ML2023 Spring HW3 - CNN

ML TAs
mlta-2023-spring@googlegroups.com

Outline

- Task Introduction
- Dataset
- Evaluation Metrics
- Kaggle Submission
- Gradescope
- Code Submission
- Grading Policy
- Regulations
- Baseline Hints
- Useful links

Task Introduction - Image Classification

- Solve image classification with **convolutional neural networks(CNN)**.

Dataset

- The images are collected from the food-11 dataset splitted into 11 classes.
- Training set: 10000 labeled images
- Validation set: 3643 labeled images
- Testing set: 3000 images without labeled

Evaluation Metrics

- Accuracy on testing set

$$\text{Acc} = \frac{\text{pred} == \text{label}}{len(\text{data})} \cdot 100\%$$

Kaggle Submission (1 / 3) - Links

- Display name: **<student ID>_<anything>**
  - e.g. b10901666_MLmaster
  - For auditing, don't put student ID in your display name.
- Kaggle links
- **HW Deadline: 2023/3/31 23:59 (UTC+8)**

Kaggle Submission (2 / 3) - Rules

- You may submit up to **5** results each day (UTC+8, AM 8:00)
- Up to **2** submissions will be considered for the private leaderboard

prediction_large.csv 2 years ago by ntuee_jizz model_large3_684_compressed.pth, size = 201KB, params: 93139 (rabbit ensemble)	0.65059	0.66341	☑
prediction_large.csv 2 years ago by ntuee_jizz model_large3_676_compressed.pth, size = 201KB, params: 93139 (rabbit ensemble)	0.65282	0.65422	☐
prediction_large.csv 2 years ago by ntuee_jizz model_large2_669_compressed.pth, size = 222KB, params: 103623	0.65394	0.65254	☑

remember to select **2** results for your final scores before the competition ends!

Kaggle Submission (3 / 3) - Format

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

```
Id,Category
0000,1
```

- Id corresponds to the jpg file name in test.
- Follow the sample code if you have trouble with formatting.

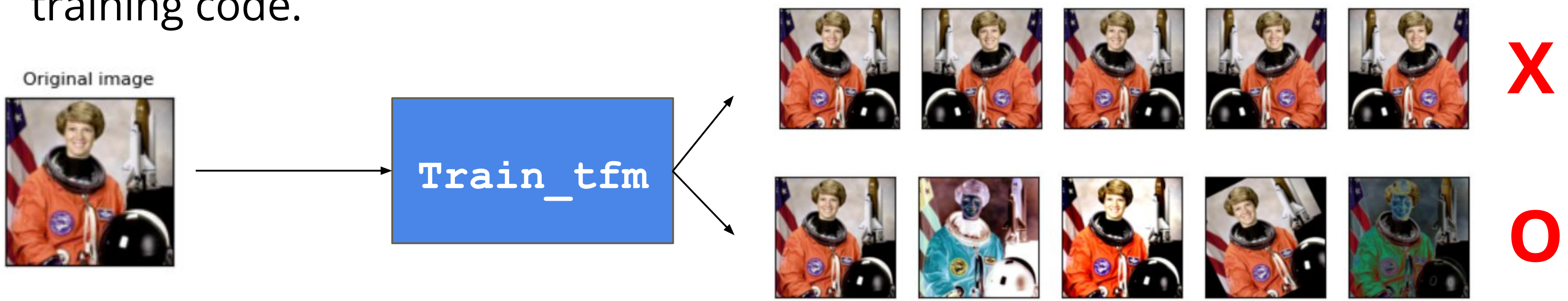
Gradescope

- Gradescope links
- Submit with gradescope, no need to upload any files.
- We can only see your last submission.
- Gradescope deadline: **2023/3/31 23:59 (UTC+8)**

Q1. Augmentation Implementation (2%)

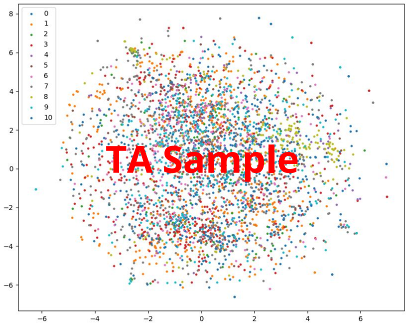
Implement augmentation by finishing **`train_tfm`** in the code with image size of your choice. Copy your **`train_tfm`** code and paste it onto the GradeScope, and explain the effects of transformations you report.

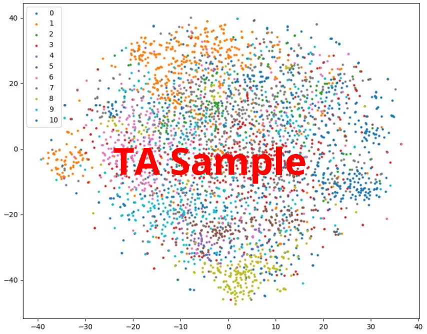
- Your **`train_tfm`** must be capable of producing 5+ different results when given an identical image multiple times.
- Your **`train_tfm`** in the report can be different from **`train_tfm`** in your training code.



Q2. Visual Representations Implementation (2%)

- Visualize the learned visual representations of the CNN model on the **validation set** by implementing t-SNE (t-distributed Stochastic Neighbor Embedding) on the output of both **top & mid** layers (You need to submit **2** images).
- Briefly explain your result of the t-SNE visualization.





Q2. Visual Representations Implementation (2%)

- Take the CNN architecture beside as example.
- The layers contained within the red box may be identified as the "Bottom layers."
- The layers contained within the green box may be identified as the "Mid layers."
- The layers contained within the blue box may be identified as the "Top layers."

```
self.cnn = nn.Sequential(
    nn.Conv2d(3, 64, 3, 1, 1),  # [64, 128, 128]
    nn.BatchNorm2d(64),
    nn.ReLU(),
    nn.MaxPool2d(2, 2, 0),      # [64, 64, 64]

    nn.Conv2d(64, 128, 3, 1, 1), # [128, 64, 64]
    nn.BatchNorm2d(128),
    nn.ReLU(),
    nn.MaxPool2d(2, 2, 0),      # [128, 32, 32]

    nn.Conv2d(128, 256, 3, 1, 1), # [256, 32, 32]
    nn.BatchNorm2d(256),
    nn.ReLU(),
    nn.MaxPool2d(2, 2, 0),      # [256, 16, 16]

    nn.Conv2d(256, 512, 3, 1, 1), # [512, 16, 16]
    nn.BatchNorm2d(512),
    nn.ReLU(),
    nn.MaxPool2d(2, 2, 0),       # [512, 8, 8]

    nn.Conv2d(512, 512, 3, 1, 1), # [512, 8, 8]
    nn.BatchNorm2d(512),
    nn.ReLU(),
    nn.MaxPool2d(2, 2, 0),       # [512, 4, 4]
)
```

Code Submission

- Compress your code, then submit it to NTU COOL.

`<student ID>_hw3.zip`

`e.g. b10901666_hw3.zip`

- If your codes are complicated, please write a README file.
- We can only see your last submission.
- Do not submit your model or dataset.
- If your code is not reasonable, your final grade will be multiplied by 0.9!
- Submission deadline:
    - **2023/3/31 23:59 (UTC+8)**

Grading Policy - Baseline (1 / 2)

Baseline	Accuracy	Hints	Training Time (Reference)
Simple	0.637	Run Sample Code	0.5hr - 1hr on Colab
Medium	0.700	Do some Data Augmentation & Train longer	1.5hr - 2hr on Colab
Strong	0.814	Use predefined CNN from torchvision or TensorFlow	10hr - 12hr on Colab (Suggest using Kaggle)
Boss	0.874	Cross Validation + Ensemble or any other methods you know	40+hr on Kaggle

Grading Policy - HW Score (2 / 2)

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

Regulations

- You should **NOT** plagiarize, if you use any other resource, you should cite it in the reference. (*) Academic Ethics Guidelines for Researchers by the Ministry of Science and Technology
- You should **NOT** modify your prediction files manually.
- **Do NOT share codes, checkpoints, and prediction files with any living creatures.**
- **Do NOT use any approaches to submit your results more than 5 times per day.**
- **Do NOT use additional data (including finding the answers of testing set) and any pre-trained model.**
- Your **assignment will not be graded** and your **final grade x 0.9** if you violate any of the above rules.
- Prof. Lee & TAs preserve the rights to change the rules & grades.

Hints for baseline

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

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.
  - Torchvision before 0.13 -> pretrained=False
  - After 0.13 -> weights=False

Classification

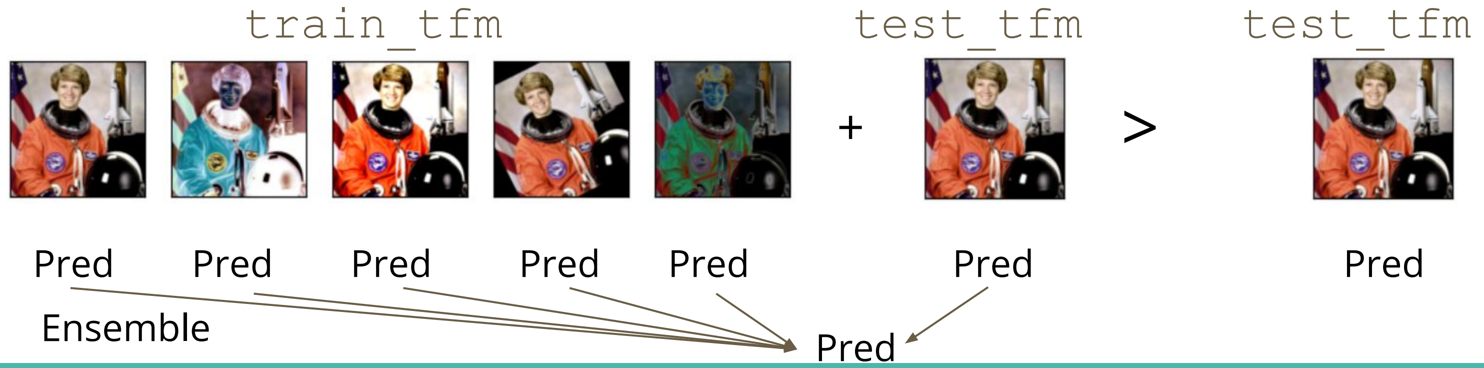
The models subpackage contains definitions for the following model architectures for image classification:

- AlexNet
- VGG
- ResNet
- SqueezeNet

Pre-defined model

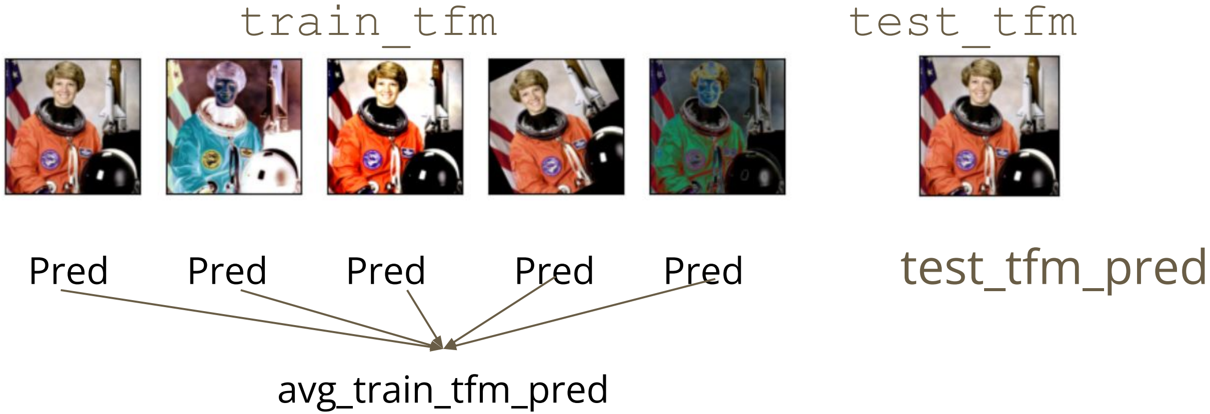
Test Time Augmentation

- The sample code tests images using a deterministic "test transformation"
- You may using 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



Test Time Augmentation

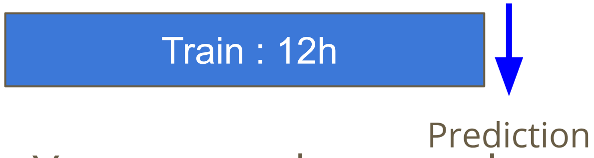
- Usually, test_tfm will produce images that are more identifiable, so you can assign a larger weight to test_tfm results for better performance.



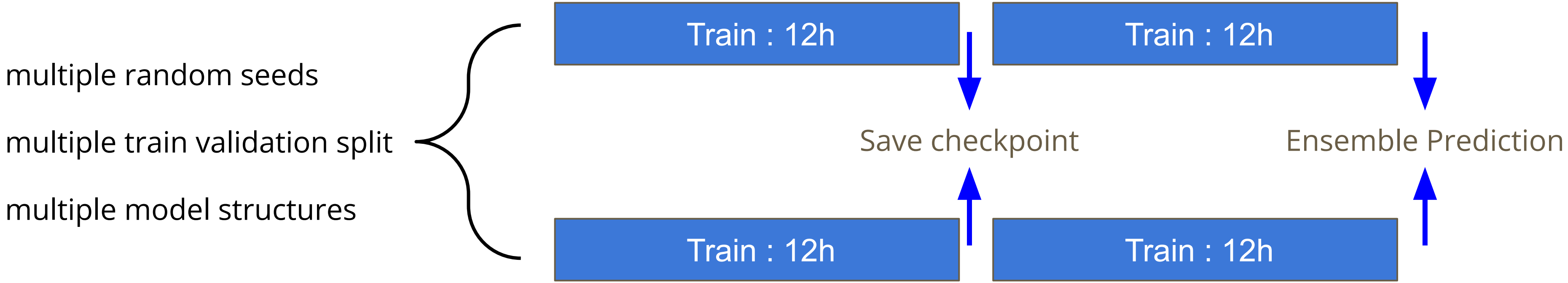
- Ex : Final Prediction = avg_train_tfm_pred * **0.2** + test_tfm_pred* **0.8**

A sample procedure for beating the boss baseline

- The boss baseline might not be beaten with a single model trained on kaggle for 12hrs

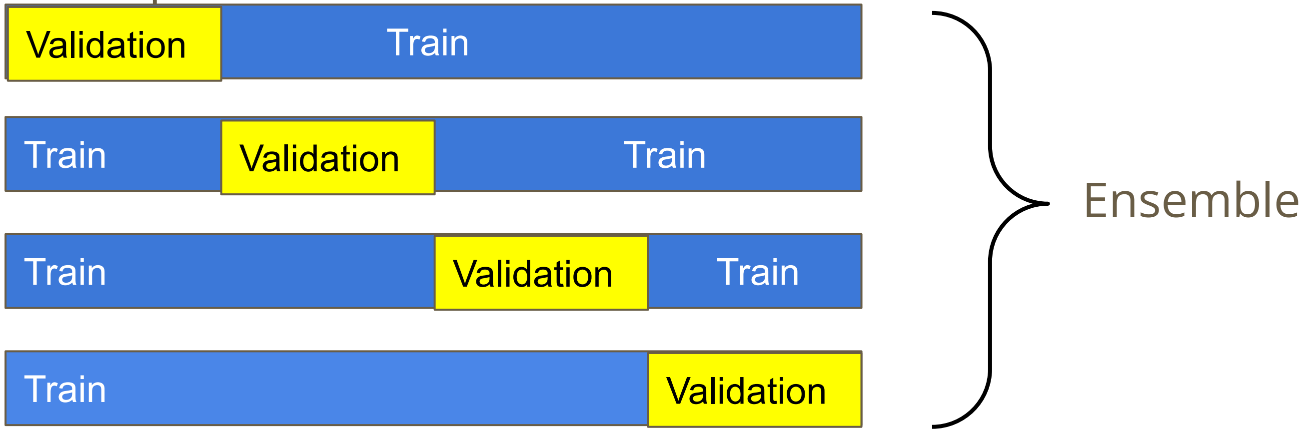


- Your procedure can be ensemble of multiple models with parallelization



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

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

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 **subsampling (max pooling)** 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
- on Optimization
  - Dropout
  - BatchNorm
  - Gradient Accumulation

Useful Links

- Course Website
- NTU COOL
- Kaggle Competition
- Gradescope
- Sample Code (Colab)
- Sample Code (Kaggle)
- Video Links
  - Introduction to CNN
  - tSNE
- Pytorch Documentation
  - torchvision.models
  - torchvision.transforms

Deadline

- Kaggle Contest
  - **2023/3/31 23:59 (UTC+8)**
- GradeScope
  - **2023/3/31 23:59 (UTC+8)**
- Code Submission to NTU COOL
  - **2023/3/31 23:59 (UTC+8)**

Common Questions

- About Submission
  - You should submit a code that produces one of your best results on Kaggle,.
  - You should compress the folder containing your codes into <student_id>_hw3.zip. If your filename is wrong, there would be a penalty on your hw3 score.
- About Kaggle
  - You should rename your display name on Kaggle by changing "Team name".
- About GradeScope
  - You should answer the questions on GradeScope, and no file about answer have to be submitted to NTU COOL.
- These would be updated on NTU COOL, please read the FAQ before you ask questions.

If you have any questions, you can ask us via...

- **Please see the FAQ before you ask questions!!!!!**
- NTU COOL (recommended)
    - https://cool.ntu.edu.tw/courses/24108/discussion_topics/184307
- Email
    - mlta-2023-spring@googlegroups.com
    - The title should begin with “[hw3]”
- TA hour
    - meet.google.com/dzv-ppjx-qtq
        - Monday 19:00 - 20:00（中文）
        - Monday 20:00 - 21:00（English)
    - Friday 上課課餘時間