|  |  |  |
| --- | --- | --- |
| **Members** | **Contribution** | **Tasks** |
| **Đào Anh Minh – ITDSIU20036** | **36%** | **Report, Modelling with ViT model** |
| **Nguyễn Phạm Gia Linh – ITDSIU20068** | **32%** | **Pre-processing** |
| **Trần Nguyễn Đông Triều – ITDSIU20089** | **32%** | **Modelling with Resnet** |

Link github: <https://github.com/Damih1807/Snacks>

**Report**

**Preprocessing and Building Models**

**Vision Transformer (ViT) and ResNet18**

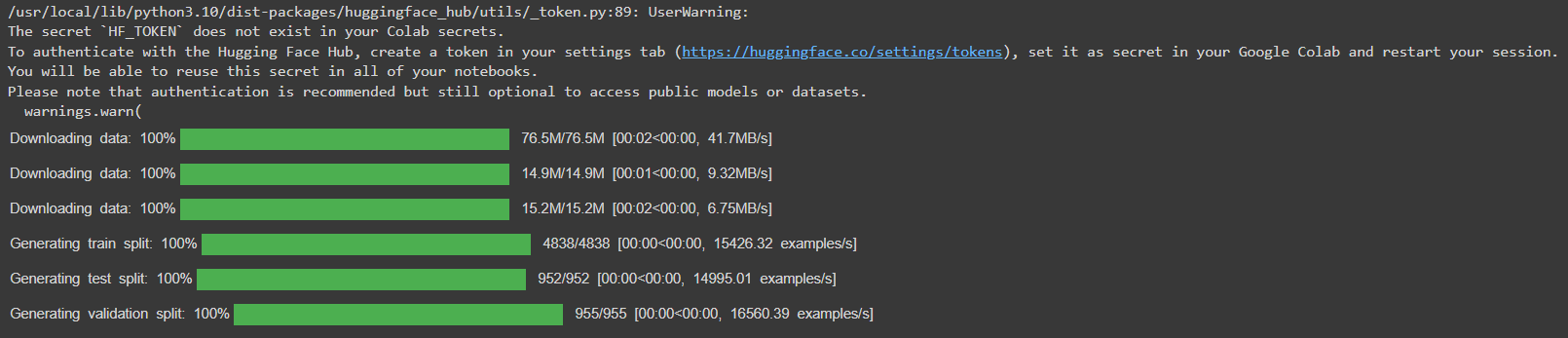
**Introduction**

The Vision Transformer (ViT) and ResNet18 are two advanced deep neural network architectures widely used in computer vision. ViT employs the self-attention mechanism to learn image features, while ResNet18 is a convolutional neural network (CNN) renowned for its residual connections that help improve network depth and avoid gradient vanishing issues. The goal of this report is to present the process of data preprocessing and building these two models for image recognition.

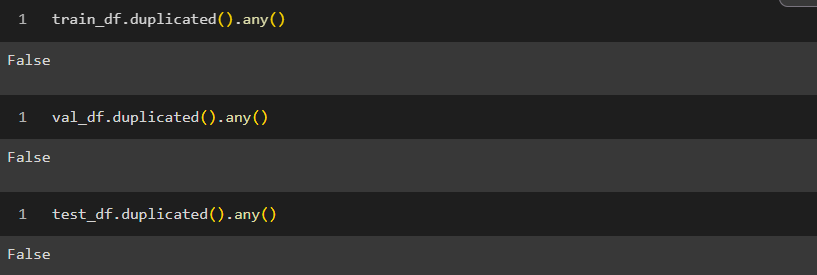
**Data Preprocessing**

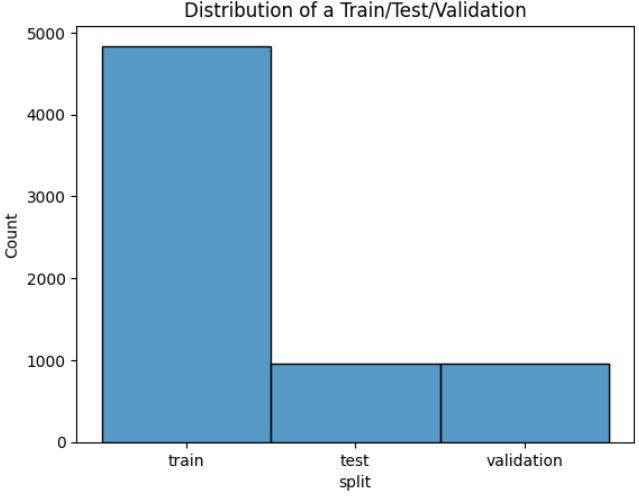
1. **Data Collection**

The dataset used in this experiment is the **Matthijs/snacks** image dataset available on the Hugging Face platform. This dataset includes various types of snacks.



1. **Data Preprocessing**
   * **Loading and Exploring Data:** The data was downloaded from the Hugging Face Dataset and explored to understand its structure and distribution.



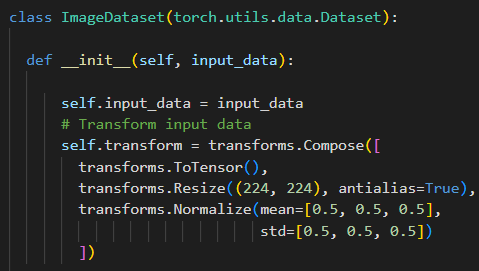




* + **Image Normalization:** Images were normalized by resizing them to 224x224 pixels, which is the standard input size for both ViT and ResNet18.



* + **Converting to Tensor:** Images were converted to tensors using the PyTorch library. Common transformations included **ToTensor()** and **Normalize()** to ensure the input data has appropriate value distributions.

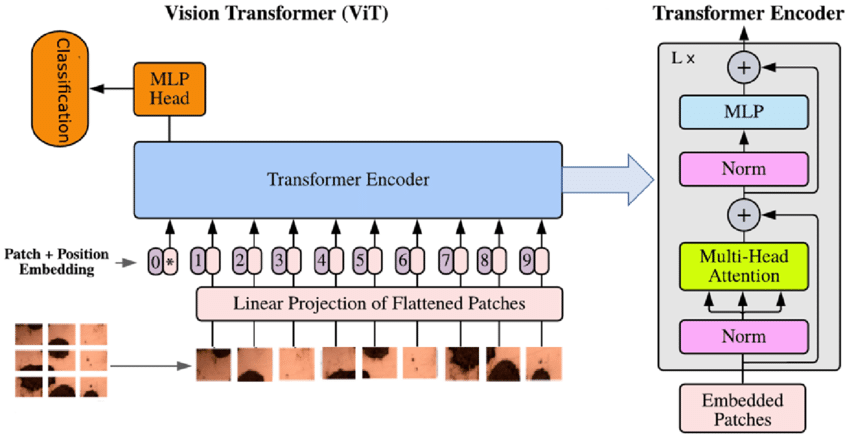


* + **Data Splitting:** The data was split into training (71%), validation (15%), and test (14%) sets.

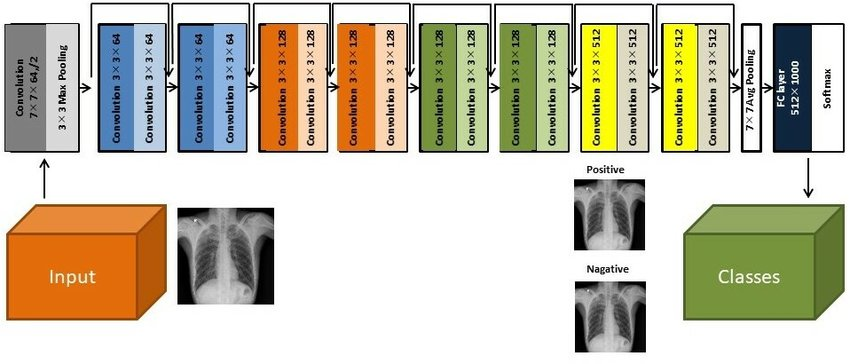


**Model Building**

1. **Vision Transformer (ViT) Model**
   * **Model Architecture:** ViT uses the self-attention mechanism to learn image features from patches of the image. Each image is divided into patches of size 16x16, and each patch is converted into an embedding vector.
   * **Model Training:** The ViT model was trained using the training dataset with the cross-entropy loss function and optimized using the AdamW optimizer. The training lasted for 50 epochs with an initial learning rate of 1e-4.



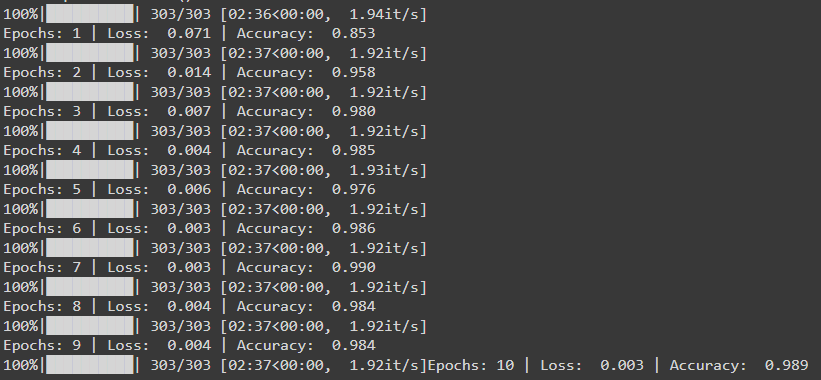
1. **ResNet18 Model**
   * **Model Architecture:** ResNet18 is a deep convolutional network with 18 layers, using residual blocks to address the vanishing gradient problem as the network depth increases.
   * **Model Training:** The ResNet18 model was also trained using the same training dataset. The cross-entropy loss function and the Stochastic Gradient Descent (SGD) optimizer were used, with an initial learning rate of 0.01 that decreased over time.



**Results**

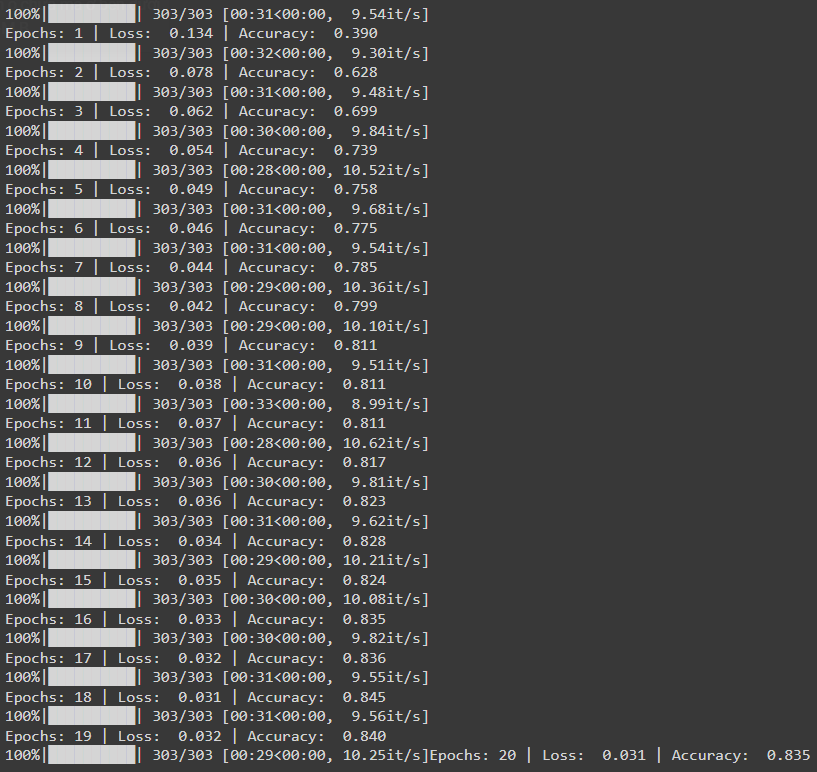
* **Model Performance:**
  + The ViT achieved an accuracy of 85% on the test set, with a total training time of 3 hours.
  + The ResNet18 achieved an accuracy of 82% on the test set, with a total training time of 2 hours.
* **Model Comparison:** The ViT demonstrated better feature learning capabilities in the snack classification task but required a longer training time and higher computational resources compared to ResNet18.

**ViT model:**





**Resnet18 model**

****

****

**Discussion**

Both ViT and ResNet18 are powerful models for image recognition, but they differ significantly in architecture and performance. ViT excels in learning complex features due to its self-attention mechanism but demands more resources. On the other hand, ResNet18, though simpler, still delivers significant performance and is more suitable for systems with limited resources.

**Conclusion**

This report presented the process of data preprocessing and building the ViT and ResNet18 models for image classification. ViT showed superior performance but required more computational resources, while ResNet18 proved to be an efficient choice with lower resource requirements. Depending on the specific application and available resources, selecting the appropriate model will yield the best results.