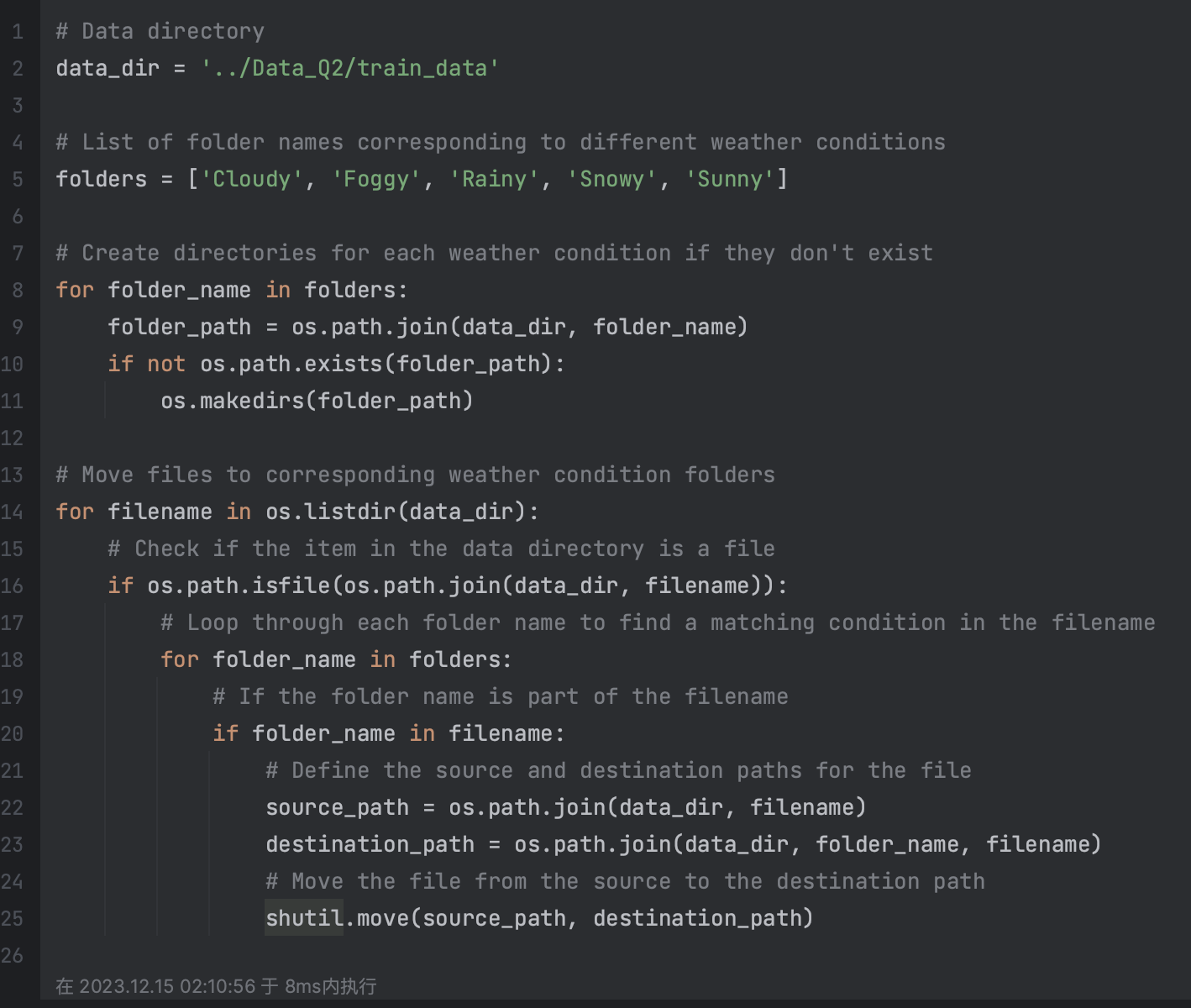
﻿**DSAA 5002 - Data Mining and Knowledge Discovery in Data Science**

**Final Exam Report – Q2 ﻿Weather Recognition**

**50015940 Jiaxiang Gao**

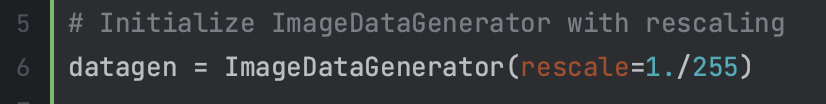
1. **Design of data loader**
2. **Data Organization and Standardization:**

The script begins by structuring the data into subdirectories named according to different weather conditions (Cloudy, Foggy, Rainy, Snowy, Sunny). This structure is integral for the automatic labeling of data during the loading process, as these folder names are used as class labels.

Each image file is sorted into an appropriate subfolder based on its filename. This organization ensures that the data is systematically arranged, facilitating easier label assignment by the data loader.

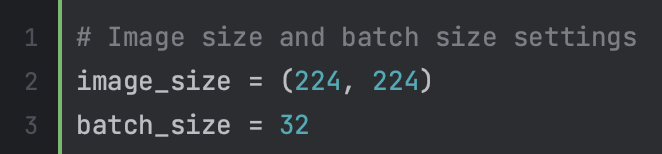
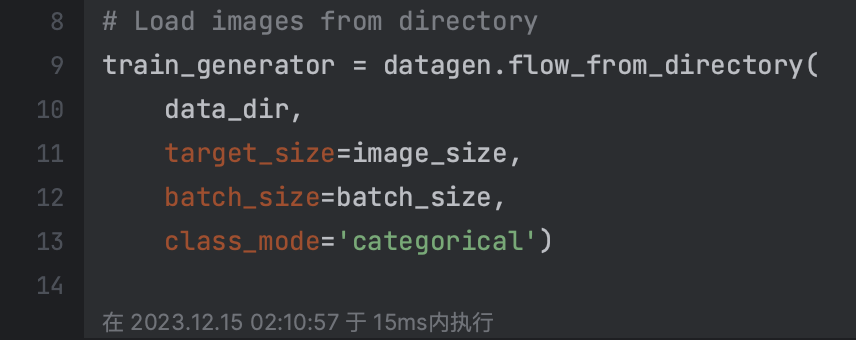
1. **Image Preprocessing and Rescaling:**

An ImageDataGenerator is utilized for image preprocessing, with rescaling set to 1./255. This step is crucial for standardizing the input data. Rescaling the images to a range of 0-1 helps in normalizing the pixel values, making it easier for the model to process and learn from the data.



1. **Data Loader Configuration:**

The flow\_from\_directory method of ImageDataGenerator is employed to load images from the organized directories. This method is beneficial as it automatically assigns labels based on the directory structure and handles image loading efficiently.

The target size for each image is set to 224x224 pixels, ensuring uniformity in input dimensions. Standardizing the size of input images is important for consistent processing by the neural network.

A batch size of 32 is chosen, which determines the number of images processed at a time during training. This size is a balance between computational efficiency and memory constraints.

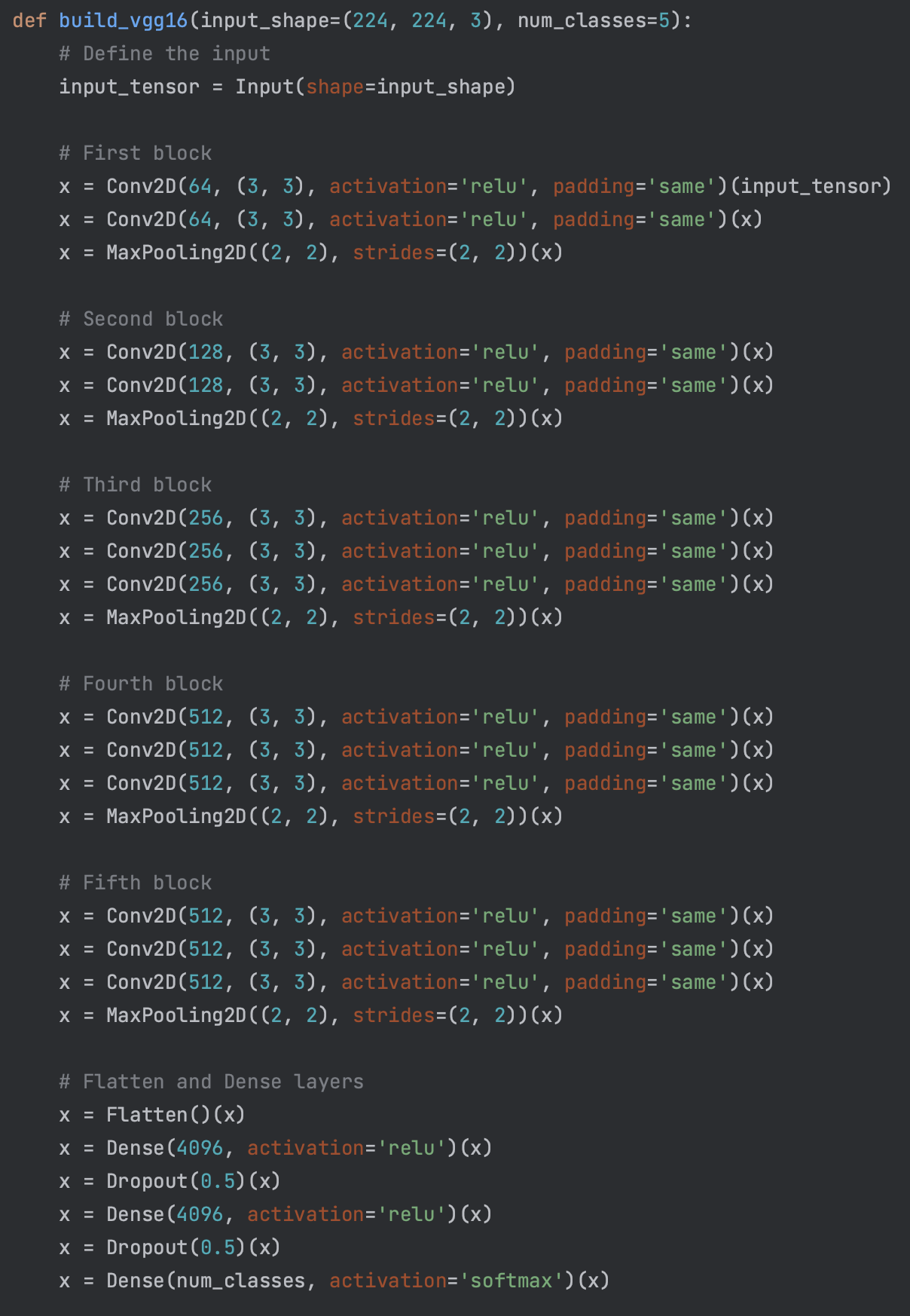
1. **Label Assignment and Categorization:**

The class mode is set to 'categorical', which means the labels are one-hot encoded. This encoding is a standard approach for multi-class classification problems.

1. **Shuffling and Additional Considerations:**

Although not explicitly mentioned in the code, the flow\_from\_directory method typically shuffles the data by default for each epoch, which helps in reducing model overfitting and ensuring that the model generalizes well.

1. **Model ﻿introduction**

The defined model is a custom implementation of the VGG16 architecture, a widely recognized convolutional neural network (CNN) used for image classification tasks. Here's a brief introduction to this model:

VGG16 is known for its simplicity, using only 3x3 convolutional layers stacked on top of each other in increasing depth. This specific implementation has been adapted for a classification task with num\_classes categories.

The model accepts input images of size 224x224 pixels with 3 channels (RGB).

Architecture:

The model consists of five main blocks of convolutional layers. Each block contains 2-3 convolutional layers with 'relu' activation and 'same' padding, followed by a max-pooling layer. The number of filters in the convolutional layers starts at 64 and doubles with each block, reaching up to 512 in the fourth and fifth blocks.

After the convolutional blocks, the model includes a flattening step to transform the 2D features into a 1D vector.

This is followed by three dense (fully connected) layers. The first two dense layers have 4096 units each and 'relu' activation, interspersed with dropout layers (with a dropout rate of 0.5) to reduce overfitting.

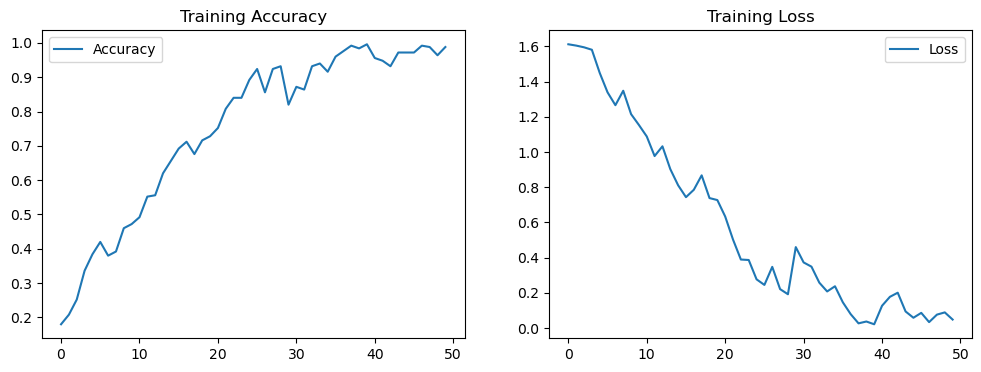
The final dense layer has a number of units equal to num\_classes and uses a 'softmax' activation function for multi-class classification.

The output is a probability distribution across the num\_classes categories, indicating the likelihood of the input image belonging to each category.

1. ﻿**Training process**

文本

描述已自动生成



1. ﻿**Accuracy on the training set**

Accuracy on training set: 0.9919999837875366

图形用户界面, 文本

描述已自动生成