# Homework 2: Convolutional Neural Networks and Beyond

**Deep Learning (84100343-0)**
Spring 2022
Tsinghua University

## 1 Introduction

As we have learned in class, CNNs are made up of layers with learnable parameters including weights and bias. Each layer takes the output of the previous layer to perform some operations and produces an output. In recent years, main-stream CNNs, such as AlexNet [4], VGG [7], GoogleNet [8], ResNet [1], DenseNet [3], EfficientNet [9] and so on, have achieved increasingly better performance on ImageNet dataset and have been leveraged to diverse applications across many fields. In this homework, a remote sensing dataset named EuroSAT [2] is provided. As shown in Figure 1, EuroSAT is a dataset based on Sentinel-2 satellite images covering 13 spectral bands and consisting out of 10 classes within a total $27,000$ labeled and geo-referenced images. You are required to solve the problem of land cover image classification by modern convolutional neural networks (CNNs).

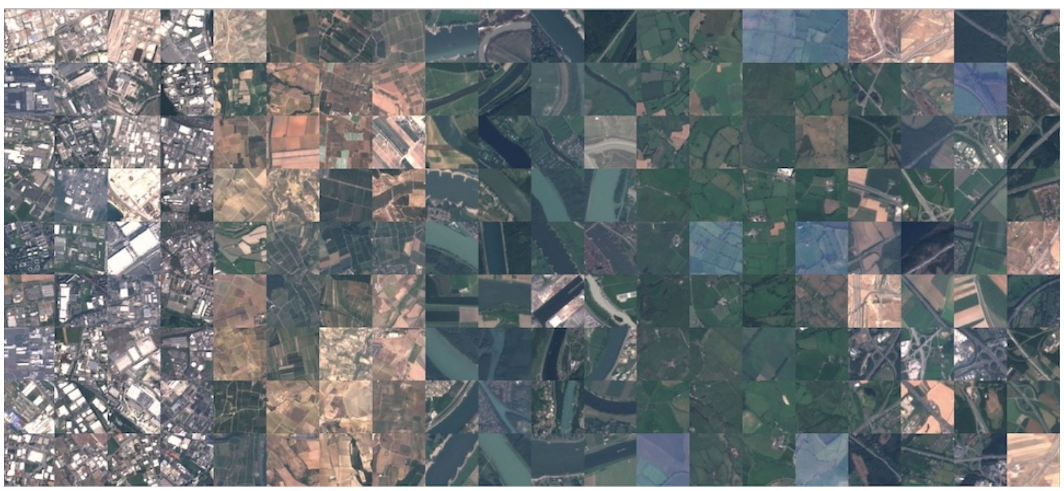

Figure 1: Example images of EuroSAT

## 2 Dataset

**Dataset Description.** The dataset contains 10 categories of remote sensing images with different land covers including "*AnnualCrop*", "*Highway*", "*PermanentCrop*", "*SeaLake*" , "*Forest*", "*Industrial*", "*Residential*", "*HerbaceousVegetation*", "*Pasture*" and "*River*". We first randomly split the whole dataset into two parts, in which the first $80\%$ is used for training while the remaining $20\%$ is for testing. We reorganize the training set into 5 tasks: **(1)** Large-Scale Learning: This task uses the original large-scale training set in which each class has more than 1000 samples. **(2)** Medium-Scale Learning: We sample the training set to generate a new medium-scale dataset in which each class has 100 samples. **(3)** Semi-Supervised Learning: This task has two subsets in which the labeled subset has 25 samples per class and the unlabeled one contains the remaining unlabeled samples.

**(4)** Long-Tailed Learning: In this task, we make the label distribution a long-tailed one. The head class with the most samples has 2430 samples while the tailed class with the least samples has only 10 samples. **(5)** Weakly-Supervised Learning: In this task, a weakly-supervised dataset is given, in which about 20% samples of per class have wrong labels.

**Directory structure.** In this part, we specify the directory structure of each task.

- The common test set:
  `"./test/{AnnualCrop,...,River}/xxx.jpg"`
- Large-Scale Learning:
  `"./1-Large-Scale/train/{AnnualCrop,...,River}/xxx.jpg"`
- Medium-Scale Learning:
  `"./2-Medium-Scale/train/{AnnualCrop,...,River}/xxx.jpg"`
- Semi-Supervised Learning:
  `"./3-Semi-Supervised/labeled/{AnnualCrop,...,River}/xxx.jpg"`
  `"./3-Semi-Supervised/unlabeled/xxx.jpg"`
- Long-Tailed Learning:
  `"./4-Long-Tailed/train/{AnnualCrop,...,River}/xxx.jpg"`
- Weakly-Supervised Learning:
  `"./5-Weakly-Supervised/train/{AnnualCrop,...,River}/xxx.jpg"`

**Download Link.** `https://cloud.tsinghua.edu.cn/f/982abd6ef7d84d49b630/`.

# 3 Requirements

## 3.1 Programming Language and Framework

Python only. We recommend PyTorch and TensorFlow. If using other frameworks, please contact TA.

## 3.2 Tutorials

- **pytorch**
  - `https://pytorch.org/tutorials/`
  - `https://github.com/utkuozbulak/pytorch-cnn-visualizations`
- **tensorflow**
  - `https://www.tensorflow.org/tutorials`
  - `https://github.com/tensorflow/lucid`

# 4 Tasks and Scoring

You need to finish the following three tasks on the given dataset:

- **Task A**: Large-Scale Learning. You are required to use the architecture of ResNet-18 to train a new model from scratch on *1-Large-Scale* and then evaluate on the given *test* set.
- **Task B**: Medium-Scale Learning. Design and implement your own convolutional neural networks (CNNs) to train a new model from scratch on *2-Medium-Scale* and then evaluate on the given *test* set. Your proposed model needs to be different from the main-stream ones, such as AlexNet [4], VGG [7], GoogleNet [8], ResNet [1], DenseNet [3], EfficientNet [9] and so on. Some recipes can be borrowed from ResNeXt [10] or ConvNeXt [5].
- **Task C**: Select a topic that you are interested in from Semi-Supervised Learning, Long-Tailed Learning, and Weakly-Supervised Learning. You can either adopt the mainstream CNNs or design a new one to train a new model from scratch on the corresponding dataset. For example, if you are interested in semi-supervised learning, please train on *3-Semi-Supervised* and then evaluate on the given *test* set. You should only finish your selected task, instead of all tasks. Note that, please DO NOT use additional data resources.

- **Train and evaluate your models**:
  - Train a model **A** for Task A, evaluate its accuracy on *test* set, and report training and test curves [**10pts**]. Note that, the *start code* of this task is provided and it is runnable, you can directly run it and report the results. ("`python main.py`")
  - Train a model **B** for Task B, evaluate its accuracy on *test* set, and report training and test curves. [**10pts**].
  - Train a model **C** for Task C, evaluate its accuracy on *test* set, and report training and test curves [**20pts**]. Besides, plot the confusion matrix of your model C on *test* set [**10pts**].
- **Dig into your model A**:
  - Substitute the SGD optimizer to *RMSprop*, *Adam*. Give a detailed analysis on the model performance and the training curve under different optimization strategies [**4pts**].
  - Visualize the features before the last *fully-connected* layer using t-SNE [6] [**8pts**].
  - Leverage a proper *neural network visualization toolkit* to visualize some *conv* features [**8pts**].
    - **tensorflow**: `https://github.com/tensorflow/lucid`
    - **pytorch**: `https://github.com/utkuozbulak/pytorch-cnn-visualizations`
    - **others**: Any other toolkits if you think they are helpful...
- **Babysit your model B and model C**:
  - Use the techniques of **data augmentation** and **learning rate strategy** to improve the performance of your model **B** and give a detailed ablation study in your report [**10pts**].
  - You should either use extra techniques you find in other materials or propose a novel technique to improve your model **C**. Please explain why it works in your report [**20pts**].

### 4.1 Notifications

- Please submit your *code and report* as an Archive (zip or tar). The document is supposed to cover your **insights** of the proposed model, the **technical details**, the **experimental results** (including training and validation curves), and the necessary **references**.
- We will focus on your code and document to decide your score. Still, under equal conditions including novelty, code quality, and document quality, a higher accuracy along with reasonable computation efficiency contributes a higher score.

## References


[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[2] P. Helber, B. Bischke, A. Dengel, and D. Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE J-STARS*, 2019.

[3] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[5] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. In *CVPR*, 2022.

[6] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[9] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019.

[10] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CVPR*, 2017.