

# Term Project

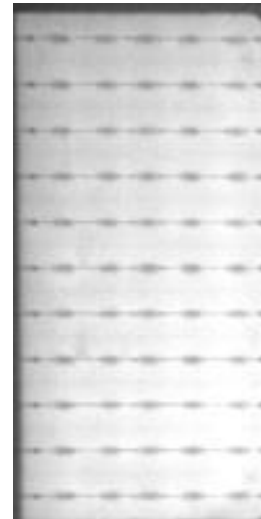
---

## Deep Learning

# Task

- EL-Image Classification
  - Electroluminescence (EL) images are used to detect defects or evaluate quality of solar panels and LEDs.
  - We are going to categorize these images into normal and fault types by training a neural network.
- Dataset
  - Input size: 100x200
  - Classes: Normal/Fault

	Normal	Fault
Train	8000	2014
Test	2000	503



Normal cell



Fault cell



# Submission

---

- Due: 24.11.29 (Fri) 23:59
- Submit file
  - Report (studentid\_name.pdf)
  - Source code and state dict(model.pth)
    - Must submit in student\_id.zip / ex) 2024111111.zip
    - Size of state\_dict  $\leq$  60MB
  - Don't forget to remove all the training images when you submit the zip file!

# Test Process

- Execute run.py
  - Test data will be in test\_data folder.
  - You must designate trained model path in argument.

```
if __name__ == '__main__':  
    parser = argparse.ArgumentParser(description='2024 DL Term Project')  
    parser.add_argument('--load-model', default='./checkpoints/model.pth', help="Model's state_dict")  
    parser.add_argument('--batch-size', default=1, help='test loader batch size')  
    parser.add_argument('--fault-dir', default='./test_data/fault_image', help='Directory for fault images')  
    parser.add_argument('--normal-dir', default='./test_data/normal_image', help='Directory for normal images')
```

- Result.txt will be generated.
- We will rank based on the result.txt file
- Do not modify run.py

# Skeleton code

- Term\_Project
  - checkpoints
  - term\_project\_train\_data
  - utils
    - \_utils.py
  - model.py
  - test.py
  - train.py
  - run.py

execute run.py



- Term\_Project
  - checkpoints
  - term\_project\_train\_data
  - utils
    - \_utils.py
  - model.py
  - test.py
  - train.py
  - run.py
  - result.txt

# Grading

---

- 100 points total
  - Performance: 60 points
    - Points:  $60 - (\text{your rank})$
  - Report: 40 points
- Non-runnable code will be big minus point!
- The run.py file must only perform inference, not training => huge minus!

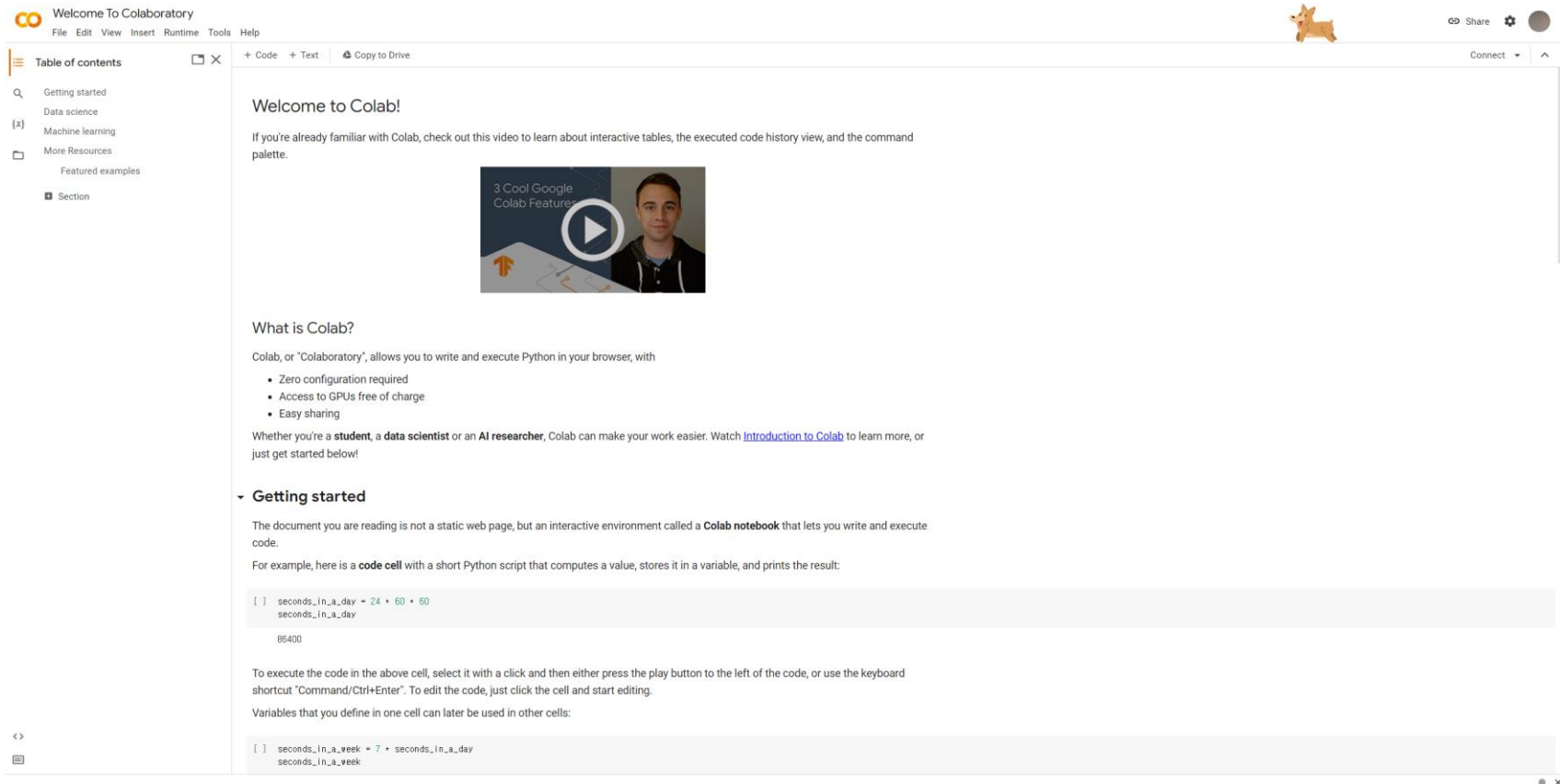
# Google Colaboratory

---

## Deep Learning

# Colab

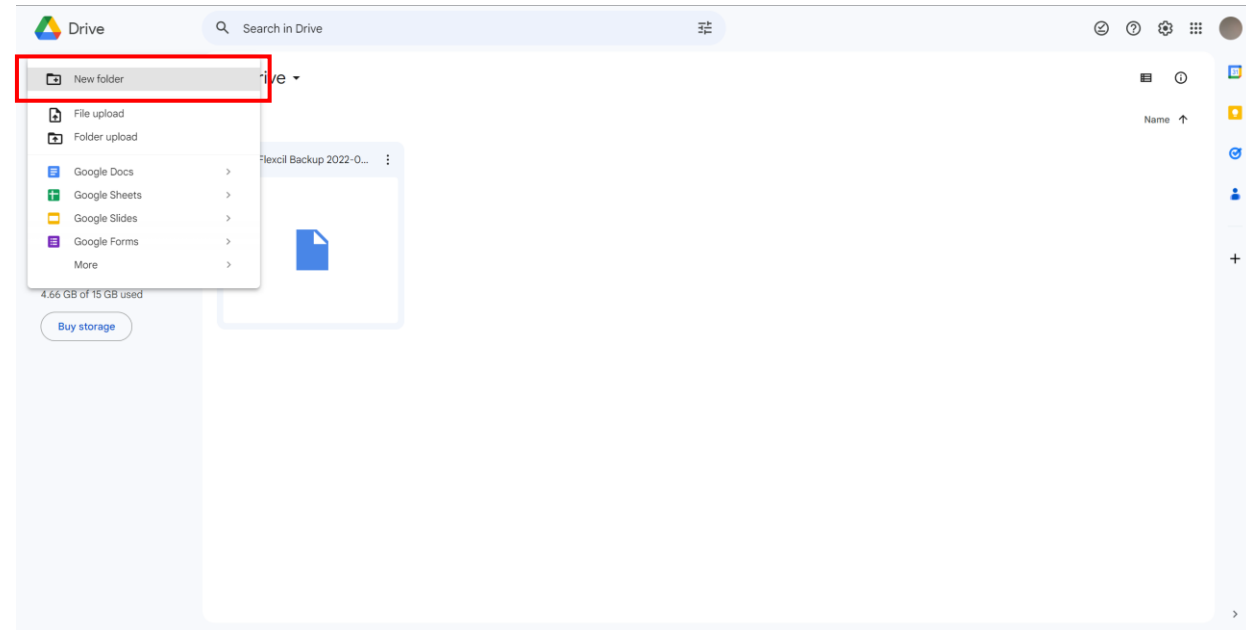
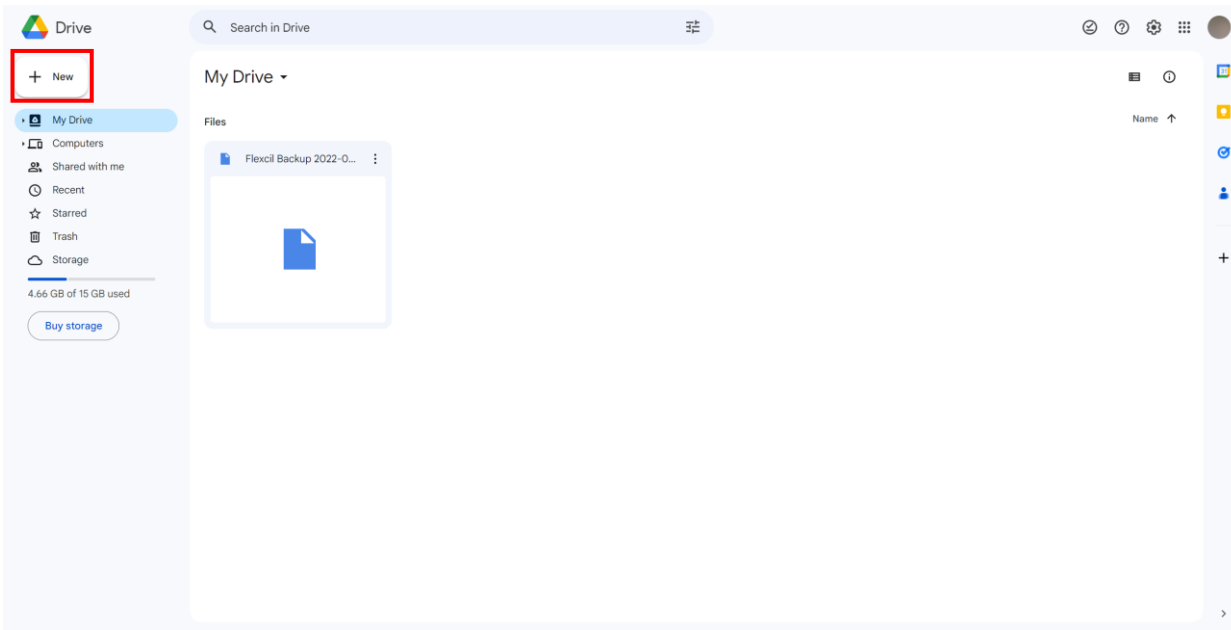
- Colab, or "Colaboratory", allows you to write and execute Python in your browser, with
  - Zero configuration required
  - Access to GPUs free of charge





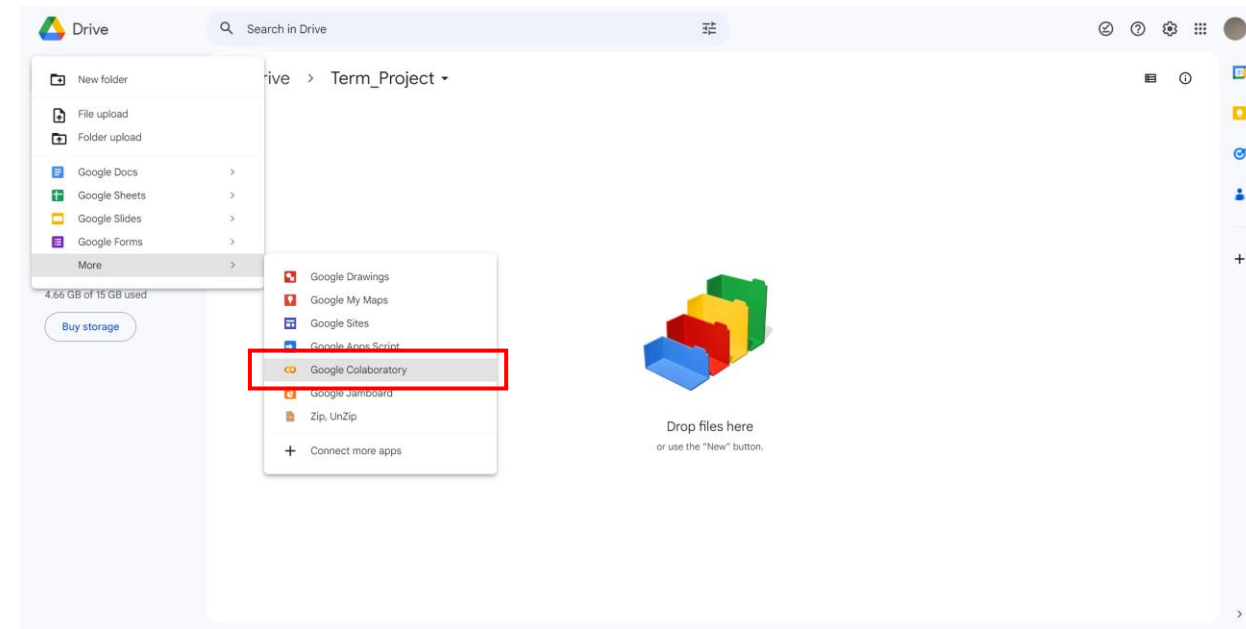
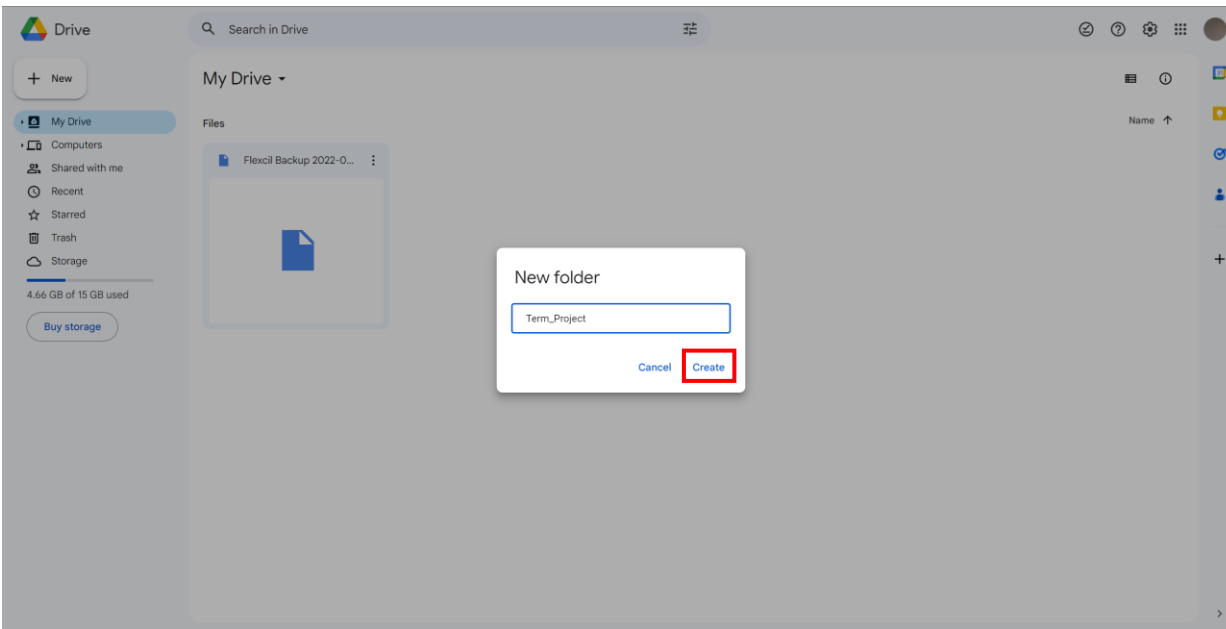
# Term Project

- Get into your Google Drive.
- Make a new folder for term project.



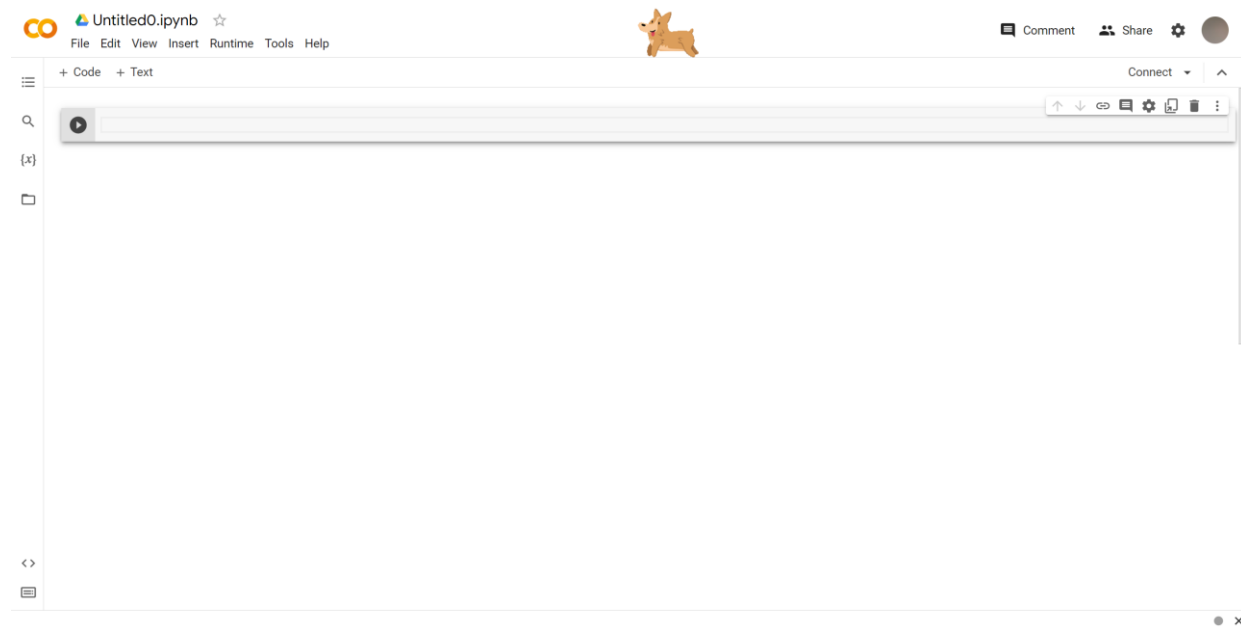
# Term Project

- Make a new folder for term project.
- Make a new google colaboratory file inside of the folder.



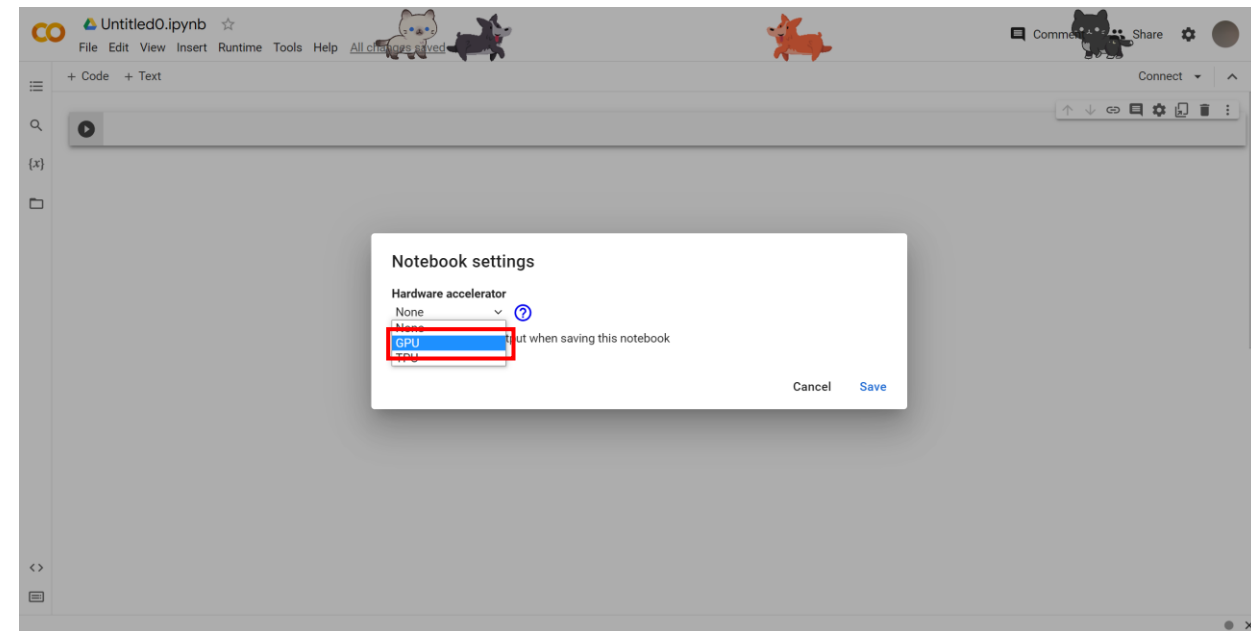
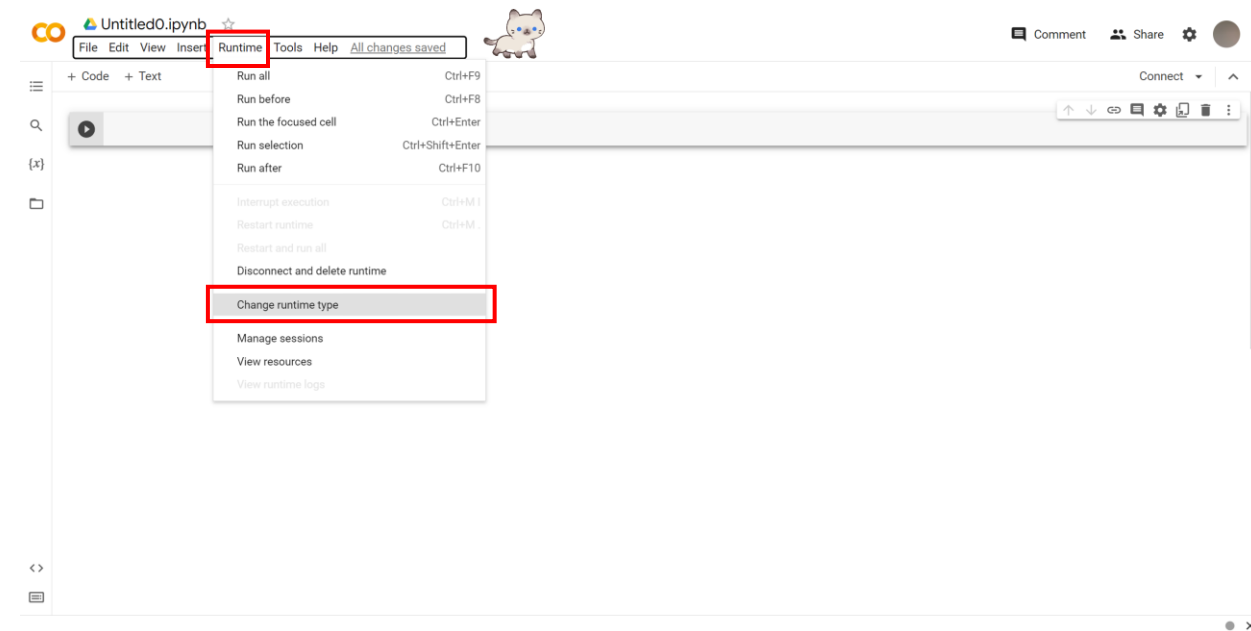
# Term Project

- Main page of colab.



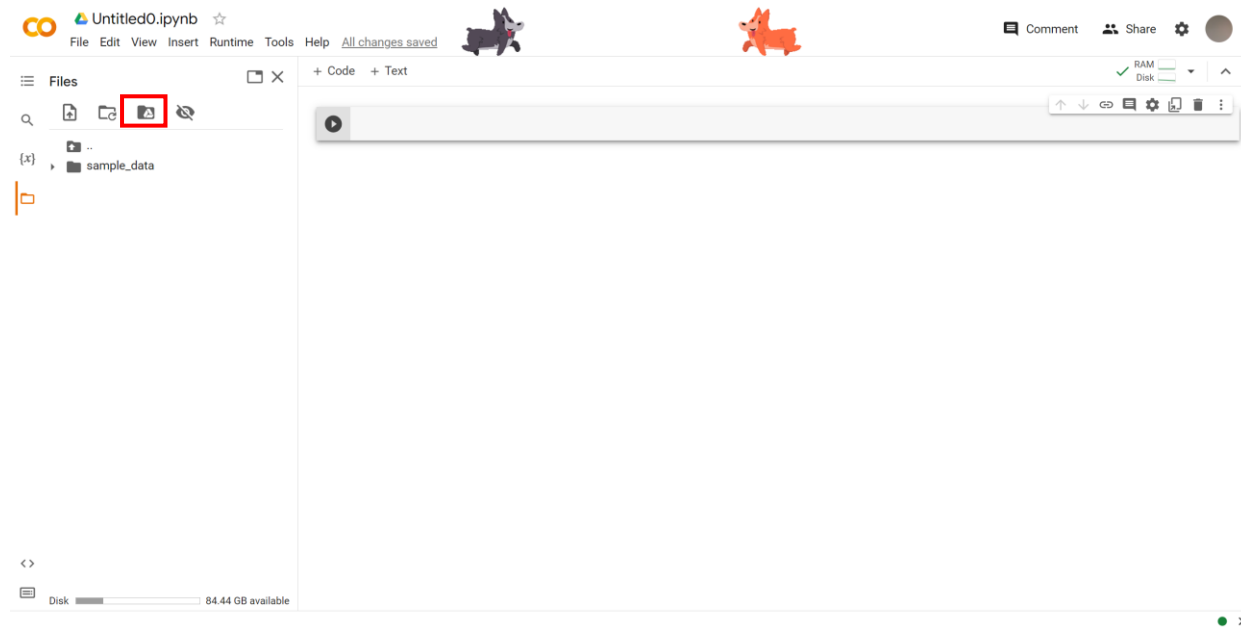
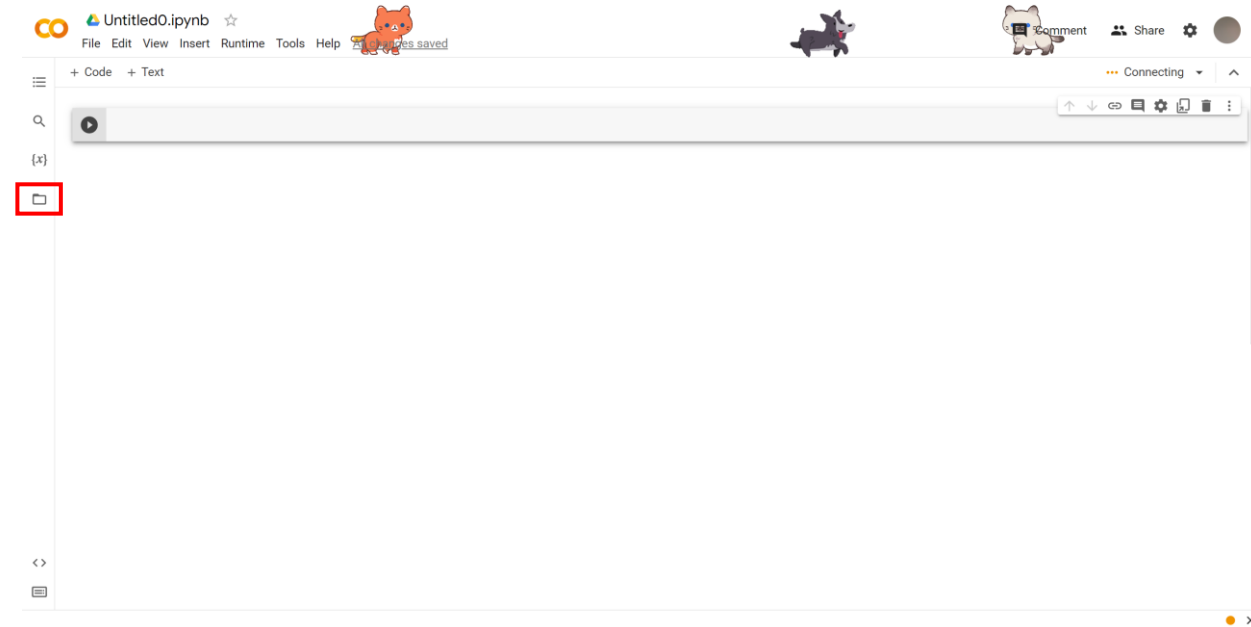
# Term Project

- Change runtime for using GPU.
  - You can use Tesla T4 GPU!!



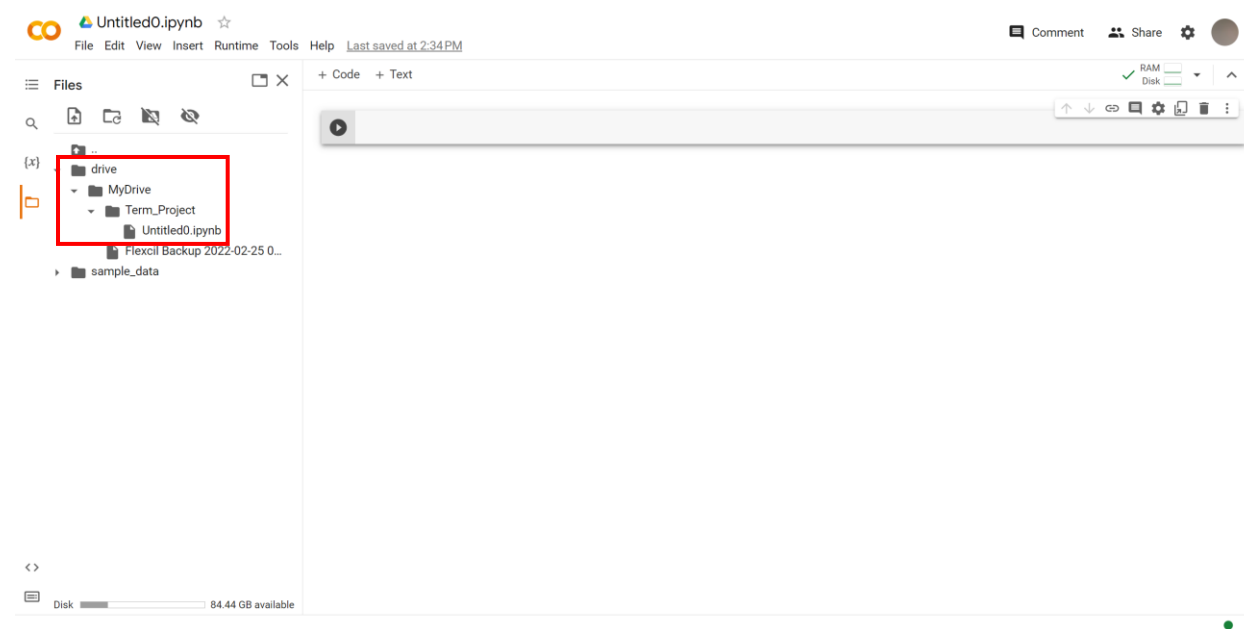
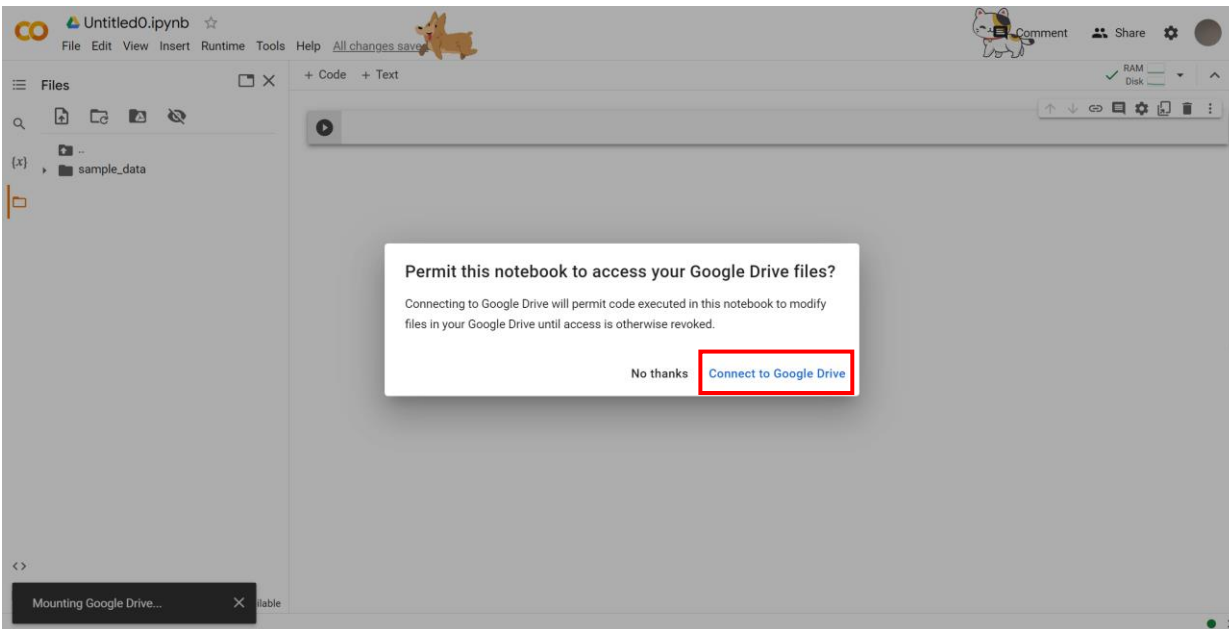
# Related Work

- Mount google drive.
  - Or you will lose all results when runtime is over.



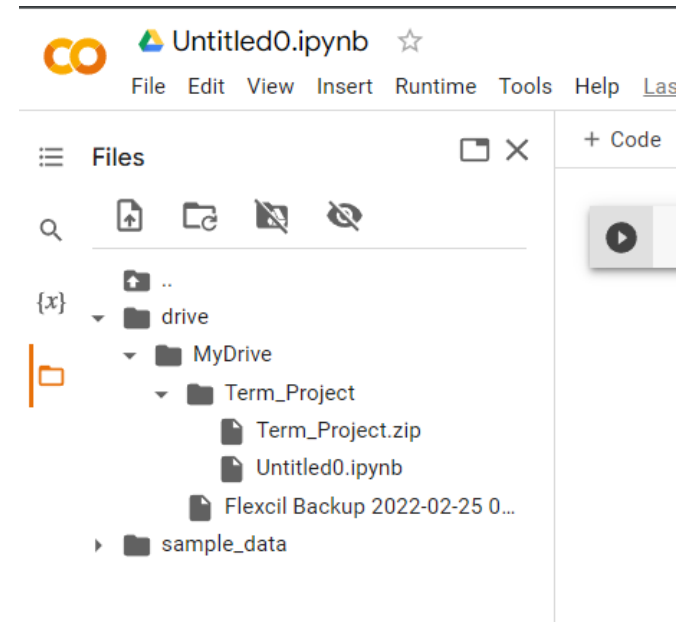
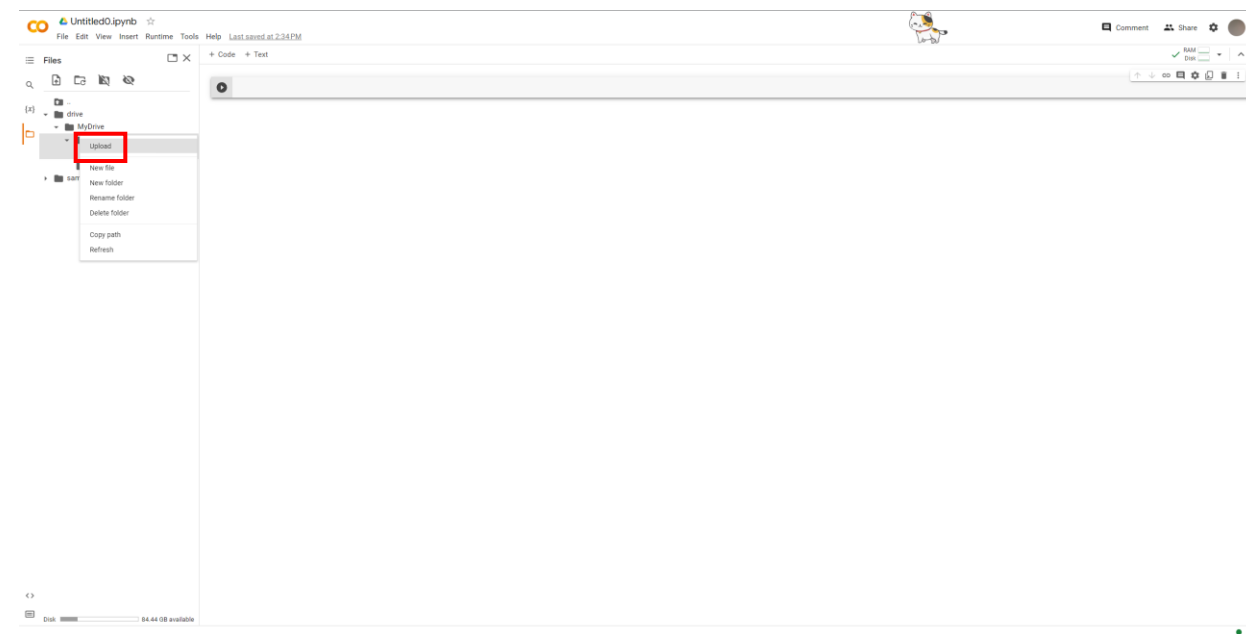
# Term Project

- Mount google drive.
  - Or you will lose all results when runtime is over.



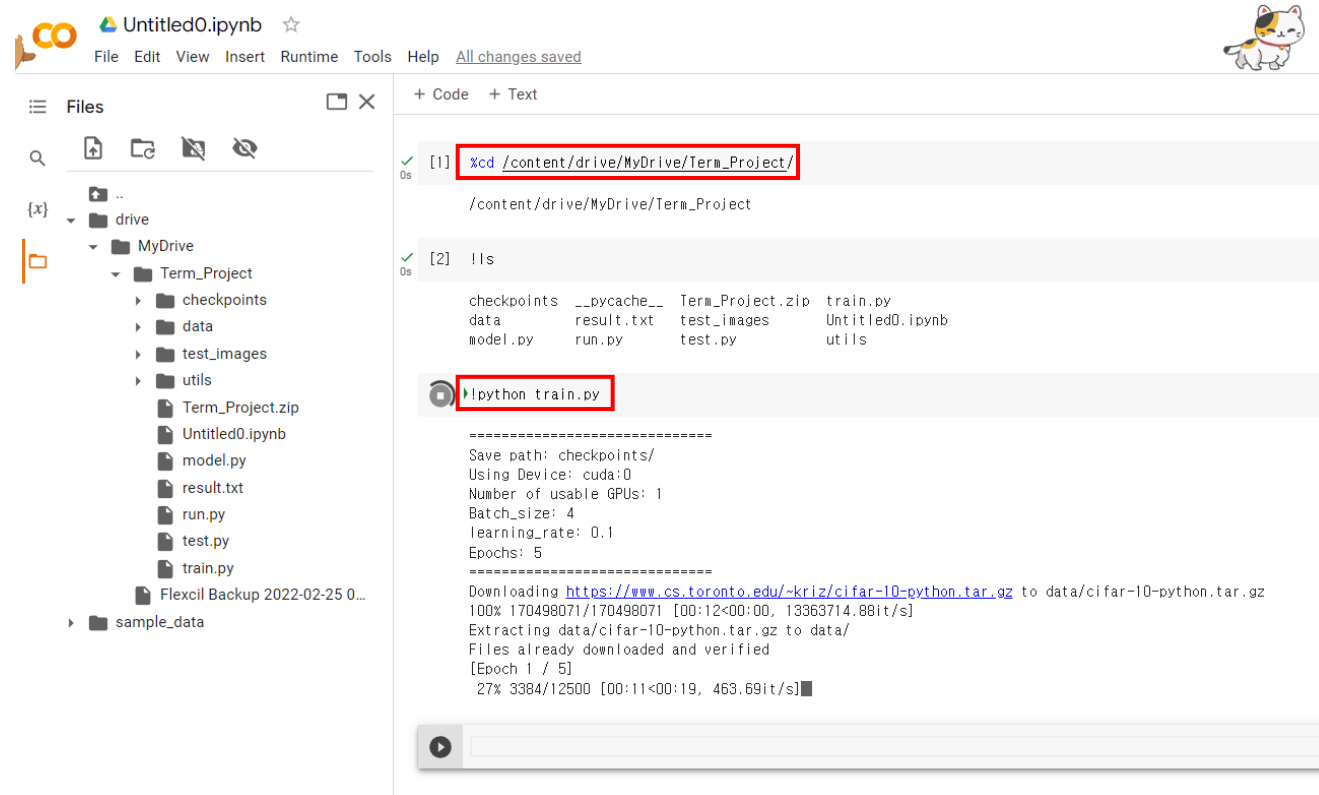
# Term Project

- Upload project folder.



# Term Project

- Unzip folder (!unzip Term\_Project.zip).
- Move to project folder.
- You can execute python file with "!python".
  - Also terminal commands!
- Ctrl + enter will execute current shell.



The screenshot shows the Google Colaboratory interface. On the left, the 'Files' pane displays a directory structure: 'drive' > 'MyDrive' > 'Term\_Project'. Inside 'Term\_Project', there are subdirectories 'checkpoints', 'data', 'test\_images', and 'utils', along with files 'Term\_Project.zip', 'Untitled0.ipynb', 'model.py', 'result.txt', 'run.py', 'test.py', and 'train.py'. Below this is a 'sample\_data' folder. The main code area on the right shows two executed cells. Cell [1] contains the command `xcd /content/drive/MyDrive/Term_Project/`, which has been executed successfully. Cell [2] contains the command `!ls`, which has been executed successfully, displaying a list of files and directories: 'checkpoints', 'data', 'model.py', 'result.txt', 'run.py', 'test\_images', 'test.py', 'Term\_Project.zip', 'train.py', 'Untitled0.ipynb', and 'utils'. Below the code cells, a terminal window shows the output of the `!python train.py` command, which has been executed successfully. The output includes a progress bar and a message: 'Downloaded https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to data/cifar-10-python.tar.gz'.

```
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved

Files
{X} drive
  MyDrive
    Term_Project
      checkpoints
      data
      test_images
      utils
      Term_Project.zip
      Untitled0.ipynb
      model.py
      result.txt
      run.py
      test.py
      train.py
      Flexcil Backup 2022-02-25 0...
    sample_data

+ Code + Text

[1] xcd /content/drive/MyDrive/Term_Project/
    /content/drive/MyDrive/Term_Project

[2] !ls
    checkpoints __pycache__ Term_Project.zip train.py
    data result.txt test_images Untitled0.ipynb
    model.py run.py test.py utils

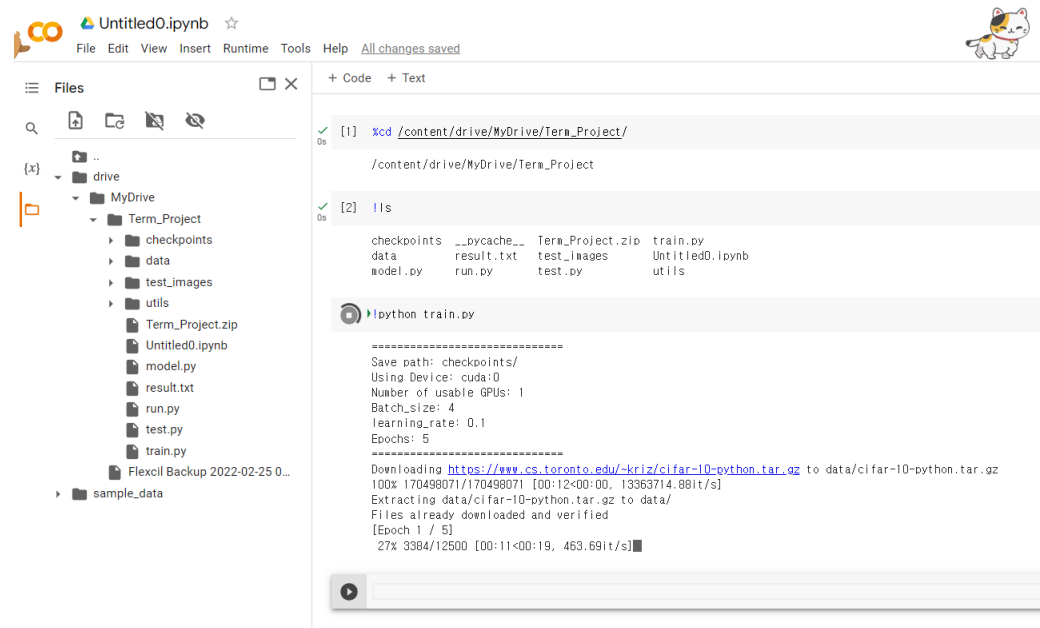
!python train.py

=====
Save path: checkpoints/
Using Device: cuda:0
Number of usable GPUs: 1
Batch_size: 4
learning_rate: 0.1
Epochs: 5
=====
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to data/cifar-10-python.tar.gz
100% 170498071/170498071 [00:12<00:00, 13363714.88it/s]
Extracting data/cifar-10-python.tar.gz to data/
Files already downloaded and verified
[Epoch 1 / 5]
27% 3384/12500 [00:11<00:19, 463.69it/s]
```



# Term Project

- You can execute python file with "!python".
  - Also terminal commands!



Untitled0.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

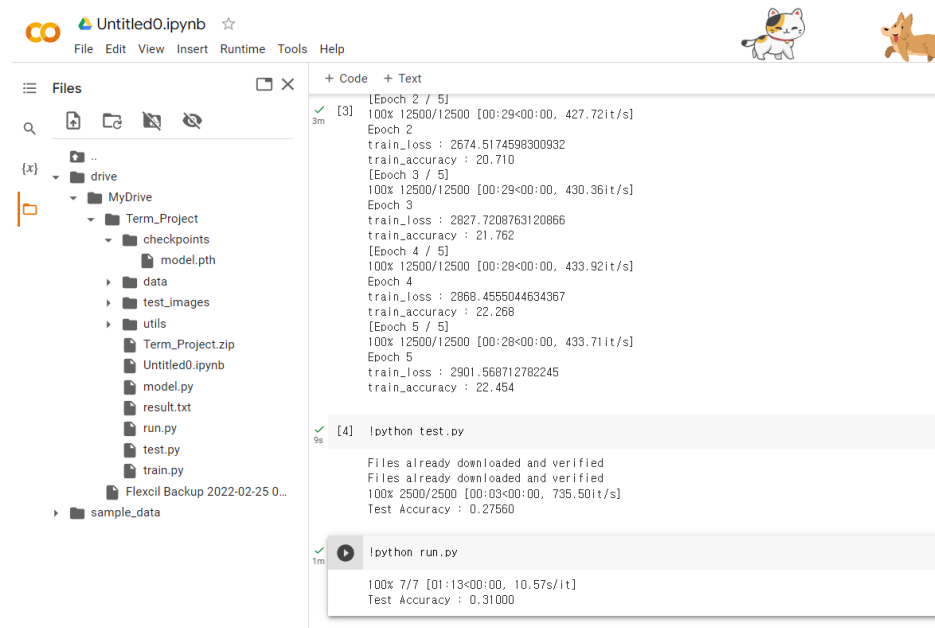
- drive
  - MyDrive
    - Term\_Project
      - checkpoints
      - data
      - test\_images
      - utils
      - Term\_Project.zip
      - Untitled0.ipynb
      - model.py
      - result.txt
      - run.py
      - test.py
      - train.py
    - Flexcil Backup 2022-02-25 0...
    - sample\_data

```
[1] %cd /content/drive/MyDrive/Term_Project/
/content/drive/MyDrive/Term_Project

[2] !ls
checkpoints  __pycache__  Term_Project.zip  train.py
data         result.txt    test_images      Untitled0.ipynb
model.py     run.py       test.py          utils

!python train.py

=====
Save path: checkpoints/
Using Device: cuda:0
Number of usable GPUs: 1
Batch_size: 4
learning_rate: 0.1
Epochs: 5
=====
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to data/cifar-10-python.tar.gz
100% 170498071/170498071 [00:12<00:00, 13363714.88it/s]
Extracting data/cifar-10-python.tar.gz to data/
Files already downloaded and verified
[Epoch 1 / 5]
27% 3384/12500 [00:11<00:19, 463.69it/s]
```



Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Files

- drive
  - MyDrive
    - Term\_Project
      - checkpoints
      - data
      - test\_images
      - utils
      - Term\_Project.zip
      - Untitled0.ipynb
      - model.py
      - result.txt
      - run.py
      - test.py
      - train.py
    - Flexcil Backup 2022-02-25 0...
    - sample\_data

```
[3] [Epoch 2 / 5]
100% 12500/12500 [00:29<00:00, 427.72it/s]
Epoch 2
train_loss : 2674.5174598300932
train_accuracy : 20.710
[Epoch 3 / 5]
100% 12500/12500 [00:29<00:00, 430.36it/s]
Epoch 3
train_loss : 2827.7208763120866
train_accuracy : 21.762
[Epoch 4 / 5]
100% 12500/12500 [00:28<00:00, 433.92it/s]
Epoch 4
train_loss : 2868.4555044634367
train_accuracy : 22.268
[Epoch 5 / 5]
100% 12500/12500 [00:28<00:00, 433.71it/s]
Epoch 5
train_loss : 2901.568712782245
train_accuracy : 22.454

[4] !python test.py

Files already downloaded and verified
Files already downloaded and verified
100% 2500/2500 [00:03<00:00, 735.50it/s]
Test Accuracy : 0.27560

!python run.py

100% 7/7 [01:13<00:00, 10.57s/it]
Test Accuracy : 0.31000
```

# Term Project

---

- Basic commands
  - `!cd =>` move to designated folder
  - `!ls =>` list files in current folder
  - `!nvidia-smi =>` current gpu status
  - `!python file.py =>` excute python file

# Pretrained model

---

Deep Learning

# Pretrained model

- NN model with pre-learned weights from training on large datasets
  - During its initial training phase, it learns from a vast amount of images, text, or data and captures general features and patterns.
- Training can be performed quickly
- Better test accuracy in small dataset
- Better generalization performance



14,197,122 images, 21841 synsets indexed

[Home](#) [Download](#) [Challenges](#) [About](#)

Not logged in. [Login](#) | [Signup](#)

**ImageNet** is an image database organized according to the **WordNet** hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. The project has been **instrumental** in advancing computer vision and deep learning research. The data is available for free to researchers for non-commercial use.

# torchvision

- Torchvision is a robust library for image tasks, efficiently supporting the entire process from data loading and preprocessing to model construction
- The *torchvision.models* model offers pre-trained model architectures, facilitating the straightforward construction of image classification models

PyTorch

Learn ▾ Ecosystem ▾ Edge ▾ Docs ▾ Blogs & News ▾ About ▾ Become a Member

0.20 ▾

Search Docs

Package Reference

Transforming and augmenting images

TV Tensors

**Models and pre-trained weights**

Datasets

Utils

Operators

Decoding / Encoding images and videos

Feature extraction for model inspection

Examples and training references

Examples and tutorials

Training references

Docs > torchvision

Shortcuts

## torchvision

Indices

This library is part of the **PyTorch** project. PyTorch is an open source machine learning framework.

Features described in this documentation are classified by release status:

**Stable:** These features will be maintained long-term and there should generally be no major performance limitations or gaps in documentation. We also expect to maintain backwards compatibility (although breaking changes can happen and notice will be given one release ahead of time).

**Beta:** Features are tagged as Beta because the API may change based on user feedback, because the performance needs to improve, or because coverage across operators is not yet complete. For Beta features, we are committing to seeing the feature through to the Stable classification. We are not, however, committing to backwards compatibility.

**Prototype:** These features are typically not available as part of binary distributions like PyPI or Conda, except sometimes behind run-time flags, and are at an early stage for feedback and testing.

[torchvision — Torchvision 0.20 documentation](#)

# torchvision

- Pretrained weights for various models are available

## Classification [🔗](#)

The following classification models are available, with or without pre-trained weights:

- AlexNet
- ConvNeXt
- DenseNet
- EfficientNet
- EfficientNetV2
- GoogLeNet
- Inception V3
- MaxViT
- MNASNet
- MobileNet V2
- MobileNet V3
- RegNet
- ResNet
- ResNeXt
- ShuffleNet V2
- SqueezeNet
- SwinTransformer
- VGG
- VisionTransformer
- Wide ResNet

## Table of all available classification weights

Accuracies are reported on ImageNet-1K using single crops:

Weight	Acc@1	Acc@5	Params	GFLOPS	Recipe
<a href="#">AlexNet_Weights.IMAGENET1K_V1</a>	56.522	79.066	61.1M	0.71	<a href="#">link</a>
<a href="#">ConvNeXt_Base_Weights.IMAGENET1K_V1</a>	84.062	96.87	88.6M	15.36	<a href="#">link</a>
<a href="#">ConvNeXt_Large_Weights.IMAGENET1K_V1</a>	84.414	96.976	197.8M	34.36	<a href="#">link</a>
<a href="#">ConvNeXt_Small_Weights.IMAGENET1K_V1</a>	83.616	96.65	50.2M	8.68	<a href="#">link</a>
<a href="#">ConvNeXt_Tiny_Weights.IMAGENET1K_V1</a>	82.52	96.146	28.6M	4.46	<a href="#">link</a>
<a href="#">DenseNet121_Weights.IMAGENET1K_V1</a>	74.434	91.972	8.0M	2.83	<a href="#">link</a>
<a href="#">DenseNet161_Weights.IMAGENET1K_V1</a>	77.138	93.56	28.7M	7.73	<a href="#">link</a>
<a href="#">DenseNet169_Weights.IMAGENET1K_V1</a>	75.6	92.806	14.1M	3.36	<a href="#">link</a>
<a href="#">DenseNet201_Weights.IMAGENET1K_V1</a>	76.896	93.37	20.0M	4.29	<a href="#">link</a>
<a href="#">EfficientNet_B0_Weights.IMAGENET1K_V1</a>	77.692	93.532	5.3M	0.39	<a href="#">link</a>
<a href="#">EfficientNet_B1_Weights.IMAGENET1K_V1</a>	78.642	94.186	7.8M	0.69	<a href="#">link</a>
<a href="#">EfficientNet_B1_Weights.IMAGENET1K_V2</a>	79.838	94.934	7.8M	0.69	<a href="#">link</a>
<a href="#">EfficientNet_B2_Weights.IMAGENET1K_V1</a>	80.608	95.31	9.1M	1.09	<a href="#">link</a>
<a href="#">EfficientNet_B3_Weights.IMAGENET1K_V1</a>	82.008	96.054	12.2M	1.83	<a href="#">link</a>
<a href="#">EfficientNet_B4_Weights.IMAGENET1K_V1</a>	83.384	96.594	19.3M	4.39	<a href="#">link</a>
<a href="#">EfficientNet_B5_Weights.IMAGENET1K_V1</a>	83.444	96.628	30.4M	10.27	<a href="#">link</a>
<a href="#">EfficientNet_B6_Weights.IMAGENET1K_V1</a>	84.008	96.916	43.0M	19.07	<a href="#">link</a>
<a href="#">EfficientNet_B7_Weights.IMAGENET1K_V1</a>	84.122	96.908	66.3M	37.75	<a href="#">link</a>
<a href="#">EfficientNet_V2_L_Weights.IMAGENET1K_V1</a>	85.808	97.788	118.5M	56.08	<a href="#">link</a>
<a href="#">EfficientNet_V2_M_Weights.IMAGENET1K_V1</a>	85.112	97.156	54.1M	24.58	<a href="#">link</a>
<a href="#">EfficientNet_V2_S_Weights.IMAGENET1K_V1</a>	84.228	96.878	21.5M	8.37	<a href="#">link</a>

[torchvision](#) — Torchvision 0.20 documentation

# torchvision

- Several usage methods

```
from torchvision.models import resnet50, ResNet50_Weights

# Old weights with accuracy 76.130%
resnet50(weights=ResNet50_Weights.IMAGENET1K_V1)

# New weights with accuracy 80.858%
resnet50(weights=ResNet50_Weights.IMAGENET1K_V2)

# Best available weights (currently alias for IMAGENET1K_V2)
# Note that these weights may change across versions
resnet50(weights=ResNet50_Weights.DEFAULT)

# Strings are also supported
resnet50(weights="IMAGENET1K_V2")

# No weights - random initialization
resnet50(weights=None)
```

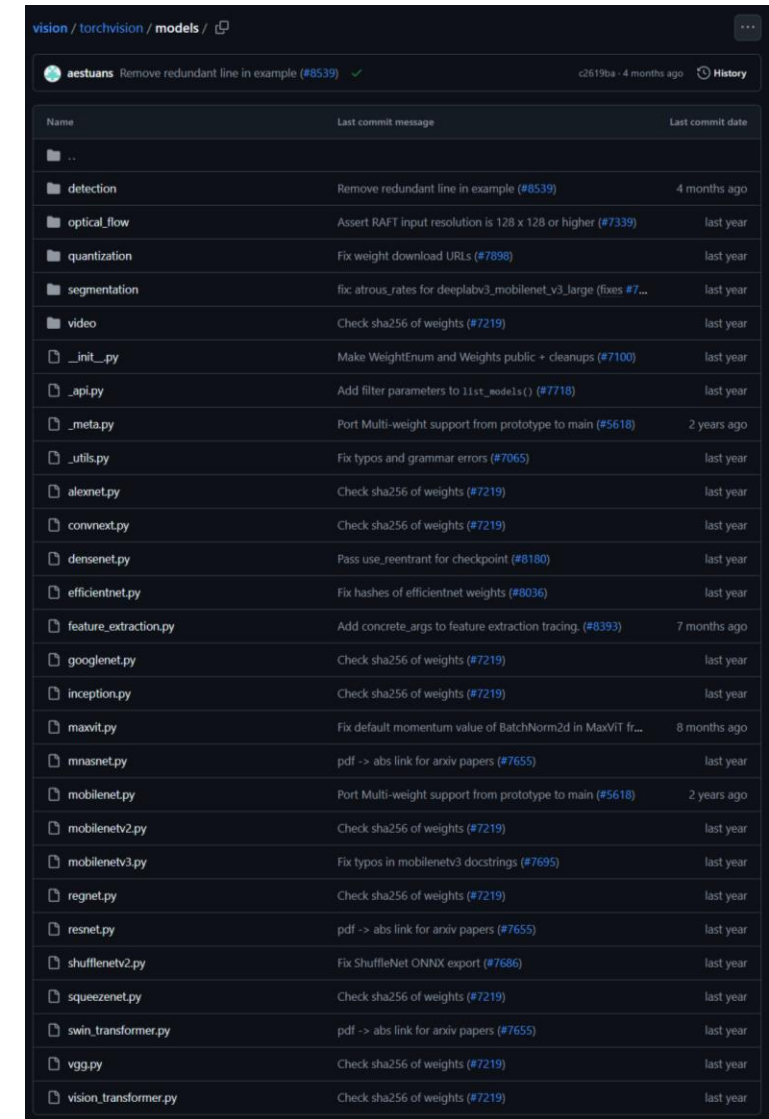
- train.py

```
85     # Print Hyperparameter
86     print("Batch_size:", args.batch_size)
87     print("learning_rate:", args.learning_rate)
88     print("Epochs:", args.epochs)
89     print("=====")
90
91     # Make Data loader and Model
92     train_loader, _ = make_data_loader(args)
93
94     # custom model
95     # model = BaseModel()
96
97     # torchvision model
98     model = resnet18(weights=ResNet18_Weights)
99
100    # you have to change num_classes to 2
101    num_features = model.fc.in_features # edited
102    model.fc = nn.Linear(num_features, num_classes) # edited
103    model.to(device)
104    print(model)
105
106    # Training The Model
107    train(args, train_loader, model)
```

[torchvision — Torchvision 0.20 documentation](#)

# torchvision

- You can download the source code of model from the official GitHub repository



The screenshot shows the GitHub repository for torchvision, specifically the 'models' directory. The repository is owned by 'aestuan' and has a commit 'c2619ba' from 4 months ago. The table lists various model files and their commit history.

Name	Last commit message	Last commit date
..		
detection	Remove redundant line in example (#8539)	4 months ago
optical_flow	Assert RAFT input resolution is 128 x 128 or higher (#7339)	last year
quantization	Fix weight download URLs (#7898)	last year
segmentation	fix atrous_rates for deeplabv3_mobilenet_v3_large (fixes #7...	last year
video	Check sha256 of weights (#7219)	last year
_init_.py	Make WeightEnum and Weights public + cleanups (#7100)	last year
_api.py	Add filter parameters to list_models() (#7718)	last year
_meta.py	Port Multi-weight support from prototype to main (#5618)	2 years ago
_utils.py	Fix typos and grammar errors (#7065)	last year
alexnet.py	Check sha256 of weights (#7219)	last year
convnext.py	Check sha256 of weights (#7219)	last year
densenet.py	Pass use_reentrant for checkpoint (#8180)	last year
efficientnet.py	Fix hashes of efficientnet weights (#8036)	last year
feature_extraction.py	Add concrete_args to feature extraction tracing. (#8393)	7 months ago
googlenet.py	Check sha256 of weights (#7219)	last year
inception.py	Check sha256 of weights (#7219)	last year
maxvit.py	Fix default momentum value of BatchNorm2d in MaxViT fr...	8 months ago
mnasnet.py	pdf -> abs link for arxiv papers (#7655)	last year
mobilenet.py	Port Multi-weight support from prototype to main (#5618)	2 years ago
mobilenetv2.py	Check sha256 of weights (#7219)	last year
mobilenetv3.py	Fix typos in mobilenetv3 docstrings (#7695)	last year
regnet.py	Check sha256 of weights (#7219)	last year
resnet.py	pdf -> abs link for arxiv papers (#7655)	last year
shufflenetv2.py	Fix ShuffleNet ONNX export (#7686)	last year
squeezenet.py	Check sha256 of weights (#7219)	last year
swin_transformer.py	pdf -> abs link for arxiv papers (#7655)	last year
vgg.py	Check sha256 of weights (#7219)	last year
vision_transformer.py	Check sha256 of weights (#7219)	last year



# torchvision

- You can also modify the code of the installed library

```

85     # Print Hyperparameter
86     print("Batch_size:", args.batch_size)
87     print("learning_rate:", args.learning_rate)
88     print("Epochs:", args.epochs)
89     print("=====")
90
91     # Make Data loader and Model
92     train_loader, _ = make_data_loader(args)
93
94     # custom model
95     # model = BaseModel()
96
97     # torchvision model
98     model = resnet18(weights=ResNet18_Weights)
99
100    # you have to change num_classes to 2
101    num_features = model.fc.in_features # edited
102    model.fc = nn.Linear(num_features, num_classes) # edited
103    model.to(device)
104    print(model)
105
106    # Training The Model
107    train(args, train_loader, model)

```

```

648 @register_model()
649 @handle_legacy_interface(weights=("pretrained", ResNet18_Weights.IMAGENET1K_V1))
650 def resnet18(*, weights: Optional[ResNet18_Weights] = None, progress: bool = True, **kwargs: Any) -> ResNet:
651     """ResNet-18 from `Deep Residual Learning for Image Recognition <https://arxiv.org/pdf/1512.03385.pdf>`_.
652
653     Args:
654         weights (:class:`~torchvision.models.ResNet18_Weights`, optional): The
655             pretrained weights to use. See
656             :class:`~torchvision.models.ResNet18_Weights` below for
657             more details, and possible values. By default, no pre-trained
658             weights are used.
659         progress (bool, optional): If True, displays a progress bar of the
660             download to stderr. Default is True.
661         **kwargs: parameters passed to the ``torchvision.models.resnet.ResNet``
662             base class. Please refer to the `source code
663             <https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py>`_
664             for more details about this class.
665
666     .. autoclass:: torchvision.models.ResNet18_Weights
667        :members:
668        """
669     weights = ResNet18_Weights.verify(weights)
670
671     return _resnet(BasicBlock, [2, 2, 2, 2], weights, progress, **kwargs)

```

# Term Project

---

- Feel free to email TA if the bug isn't fixed or if you have any questions!
- Email: [seokjinoh@hanyang.ac.kr](mailto:seokjinoh@hanyang.ac.kr)  
[wjdeodbs386@hanyang.ac.kr](mailto:wjdeodbs386@hanyang.ac.kr)
- Please try searching on Google before asking any questions regarding the bug.
- Please specify which class you are taking and who you are when sending an email.