Assigned: 17 November 2021

Homework #5 – Convolutional Neural Networks (CNN)

EE 541: Fall 2021

Due: Friday, 03 December 2021 at 23:59. No late submissions. Submission instructions will follow separately on canvas.

In this assignment you will apply deep-learning techniques to computer vision. Your task is to create a compatibility classifier using the Polyvore fashion dataset.

Find (minimal) starter code for this project at https://github.com/franzke-usc-21fa/ee541-hw5-starter. You are expected to explore and tune hyper-parameters and model architecture beyond the starter code. Review the `README` file to understand the archive content and the included supplemental files.

Dataset

The *Polyvore Outfits* [1] dataset is *real-world* data created from users' outfit preferences at polyvore.com. Paired items from outfits that receive high user-ratings as *compatible*. The Polyvore dataset contains 68,306 outfits created from 365,054 items. There are a maximum 19 items per outfit. Figure 1 shows an example outfit.

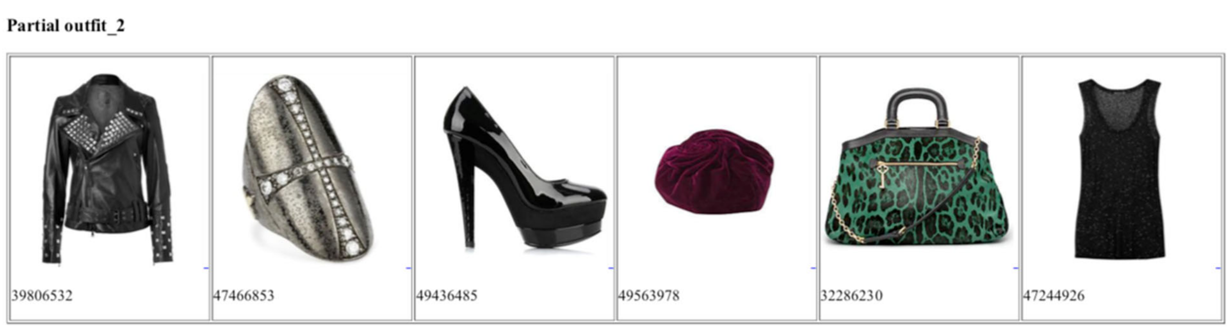

Figure 1: A visualization of a partial outfit in the dataset. The number at the bottom of each image is the ID of this item.

Category classification

- The starter repository includes several files. Some require additions but give structure to follow. First: make sure that you set the dataset location in `utils.py::Config['root_path']`.
    1. `train_category.py`: training script
    2. `model.py`: CNN classifier model (incomplete).
    3. `data.py`: dataset handlers (incomplete).

4. `utils.py`: utility functions and configuration

- The `train_model` function in `train_*.py` handles model training. The driver applies a data batch from the dataloader (`data.py`) to the model during each iteration. Extend the driver to record training accuracy so you can generate learning curves.
- Construct and compare two separate models:
  1. Finetune a pretrained ImageNet model (*e.g.*, ResNet50). The PyTorch model zoo provides many standard pretrained CNN architectures. Use transfer learning to repurpose an ImageNet model (single image input, multi-class output) to this problem (two image input, binary class output).
  2. Construct your own "custom" model and train from scratch.

  Compare the two models (pretrained vs. custom) and comment on the results. What is the advantage of fine-tuning a pretrained model vs. creating a custom network? Did you require learning rates for each model? If so, how did you determine their value and rate schedules?

  Note: the `images` folder contains images coded by id. Refer to `polyvore_item_metadata.json` for more information about each sample.
- Modify `data.py`. Create a dataset and dataloader suitable for this compatibility task. They should prepare supervised training pairs `([image1, image2], label)`. The `get_data_transforms` function applies input normalization. You may modify or extend the data transforms.
- Split **no less than 10% data for testing** your final model. Generate a `compatibility.txt` file that lists item pairwise ids from your test set and the predicted vs ground-truth compatibility.

Tips

1. Expect over-fitting. Adjust and tune the model structure, hyper-parameters, and regularization to reduce over-fitting. You may design any custom model structures you like.
2. Set `to(device)` flag (`device=cuda:0`) and increase the `batch_size` defined in `utils.py` to speed up training.
3. To debug try reducing the data-set size. Set `debug=True` in `utils.py`. You can also reduce the number of epochs and set a random `seed` to aid debugging.
4. It will take many epochs to reach a performance plateau. But this will depend on the network structure and the learning rate(s). Explore whether training further increases generalization or tends toward over-fitting.

References

[1] Mariya I Vasileva, Bryan A Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David Forsyth. Learning type-aware embeddings for fashion compatibility. *Proceedings of the European Conference on Computer Vision* (ECCV), pages 390-405, 2018.