos and altered it in a way that all objects are unique. There should not be any duplicated photo or the same object twice.

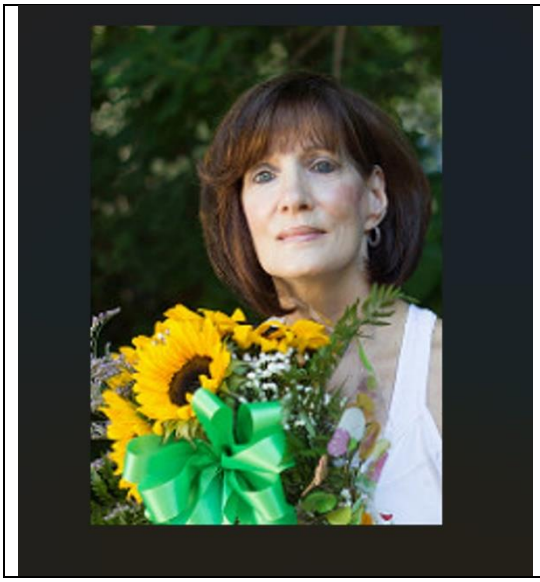
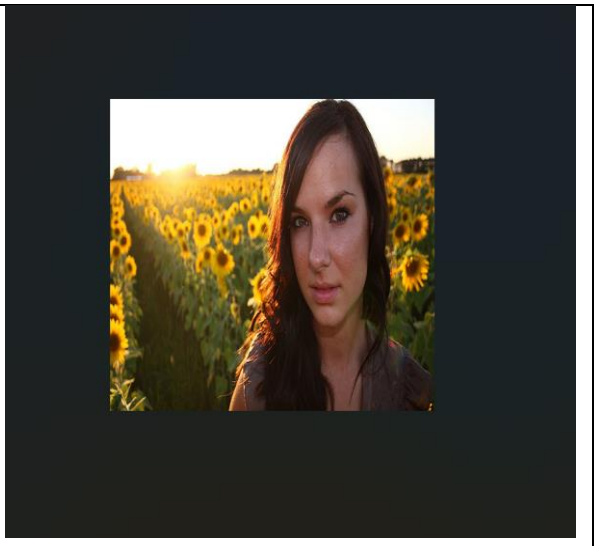
Very significant for our dataset is the fact that there are many photos of a flower with some kind of green/grass background. Even though there are different flowers on them, ML model might be prone to neglect it and pick the **“Dandelion”** group, as it contains the most photos with such a background.

Example of such case:

# Errors and noise

The labelling for images was done correctly with no errors found. However, we can find some pictures containing more than one type of flower. This brings an issue of more than one characteristic for an image. For example: daisy flower with poppy flower or tulips with red rose-like flowers. This is the case for estimably less than 0.1%.


Another problem was that many images contained people holding flowers or standing next to them. It may cause ML model to pick a group based on the number of such images in a set rather than recognizing the actual flower. We altered the data set but there can still be some pictures (estimetly less than 1%) as such:

## Data difficulty

## Data representation

The model will take raw RGB images as input. This decision ensures that all significant features of the images are retained, which might play a key role in flower classification because they provide critical information that distinguishes one species from another. Different hues can indicate unique genetic traits necessary for accurate identification and categorization.

The process of transforming data into NumPy arrays involves loading each image using the Pillow (PIL) library and converting it into a three-dimensional NumPy array. This array represents the image in terms of its rows, columns, and color channels (RGB). Additionally, a corresponding numerical label is assigned to each image based on the previously established mapping. Below is an illustrative example of how this transformation is implemented in Python:

```python
def generate_data_paths(data_dir):
    filepaths = []
    labels = []

    folds = os.listdir(data_dir)
    for fold in folds:
        foldpath = os.path.join(data_dir, fold)
        filelist = os.listdir(foldpath)
        for file in filelist:
            fpath = os.path.join(foldpath, file)
            filepaths.append(fpath)
            labels.append(category_to_number[fold])

    return filepaths, labels


filepaths, labels = generate_data_paths(data_dir)


1 usage
def load_image(image_path):
    image = Image.open(image_path)
    image = image.resize((img_size, img_size))
    image = np.array(image) / 255
    return image


for i in filepaths:
    images.append(load_image(i))
```

Label conversion involves mapping each class label contained in the folder name to a numeric identifier. This mapping is done using a simple dictionary-based approach in which each unique class label (e.g. daisy, dandelion, sunflower, tulip, rose) is assigned a specific numeric value. This

conversion makes the labels easier to handle during training, enabling efficient processing within machine learning algorithms.

```
category_to_number = {
    'daisy': 0,
    'dandelion': 1,
    'sunflower': 2,
    'rose': 3,
    'tulip': 4
}

number_to_category = {
    0: 'daisy',
    1: 'dandelion',
    2: 'sunflower',
    3: 'rose',
    4: 'tulip'
}
```

## Data normalization

The data standardization and normalization we used include scaling each pixel value to a new value in the range <0; 1>, and scaling the dimensions of the images to be the same for all occurring in the dataset.

```
image = image.resize((img_size, img_size))
image = np.array(image) / 255
```

In addition, the number of images per class can be normalized with a simple conditional statement that will limit the number of images during import.

We decided to standardize our data to increase stability and potentially speed up the training process. With unscaled data, the values of the weights in the solutions (such as neural networks) can become quite large, which can introduce instability into the model.

## Data augmentation

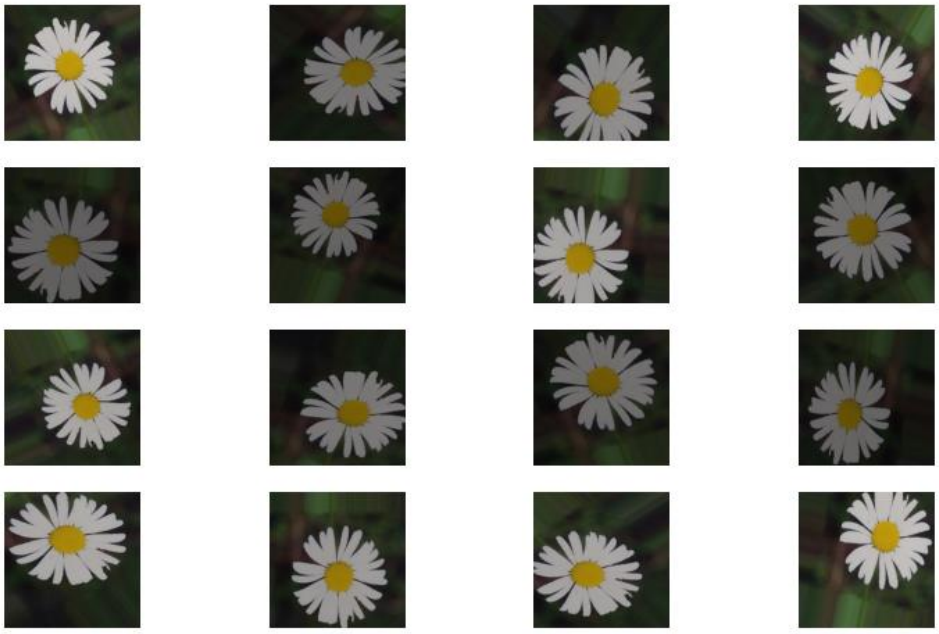
Thanks to the ImageDataGenerator class, which is available in the Python library keras.preprocessing.image, we were able to perform data expansion. In the aforementioned class, there are several parameters that we used, such as random rotation, inversion, translation,

brightness changes and magnification within a certain range. Data expansion allows us to artificially expand the dataset by creating modified copies of existing images. By applying various transformations, we generate new examples that increase the diversity of our dataset.

```
ImageDataGenerator(preprocessing_function=scalar,
                   rotation_range=40,
                   width_shift_range=0.2,
                   height_shift_range=0.2,
                   brightness_range=[0.4, 0.8],
                   zoom_range=0.2,
                   horizontal_flip=True,
                   vertical_flip=True)
```

Data augmentation also acts as a form of regularization. It prevents the model from overfitting to the training data by introducing variations. Regularization helps the model avoid memorizing specific details of individual images and encourages it to focus on more discriminative features.

Example of data augmentation for daisy flower:



# Data splits

The first split of the data using the original data without modification. No augmentation, normalization, standardization of data was applied. The figure below shows part of the source code

responsible for splitting the data. We decided to put 80% of the images in the training set, 10% in the correct set and the remaining 10% in the test set.

```
# train
train_df, dummy_df = train_test_split( *arrays: df, train_size=0.8, shuffle=True, random_state=123)

# validation and test
valid_df, test_df = train_test_split( *arrays: dummy_df, train_size=0.5, shuffle=True, random_state=123)
```

The second data split will use data augmentation and pixel values normalization. The proportions between data sets will remain the same

In the third division of the data, the proportions between data sets will remain the same while the distribution of images between classes will be equalized.

# REPORT 1

## FLOWER RECOGNITION

Michał Łomowski 184790

Ludwik Piksa 184775

Łukasz Kulesza 184764

## Problem definition

The objective of this project is to develop a system capable of accurately recognizing 5 types for flowers such as:

- Tulips
- Roses
- Dandelions
- Sunflowers
- Daisies

The system should be robust against variations in factors such as lighting conditions, background clutter, and variations in flower appearance.

# System input

The system accept input images of flowers in standard image formats such as JPEG, PNG, or BMP. Each image should have a minimum resolution of 400 x 400 pixels to ensure sufficient detail for accurate flower recognition

Training and testing data will be collected from various sources on the internet, acknowledging the possibility of some erroneous images or labels.

Examples of input data:

## System output

The system’s output system is a classification label indicating the recognized type of flower present in the input image. The label will be visible on the image. Each image is classified into one of the predefined categories, including Tulips, Roses, Dandelions, Sunflowers, Daisies, based on the analysis performed by the model.

Example outputs for inputs: