Explanations

Helper doc that summarizes the work and contains conclusions.

Data

Python file ‘analyze\_data.py’ based on:

[ElephClass - Asian vs African Elephants Classifier (kaggle.com)](https://www.kaggle.com/code/nasruddinaz/elephclass-asian-vs-african-elephants-classifier)

How to distinguish African and Asian elephants:

[What's the Difference between African and Asian Elephants? | Thomson Safaris](https://thomsonsafaris.com/blog/difference-african-asian-elephant/)

What features may the network learn? Ears, skin, environment, head ‘dome’, …

Data characteristics:

1. Different image shapes – ranges from (100, 100) to (4992, 3328).
2. Train set contains less than 5% mislabeled images.
3. All images in the test set have the correct label.
4. Balanced distribution for the train and test sets.

Binary classification Task

best results achieved on this dataset (from Kaggle) using:

1. Machine Learning –

Extracting features using HoG (object detection method) and concatenate them to RGB values 🡪 reduce dimensions using PCA 🡪 apply SVM 🡪 72% accuracy.

[Image Classification using ML + Hog features(71%) (kaggle.com)](https://www.kaggle.com/code/yassineyahyaouii/image-classification-using-ml-hog-features-71)

2. Deep learning: CNN –

Using transfer learning with ImageNet pretrained networks.

The reference: [ElephClass - Asian vs African Elephants Classifier (kaggle.com)](https://www.kaggle.com/code/nasruddinaz/elephclass-asian-vs-african-elephants-classifier) shows best results (91% accuracy) for MobileNet with 20 epochs, batch size of 32 and implementation with tensor flow.

I separated train set to train and validation with ratio of 20%.

I tried different options (using torch) to find the best model:

Batch size = {32, 64, 128}

Epochs = {15, 20, 25}

Update all parameters or just final layer - Feature Extraction (FE) or Fine Tuning (FN)

Models = {MN\_v2, MN\_v3\_small, MN\_v3\_large, squeezenet, resnet50} (MN refers to MobileNet)

The results (for validation set) are summarized in the table below –

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Number | Batch size | Epochs | FT/FE | Pretrained model | Validation accuracy | Time [min.sec] | Use LoRA/DoRA | comments |
| 1 | 32 | 15 | FE | MN\_v2 |  |  | No |  |
| 2 | 32 | 15 | FE | MN\_v3\_small |  |  | No |  |
| 3 | 64 | 15 | FE | MN\_v3\_small |  |  | No |  |
| 4 | 64 | 25 | FE | MN\_v3\_large |  |  | No |  |
| 5 | 128 | 15 | FE | MN\_v3\_small |  |  | No |  |
| 6 | 128 | 15 | FE | squeezenet |  |  | No | Conv 1X1 instead of FC |
| 7 | 128 | 100 | FE | Resnet50 |  |  | No | Same network as for WikiArt |
| 8 | 128 | 20 | FE | Resnet50 |  |  | No | Same network as for WikiArt |
| 9 | 128 | 15 | FT | MN\_v3\_small |  |  | No | May take some time |
| 10 | 128 | 15 | FE | MN\_v3\_small |  |  | LoRA |  |
| 11 | 128 | 15 | FE | MN\_v3\_small |  |  | DoRA | Best model? |

I don’t have GPU on my computer, so I did it in colab’s GPU but ran out of time so need to complete cases #1-11.

2. Deep learning: ViT –

Using transfer learning with ImageNet pretrained network – ‘vit base patch16 224’.

I tried 2 different implementations:

1. Add this pretrained network to the code of CNN.

2. write separate code to run only ViT, based upon Tal’s references –

[Fine\_tuning\_the\_Vision\_Transformer\_on\_CIFAR\_10\_with\_the\_🤗\_Trainer.ipynb - Colaboratory (google.com)](https://colab.research.google.com/github/NielsRogge/Transformers-Tutorials/blob/master/VisionTransformer/Fine_tuning_the_Vision_Transformer_on_CIFAR_10_with_the_%F0%9F%A4%97_Trainer.ipynb)

[Fine-Tune ViT for Image Classification with 🤗 Transformers (huggingface.co)](https://huggingface.co/blog/fine-tune-vit)

Since this references used data from HuggingFace and not from Kaggle at first I converted our data to be the same type as HuggingFace’s data and then ran their code, which based on HuggingFace build-in implementation for ViT. Somehow it takes much more time to train (about 4 times than the time for option 1…)

I ran it for 2 or 3 epochs because I didn’t have GPU and it seems promising. I think that running it for 10 epochs will outperform CNN and get above 95% accuracy.