Topic:

Re-implementation of Paper CvT: Introducing Convolutions to Vision Transformers

Submitted by:

Name: Shahkar Javid

Matricola: 2047305

Email: javid.2047305@studenti.uniroma1.it

Name: Sameer Ahmed

Matricola: 2047250

Email: ahmed.2047250@studenti.uniroma1.it

Submitted To:

Prof. Simone Scardapane

Prof. Danilo Comminiello

1. INTRODUCTION

In this report we try to re-implement the model proposed in CvT: Introducing Convolutions to Vision Transformers –(Haiping Wu. 2021) [1] for our Neural Network project. This section contains a description of the methods we have implemented, both from the paper perspective and from our point of view.

2. Description

2.1 PAPER DESCRIPTION

The paper introduces a new architecture, named Convolutional vision transformer (CvT), that improves Vision Transformer (ViT) in performance and efficiency by introducing convolutions into ViT to yield the best of both designs. This is accomplished through two primary modifications: a hierarchy of Transformers containing a new convolutional token embedding, and a convolutional Transformer block.

According to paper these modification convolutional neural networks (CNNs) to the ViT architecture (i.e. shift, scale, and distortion invariance). The authors of the paper conduct extensive experiments, showing that this approach achieves state-of-the-art performance over other Vision Transformers and ResNets on ImageNet-1k, with fewer parameters but only when pretrained on larger datasets (e.g. ImageNet-22k) and fine-tuned to downstream tasks.

The paper also proposed to remove positional encoding, a crucial component in existing Vision Transformers; which has no effect on the model accuracy hence simplifying the design for higher resolution vision tasks.

Before going into details of CVT, the paper starts by giving slight background of original vision transformers (ViT) proposed in An Image is Worth 16x16 Words paper [2] in 2020. The ViT design adapts Transformer architectures; First, images are split into discrete nonoverlapping patches (e.g. 16 × 16). Then, these patches are treated as tokens (analogous to tokens in NLP), summed with a special positional encoding to represent coarse spatial information, and input into repeated standard Transformer layers to model global relations for classification. The Vision Transformer (ViT) is the first to prove that a pure Transformer architecture can attain state-of-the-art performance (e.g. ResNets, EfficientNet) on image classification when the data is large enough (i.e. on ImageNet-22k, JFT-300M).

Despite the success of vision Transformers at large scale, the performance is still below similarly sized convolutional neural network (CNN counterparts (e.g., ResNets) when trained on smaller amounts of data. One possible reason may be that ViT lacks certain desirable properties inherently built into the CNN architecture that make CNNs uniquely suited to solve vision tasks. For example, images have a strong 2D local structure: spatially neighboring pixels are usually highly correlated.

So in the paper, authors hypothesize that convolutions can be strategically introduced to the ViT structure to improve performance and robustness, while concurrently maintaining a high degree of computational and memory efficiency. To verify, paper present aforementioned architecture, called the Convolutional vision Transformer (CvT), which incorporates convolutions into the Transformer.

2.2 CvT ARCHITECTURE

The CvT design introduces convolutions to two core sections of the ViT architecture. First, we partition the Transformers into multiple stages that form a hierarchical structure of Transformers. The beginning of each stage consists of a convolutional token embedding that performs an overlapping convolution operation with stride the (degree of overlap can be controlled via the stride length) on a 2D-reshaped token map (i.e., reshaping flattened token sequences back to the spatial grid), followed by layer normalization. Formally, given a 2D image or a 2D-reshaped output token map from a previous stage as the input to stage i, we learn a function f(·)that maps into new tokens with a channel size Ci, where f(·) is 2D convolution operation of kernel size s × s, strides − o and p padding (to deal with boundary conditions).

The new token map Hi×Wi×Ci has height and width is then flattened into size HiWi × Ci and normalized by layer normalization for input into the subsequent Transformer blocks of stage i.

Second, the linear projection prior to every self-attention block in the Transformer module is replaced with our proposed convolutional projection. This allows the model to further capture local spatial context and reduce semantic ambiguity in the attention mechanism.

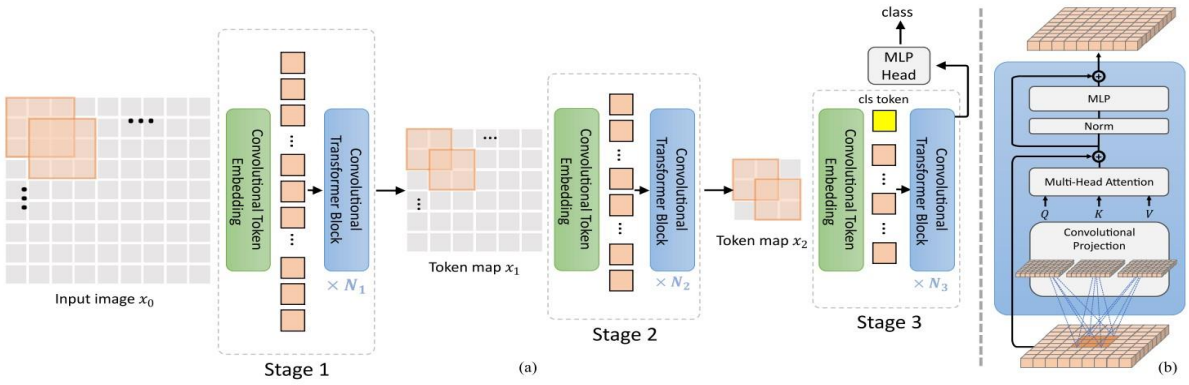


Figure 2: The pipeline of the proposed CvT architecture. (a) Overall architecture, showing the hierarchical multi-stage structure facilitated by the Convolutional Token Embedding layer. (b) Details of the Convolutional Transformer Block, which contains the convolution projection as the first layer.

In the paper, author's implemented three configuration of Convolutional Vision Transformer i.e CvT-13, CvT-21 and CvT-W24 with difference of only some hyperparameters. For evaluation author pretrained the model with ImageNet datasets with batch size of 2028 and then fine-tuned it on Cifar10, Cifar100, Oxford-IIIT Pets and Oxford-IIIT Flowers with batch size of 512. Cvt-W24 was found to be best performing among other with comparatively lower parameters.

	Output Size	Layer Name	CvT-13	CvT-21	CvT-W24
Stage1	56×56	Conv. Embed.	7×7, 64, stride 4		7×7, 192, stride 4
	56×56	Conv. Proj. MHSA MLP	$\begin{bmatrix} 3 \times 3, 64 \\ H_1 = 1, D_1 = 64 \\ R_1 = 4 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 64 \\ H_1 = 1, D_1 = 64 \\ R_1 = 4 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 192 \\ H_1 = 3, D_1 = 192 \\ R_1 = 4 \end{bmatrix} \times 2$
Stage2	28×28	Conv. Embed.	3×3, 192, stride 2		3×3, 768, stride 2
	28×28	Conv. Proj. MHSA MLP	$\begin{bmatrix} 3 \times 3, 192 \\ H_2 = 3, D_2 = 192 \\ R_2 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 192 \\ H_2 = 3, D_2 = 192 \\ R_2 = 4 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 3, 768 \\ H_2 = 12, D_2 = 768 \\ R_2 = 4 \end{bmatrix} \times 2$
Stage3	14×14	Conv. Embed.	3×3, 384, stride 2		3×3, 1024, stride 2
	14×14	Conv. Proj. MHSA MLP	$\begin{bmatrix} 3 \times 3, 384 \\ H_3 = 6, D_3 = 384 \\ R_3 = 4 \end{bmatrix} \times 10$	$\begin{bmatrix} 3 \times 3, 384 \\ H_3 = 6, D_3 = 384 \\ R_3 = 4 \end{bmatrix} \times 16$	$\begin{bmatrix} 3 \times 3, 1024 \\ H_3 = 16, D_3 = 1024 \\ R_3 = 4 \end{bmatrix} \times 20$
Head	1×1	Linear	1000		
Params			19.98 M	31.54 M	276.7 M
FLOPs			4.53 G	7.13 G	60.86 G

3. OUR APPROACH

After reading the paper we decided to re-implement the CvT-13 version of convolutional vision transformer on a low scale. In the paper, CvT model was pretrained on ImageNet dataset for 300 epoch and then further finetuned on relatively small dataset like Cifar10. As due to our GPU limitation we decided to pre-trained our model on Cifar100 and then finetuned it on other small datasets [3] [5] for a binary and multi classification problem. We also tried our best to follow similar architecture as proposed in original CvT but with a lot of hyperparameter changes to run it on our machine faster and efficiently.

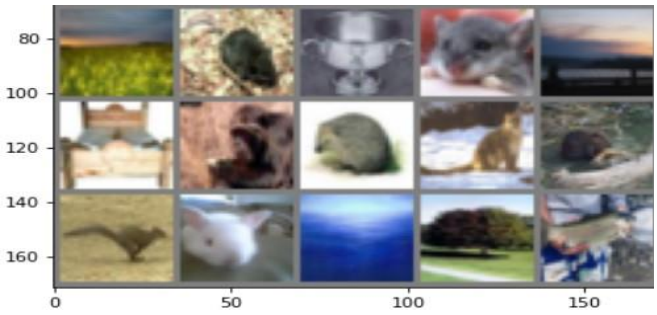
In the paper, model was pre-trained with an input resolution of "224 X 224" and fine-tuned resolution of "384 X 384"; but in our model we decided to feed input of 32 X 32 both at pre-training and fine tuning. we pre-trained our model for 350 epoch at 512 batch size.

After implementing all the architecture we tried different experiments to compare our result with different dataset that our machine could handle.

4. Datasets Details

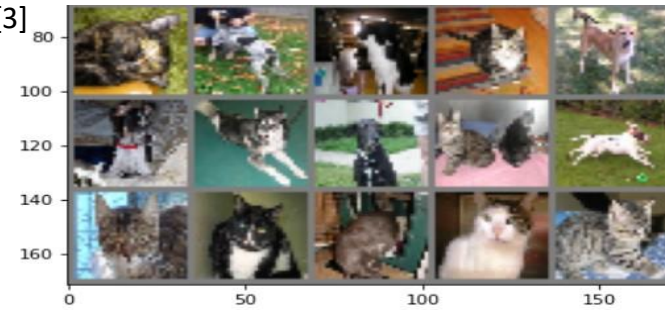
Cifar100 was used for pretraining.

- Classes: 100
- Total images: 60000
- Train: 50000
- Test: 10000



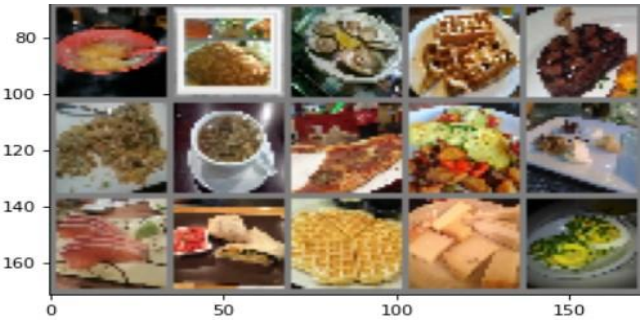
Cat Vs Dog dataset was used for fine-tuning [3]

- Classes: 2
- Total images: 24998
- Train: 17499
- Test: 7499



Food dataset was used for fine-tuning [5]

- Classes: 11
- Total images: 13434
- Train: 9404
- Test: 4030



5. Pre-processing of Datasets

Before feeding data into our model, we preprocess our dataset i.e. Cifar100 for pre-training and CatVDog [3] & Food [5] for fine tuning. First by using compose transform function of pytorch we normalize and resize (3,32,32) our dataset; afterwards dataset is split between training and testing sets (70% train and 30 % test).

6. Hyperparameters

we have selected below hyperparameter for pre-training and fine tuning considering our GPU capabilities.

6.1 pretraining

Number of Epochs = 350

Batch Size = 512
Input Size = 32
Initial Learning rate (scheduler cosine annealing) = 0.02(as per paper [1])
Optimizer = AdamW optimizer (as per paper [1])
Loss Function= Cross Entropy loss

6.2 Fine-Tuning

Food Dataset

Number of Epochs = 50
Batch Size = 256
Input Size = 32
Learning rate = 0.1
Optimizer = SGD, momentum =0.9
Loss Function = Cross Entropy loss

Cat Vs Dog Dataset

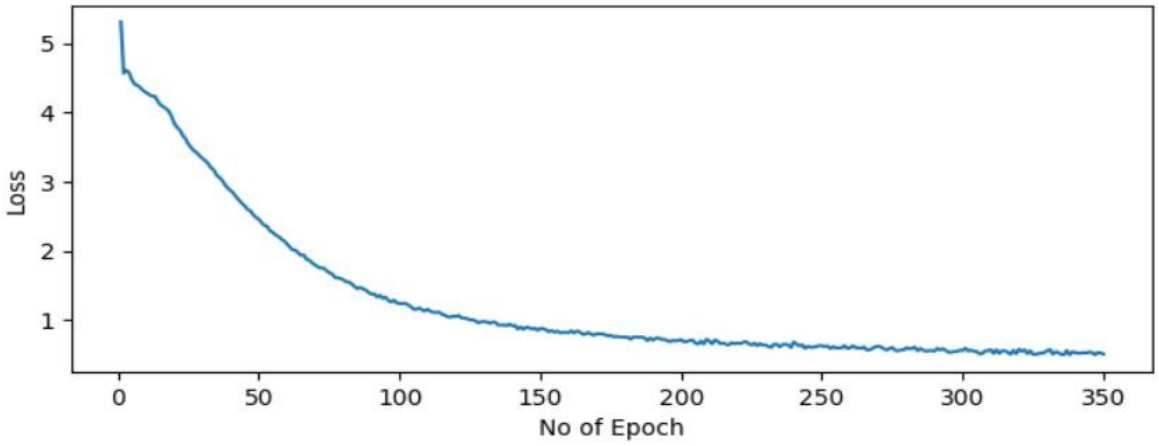
Number of Epochs = 20
Batch Size = 256
Input Size = 32
Learning rate = 0.1
Optimizer = SGD, momentum =0.9
Loss Function = Binary cross Entropy

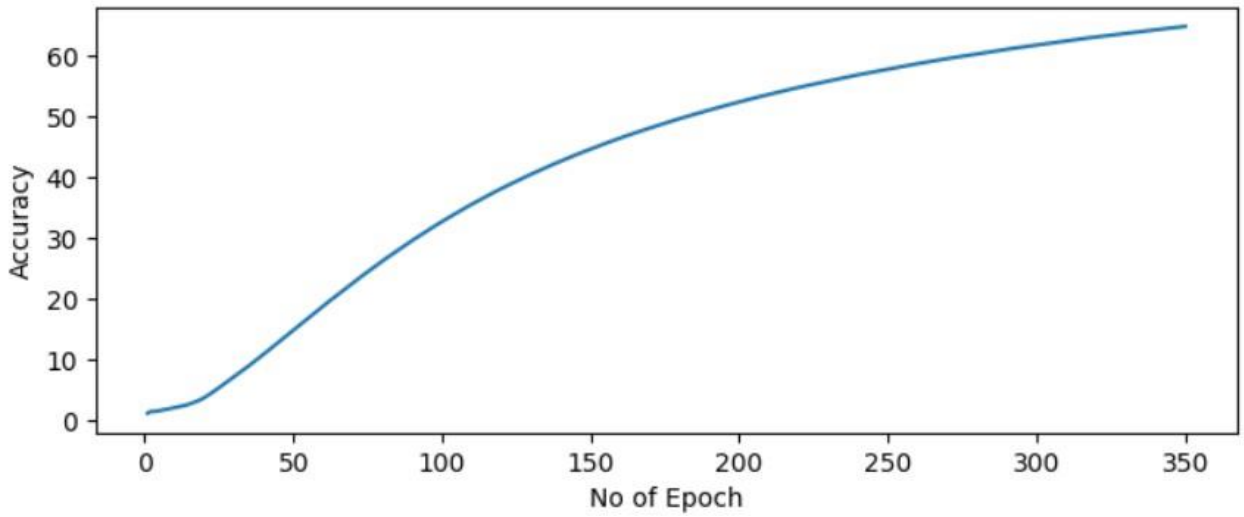
7. Evaluation

We have evaluated our model accuracy by pretraining again and again and we have observed that accuracy keep on increasing on every epoch, so if we increase number of epoch from 350 to a large number there is a chance of decent accuracy

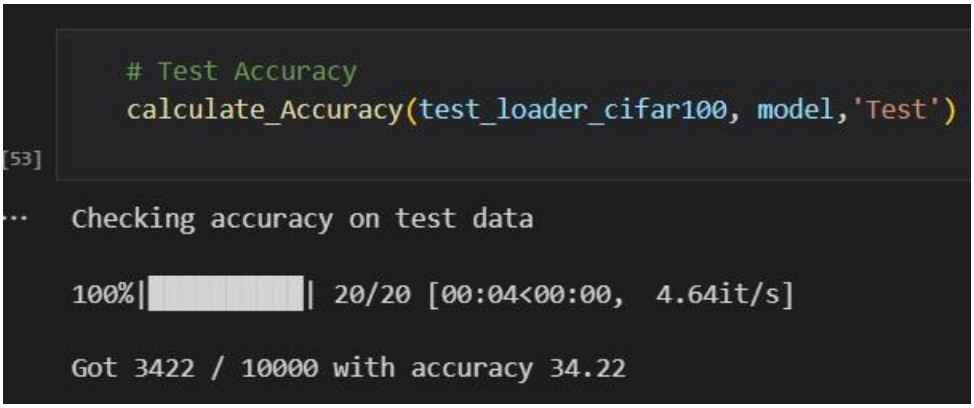
7.1 After pretraining

Below are the loss and accuracy plot against number of epochs.



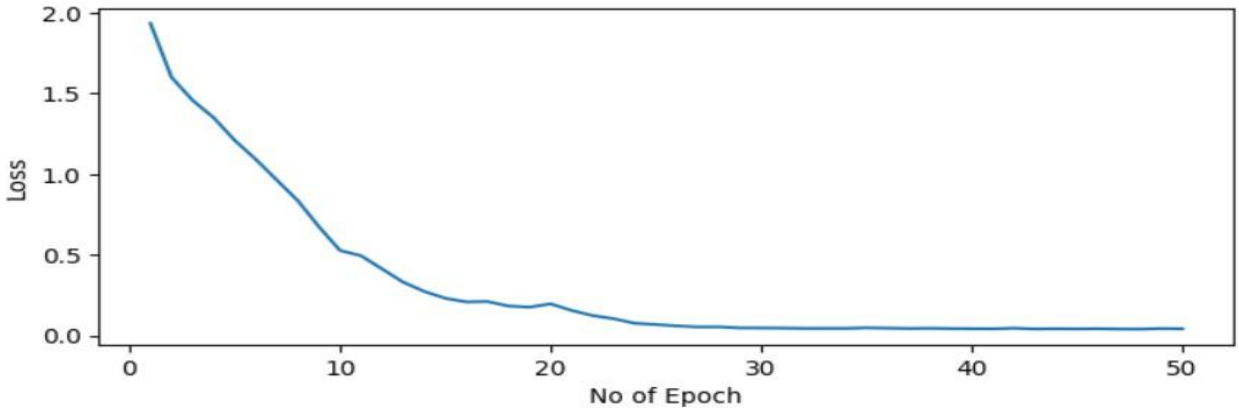


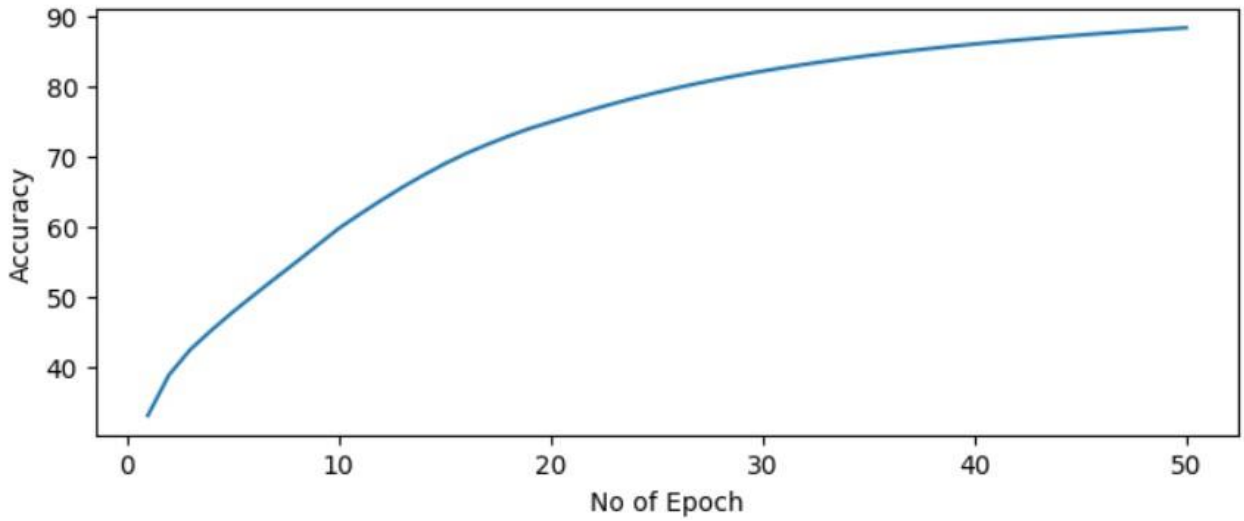
After pre-training for 350 epoch we have observed the train accuracy of above 70 % and test accuracy of 34.22%. which can also be observed from our Jupyter notebook.



7.2 After fine tuning

Below can be seen train loss and accuracy plot against number of epochs for our food dataset.





From below block of code over test accuracy can be observed i.e **41.59%** . We tried several hyperparameter changes, but no significant result was achieved in term of increasing accuracy.

```
# Train Accuracy
calculate_Accuracy(train_loader_Food, model_Food,"Train")
# Test Accuracy
calculate_Accuracy(test_loader_Food, model_Food,'Test')
```

✓ 1m 33.4s

```
Checking accuracy on training data

100%|██████████| 37/37 [01:05<00:00,  1.77s/it]

Got 9216 / 9404 with accuracy 98.00
Checking accuracy on test data

100%|██████████| 16/16 [00:27<00:00,  1.73s/it]

Got 1676 / 4030 with accuracy 41.59
```

While for our cat and dog data set our test accuracy was about 50 %. Both training and test accuracy is almost similar, nut there was no accuracy increase even after training for many epoch so we just stopped at 20 epoch.

8. CONCLUSION AND SUMMARY

In Summary we have implemented low scale version of Convolutional vision transformer proposed in paper [1] with low resolution pictures and a very small dataset due to our machine capability. we conclude that if we pretrained our model as well on a large dataset we will also be able to achieve a decent accuracy. On CatVsDog dataset we were able to get accuracy around 50% and in Food dataset with 13000 images and 11 classes accuracy was 42.2%. below mentioned action could increase our accuracy further.

- We are feeding an input of 32x32 which in the last stage of model down sampled to 2x2, this can be one of the reasons of low accuracy. By feeding a dataset of high-resolution pictures, accuracy can be improved.
- Model must be trained on a larger dataset.
- If dataset is not large enough, we should increase number of epoch for pretraining for better accuracy.
- Batch size can be reduced for better accuracy.

This project allow us to understand state of the art method for image classification and also the importance of setting the correct hyperparameters to balance between computational time and accuracy.

9. REFRENCES

Below are the references that were used in this project;

[1]. CvT: Introducing Convolutions to Vision Transformers -Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, Lei Zhang (https://arxiv.org/pdf/2103.15808)

[2]. AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE - Alexey Dosovitskiy∗,† Lucas Beyer∗, Alexander Kolesnikov∗ , Dirk Weissenborn∗ , Xiaohua Zhai∗, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer,Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby∗ (https://arxiv.org/pdf/2010.11929v2.pdf)

[3]. Cats and Dogs Dataset (https://www.microsoft.com/enus/download/details.aspx?id=54765)

[4]. Code refrence for fine tuning function (https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html)

[5]. Kaggle Food dataset-11 classes (https://www.kaggle.com/datasets/trolukovich/food11-image-dataset?select=training)