Discussion 10

EE599 Deep Learning

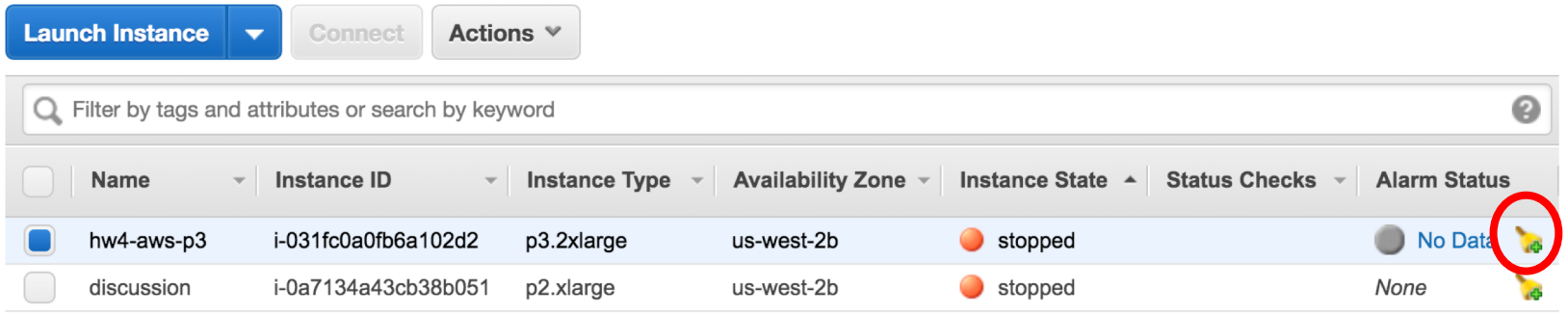
Kuan-Wen (James) Huang

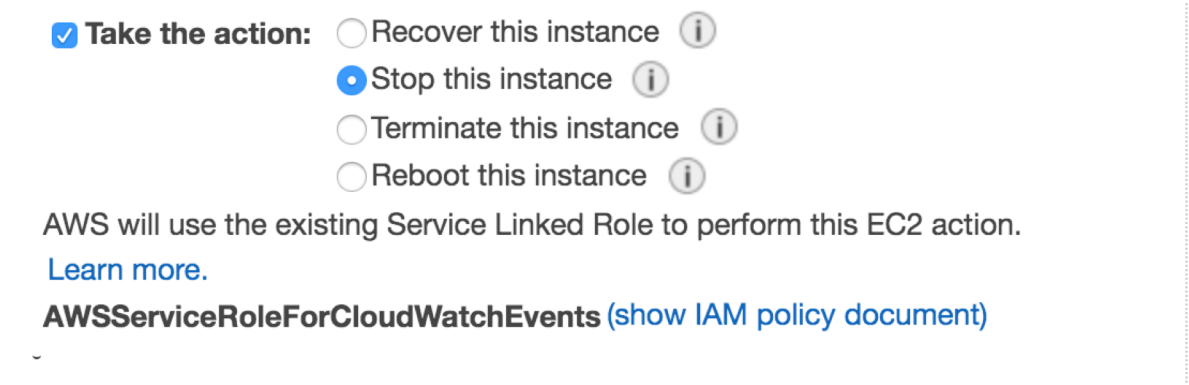
Spring 2020

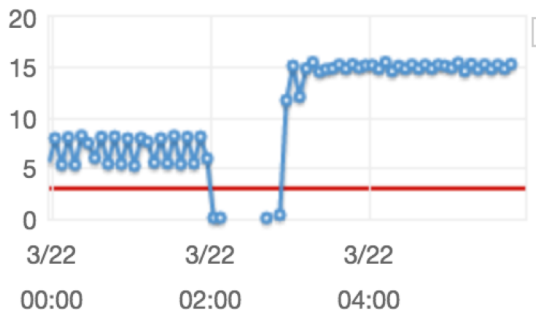
How to save your AWS credit

- Work/debug your codes locally or with a cheap instance (p2) before running it on p3.
- Set an alarm which stops the instance when the work is done.
- Use spot instance (at late night).

How to save your AWS credit - Set an Alarm







How to save your AWS credit - Use Spot Instance

- AWS has updated their rules and new user has 0 limit on spot p-type instance. Please go to https://console.aws.amazon.com/support/home#/case/create?issueType=service-limit-increase&limitType=service-code-ec2-spot-instances

  and ask for a limit increase.
- Reference: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-limits.html

Pricing and Cost Report

- https://aws.amazon.com/ec2/instance-types/

- Pricing
  - https://aws.amazon.com/ec2/pricing/on-demand/
  - https://aws.amazon.com/ec2/spot/pricing/

- Cost Report
  - https://console.aws.amazon.com/cost-reports/home?#/dashboard

Set up the Environment of Remote Instance

- Virtual Environment with latest python and tensorflow/pytorch
  - `conda create -n new_env python=3.7 anaconda`
  - `source activate new_env`
  - `pip install tensorflow pytorch`
  - `pip install torchvision sklearn`
- Update cuDNN libraries (NVIDIA CUDA® Deep Neural Network libraries)
  - `gdown https://drive.google.com/uc?id=1hi2HBzCyD3xPUU2F2xe2XoPXPPLhHef9`
  - `tar -xzvf cudnn-10.0-linux-x64-v7.6.5.32.tgz`
  - `sudo cp cuda/include/cudnn.h /usr/local/cuda/include`
  - `sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64`
  - `sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn*`
  - Reference: https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html#download and piazza post 198

Download Codes and Dataset (HW4)

- Download the starter code from Jiali's github
  - `git clone https://github.com/davidsonic/EE599-CV-Project.git`

- Download and upzip the dataset
  - `gdown https://drive.google.com/uc?id=1ZCDRRh4wrYDq0O3FOptSlBpQFoe0zHTw`
  - `unzip polyvore_outfits_hw.zip`

Category Classification (HW4)

- Raw images of items are in `polyvore_outfits/images/`
- The file `polyvore_item_metadata.json` provides you the ID of image and its category
- You are asked to train a CNN classifier with
  1. Pre-trained net (either ResNet50 or MobileNet) + extra dense layers
  2. Your own model

Category Classification (HW4)

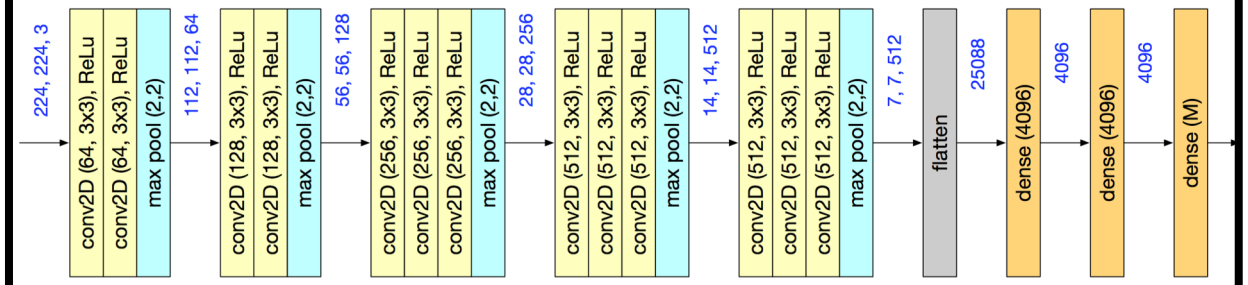
- Raw images of items are in `polyvore_outfits/images/`
- The file `polyvore_item_metadata.json` provides you the ID of image and its category_id
- You are asked to train a CNN classifier with
  1. Pre-trained net (either ResNet50 or MobileNet) + extra dense layers
  2. Your own model

Category Classification (HW4)

- There are two parts you (might) need to code
  - Dataloader and deep learning model

data.py

train_category.py

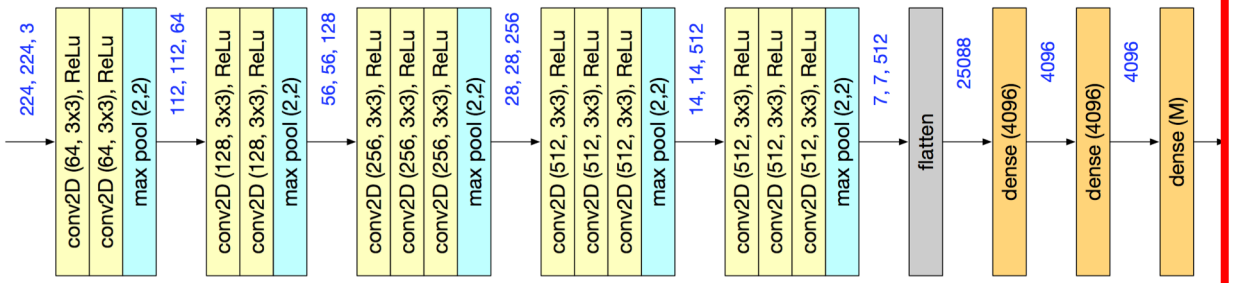


Category Classification (HW4)

- There are two parts you (might) need to code
  - Dataloader and deep learning model

data.py

train_category.py



< Build you model >
model = tf.keras.models.Model(…)
model.compile(…)
model.fit(…)

Category Classification (HW4)

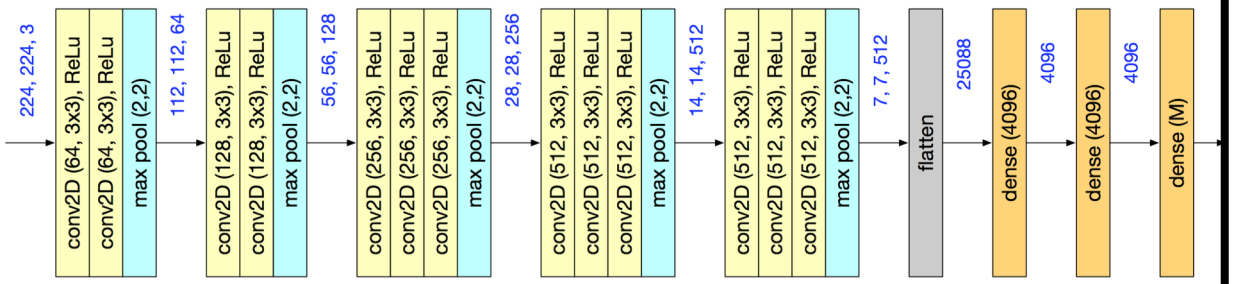
- There are two parts you (might) need to code
  - Dataloader and deep learning model

data.py

For efficient data input pipelines,
use tf.data API

1. Create a tf.data.Dataset object which iterates over all samples
2. Use map() for preprocessing
3. Use batch() to create batches
4. Use prefetch() to let it collect next batch when processing the current batch

train_category.py



< Build you model >
model = tf.keras.models.Model(…)
model.compile(…)
model.fit(…)

Category Classification (HW4)

- Dataloader
  - Zixuan posted a `data2.py` with `train_category.py` on piazza.
  - Check: https://piazza.com/class/k5cxqizvkb8236?cid=203
  - For a complete tutorial: https://www.tensorflow.org/guide/data#basic_mechanics

Compatibility Prediction (HW4)

- Grand goal: Predict whether an outfit (a set of items) is compatible or not.
- Easier one: given a pair of items, predict whether they are compatible, i.e. belong to the same outfit.

Compatibility Prediction (HW4)

- Two parts you need to code

dataloader.py

The model should be similar to the previous problem but takes two input images and the output is a binary (compatible or not) label.

train_compatibility.py

The model should be similar to the previous problem but takes two input images and the output is a binary (compatible or not) label.

Compatibility Prediction (HW4)

- Dataloader for a pair of images and a compatibility label
  - `train.json` and `valid.json` contain sets with their set_id. Each set has items with their corresponding item_id linked to the images.
  - `compatibility_train.txt` and `compatibility_valid.txt` contains the training and validation data. The first column is either 1 or 0 (binary label). However, these files are created directly for compatibility prediction for a set (outfit), not for pairwise compatibility.

  - In order to construct your training and validation dataset, you need to pre-process the data. E.g., take pair of items of each row; they are compatible if label = 1; from set_id + index to get their item_id in json file (item_id helps you find the image).

Compatibility Prediction Bonus (HW4)

- Use your pairwise compatibility prediction net on the outfits (set of items) listed in `compatibility_test_hw.txt`. The prediction is based on the average of all combinations of items.