Visual Analytics Portfolio

Assignment 3: Using pretrained CNNs for image classification

Cultural Data Science, 2023

Author: Aleksander Moeslund Wael
Student no. 202005192

Assignment notes (Ross)

In the previous assignments involving classification, we were performing a kind of simple feature extraction on images by making them greyscale and flattening them to a single vector. This vector of pixel values was then used as the input for some kind of classification model.

For this assignment, we're going to be working with an interesting kind of cultural phenomenon - fashion. On UCloud, you have access to a dataset of *Indo fashion* taken from this Kaggle dataset. There is a paper which goes along with it on *arXiv.org*, which you can read here.

Your instructions for this assignment are short and simple:

- You should write code which trains a classifier on this dataset using a *pretrained CNN like VGG16*
- Save the training and validation history plots
- Save the classification report

About the project

This repo contains code for finetuning a pretrained convolutional neural network, and conducting a 15-label image classification task.

Data

The data used in the project is the Indo fashion dataset available on Kaggle. The dataset consists of ~106K images displaying 15 unique cloth categories. The data is pre-split into a training, test and validation set. Make sure to download and structure the data is shown in the `repository structure` segment.

Model

The `VGG16` model was used. The model is 16 layers deep, and has approx 138 million parameters and is trained on the ImageNet dataset. (source).

Pipeline

The `cnn_fashion.py` script in the `src` folder contains the main code pipeline. The structure is as follows:

1. Import and preprocess images and metadata

2. Augment image data
3. Load the `VGG16` model
4. Train the model on the Indo fashion training set.
5. Save the learning curve plots from training.
6. Predict labels of the test set.
7. Print and save a classification report.

Requirements

The code is tested on Python 3.11.2. Futhermore, if your OS is not UNIX-based, a bash-compatible terminal is required for running shell scripts (such as Git for Windows).

Usage

The repo was setup to work with Windows (the WIN_ files), MacOS and Linux (the MACL_ files).

1. Clone repository to desired directory

```
git clone https://github.com/AU-CDS/assignment3-pretrained-cnns-alekswael
cd assignment3-pretrained-cnns-alekswael
```

2. Run setup script

NOTE: Depending on your OS, run either `WIN_setup.sh` or `MACL_setup.sh`.

The setup script does the following:

1. Creates a virtual environment for the project
2. Activates the virtual environment
3. Installs the correct versions of the packages required
4. Deactivates the virtual environment

```
bash WIN_setup.sh
```

3. Run pipeline

NOTE: Depending on your OS, run either `WIN_run.sh` or `MACL_run.sh`.

Run the script in a bash terminal.

The script does the following:

1. Activates the virtual environment
2. Runs `cnn_fashion.py` located in the `src` folder
3. Deactivates the virtual environment

```

```

```
bash WIN_run.sh
```

Note on model tweaks

Some model parameters can be set through the `argparse` module. However, this requires running the Python script seperately OR altering the `run*.sh` file to include the arguments. The Python script is located in the `src` folder. Make sure to activate the environment before running the Python script.

```
cnn_fashion.py [-h] [-bs BATCH_SIZE] [--train_subset TRAIN_SUBSET] [--val_subset
VAL_SUBSET] [--test_subset TEST_SUBSET] [-e EPOCHS]

options:
  -h, --help            show this help message and exit
  -bs BATCH_SIZE, --batch_size BATCH_SIZE
                        Batch size for training. (default: 64)
  --train_subset TRAIN_SUBSET
                        Number of training images to use. If not specified, all
images are used. (default: 91166)
  --val_subset VAL_SUBSET
                        Number of validation images to use. If not specified, all
images are used. (default: 7500)
  --test_subset TEST_SUBSET
                        Number of test images to use. If not specified, all images
are used. (default: 7500)
  -e EPOCHS, --epochs EPOCHS
                        Number of epochs to train for. (default: 10)
```

Repository structure

This repository has the following structure:

```
│   MACL_run.sh
│   MACL_setup.sh
│   README.md
│   requirements.txt
│   WIN_run.sh
│   WIN_setup.sh
│
├───images
│   ├───metadata
│   │       test_data.json
│   │       train_data.json
│   │       val_data.json
│   │
│   ├───test
│   │       7500.jpeg
│   │
│   ├───train
```

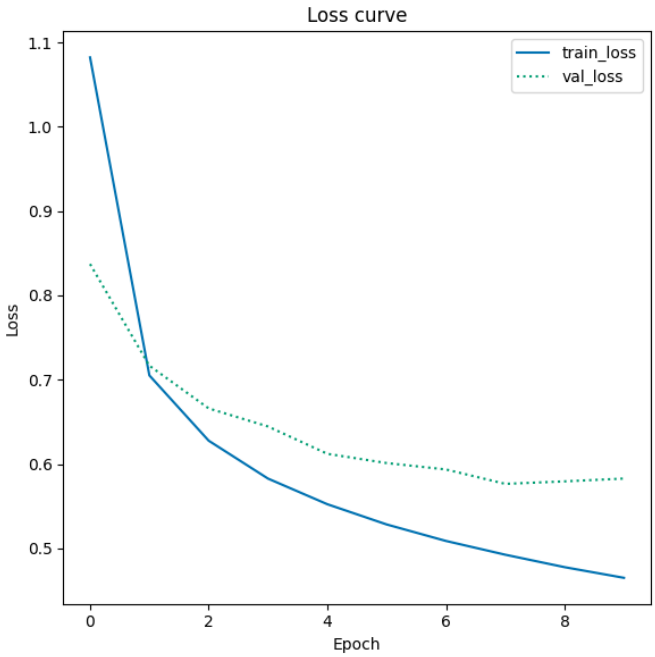
```
│   │      91166.jpeg
│   │
│   └───val
│           7500.jpeg
│
├───out
│       cnn_fashion_SUBSET.png
└───src
        cnn_fashion.py
```

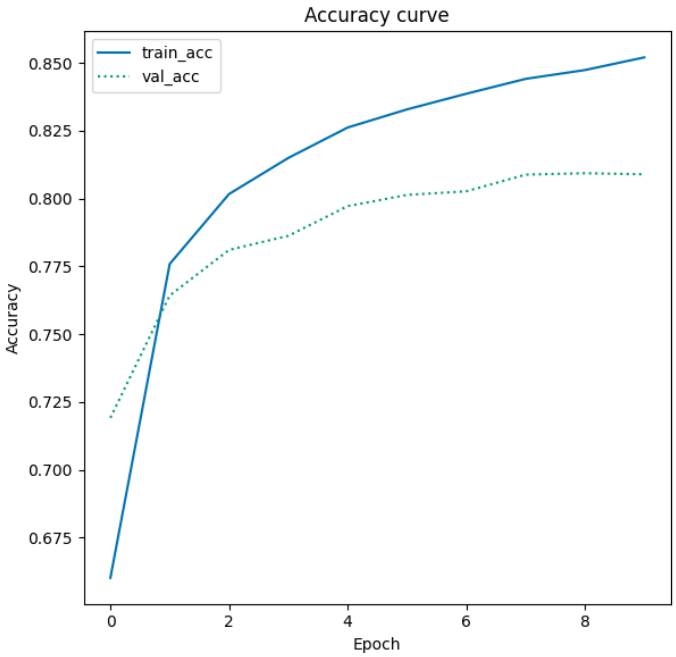
Remarks on findings

The model was trained using the default arguments (all data included, batch size of 64, 10 epochs). This yielded an avg accuracy og 56%, with high intercategory variance. Palazzos do not seem to be predicted at all (neither correct or false), and categories like lehenga, petticoats and curta have decent precision but very low recall (i.e. underpredicts the category). The categories most succesfully predicted are blouse, mojaris_men and saree, with an f1-score of 0.61-0.67.

The learning curves for the model show signs of overfitting quite early on in the fit history, so perhaps the data augmentation, model architecture or data split could be altered to improve the fit.

	precision	recall	f1-score	support
blouse	0.68	0.65	0.67	500
dhoti_pants	0.42	0.28	0.34	500
dupattas	0.34	0.25	0.29	500
gowns	0.11	0.89	0.20	500
kurta_men	0.33	0.00	0.01	500
leggings_and_salwars	0.38	0.69	0.49	500
lehenga	1.00	0.01	0.02	500
mojaris_men	0.88	0.47	0.62	500
mojaris_women	0.75	0.25	0.37	500
nehru_jackets	0.96	0.18	0.31	500
palazzos	0.00	0.00	0.00	500
petticoats	0.71	0.01	0.02	500
saree	0.67	0.55	0.61	500
sherwanis	0.80	0.01	0.02	500
women_kurta	0.31	0.29	0.30	500
accuracy			0.30	7500
macro avg	0.56	0.30	0.28	7500
weighted avg	0.56	0.30	0.28	7500


Loss curve
train_loss
val_loss
Loss
Epoch
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0
2
4
6
8


Accuracy curve
train_acc
val_acc
Accuracy
Epoch
0.850
0.825
0.800
0.775
0.750
0.725
0.700
0.675
0
2
4
6
8