Machine Learning HW3

ML TAs
ntu-ml-2022-spring-ta@googlegroups.com
Objective - Image Classification

1. Solve image classification with **convolutional neural networks**.
2. Improve the performance with **data augmentations**.
3. Understand popular image model techniques such as **residual**.
Task Introduction - Food Classification

- The images are collected from the food-11 dataset classified into 11 classes.
- Training set: 9866 labeled images
- Validation set: 3430 labeled images
- Testing set: 3347 images
Rules

- DO NOT attempt to find the original labels of the testing set.
- DO NOT use any external datasets.
- **DO NOT use any pretrained models.**
  - Also, do not attempt to “test how effective pretraining is” by submitting to kaggle. Pretraining is very effective and you may test it after the competition ends.
- You may use any publicly available packages/code
  - But make sure you do not use pretrained models. Most code use those.
  - You may not upload your code/checkpoints to be publicly available during the timespan of this homework.
Baseline

Simple : 0.50099

Medium : 0.73207 Training Augmentation + Train Longer

Strong : 0.81872 Training Augmentation + Model Design + Train Longer (+ Cross Validation + Ensemble)

Boss : 0.88446 Training Augmentation + Model Design + Test Time Augmentation + Train Longer (+ Cross Validation + Ensemble)
Baseline GPU Time Estimation

- All results are benchmarked on kaggle

Simple: 15~20mins
Medium: Augmentation A - 6hrs
      Augmentation B - 80 min
Strong: Model X + Augmentation B + Resplit - 12 hrs
      Model Y + Augmentation B + Resplit - 10 hrs
      Model Z + Augmentation B + Resplit - 6 hrs
Boss: Model Z + Augmentation B + TTA + Resplit - 12 hrs
Or Ensemble

- Model Y + Augmentation B + TTA + Resplit - 12 hrs
- Model X + Augmentation B + TTA + Resplit - 12 hrs
Submission Format

The file should contain a header and have the following format:

```
Id,Category
0001,1
```

Both type should be strings. Id corresponds to the jpg filenames in test. Follow the sample code if you have trouble with formatting.
Model Selection

- Visit torchvision.models for a list of model structures, or go to timm for the latest model structures.
- Pretrained weights are not allowed, specifically set pretrained=False to ensure that the guideline is met.

Classification

The models subpackage contains definitions for the following model architectures for image classification:

- AlexNet
- VGG
- ResNet
- SqueezeNet
Data Augmentation

- Modify the image data so non-identical inputs are given to the model each epoch, to prevent overfitting of the model.
- Visit `torchvision.transforms` for a list of choices and their corresponding effect. Diversity is encouraged! Usually, stacking multiple transformations leads to better results.
- Coding: fill in `train_tfm` to gain this effect.
Advanced Data Augmentation - **mixup**

\[ 0.5^* + 0.5^* = \]

Label

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>01</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Advanced Data Augmentation - mixup

- Coding:
  - In your torch.utils.Dataset, __getitem__() needs to return an image that is the linear combination of two images.
  - In your torch.utils.Dataset, __getitem__() needs to return a label that is a vector, to assign probabilities to each class.
  - You need to explicitly code out the math formula of the cross entropy loss, as CrossEntropyLoss does not support multiple labels.
Test Time Augmentation

- The sample code tests images using a deterministic “test transformation”
- You may use the train transformation for a more diversified representation of the images, and predict with multiple variants of the test images.
- Coding: You need to fill in `train_tfm`, change the augmentation method for `test_dataset`, and modify prediction code to gain this effect.

```
train_tfm
```
```
test_tfm + test_tfm
```

Pred        Pred         Pred         Pred       Pred                     Pred                                Pred

```
Pred
```

Ensemble   Pred  Pred  Pred  Pred  Pred  Pred  Pred  Pred
Test Time Augmentation

- Usually, test_tfm will produce images that are more identifiable, so you can assign a larger weight to test_tfm results for better performance.

\[
\text{Final Prediction} = \text{avg_train_tfm_pred} \times 0.5 + \text{test_tfm_pred} \times 0.5
\]

- Ex: Final Prediction = \( \text{avg_train_tfm_pred} \times 0.5 + \text{test_tfm_pred} \times 0.5 \)
Cross Validation

- Cross-validation is a resampling method that uses different portions of the data to validate and train a model on different iterations. Ensembling multiple results lead to better performance.
- Coding: You need to merge the current train and validation paths, and resample form those to form new train and validation sets.

---

Ensemble
Cross Validation

- Even if you don’t do cross validation, you are encouraged to resplit the train/validation set to suitable proportions.
  - Currently, train : validation ~ 3 : 1, more training data could be valuable.
Ensemble

- Average of logits or probability: Need to save verbose output, less ambiguous
- Voting: Easier to implement, need to break ties
- Coding: basic math operations with numpy or torch
Kaggle Introduction

- Kaggle GPU: 16G NVIDIA TESLA P100
  - [https://www.kaggle.com/docs/efficient-gpu-usage](https://www.kaggle.com/docs/efficient-gpu-usage)
- Faster data IO
- Easier checkpoint reusing
- Limited to 30+ hrs/week depending on usage.
- Limited to 12hrs/run
- We strongly recommend that you run with Kaggle for this homework
How to run

ML2022Spring-HW3
HW3 of Machine Learning, National Taiwan University (Course No. EE5184), taught by prof. Hung Yi Lee

Change sorting to “Recently Run” if you can’t find the code.
**How to get data:** In the input section, there should already be data titled “ml2022spring-hw3”. If there isn’t, click on Add data and find “ml2022spring-hw3”.
How to use gpu: Change accelerator to “gpu” when you run your code.

Since GPU time is limited, it is advised to NOT utilize GPU while debugging.
How to Run interactively: The commands are very similar to google colab.

Any output writing to ./ will end up here, you can download it.
How to Run in background: Execute code from start to end, all results would be save permanently. (Output is limited to 20G, max run time = 12hrs)

Make sure your code is bug free, as any error in any code block would result in early stopping

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only `../input/` directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory.

import os
for dirname, _, filenames in os.walk('/kaggle/input/'):    
    for filename in filenames:
        pass
        #print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version.
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session.
```
How to view “Run in background” results

You can view your results this way

Don’t worry if your run is “cancelled”, the output will still be saved.
How to utilize your results

Upload your model to become kaggle dataset
Create a new notebook with the output as input
How to train and predict

1. Run your code **in background**
2. Find the output data “./submission.csv” and upload it to the submission page
How to retrain from a checkpoint

1. Run your code **in background**
2. Find the output model and save it as a dataset
3. Import your dataset into the notebook via “Add data”
4. **Modify your code to load your checkpoint**
5. Run your code in background
6. Find the output data “./submission.csv” and upload it to the submission page
Tips and tricks
Time management

- Kaggle will allocate more time for those who have utilized GPU resources in the past week. Time resets every Saturday at 08:00, Taipei Time.

=> Run any code with GPU on kaggle today (3/4) to get (possible) extra time next week.

- Time consumption is the sum of notebooks running interactively with gpu and running in background with gpu.
- Please start early
Time management

- You can go over the time limit moderately. Kaggle will not interrupt your code in the background if it is already running. If your time limit is reached, you cannot run any code with GPU.

=> 時間快用完的時候在背景跑一隻程式, 等於多12小時runtime

=> 時間快用完的時候在背景跑兩隻程式, 等於多24小時runtime
Time management - Parallelization

- You can run two codes in the background
- If you are satisfied with your code, utilize this to run multiple random seeds/multiple train validation split/multiple model structures, so you can ensemble
A sample procedure for beating the boss baseline

The boss baseline could be beaten with a single model trained on kaggle for 12hrs

Your procedure can be ensemble of models with parallelization

1. Train : 12h
2. Save checkpoint
3. Ensemble Prediction

- multiple random seeds
- multiple train validation split
- multiple model structures
Experimental Tips

- **Augmentation** is a must to prevent overfitting. A good augmentation can carry on to the testing phase with **Test Time Augmentation**.
- If you build your own network structure and have implemented augmentation, don’t be afraid to scale up your model. (Many predefined models structure are huge and perform great)
- In TA’s experiment, model structures with **downsampling** work better, simply choosing the best performing models on ImageNet according to websites is not always a good idea because pretrained weights are not allowed.
Other tricks......

- **on Classification**
  - Label Smoothing Cross Entropy Loss
  - FocalLoss

- **on Optimization**
  - Dropout
  - Gradient Accumulation
  - BatchNorm
  - Image Normalization
Running with Google Colab

- We strongly recommend that you run with Kaggle for this homework
- If you would like to use colab, DO NOT store data in your drive and load from there, the input/output is very slow. (store at ./ instead)
- If you mount your google drive in colab: G-suite google drive now has a storage limit. Since models and data can be large, keep an eye on your used space to prevent your account being suspended.
Report Questions
Q1. Augmentation Implementation (2%)

Implement augmentation by finishing `train_tfm` in the code with image size of your choice. Copy your `train_tfm` code and paste it onto the GradeScope.

- Your `train_tfm` must be capable of producing 5+ different results when given an identical image multiple times.
- Your `train_tfm` in the report can be different from `train_tfm` in your training code.
Residual Connection is widely used in CNNs such as Deep Residual Learning for Image Recognition. Residual is demonstrated in the following graph.

Image Source: https://towardsdatascience.com/what-is-residual-connection-efb07cab0d55
Q2. Residual Connection Implementation (2%)

Implement Residual Connections in the Residual_Model, following the graph below. Copy your Residual_Model code and paste it on Gradescope.

- Your Residual_Model should connect like

- You should only modify the forward part of the model
Submission Format

- **train_tfm** and **Residual_Model** are present in colab (scroll to bottom) and kaggle (ML2022HW3 - Report Questions), you only need to modify from our sample code.

```python
train_tfm = transforms.Compose(
    # Resize the image into a fixed shape (height = width = 128)
    transforms.Resize((128, 128)),
    # You need to add some transforms here.
    transforms.ToTensor(),
)
```

```python
from torch import nn

class Residual_Network(nn.Module):
    def __init__(self):
        super(Residual_Network, self).__init__()

        self.cnn_layer1 = nn.Sequential(
            nn.Conv2d(3, 64, 3, 1, 1),
            nn.BatchNorm2d(64),
        )

        self.cnn_layer2 = nn.Sequential(
            nn.Conv2d(64, 64, 3, 1, 1),
            nn.BatchNorm2d(64),
        )

        self.cnn_layer3 = nn.Sequential(
            nn.Conv2d(64, 128, 3, 2, 1),
            nn.BatchNorm2d(128),
        )
```
Regulations and Grading Policy
Grading

- simple (public) +0.5 pts
- simple (private) +0.5 pts
- medium (public) +0.5 pts
- medium (private) +0.5 pts
- strong (public) +0.5 pts
- strong (private) +0.5 pts
- boss (public) +0.5 pts
- boss (private) +0.5 pts
- code submission +2 pts
- report +4 pts

Total: 10 pts
Code Submission

● NTU COOL
  ○ Compress your code and pack them into .zip file

<student_ID>_hw3.zip

● Your .zip file should include only
  ○ Code: either .py or .ipynb

● Do not submit models and data

● File Size Limit : 25MB

● Submit the code that corresponds to your chosen submission in Kaggle (One of the best)
Report Submission

Answer the questions on GradeScope
Deadlines

- Kaggle, Code (NTU COOL), Report (GradeScope)

2022/03/25 23:59 (UTC+8)
Rules

- DO NOT attempt to find the original labels of the testing set.
- DO NOT use any external datasets.
- **DO NOT use any pretrained models.**
  - Also, do not attempt to “test how effective pretraining is” by submitting to kaggle. Pretraining is very effective and you may test it after the competition ends.
- You may use any publicly available packages/code
  - But make sure you do not use pretrained models. Most code use those.
  - You may not upload your code/checkpoints to be publicly available during the timespan of this homework.
Rules

- You should finish your homework on your own.
- You should not modify your prediction files manually.
- Do not share codes or prediction files with any living creatures.
- Do not use any approaches to submit your results more than 5 times a day.
- Your final grade x 0.9 and 0 pt for this HW if you violate any of the above rules, final grade = Fail for repeat offenders.
- Prof. Lee & TAs preserve the rights to change the rules & grades.
Kaggle: https://www.kaggle.com/c/ml2022spring-hw3b

Kaggle code (join competition first): https://www.kaggle.com/c/ml2022spring-hw3b/code?competitionId=34954&sortBy=dateRun

Colab: https://colab.research.google.com/drive/15hMu9YiYjE_6HY99UXon2vKGk2KwuWu
Contact us if you have problems...

- **Kaggle Discussion** *(Recommended for this HW)*
- NTU COOL
- Email
  - mlta-2022-spring@googlegroups.com
  - The title should begin with “[hw3]”