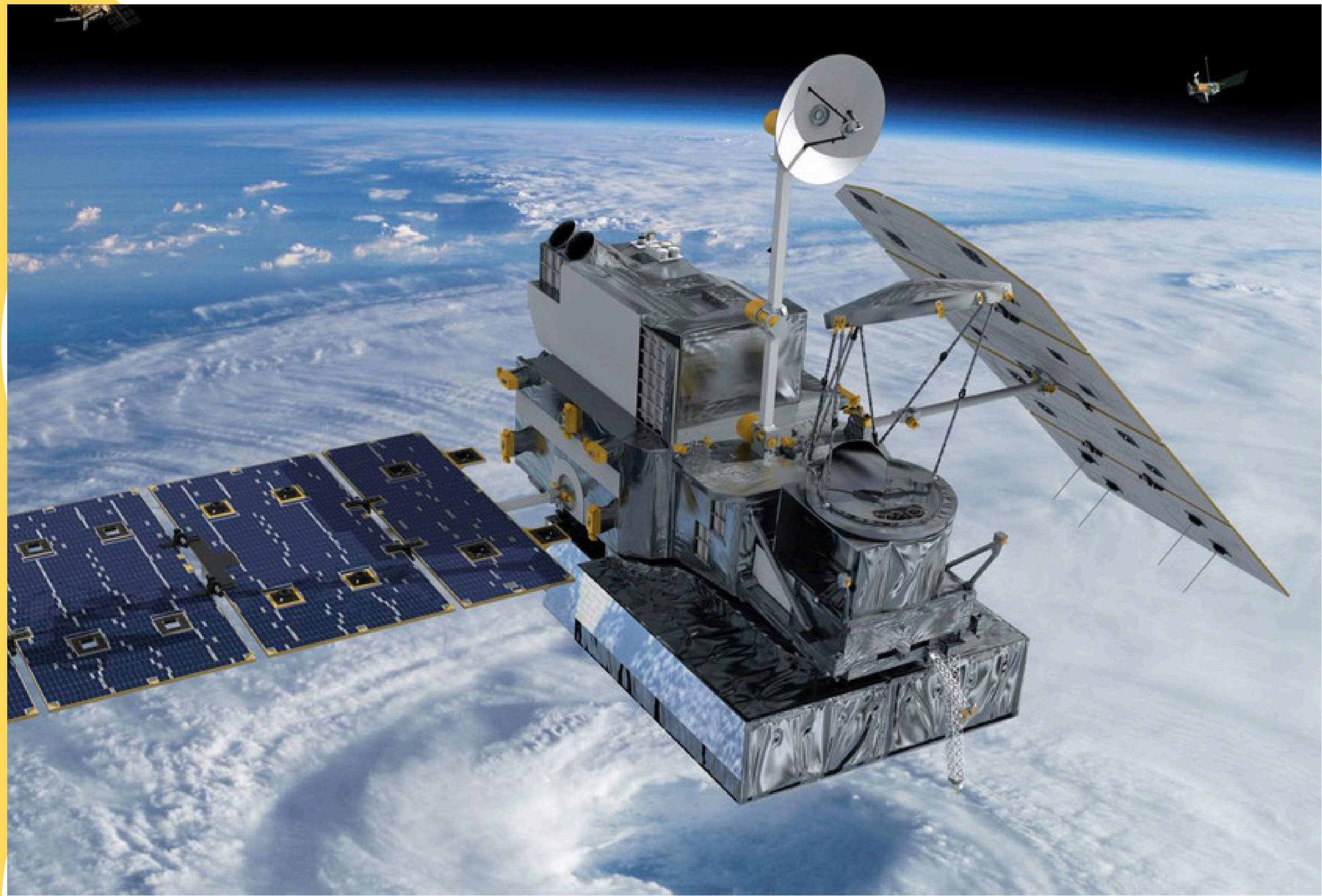
Satellite Image Classification

BootCamp in Data Science, Concordia University

FRIZAT DENIS



Hugging Face

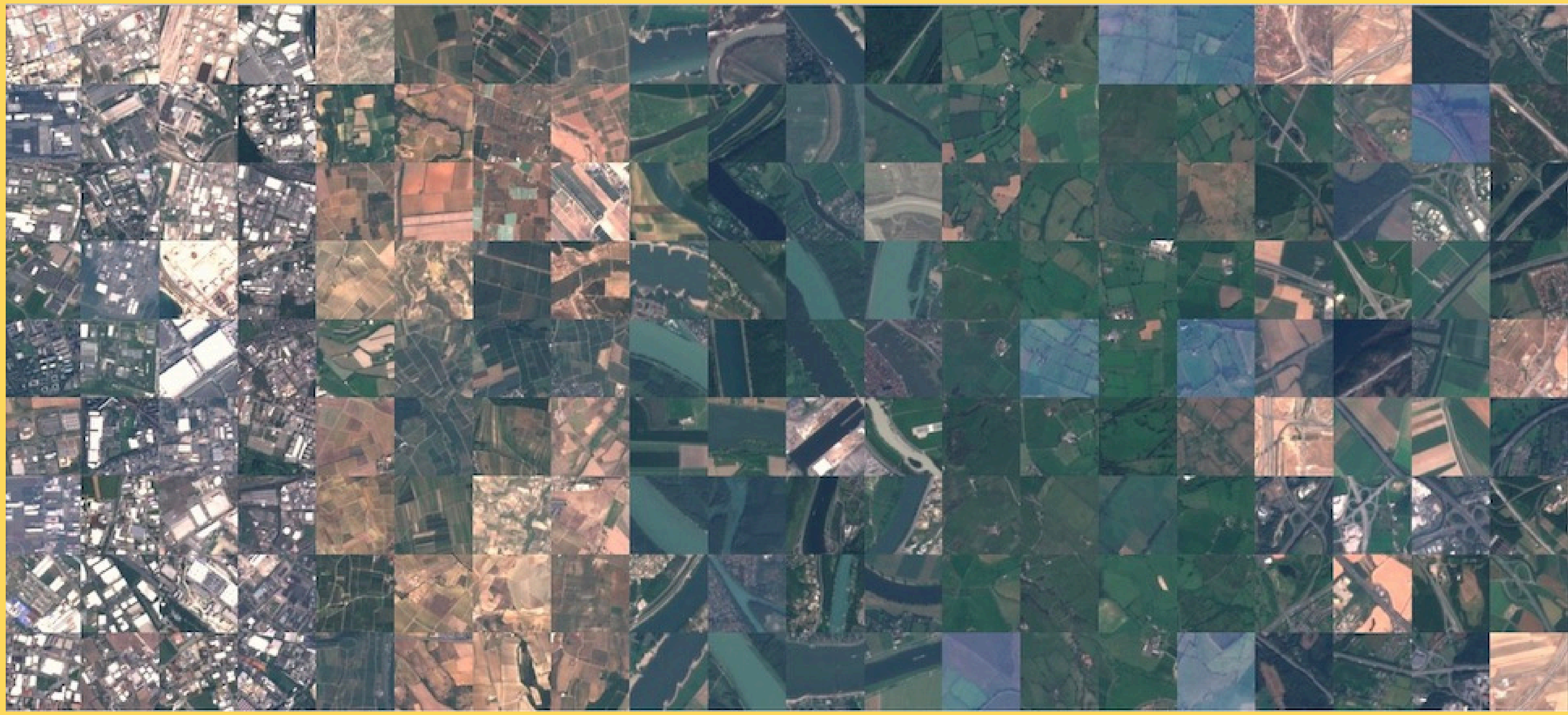
- Pre-trained model (lot of tasks in NLPor computer Vision)
- amazing hub for machine learning model (host thousands of model, dataset and demos) in open source community
- lot of documentation
- Space for train and deploy your model

Dataset

Satellite Image from EuroSAT

- Image classification dataset based on satellite images captured by the Sentinel-2

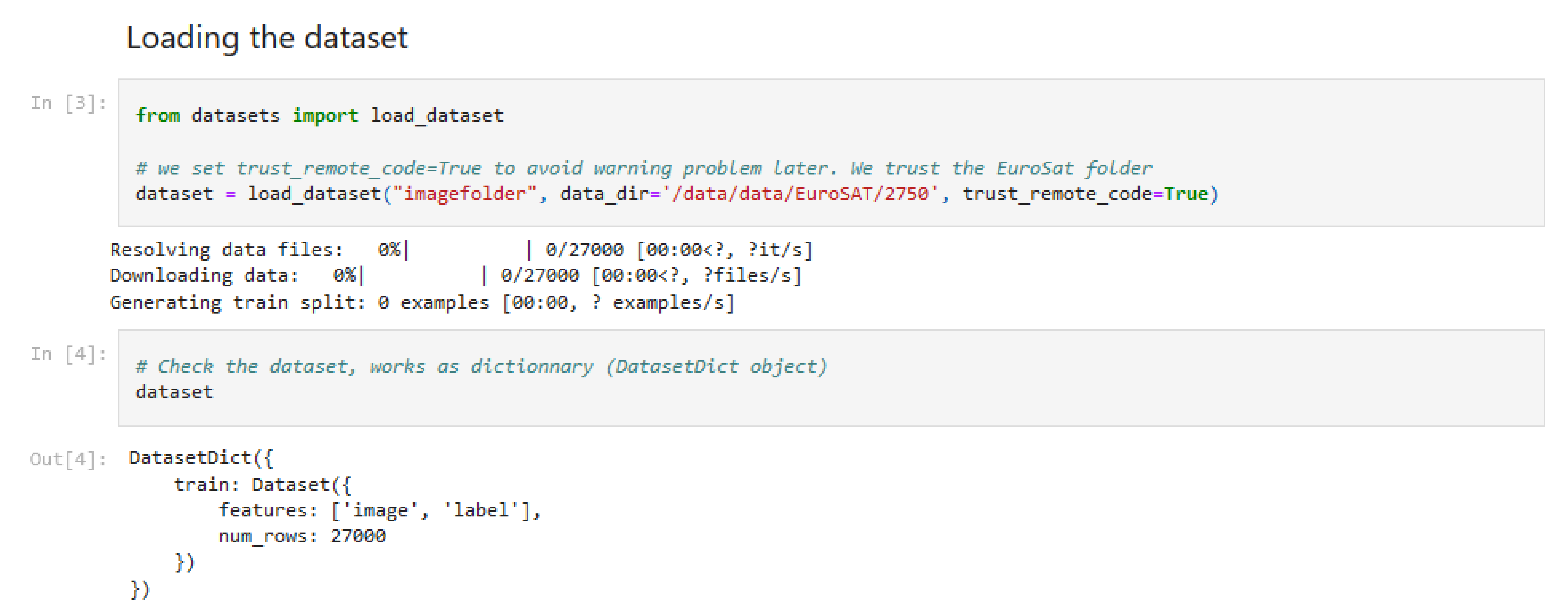
- The dataset consisting out of 10 classes :
  - Forest
  - River
  - Highway
  - Annual Crop
  - Sea Lake
  - Herbaceous Vegetation
  - Industrial
  - Residential
  - Permanent Crop
  - Pasture



- In total 27 000 labelled images

Source: EuroSAT

Importing data set



Loading the dataset

In [3]:

```
from datasets import load_dataset

# we set trust_remote_code=True to avoid warning problem later. We trust the EuroSat folder
dataset = load_dataset("imagefolder", data_dir='/data/data/EuroSAT/2750', trust_remote_code=True)
```

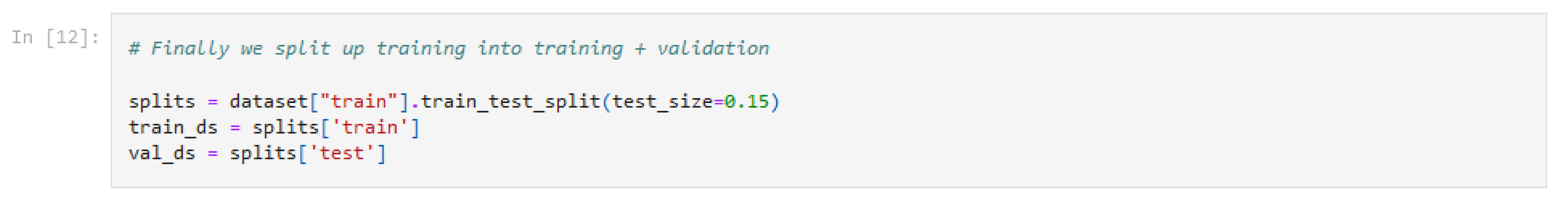
```
Resolving data files:   0%|          | 0/27000 [00:00<?, ?it/s]
Downloading data:   0%|          | 0/27000 [00:00<?, ?files/s]
Generating train split: 0 examples [00:00, ? examples/s]
```

In [4]:

```
# Check the dataset, works as dictionnary (DatasetDict object)
dataset
```

Out[4]:

```
DatasetDict({
    train: Dataset({
        features: ['image', 'label'],
        num_rows: 27000
    })
})
```



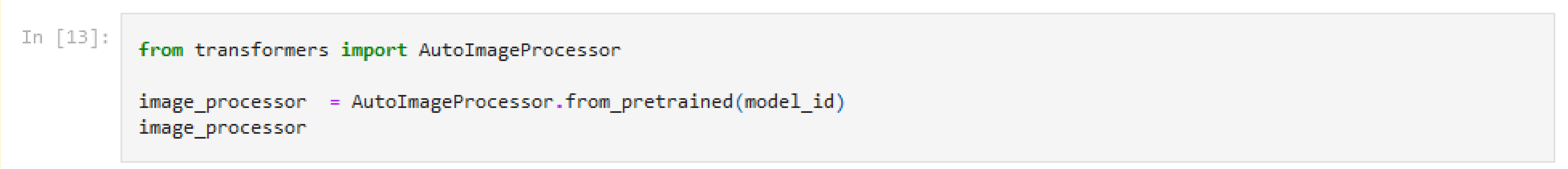
In [12]:

```
# Finally we split up training into training + validation

splits = dataset["train"].train_test_split(test_size=0.15)
train_ds = splits['train']
val_ds = splits['test']
```

Processing Data

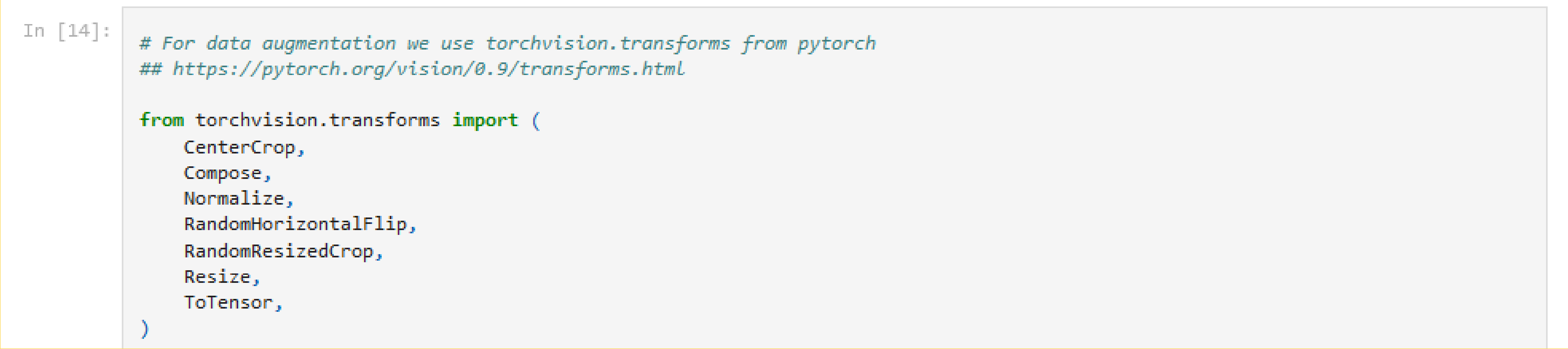
Pre-processing / Data augmentation

To make sure we use the appropriate image mean and standard deviation for the model architecture we are going to use, we instantiate what is called an image processor with the ‘AutoImageProcessor.from_pretrained’ method.



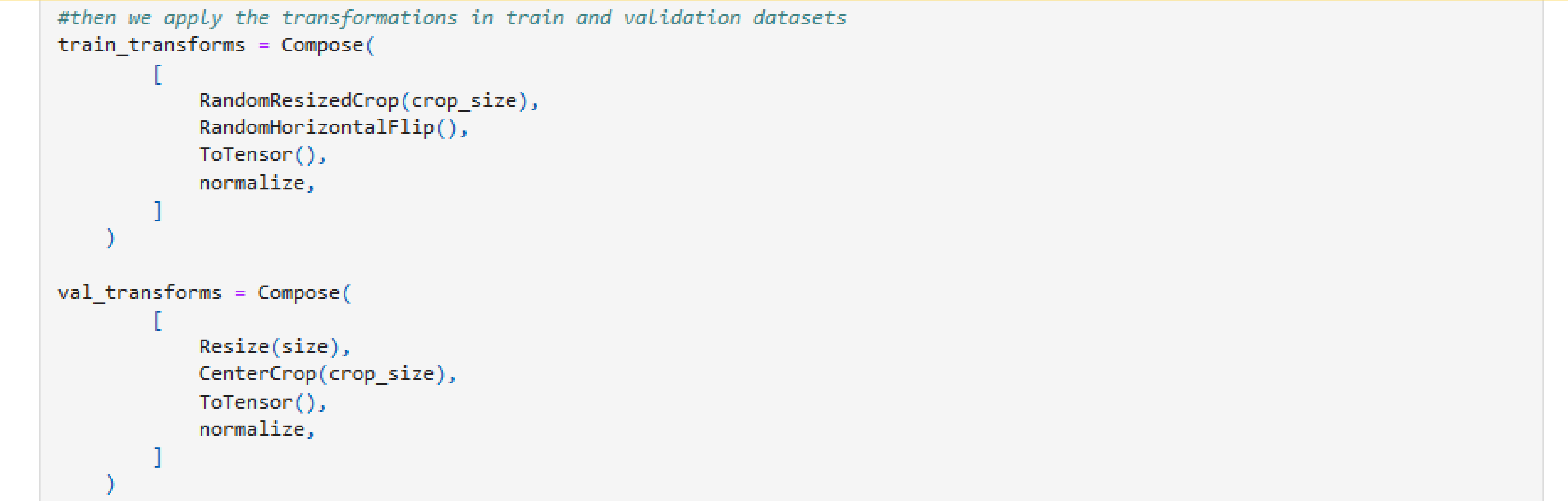
In [13]:

```
from transformers import AutoImageProcessor

image_processor  = AutoImageProcessor.from_pretrained(model_id)
image_processor
```



In [14]:

```
# For data augmentation we use torchvision.transforms from pytorch
## https://pytorch.org/vision/0.9/transforms.html

from torchvision.transforms import (
    CenterCrop,
    Compose,
    Normalize,
    RandomHorizontalFlip,
    RandomResizedCrop,
    Resize,
    ToTensor,
)
```



```python
#then we apply the transformations in train and validation datasets
train_transforms = Compose(
        [
            RandomResizedCrop(crop_size),
            RandomHorizontalFlip(),
            ToTensor(),
            normalize,
        ]
    )

val_transforms = Compose(
        [
            Resize(size),
            CenterCrop(crop_size),
            ToTensor(),
            normalize,
        ]
    )
```

Which model do we want to use ?

Models 1/3

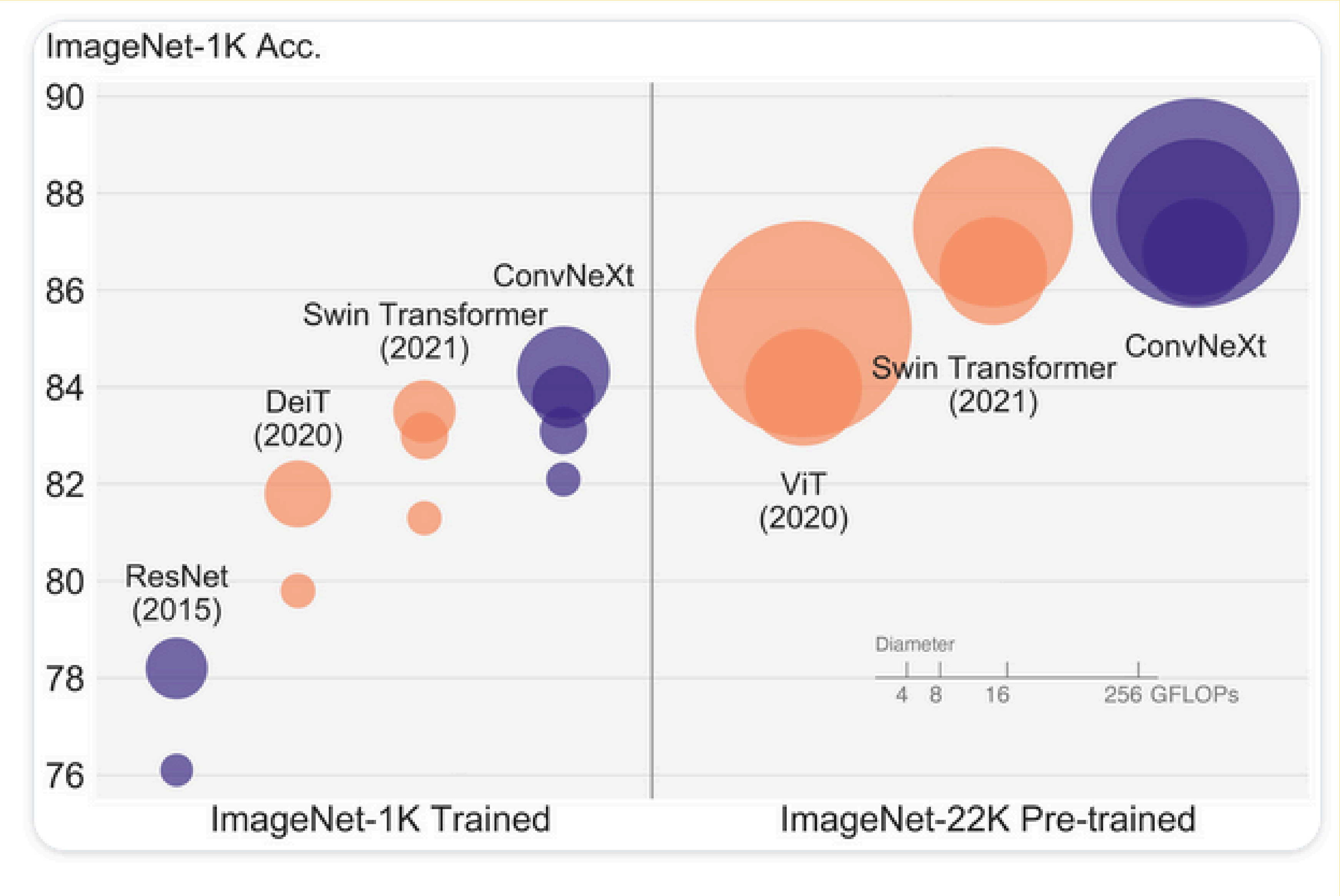
ConvNeXT

A ConvNet for the 2020s by Zhuang Liu ets al

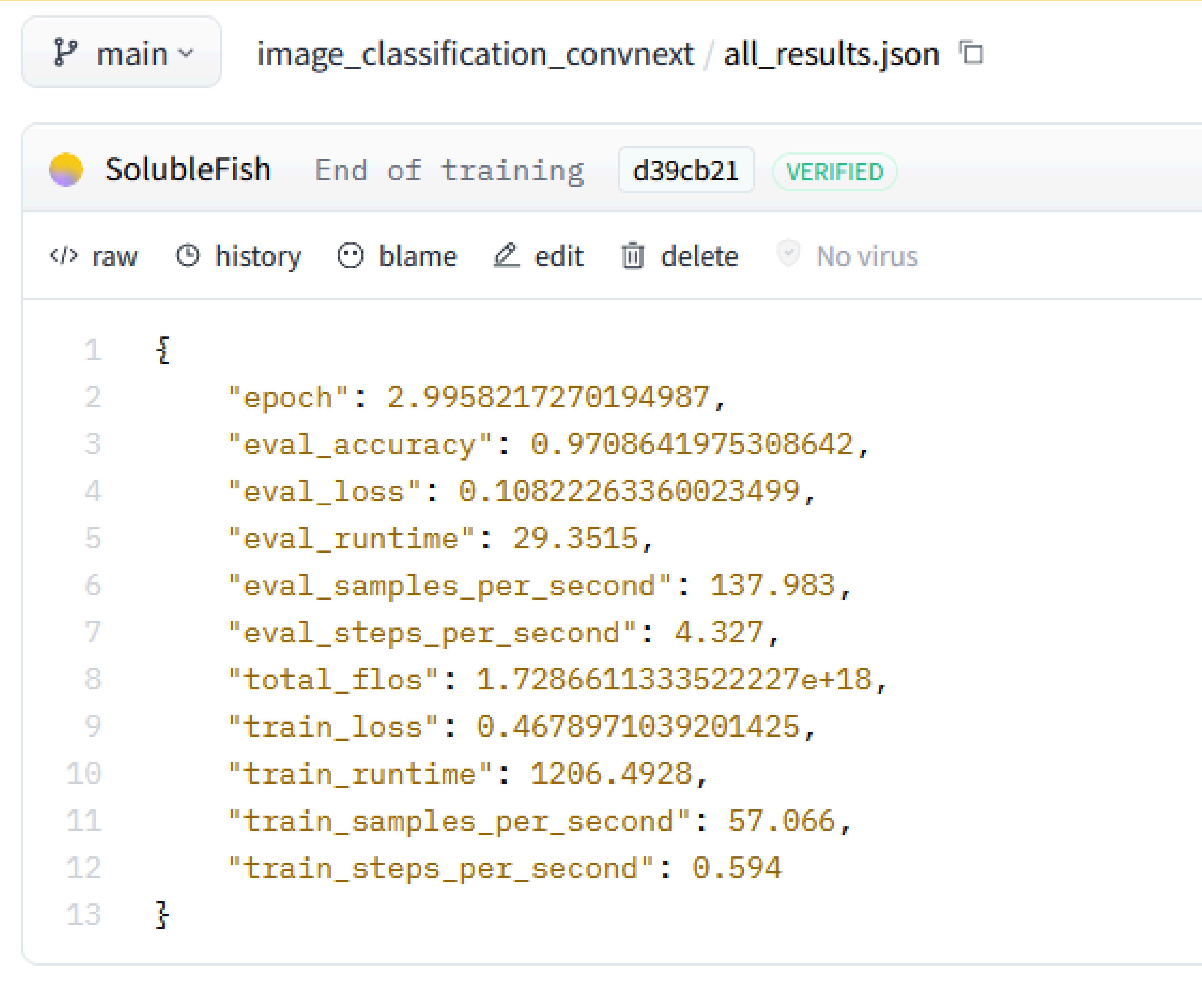
- Convolutional Neural Network Model that come from ConvNet (state-of-the-art image classification model)

- Exploration into the structure of ConvNet, design spaces, test the limits. Discover several key components that contribute to improve the performance in **classification task**

- Constructed entirely from standard ConvNet modules, gives high results in terms of accuracy and scalability



ConvNeXT results



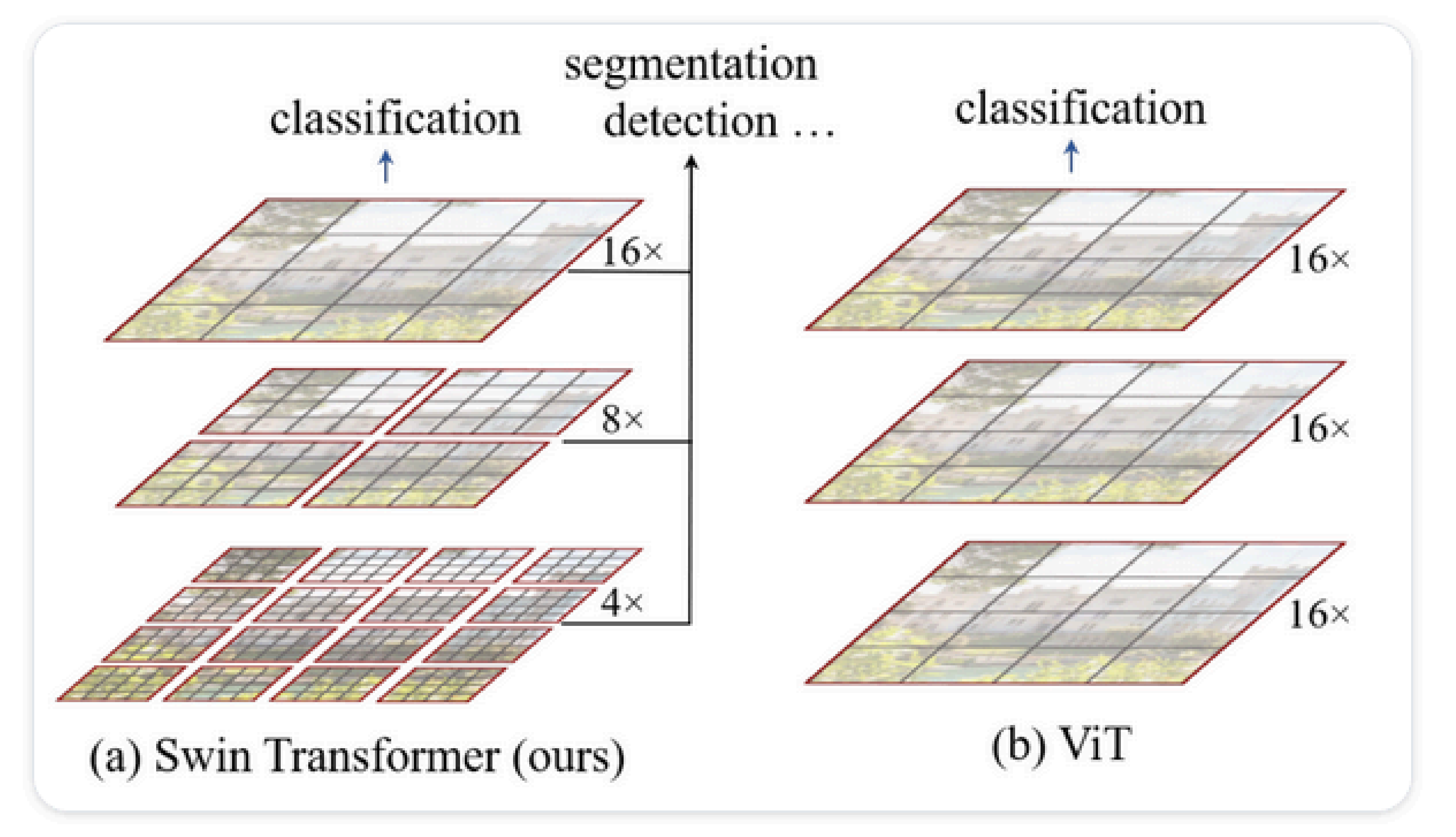
```
{
    "epoch": 2.9958217270194987,
    "eval_accuracy": 0.9708641975308642,
    "eval_loss": 0.10822263360023499,
    "eval_runtime": 29.3515,
    "eval_samples_per_second": 137.983,
    "eval_steps_per_second": 4.327,
    "total_flos": 1.7286611333522227e+18,
    "train_loss": 0.4678971039201425,
    "train_runtime": 1206.4928,
    "train_samples_per_second": 57.066,
    "train_steps_per_second": 0.594
}
```

Models 2/3

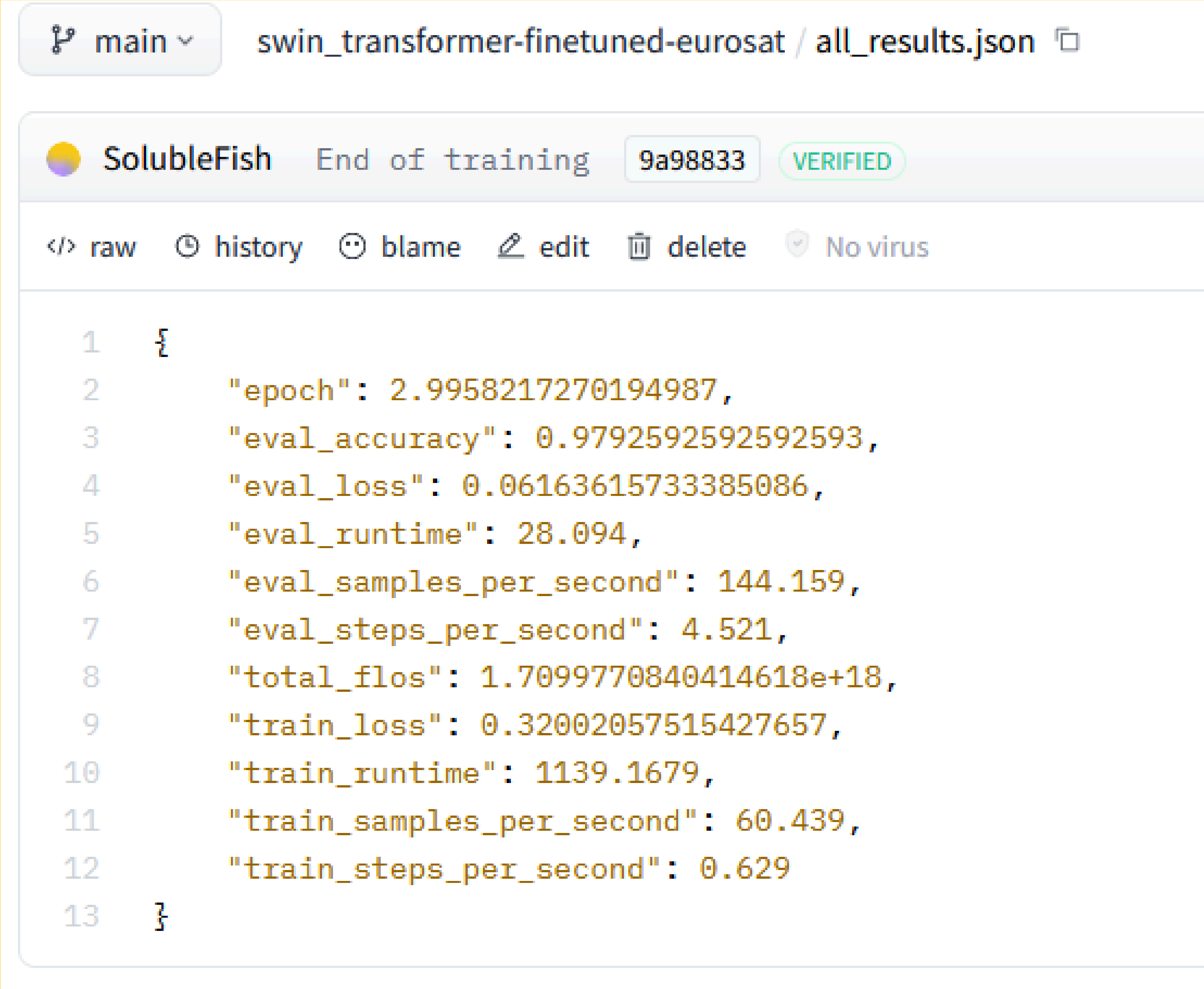
Swin Transformer

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows by Ze Liu ets al

- The shifted windowing scheme brings greater efficiency by limiting self-attention computation to non-overlapping local windows while also allowing for cross-window connection.
- This hierarchical architecture has the flexibility to model at various scales and has linear computational complexity with respect to image size.



Swin results



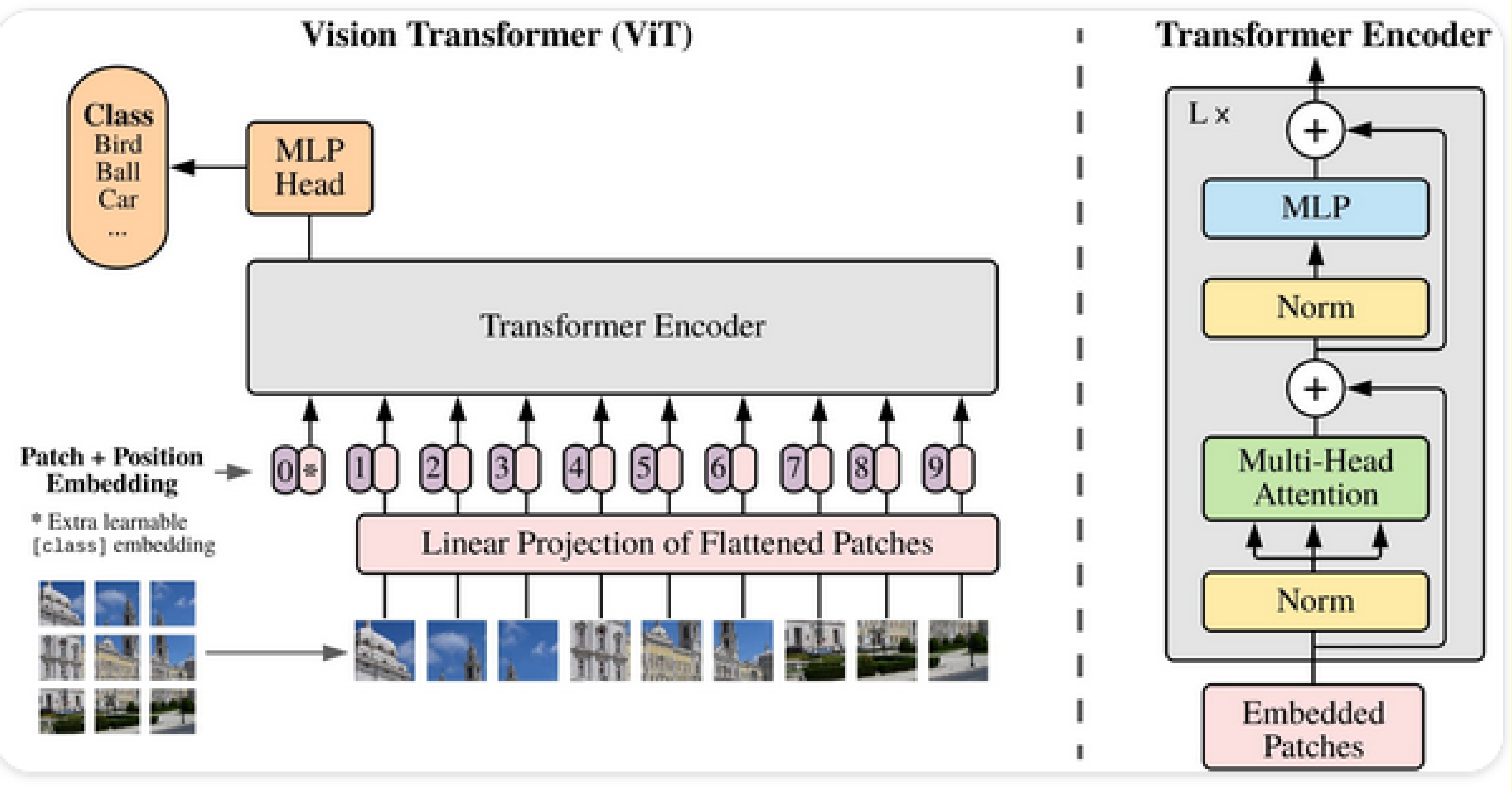
```
{
    "epoch": 2.9958217270194987,
    "eval_accuracy": 0.9792592592592593,
    "eval_loss": 0.06163615733385086,
    "eval_runtime": 28.094,
    "eval_samples_per_second": 144.159,
    "eval_steps_per_second": 4.521,
    "total_flos": 1.7099770840414618e+18,
    "train_loss": 0.32002057515427657,
    "train_runtime": 1139.1679,
    "train_samples_per_second": 60.439,
    "train_steps_per_second": 0.629
}
```

Models 3/3

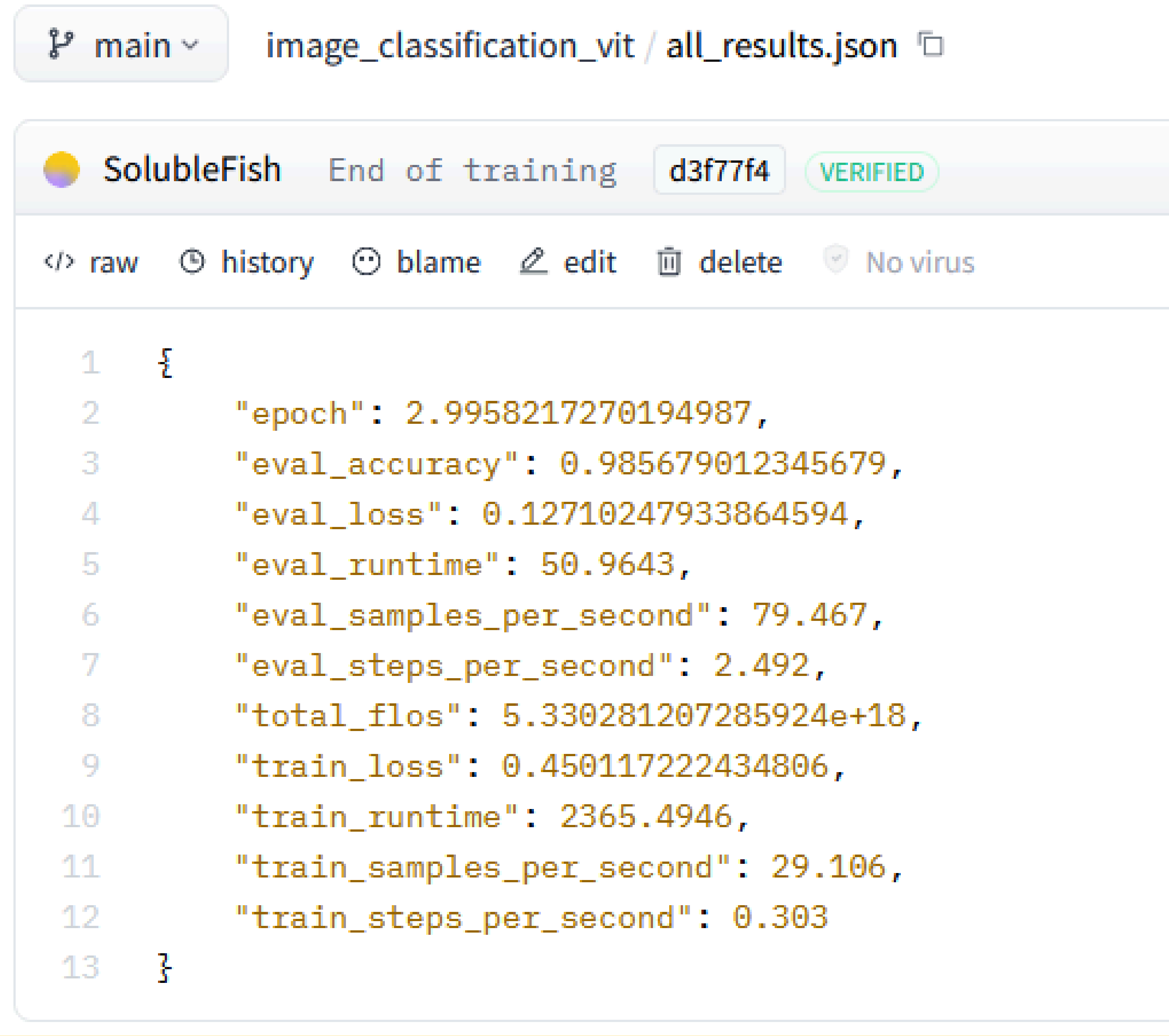
Vision Transformer (ViT)

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale by Alexey Dosovitskiy ets al

- In vision, CNN architecture model focus all attention
- This reliance in CNNs model is not necessary
- Transformer applied to sequences of image patches can perform very well on **image classification tasks**
- Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks

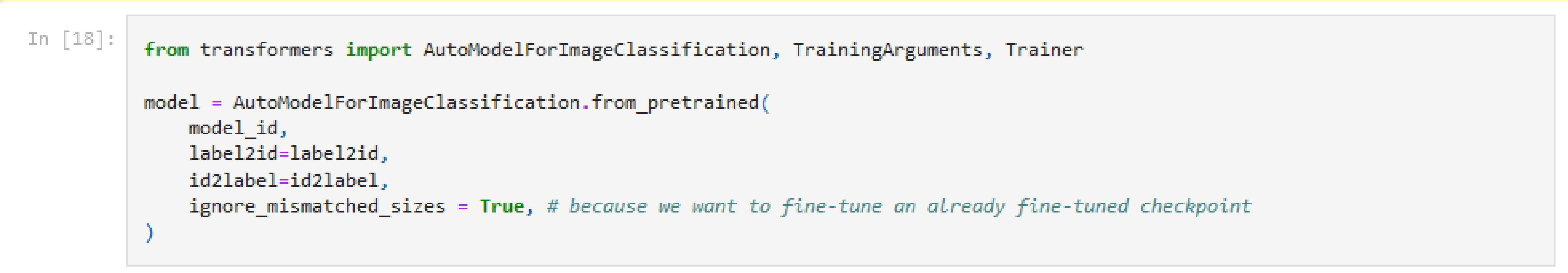


ViT results


main
image_classification_vit / all_results.json
SolubleFish End of training d3f77f4 VERIFIED
raw
history
blame
edit
delete
No virus

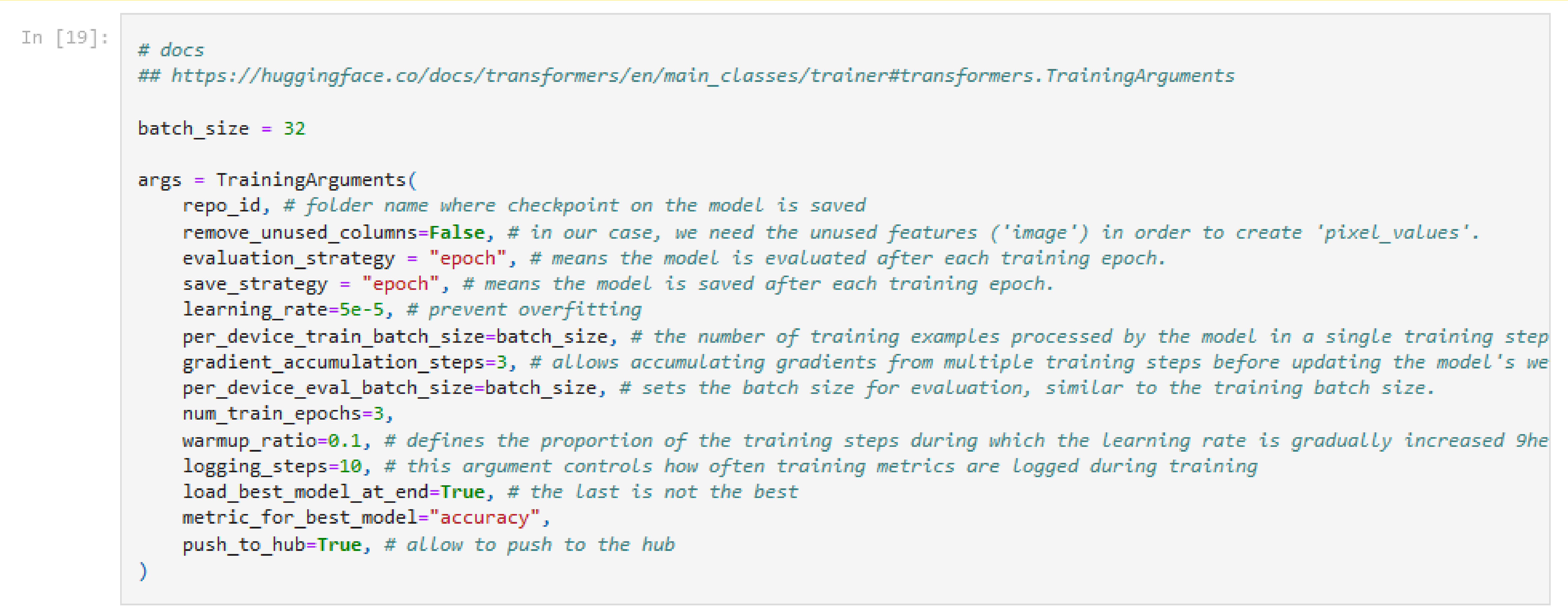
```
{
    "epoch": 2.9958217270194987,
    "eval_accuracy": 0.985679012345679,
    "eval_loss": 0.12710247933864594,
    "eval_runtime": 50.9643,
    "eval_samples_per_second": 79.467,
    "eval_steps_per_second": 2.492,
    "total_flos": 5.330281207285924e+18,
    "train_loss": 0.450117222434806,
    "train_runtime": 2365.4946,
    "train_samples_per_second": 29.106,
    "train_steps_per_second": 0.303
}
```

HuggingFace Space



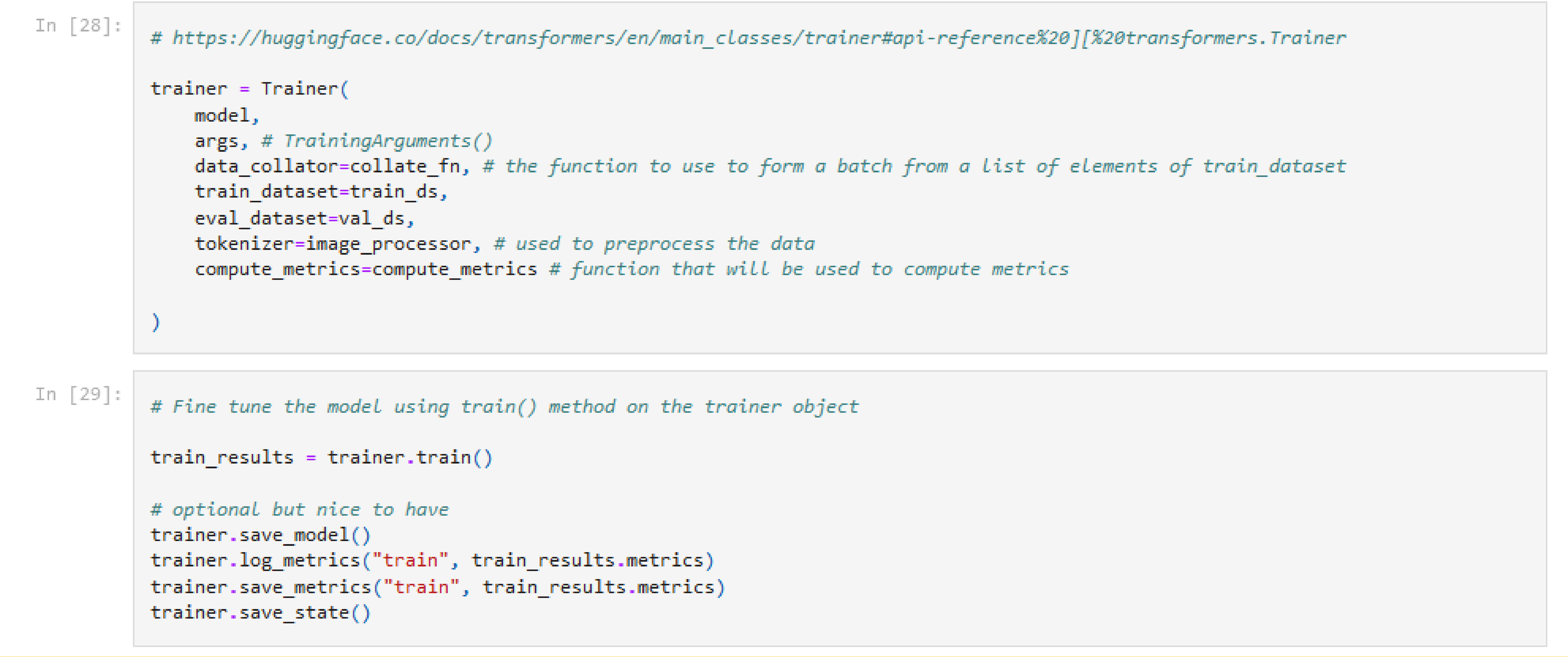
In [18]:

```python
from transformers import AutoModelForImageClassification, TrainingArguments, Trainer

model = AutoModelForImageClassification.from_pretrained(
    model_id,
    label2id=label2id,
    id2label=id2label,
    ignore_mismatched_sizes = True, # because we want to fine-tune an already fine-tuned checkpoint
)
```



In [19]:

```python
# docs
## https://huggingface.co/docs/transformers/en/main_classes/trainer#transformers.TrainingArguments

batch_size = 32

args = TrainingArguments(
    repo_id, # folder name where checkpoint on the model is saved
    remove_unused_columns=False, # in our case, we need the unused features ('image') in order to create 'pixel_values'.
    evaluation_strategy = "epoch", # means the model is evaluated after each training epoch.
    save_strategy = "epoch", # means the model is saved after each training epoch.
    learning_rate=5e-5, # prevent overfitting
    per_device_train_batch_size=batch_size, # the number of training examples processed by the model in a single training step
    gradient_accumulation_steps=3, # allows accumulating gradients from multiple training steps before updating the model's we
    per_device_eval_batch_size=batch_size, # sets the batch size for evaluation, similar to the training batch size.
    num_train_epochs=3,
    warmup_ratio=0.1, # defines the proportion of the training steps during which the learning rate is gradually increased 9he
    logging_steps=10, # this argument controls how often training metrics are logged during training
    load_best_model_at_end=True, # the last is not the best
    metric_for_best_model="accuracy",
    push_to_hub=True, # allow to push to the hub
)
```



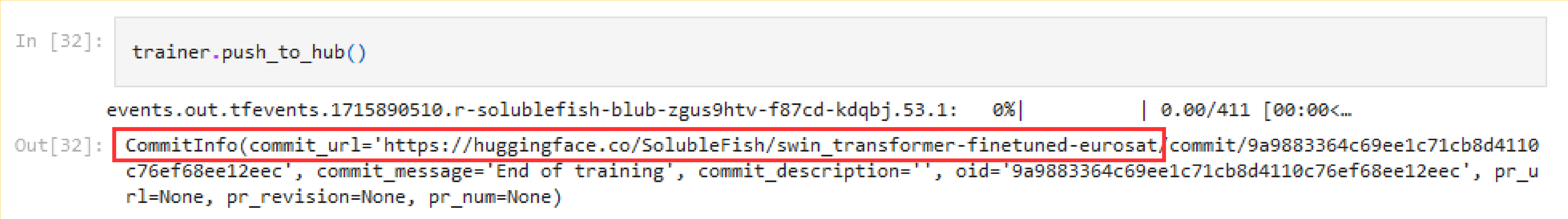
In [28]:

```
# https://huggingface.co/docs/transformers/en/main_classes/trainer#api-reference%20][%20transformers.Trainer

trainer = Trainer(
    model,
    args, # TrainingArguments()
    data_collator=collate_fn, # the function to use to form a batch from a list of elements of train_dataset
    train_dataset=train_ds,
    eval_dataset=val_ds,
    tokenizer=image_processor, # used to preprocess the data
    compute_metrics=compute_metrics # function that will be used to compute metrics

)
```

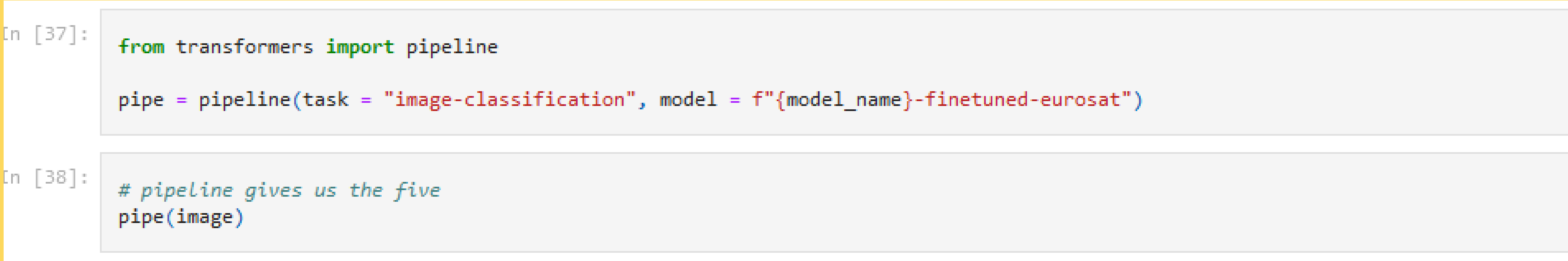
In [29]:

```
# Fine tune the model using train() method on the trainer object

train_results = trainer.train()

# optional but nice to have
trainer.save_model()
trainer.log_metrics("train", train_results.metrics)
trainer.save_metrics("train", train_results.metrics)
trainer.save_state()
```

Push to the hub



```
In [32]:  trainer.push_to_hub()

          events.out.tfevents.1715890510.r-solublefish-blub-zgus9htv-f87cd-kdqbj.53.1:   0%|          | 0.00/411 [00:00<…

Out[32]:  CommitInfo(commit_url='https://huggingface.co/SolubleFish/swin_transformer-finetuned-eurosat/commit/9a9883364c69ee1c71cb8d4110
          c76ef68ee12eec', commit_message='End of training', commit_description='', oid='9a9883364c69ee1c71cb8d4110c76ef68ee12eec', pr_u
          rl=None, pr_revision=None, pr_num=None)
```



```
In [37]:  from transformers import pipeline

          pipe = pipeline(task = "image-classification", model = f"{model_name}-finetuned-eurosat")

In [38]:  # pipeline gives us the five
          pipe(image)
```

Deployment

https://huggingface.co/spaces/SolubleFish/Concordia_project_deploy

Thank you !