深度學習系統與實現

LAB02 – Data augmentation

Dept. of Computer Science and
Information Engineering

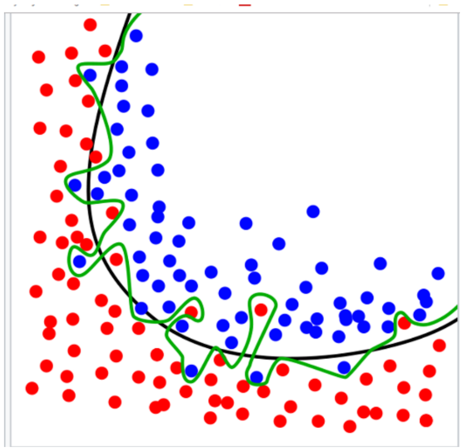
National Chiao Tung University

Outline

- Background
- Purpose
- LAB 2-1
- LAB 2-2
- Note

Background

- □ Overfitting
  - Model best follows the training data
  - But, have high error on unseen data (test)
  - Black line - regularized model
  - Green line – overfitting model
- □ Underfitting
  - Perform bad on training data
  - Ex: fit linear model to non-linear data

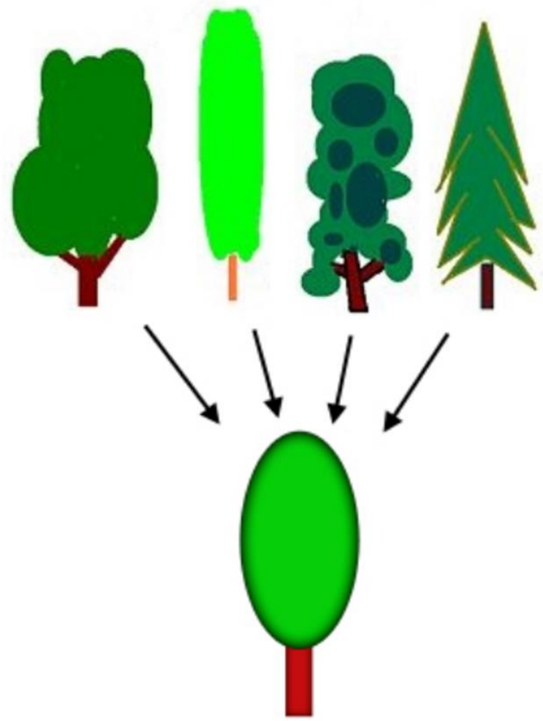


Background

- Cause of overfitting
  - Use a model specific to training data
  - The # of parameters is too much to perfectly predict the training data
- Solution
  - Increase the training data – Find the training data can cover all situations (w/o unseen)
  - Data augmentation – improve generality
  - Regularization

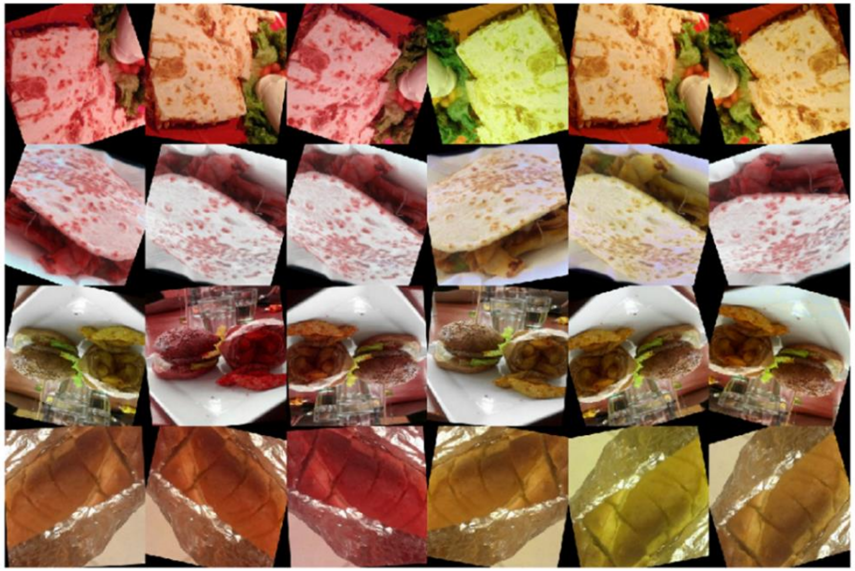
Background

- □ Generalization
  - Only extract the essential feature of the training data, don’t care other detail
- □ Specialization
  - Too detailed
  - We also called the too detailed feature in training data - Noise



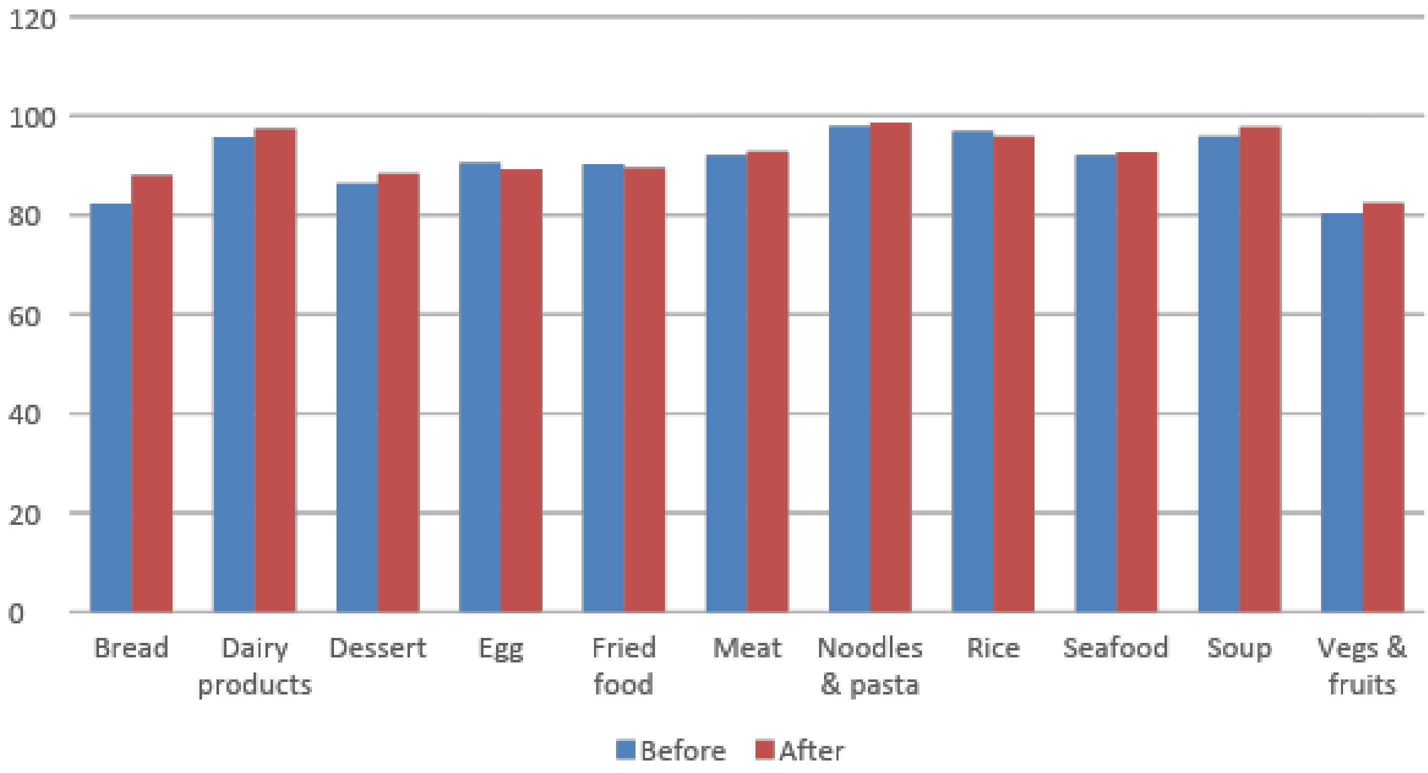
Background

- Data augmentation
  - Create more data to solve the problem of lack of data – prevent overfitting
  - Some simple augmentation like: rotate, scale, crop, flip, adjust color, add noise…et al.



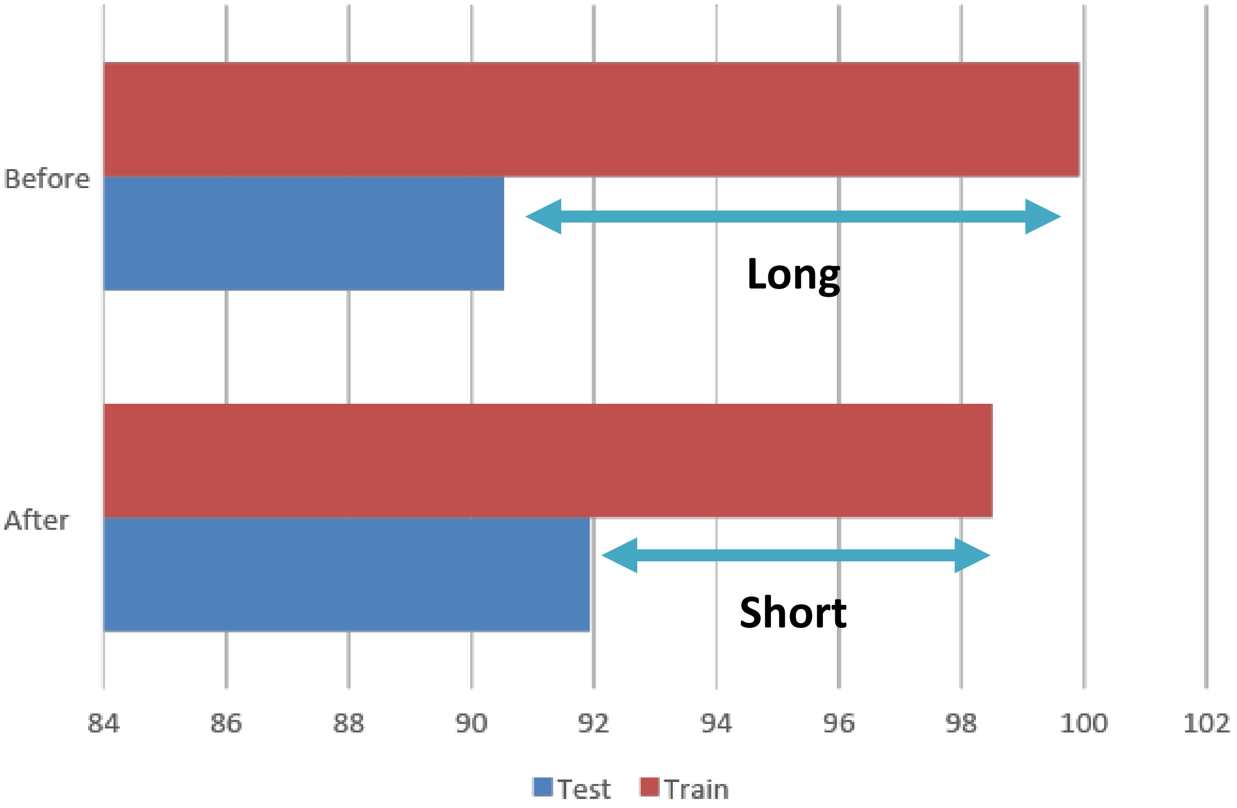
Purpose

- Improve generalization (Acc. In test ↑)
  - Total: 90.53 -> 91.93



Purpose

- Reduce overfitting



LAB 2-1 (40%)

- Use simple data augmentation to improve the performance of your model
  - Just use the transform function supplied by pytorch (torchvision.transforms)
  - Can use your own model or TA's
  - Dataset: Food11
- Please show the accuracy of each class(test set) before and after data augmentation

LAB 2-2 (60%)

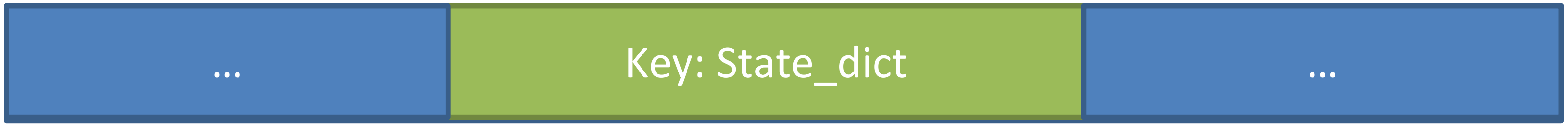
- TA will supply the pre-trained model – Resnet18

torch.models.resnet18()

  - Only provide weights/bias parameters
  - Is a checkpoint file

Python dictionary

pretrained.ckpt

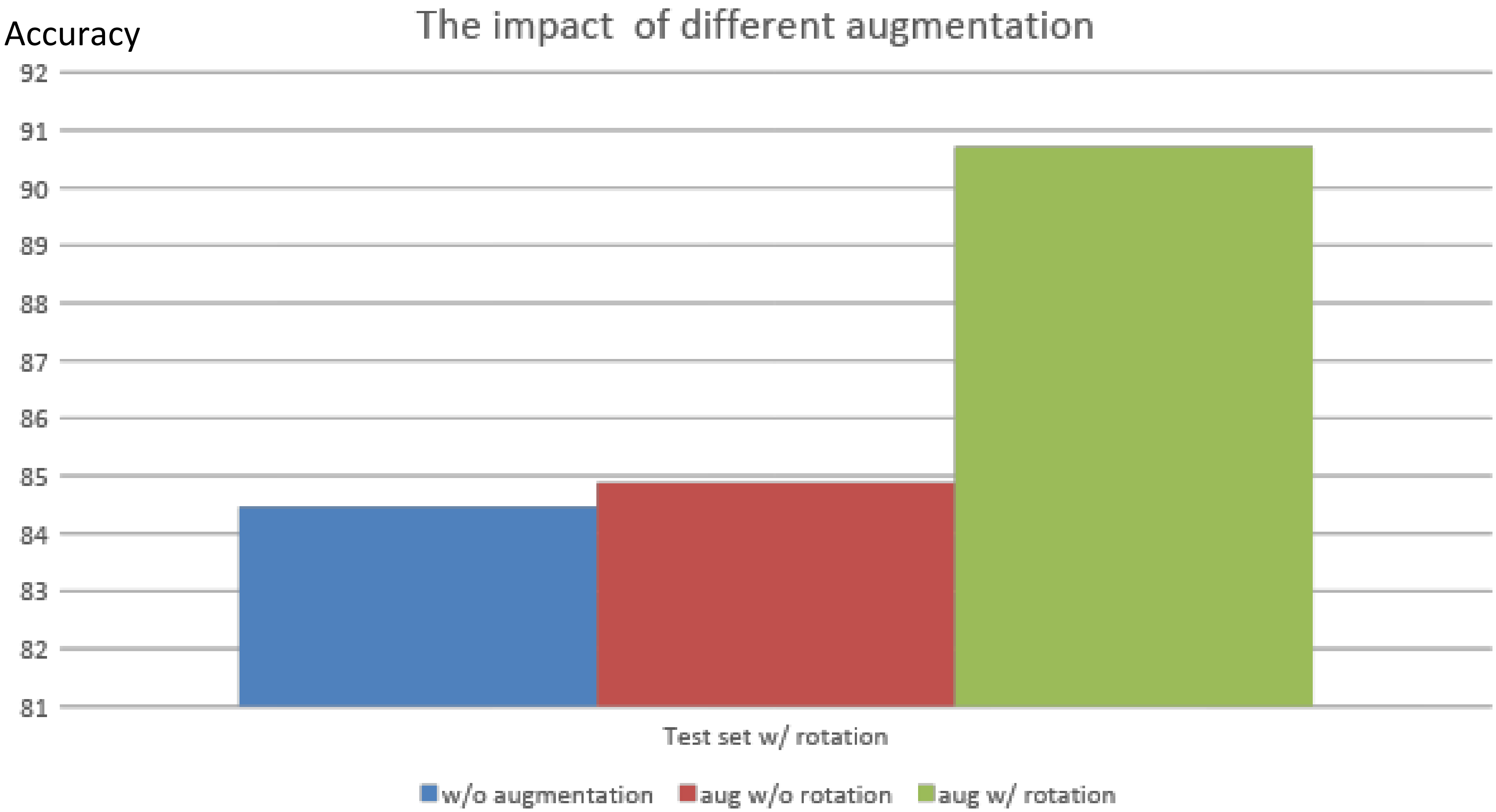


Hint: load the state_dict (param) – torch.load() & model.load_state_dict()

Hint: the output channel of fully connect layer should change to 11

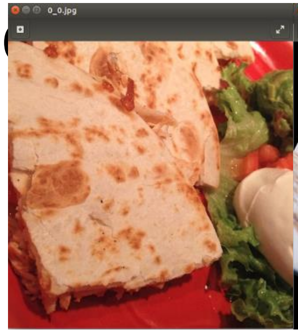
- TA will supply another testing data
  - The new data is also created via data augmentation
  - You can simply see the impact of diff. augmentation

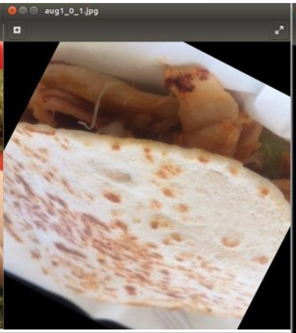
In Pratice

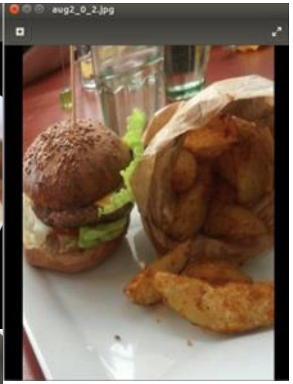

Accuracy
The impact of different augmentation
92
91
90
89
88
87
86
85
84
83
82
81
Test set w/ rotation
🟦w/o augmentation 🟥aug w/o rotation 🟩aug w/ rotation

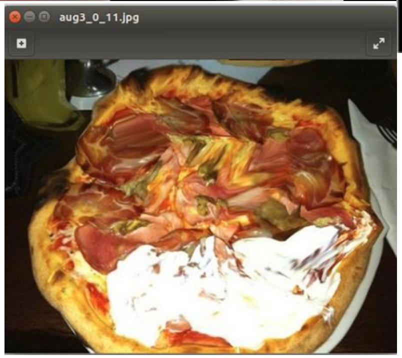
LAB 2-2 (60%)

- □ Purpose: To understand that you should adopt diff. data-augmentation depend on the cases
- □ Use TA’s pre-trained m
  - Four type of data –
    - {class no.}_XXXX.jpg
    - aug1_{class no.}_XXXX.jpg
    - aug2_{class no.}_XXXX.jpg
    - aug3_{class no.}_XXXX.jpg









- □ You can use another library
  - Suggest: imgaug

(a python library)

LAB 2-2 (60%)

- □ Use data augmentation to increase accuracy up to 87% for second test-set (30%)
- □ Please show the accuracy of each class(second test-set) before and after data augmentation (15%) (Don’t just use the torch.transforms)
- □ Please show the accuracy of each class(original test-set) before and after data augmentation

  (15%) (Don’t just use the torch.transforms)

(Note: don’t use any regularization method)

LAB 2-2 (60%)

□ Configuration suggestion:

- Optimizer: SGD
- Momentum: 0.9
- Initial learning rate: 0.001
- Lr decay: StepLR (step size = 7 , gamma = 0.5)
- Batch size: 64

Note

- Data augmentation is time consuming
- Average cost per image:
  - Pytorch (torchvision) – 0.0287 s
  - Imgaug – 0.0308 s
  - Example – a train set with 10,000 images
    - 1 epoch - 0.03 x 10,000 x 1 = 300 s = 5 min
    - 20 epoch - 5 x 20 = 100 min = 1 hr. 40 min

Grading

□ LAB 2-1 (40%)

□ LAB 2-2 (60%)

□ Bonus (5%)

- Make the accuracy higher than 91 % (lab 2-2, original test data)
- Use data augmentation and no regularization

Total: 105

□ Submission: source code + report (E3)

- zip format (ex: dllab_lab2_{group id}.zip)

□ Deadline : 2018/10/9 (二), 23:59

□ Demo : Date To Be Determined

Report Spec.

- EX:
  - Problems & solutions
  - Experiment setup
  - Results
  - Analysis

Reference

- Pytorch – torchvision:
  - https://pytorch.org/docs/stable/torchvision/index.html
- Pytorch Document:
  - https://pytorch.org/docs/stable/index.html
- Imgaug Document & git-repo :
  - https://imgaug.readthedocs.io/en/latest/
  - https://github.com/aleju/imgaug
- How to use Imgaug:
  - https://colab.research.google.com/drive/109vu3F1LTzD1gdVV6cho9fKGx7IzbFll#scrollTo=rQ6DFPNvVD8s