

---

# Machine Learning Homework 8

## Anomaly Detection

ML TAs

[ntu-ml-2021spring-ta@googlegroups.com](mailto:ntu-ml-2021spring-ta@googlegroups.com)

---

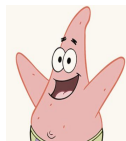
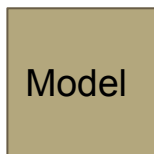
# Goal

- Unsupervised anomaly detection in computer vision: Whether a machine learning model is able to tell a testing image is of the same class (distribution) as the training images

# Goal

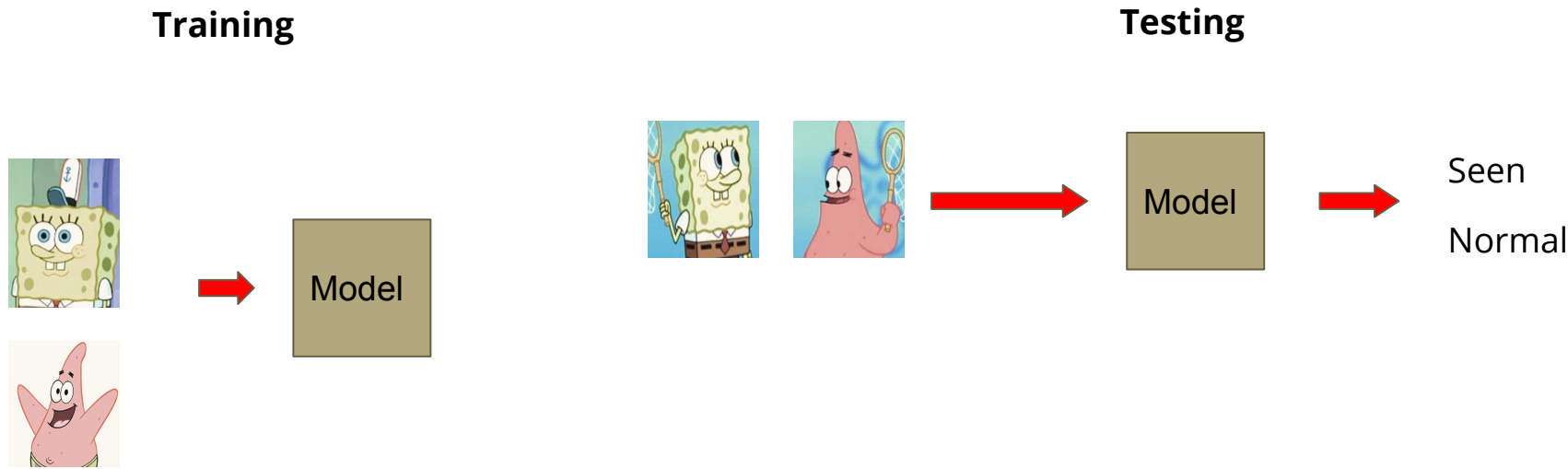
- Unsupervised anomaly detection in computer vision: Whether a machine learning model is able to tell a testing image is of the same class (distribution) as the training images

## Training



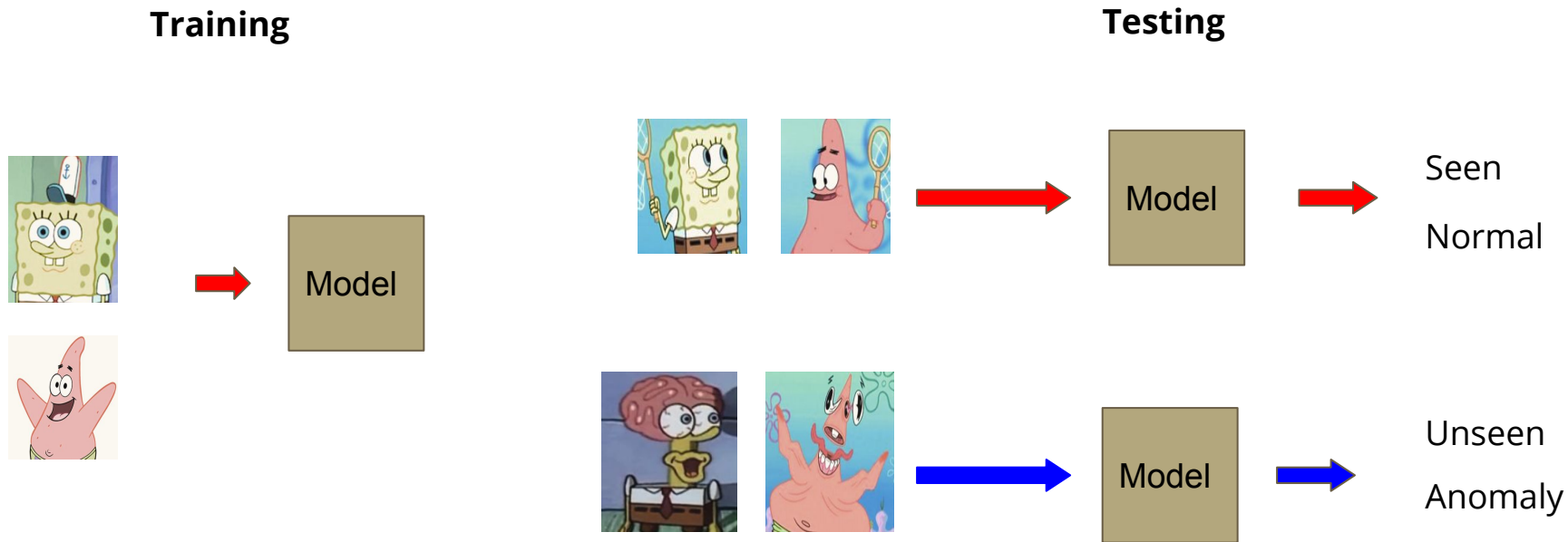
# Goal

- Unsupervised anomaly detection in computer vision: Whether a machine learning model is able to tell a testing image is of the same class (distribution) as the training images



# Goal

- Unsupervised anomaly detection in computer vision: Whether a machine learning model is able to tell a testing image is of the same class (distribution) as the training images

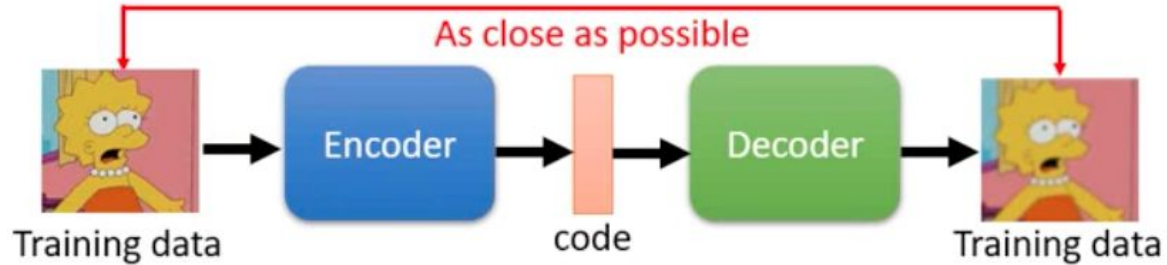


# Data

- Trainingset: About 140k human faces (size  $64*64*3$ )
- Testingset: Another 10k data from the same distributions as the trainingset (normal data, of class label 0) along with 10k human face images from the other distributions (anomalies, of class label 1)
- Notice: Additional training data and pretrained models are prohibited
- Data format: tar zxvf data-bin.tar.gz
- data-bin/
  - trainingset.npy
  - testingset.npy

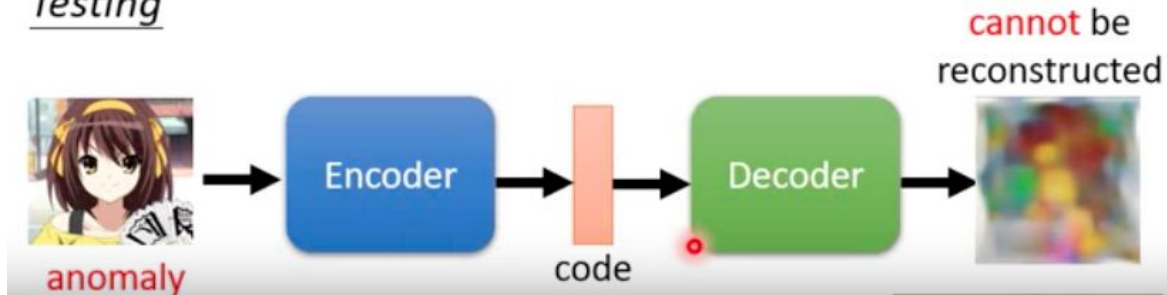
# Method - Autoencoder

## Training



Using training data to learn an autoencoder

## Testing



# Autoencoder

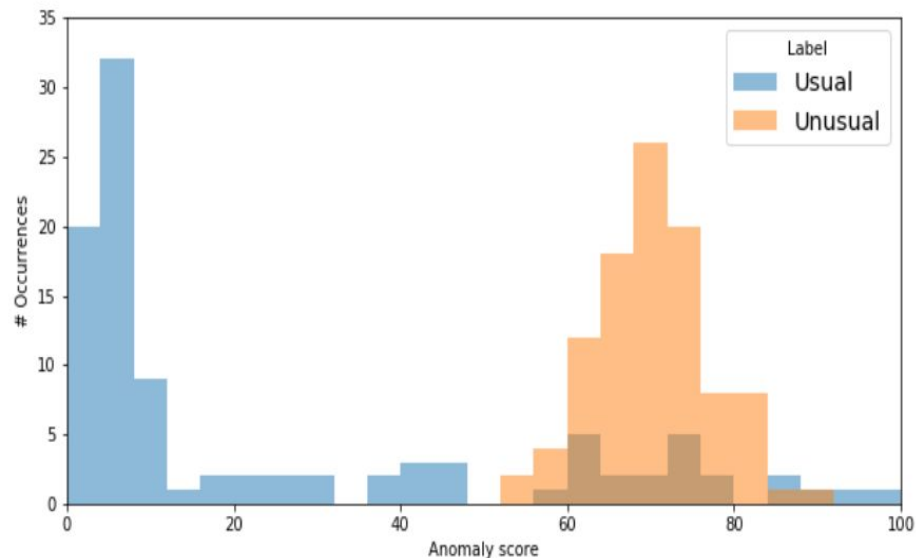
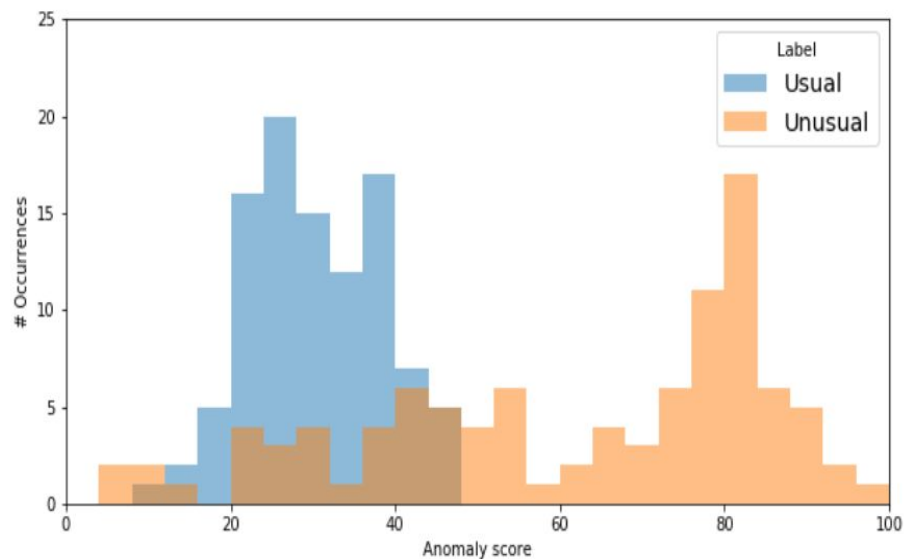
- When to stop training? Training should stop when the mse loss converges
- During inference, we calculate the reconstruction error between the input image and the reconstructed one
- The reconstruction error will be referred to as **abnormality** (anomaly score)
- The **abnormality** of an image can be a metric of the possibility that it's distribution is unseen during training
- Therefore we use the **abnormality** as our predicted values



# Accuracy score

- Usually we compute accuracy scores for classification tasks
- Here, our model functions as a **sensor** rather than a classifier
- Thus, we need a **threshold** with respect to **abnormality** (usually the reconstruction error) to determine whether a piece of data is an anomaly
- If we used accuracy score for this assignment, you would have to try every possible threshold for one single model to get a satisfactory score
- However, what we want is a **sensor** that gets the highest accuracy on the average of every possible threshold

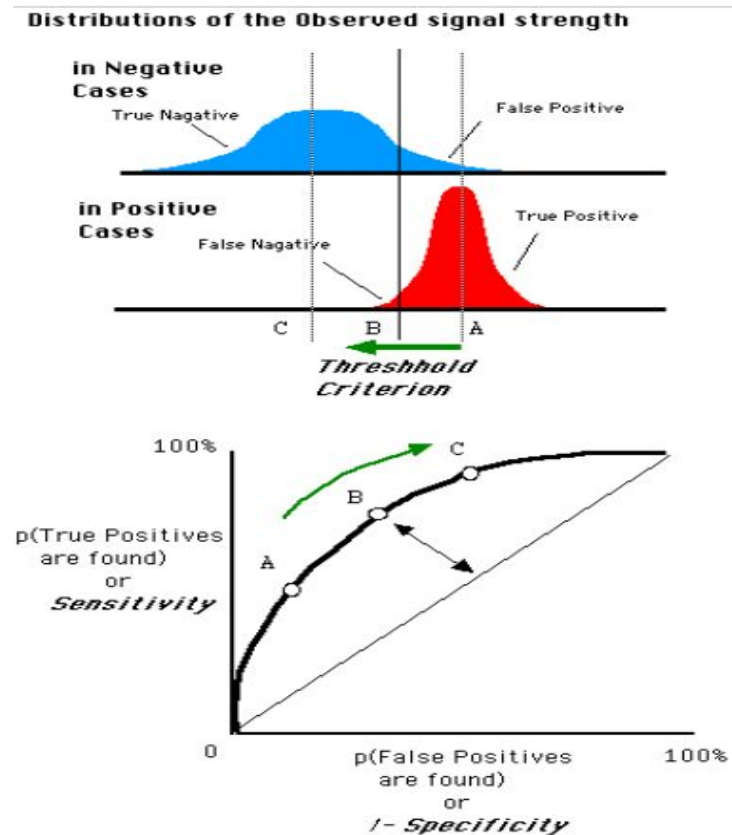
# Which sensor is better?



# Metric - ROC\_AUC score

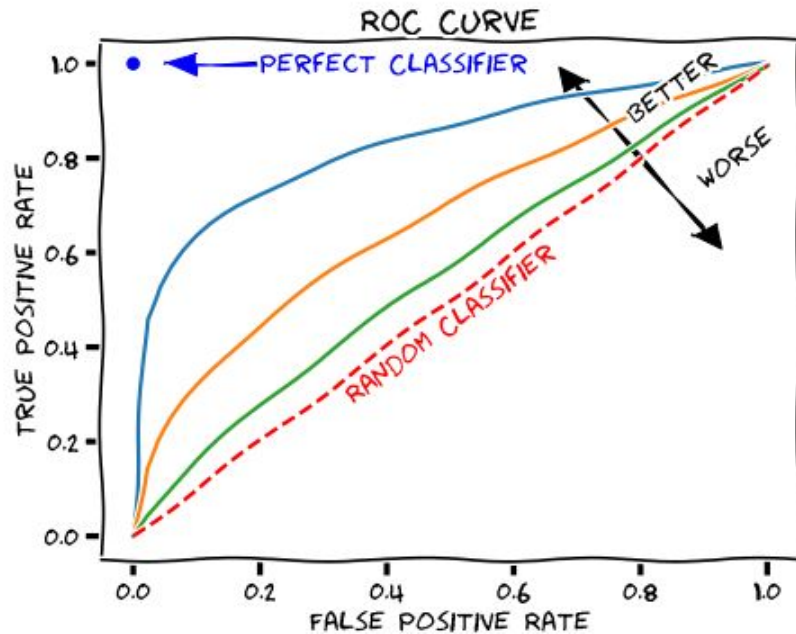
- A better sensor should
  - Give high anomaly scores to the abnormal and low scores to the normal
  - Exhibit a large gap between the scores of 2 groups
- An ROC is suitable for our task
- Each point on the ROC curve stands for true positive rate and false positive rate at certain threshold
- The Area Under the ROC curve is calculated to measure the general ability of the model

# ROC\_AUC score



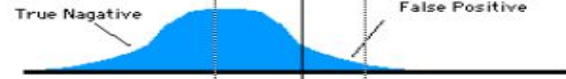
[https://en.wikipedia.org/wiki/Receiver\\_operating\\_characteristic](https://en.wikipedia.org/wiki/Receiver_operating_characteristic)

# ROC\_AUC score



Distributions of the Observed signal strength

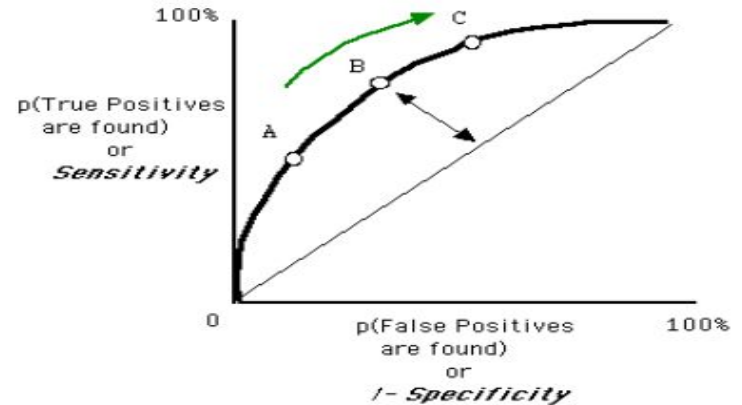
in Negative Cases



in Positive Cases



Threshold Criterion



[https://en.wikipedia.org/wiki/Receiver\\_operating\\_characteristic](https://en.wikipedia.org/wiki/Receiver_operating_characteristic)

# Kaggle

Metric: ROC\_AUC score

Sample output:

```
Id,Predicted
0,10000.0
1,10000.0
2,10000.0
3,10000.0
4,10000.0
5,10000.0
6,10000.0
7,10000.0
8,10000.0
9,10000.0
10,10000.0
11,10000.0
12,10000.0
13,10000.0
14,10000.0
```

# How ROC AUC is calculated

ID	Anomaly score	Label
0	11383	0
1	256676	1
2	862365	1
3	152435	0
4	848171	0

Sort  
by  
score



ID	Anomaly score	Label
2	862365	1
4	848171	0
1	256676	1
3	152435	0
0	11383	0

# How ROC AUC is calculated

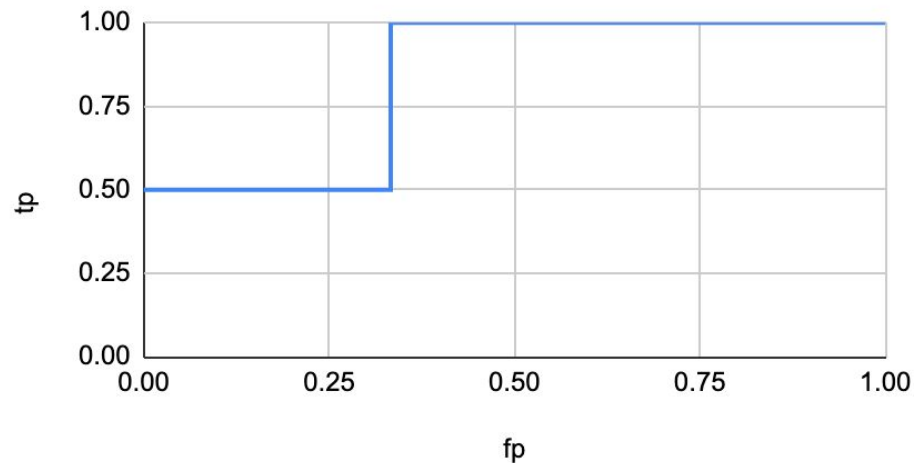
ID	Anomaly score	Label	fp before normalization	tp before normalization
2	862365	1	0	1
4	848171	0	1	1
1	256676	1	1	2
3	152435	0	2	2
0	11383	0	3	2



# How ROC AUC is calculated

ID	Anomaly score	Label	fp	tp
0	11383	0	0	0.5
3	152435	0	0.333333	0.5
1	256676	1	0.333333	1
4	848171	0	0.666667	1
2	862365	1	1	1

ROC curve



Area Under Curve:  $0.5 * \frac{1}{3} + \frac{2}{3} = 0.8333$

# Scoring

- Code submission: 4 pt
- Baselines 6 pt (3 pt for the public ones and the other 3 pt for the private ones)
  - Simple public: 1 pt (public score: 0.64046)
  - Medium public: 1 pt (public score: 0.75719)
  - Strong public: 0.5 pt (public score: 0.81304)
  - Boss public: 0.5 pt (public score: 0.86590)
  - Simple private: 1 pt
  - Medium private: 1 pt
  - Strong private: 0.5 pt
  - Boss private: 0.5 pt
- Bonus for submitting report: 0.5 pt

# Bonus

- If you succeed in beating both boss baselines, you can get extra 0.5 pt by submitting a brief report to explain your methods (in less than 100 English words), which will be made public to the whole class
- [Report Template](#)

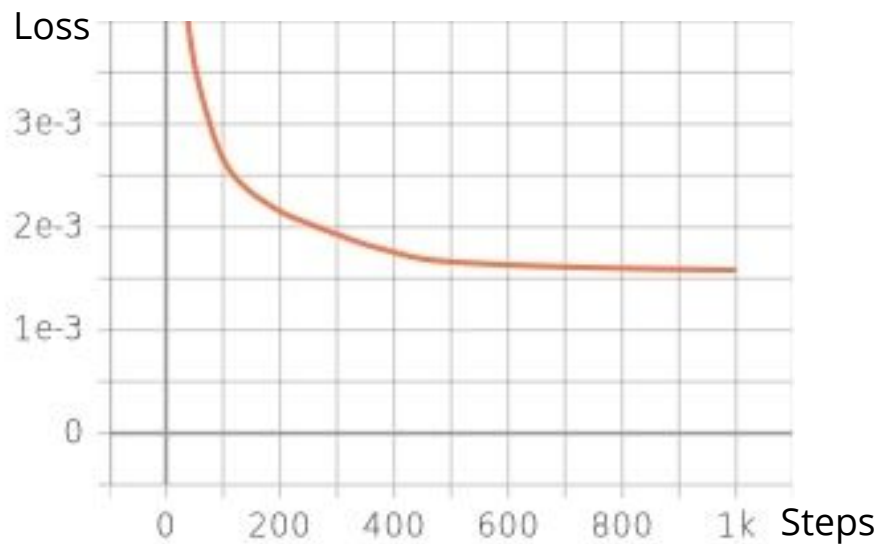
# Baseline guides

- Simple
  - FCN autoencoder
- Medium
  - CNN autoencoder
  - Try smaller models (less layers)
  - Smaller batch size
- Strong
  - Add BatchNorm
  - Train for longer
- Boss:
  - Add an extra classifier
  - Sample random noises as anomaly images
  - Or one-class-classification (OCC) with GANs: [OCGAN](#), [End-to-end OCC](#), [paper pool for Anomaly Detection](#)

# Baseline training statistics

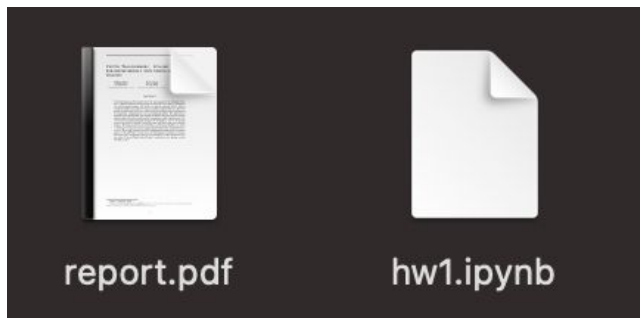
- Simple
  - Number of parameters: 3176419
  - Training time on colab: ~ 30 min
- Medium
  - Number of parameters: 47355
  - Training time on colab: ~ 30 min
- Strong
  - Number of parameters: 47595
  - Training time on colab: 4 ~ 5 hrs
- Boss:
  - Number of parameters: 4364140
  - Training time on colab: 1.5~3 hrs

# Strong baseline training curve



# Code Submission

- Your .zip file should include only
  - **Code:** either .py or .ipynb
  - **Report:** .pdf (only for those who got 10 points)
- Example:



# Code Submission

- Wrap your codes in a directory whose format should be:
- **<student\_id>\_hw8/**
  - Your codes
- And zip it as **<student\_id>\_hw8.zip**
- Note that upon unzipping, there must be a directory named **<student\_id>\_hw8/** , which must contain all your codes
- Submit **<student\_id>\_hw8.zip** via NTU COOL



# Code Submission

- DO
  - Specify the source of your code. (You may refer to [Academic Ethics Guidelines](#))
  - Organize your code and make it easy to read (not necessary)
- DO NOT
  - Submit empty or garbage files
  - Submit the dataset or model
  - Compress your codes into other formats like .rar or .7z and simply rename it to .zip
- Note
  - We can only see your last submission
  - Do not submit your model or dataset
  - If your code is not reasonable, your semester grade **x 0.9**

# Regulations

- Plagiarism is not allowed
- Do not modify your prediction file
- Do not share your prediction file with anyone
- Do not submit your prediction file more than **5** times to Kaggle in any way
- **Do NOT search or use additional data or pre-trained models.**
- Violators are subject to **x 0.9** of their semester grades
- Prof. Lee & TAs preserve the rights to change the rules & grades

# Important dates

- Kaggle deadline: 5/21 23:59 (**GMT+8**)
- Code & report deadline: 5/23 23:59 (**GMT+8**)
- **Late submissions are NOT allowed**

# Links

- Kaggle: <https://www.kaggle.com/c/ml2021spring-hw8>
- Colab:  
[https://colab.research.google.com/drive/1D\\_8lkhzLfoVhA6bTekf-Yw82o6P4g1rQ?usp=sharing](https://colab.research.google.com/drive/1D_8lkhzLfoVhA6bTekf-Yw82o6P4g1rQ?usp=sharing)

# Contact TAs

- NTU COOL (recommended)
  - <https://cool.ntu.edu.tw/login/portal>
- Email:
  - [ntu-ml-2021spring-ta@googlegroups.com](mailto:ntu-ml-2021spring-ta@googlegroups.com)
  - The title should begin with **[hw8]**
- TA hour
  - Each Friday during class