Nashra Amaan(MA23C024) DA6401 - Assignment 2

Learn how to use CNNs: train from scratch and finetune a pre-trained model as it is.

Nashra

Instructions

- The goal of this assignment is twofold: (i) train a CNN model from scratch and how to tune the hyperparameters and visualize filters (ii) finetune a pre-train just as you would do in many real-world applications
- Discussions with other students is encouraged.
- You must use `Python` for your implementation.
- You can use any and all packages from `PyTorch`, `Torchvision` or PyTorch-Lig OTHER DL library such as `TensorFlow` or `Keras` is allowed. Please confirm w before using any new external library. BTW, you may want to explore PyTorch as it includes `fp16` mixed-precision training, `wandb` integration and many oth boxes eliminating the need for boiler-plate code. Also, do look out for PyTorch
- You can run the code in a jupyter notebook on colab by enabling GPUs.
- You have to generate the report in the format shown below using `wandb.ai`. Y

start by cloning this report using the clone option above. Most of the plots tha asked for below can be (automatically) generated using the APIs provided by

- You also need to provide a link to your GitHub code as shown below. Follow g software engineering practices and set up a GitHub repo for the project on Da do not write all code on your local machine and push everything to GitHub or day. The commits in GitHub should reflect how the code has evolved during tl of the assignment.
- You have to check Moodle regularly for updates regarding the assignment.

Problem Statement

In Part A and Part B of this assignment you will build and experiment with CNN b image classifiers using a subset of the iNaturalist dataset.

Part A: Training from scratch

Question 1 (5 Marks)

Build a small CNN model consisting of 5 convolution layers. Each convolution lay be followed by an activation and a max-pooling layer.

After 5 such conv-activation-maxpool blocks, you should have one dense layer fol the output layer containing 10 neurons (1 for each of the 10 classes). The input la should be compatible with the images in the iNaturalist dataset dataset.

The code should be flexible such that the number of filters, size of filters, and act function of the convolution layers and dense layers can be changed. You should a to change the number of neurons in the dense layer.

- What is the total number of computations done by your network? (assume m each layer of size $\mathrm{k} \times \mathrm{k}$ and n neurons in the dense layer)
- What is the total number of parameters in your network? (assume m filters ir layer of size $\mathrm{k} \times \mathrm{k}$ and n neurons in the dense layer)

Question 2 (15 Marks)

You will now train your model using the iNaturalist dataset. The zip file contains a test folder. Set aside 20% of the training data, as validation data, for hyperparai tuning. Make sure each class is equally represented in the validation data. **Do not test data for hyperparameter tuning.**

Using the sweep feature in wandb find the best hyperparameter configuration. H some suggestions but you are free to decide which hyperparameters you want to

- number of filters in each layer : 32, 64, ...
- activation function for the conv layers: ReLU, GELU, SiLU, Mish, ...
- filter organisation: same number of filters in all layers, doubling in each subs layer, halving in each subsequent layer, etc
- data augmentation: Yes, No
- batch normalisation: Yes, No
- dropout: 0.2, 0.3 (BTW, where will you add dropout? You should read up a bit

Based on your sweep please paste the following plots which are automatically ger wandb:

- accuracy v/s created plot (I would like to see the number of experiments you r the best configuration).
- parallel co-ordinates plot
- correlation summary table (to see the correlation of each hyperparameter wit accuracy)

Also, write down the hyperparameters and their values that you sweeped over. Sn strategies to reduce the number of runs while still achieving a high accuracy wou appreciated. Write down any unique strategy that you tried.

Question 3 (15 Marks)

Based on the above plots write down some insightful observations. For example,

- adding more filters in the initial layers is better
- Using bigger filters in initial layers and smaller filters in latter layers is better
- ... ...

(Note: I don't know if any of the above statements is true. I just wrote some rando comments that came to my mind)

Question 4 (5 Marks)

You will now apply your best model on the test data (You shouldn't have used test

far. All the above experiments should have been done using train and validation c

- Use the best model from your sweep and report the accuracy on the test set.
- Provide a 10×3 grid containing sample images from the test data and predi made by your best model (more marks for presenting this grid creatively).
- **(UNGRADED, OPTIONAL)** Visualise all the filters in the first layer of your bes for a random image from the test set. If there are 64 filters in the first layer pl an 8×8 grid.
- **(UNGRADED, OPTIONAL)** Apply guided back-propagation on any 10 neurons CONV5 layer and plot the images which excite this neuron. The idea again is t interesting patterns which excite some neurons. You will draw a 10×1 grid one image for each of the 10 neurons.

Question 5 (10 Marks)

Paste a link to your github code for Part A

Example: https://github.com/NashraDocs/MA23C024_ASS2_DL/blob/main/partA_ass_DL_ma23c024.ipynb;

- We will check for coding style, clarity in using functions and a `README` file wit instructions on training and evaluating the model (the 10 marks will be based
- We will also run a plagiarism check to ensure that the code is not copied (0 m assignment if we find that the code is plagiarised).
- We will also check if the training and test data has been split properly and ran You will get 0 marks on the assignment if we find any cheating (e.g., adding te training data) to get higher accuracy.

Part B : Fine-tuning a pre-trained model

Question 1 (5 Marks)

In most DL applications, instead of training a model from scratch, you would use pre-trained on a similar/related task/dataset. From `torchvision`, you can load **AN**

model (`GoogLeNet`, `InceptionV3`, `ResNet50`, `VGG`, `EfficientNetV2`, `VisionTransf` pre-trained on the ImageNet dataset. Given that ImageNet also contains many an images, it stands to reason that using a model pre-trained on ImageNet maybe he this task.

You will load a pre-trained model and then fine-tune it using the naturalist data tl used in the previous question. Simply put, instead of randomly initialising the we network you will use the weights resulting from training the model on the Image (`torchvision` directly provides these weights). Please answer the following quest

- The dimensions of the images in your data may not be the same as that in the data. How will you address this?
- ImageNet has 1000 classes and hence the last layer of the pre-trained model have 1000 nodes. However, the naturalist dataset has only 10 classes. How w address this?

(Note: This question is only to check the implementation. The subsequent questi talk about how exactly you will do the fine-tuning.)

Question 2 (5 Marks)

You will notice that `GoogLeNet`, `InceptionV3`, `ResNet50`, `VGG`, `EfficientNetV2`, `VisionTransformer` are very huge models as compared to the simple model that y implemented in Part A. Even fine-tuning on a small training data may be very exp What is a common trick used to keep the training tractable (you will have to read this)? Try different variants of this trick and fine-tune the model using the iNatur dataset. For example, '___'ing all layers except the last layer, '___'ing upto k layer '___'ing the rest. Read up on pre-training and fine-tuning to understand what exa terms mean.

Write down the at least 3 different strategies that you tried (simple bullet points v fine).

Question 3 (10 Marks)

Now fine-tune the model using **ANY ONE** of the listed strategies that you discuss Based on these experiments write down some insightful inferences comparing tr from scratch and fine-tuning a large pre-trained model.

Question 4 (10 Marks)

Paste a link to your GitHub code for Part B

Example: https://github.com/NashraDocs/MA23C024_ASS2_DL/blob/main/partB_assignmnet_DL_ma23c024.ipynb

Follow the same instructions as in Question 5 of Part A.

(UNGRADED, OPTIONAL) Part C : Using a trained model as it is

Question 1 (0 Marks)

Object detection is the task of identifying objects (such as cars, trees, people, anir images. Over the past 6 years, there has been tremendous progress in object dete very fast and accurate models available today. In this question you will use a pre-t YoloV3 model and use it in an application of your choice. Here is a cool demo of Y (click on the image to see the demo on youtube).

Go crazy and think of a cool application in which you can use object detection (al
mates of monkeys loitering outside the lab, detecting cycles in the CRC corridor, .

Make a similar demo video of your application, upload it on youtube and paste a l
(similar to the demo I have pasted above).

Also note that I do not expect you to train any model here but just use an existing
it is. However, if you want to fine-tune the model on some application-specific da
you are free to do that (it is entirely up to you).

Notice that for this question I am not asking you to provide a GitHub link to your
giving you a free hand to take existing code and tweak it for your application. Fee
paste the link of your code here nonetheless (if you want).

Example: https://github.com/<user-id>/da6401_assignment2/partC

Self Declaration

I, Nashra Amaan (Roll no: MA23C024), swear on my honour that I have written th
the report by myself and have not copied it from the internet or other students.