Assignment 3 - Using pretrained CNNs for image classification

Original Assignment Description

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

Tips

- You should not upload the data to your repo - it's around 3GB in size.
    - Instead, you should document in the README file where your data comes from, how a user should find it, and where it should be saved in order for your code to work correctly.
- The data comes already split into training, test, and validation datasets. You can use these in a `TensorFlow` data generator pipeline like we saw in class this week - you can see an example of that here.
- There are a lot of images, around 106k in total. Make sure to reserve enough time for running your code!
- The image labels are in the metadata folder, stored as JSON files. These can be read into `pandas` using `read_json()`. You can find the documentation for that online.

Assignment 3 - Luke Ring (202009983)

Contribution

The code for this assignment was written independently and is my own (Luke Ring, 202009983) zeyus @ github.

Description

This repository contains code that fine tunes a VGG16 model on the Indo Fashion dataset. The training history plot, confusion matrix and classification report are saved in the `out/` folder, along with a csv file containing the training history, and a txt file containing the model summary and classification report.

The data were preprocessed using the following steps:

- Proportionally rescaled the images to `IMAGE_SIZE`x`IMAGE_SIZE` pixels, or `IMAGE_WIDTH`x`IMAGE_HEIGHT` pixels if specified.
- Images were zero-padded to match the input size of the model, with the image centered.
- Normalized the pixel values to be between 0 and 1.
- For training data, random horizontal flips were and random rotations of -20 to 20 percent were applied.

For the results below, the images were resized to 100x200 pixels, and the batch size was 256.

Setup

Windows GPU (optional)

If you have an NVIDIA GPU you can do the following before installing the prerequisites:

- Install Anaconda
- Create a new environment using `conda create -n vgg16 python=3.9`
- Activate the environment using `conda activate vgg16`
- Install cudatoolkit and cudnn with `conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.0`

Prerequisites

- Install the required packages using `pip install -r requirements.txt`
- Download the dataset from Kaggle, unzip and save it to `data/` either manually, or by doing the following:
  - Get your kaggle API token from here and save it to `~/.kaggle/kaggle.json`
  - Run `python src/cnn.py --download` to download the dataset

Usage

- Run `python src/cnn.py` to train the model with the default parameters.

Most settings can be customized, such as input resizing, batch size, number of epochs, etc. Run `python src/cnn.py --help` to see all the available options.

```
❯ python .\src\cnn.py --help
usage: cnn.py [-h] [--version] [-m MODEL_SAVE_PATH] [--download] [-d DATASET_PATH]
[-s IMAGE_SIZE] [-w IMAGE_WIDTH] [-t IMAGE_HEIGHT] [-b BATCH_SIZE] [-e EPOCHS] [-o
OUT] [-n] [-c FROM_CHECKPOINT] [-r] [-p PARALLEL]

Text classification CLI

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -m MODEL_SAVE_PATH, --model-save-path MODEL_SAVE_PATH
                        Path to save the trained model(s) (default: models)
  --download            Download the dataset from kaggle (default: False)
  -d DATASET_PATH, --dataset-path DATASET_PATH
                        Path to the dataset (default: data)
```

```
  -s IMAGE_SIZE, --image-size IMAGE_SIZE
                        The image size (width and height) (default: 32)
  -b BATCH_SIZE, --batch-size BATCH_SIZE
                        The batch size (default: 32)
  -e EPOCHS, --epochs EPOCHS
                        The number of epochs (default: 10)
  -o OUT, --out OUT     The output path for the plots and stats (default: out)
  -n, --no-train        Do not train the model (default: False)
  -c FROM_CHECKPOINT, --from-checkpoint FROM_CHECKPOINT
                        Use the checkpoint at the given path (default: None)
  -r, --resnet          Use ResNet50 as the base model. (default: False)
  -p PARALLEL, --parallel PARALLEL
                        Number of workers/threads for processing. (default: 4)
```

Results

The final VGG16 model was trained in two steps, with a bach size of 256 and 150 epochs each.

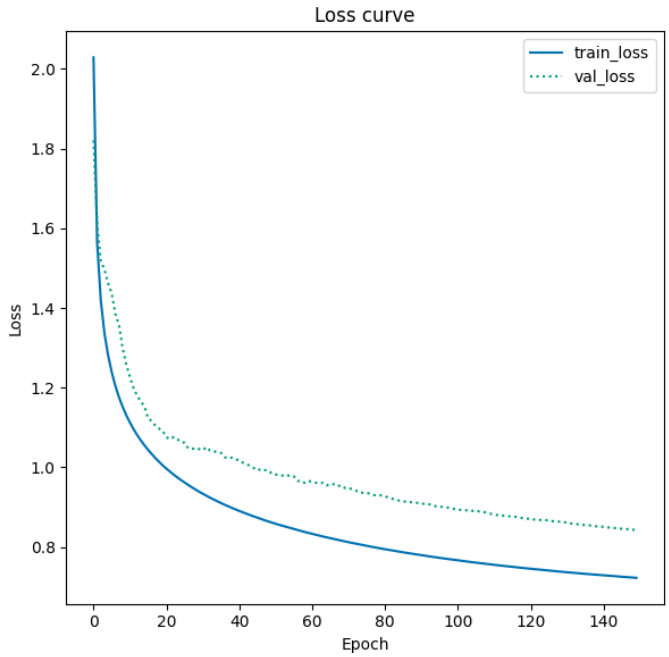
The classification report, model history and confusion matrix are described below.

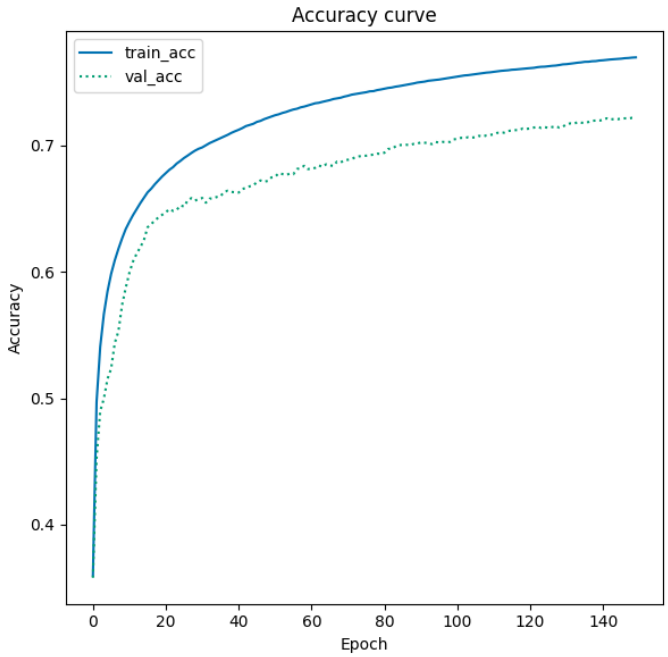
First training step

Classification report

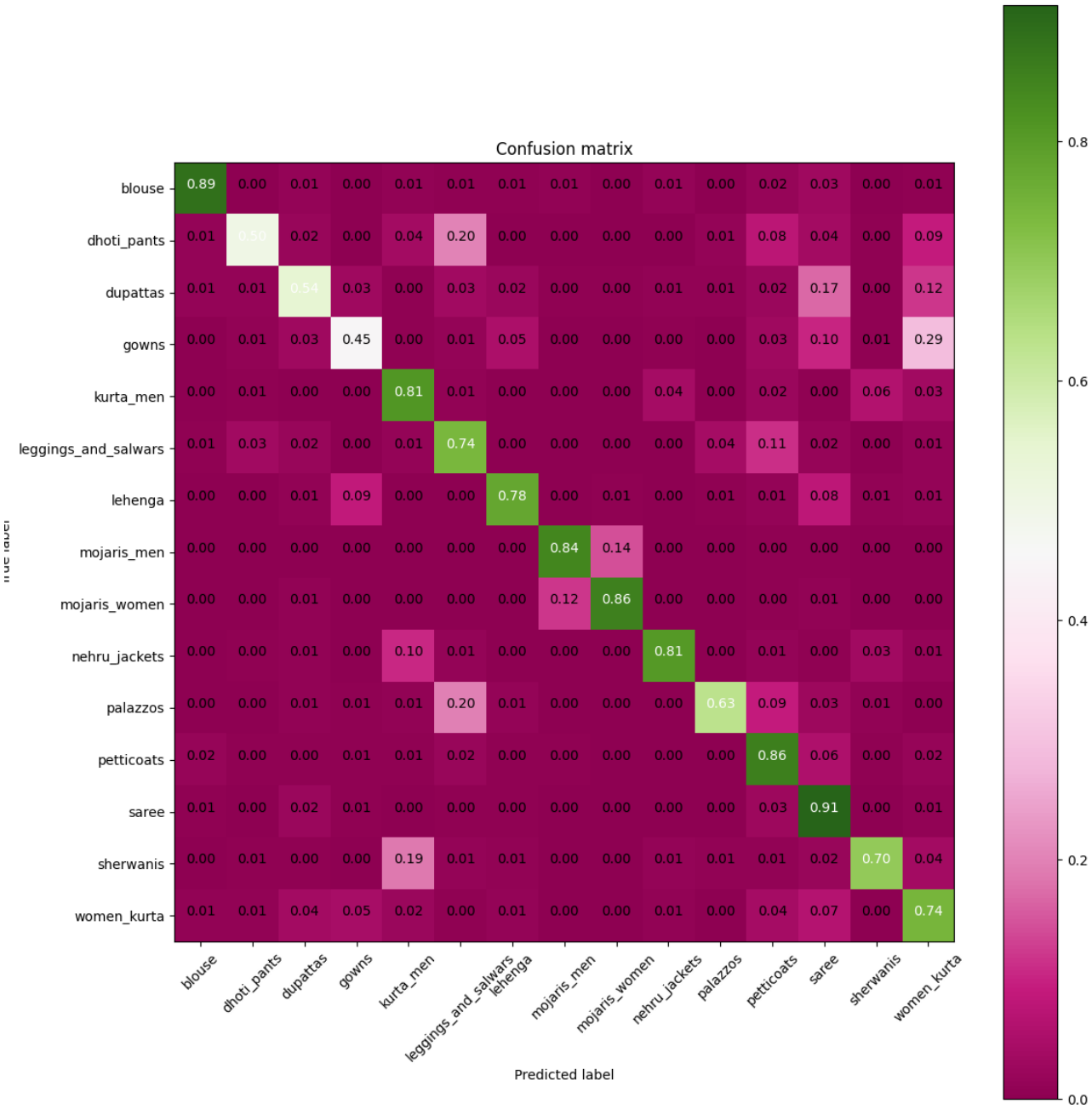
	precision	recall	f1-score	support
blouse	0.93	0.89	0.91	500
dhoti_pants	0.84	0.50	0.63	500
dupattas	0.74	0.54	0.62	500
gowns	0.71	0.45	0.55	500
kurta_men	0.67	0.81	0.73	500
leggings_and_salwars	0.59	0.74	0.66	500
lehenga	0.86	0.78	0.82	500
mojaris_men	0.86	0.84	0.85	500
mojaris_women	0.84	0.86	0.85	500
nehru_jackets	0.90	0.81	0.85	500
palazzos	0.88	0.63	0.74	500
petticoats	0.64	0.86	0.73	500
saree	0.60	0.91	0.72	500
sherwanis	0.84	0.70	0.76	500
women_kurta	0.53	0.74	0.62	500
accuracy			0.74	7500
macro avg	0.76	0.74	0.74	7500
weighted avg	0.76	0.74	0.74	7500

Model history





Confusion matrix



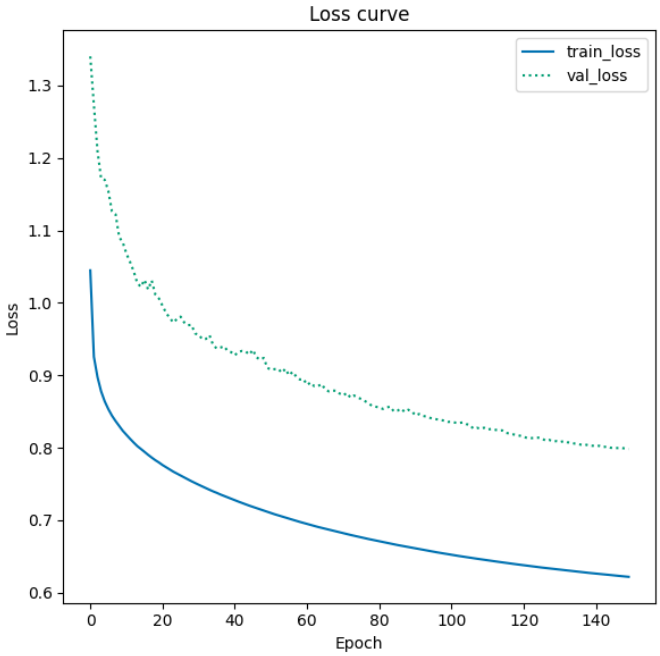
Second training step

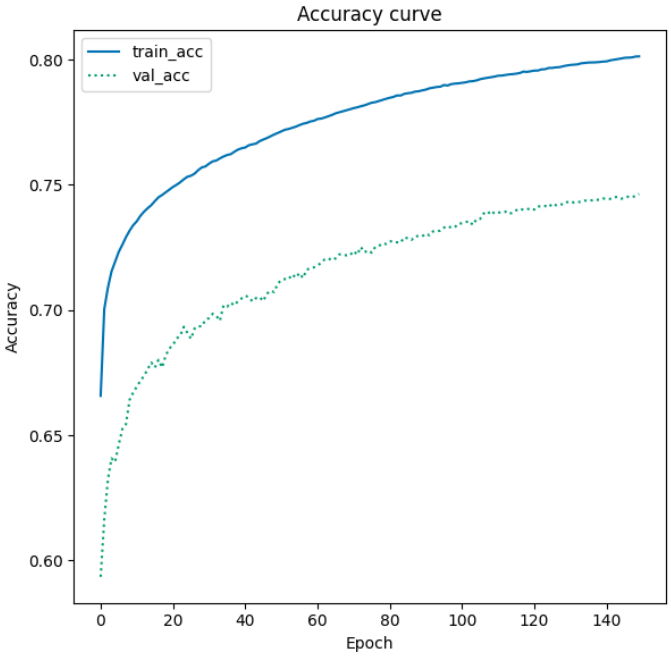
Classification report

```
precision    recall  f1-score   support

              blouse       0.90      0.94      0.92       500
         dhoti_pants       0.85      0.55      0.67       500
            dupattas       0.79      0.47      0.59       500
               gowns       0.69      0.41      0.52       500
           kurta_men       0.66      0.87      0.75       500
leggings_and_salwars       0.66      0.75      0.70       500
            lehenga       0.90      0.81      0.86       500
         mojaris_men       0.89      0.79      0.84       500
       mojaris_women       0.81      0.90      0.85       500
```

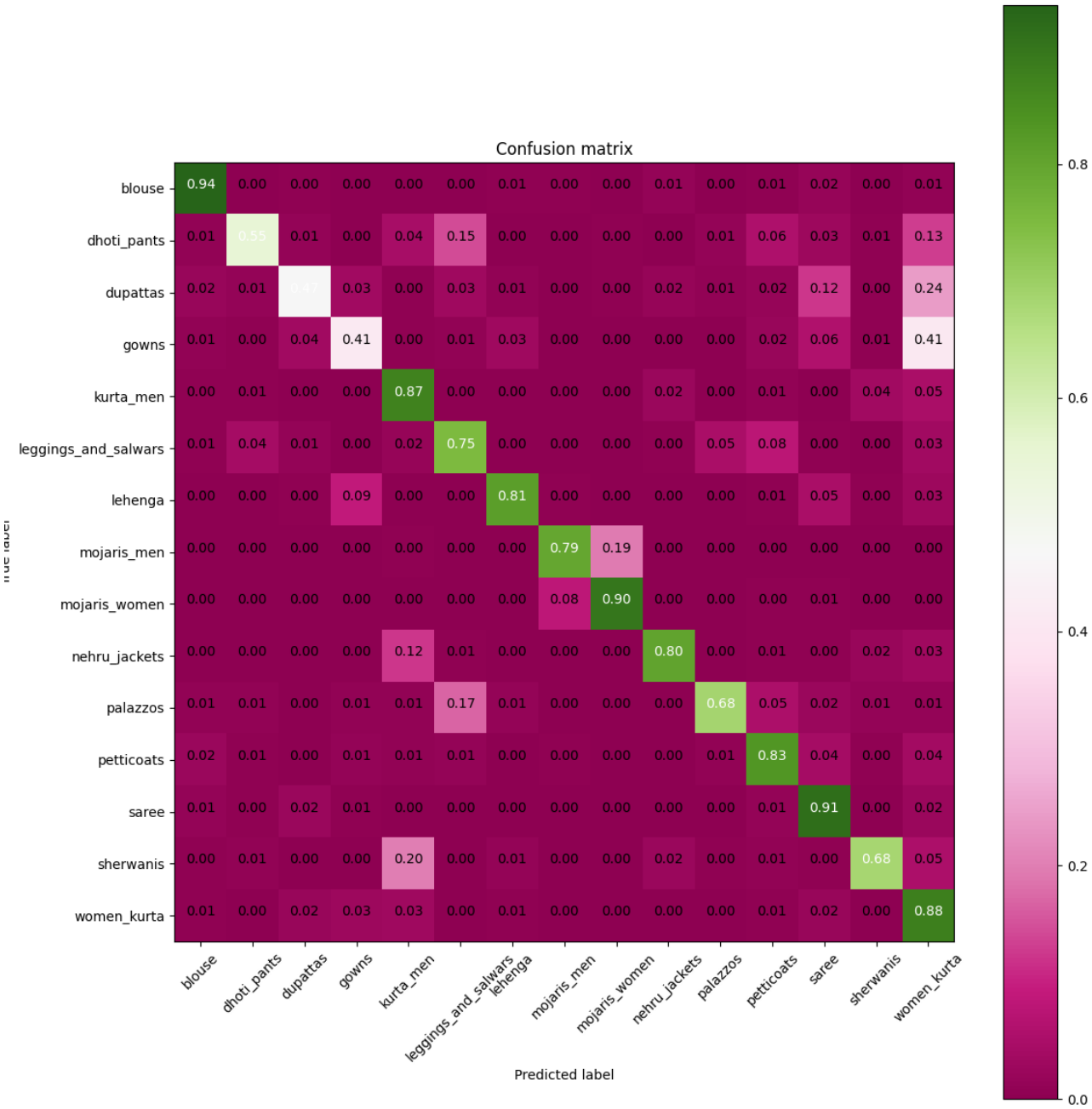
```
          nehru_jackets       0.90      0.80      0.85       500
               palazzos       0.89      0.68      0.77       500
             petticoats       0.75      0.83      0.79       500
                  saree       0.71      0.91      0.80       500
              sherwanis       0.87      0.68      0.77       500
            women_kurta       0.45      0.88      0.60       500

               accuracy                           0.75      7500
              macro avg       0.78      0.75      0.75      7500
           weighted avg       0.78      0.75      0.75      7500
```

Model history





Confusion matrix



Conclusion

The model was able to achieve an accuracy of 75% on the test set, which is not bad, but it could have been better. Notably, something must have gone wrong with the training continuation, although the fine-tuned model was loaded correctly, the model training did not seem to continue, but given more epochs it's possible that the accuracy could have been further improved.