Assignment 2

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

Mitesh Khapra

Instructions

- The goal of this assignment is threefold: (i) train a CNN model from scratch and learn how to tune the hyperparameters and visualise filters (ii) finetune a pre-trained model just as you would do in many real world applications (iii) use an existing pre-trained model for a cool application.
- We strongly recommend that you work on this assignment in a team of size 2. Both the members of the team are expected to work together (in a subsequent viva both members will be expected to answer questions, explain the code, etc).
- Collaborations and discussions with other groups are strictly prohibited.
- You must use Python (numpy and pandas) for your implementation.
- You can use any and all packages from keras, pytorch, tensorflow
- You can run the code in a jupyter notebook on colab by enabling GPUs.
- You have to generate the report in the same format as shown below using wandb.ai. You can start by cloning this report using the clone option above. Most of the plots that we have asked for below can be (automatically) generated using the apis provided by wandb.ai
- You also need to provide a link to your github code as shown below. Follow good software engineering practices and set up a

github repo for the project on Day 1. Please do not write all code on your local machine and push everything to github on the last day. The commits in github should reflect how the code has evolved during the course 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 based image classifiers using a subset of the iNaturalist dataset. In Part C you will take a pre-trained object detection model and use it for a novel application.

Part A: Training from scratch

Question 1 (5 Marks)

Build a small CNN model consisting of 5 convolution layers. Each convolution layer would be followed by a ReLU activation and a max pooling layer. Here is sample code for building one such conv-relu-maxpool block in keras.

```
model = Sequential()
model.add(Conv2D(16, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

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

The code should be flexible such that the number of filters, size of filters and activation function in each layer can be changed. You should also be able to change the number of neurons in the dense layer.

(a) What is the total number of computations done by your network? (assume m filters in each layer of size $k \times k$ and n neurons in the

dense layer)

(b) What is the total number of parameters in your network? (assume m filters in each layer of size $k \times k$ and n neurons in the dense layer)

Question 2 (10 Marks)

You will now train your model using the iNaturalist dataset. The zip file contains a train and a test folder. Set aside 10% of the training data for hyperparameter tuning. Make sure each class is equally represented in the validation data. Do not use the test data for hyperparameter tuning.

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

- number of filters in each layer : 32, 64, ...
- filter organisation: same number of filters in all layer, doubling in each subsequent layer, halving in each subsequent layer, etc
- data augmentation (easy to do in keras): Yes, No
- dropout: 20%, 30% (btw, where will you add dropout? you should read up a bit on this)
- batch normalisation: Yes, No

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

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

Also write down the hyperparameters and their values that you sweeped over. Smart strategies to reduce the number of runs while still achieving a high accuracy would be 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 random 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 data so far. All the above experiments should have been done using train and val data only).

(a) Use the best model from your sweep and report the accuracy on the test set.

(b) Provide a 10 x 3 grid containing sample images from the test data and predictions made by your best model (more marks for presenting this grid creatively).

(c) Visualise all the filters in the first layer of your best model for a random image from the test set. If there are 64 filters in the first layer plot them in an 8 x 8 grid.

Question 5 (10 Marks)

Apply guided back propagation on any 10 neurons in the CONV5 layer and plot the images which excite this neuron. The idea again is to discover interesting patterns which excite some neurons. You will draw a 10 x 1 grid below with one image for each of the 10 neurons.

Question 6 (10 Marks)

Paste a link to your github code for Part A

Example: https://github.com/<user-id>/cs6910_assignment2/partA;

- We will check for coding style, clarity in using functions and a README file with clear instructions on training and evaluating the model (the 10 marks will be based on this).
- We will also run a plagiarism check to ensure that the code is not copied (0 marks in the assignment if we find that the code is plagiarised).
- We will check the number of commits made by the two team members and then give marks accordingly. For example, if we see 70% of the commits were made by one team member then that member will get more marks in the assignment (**note that this contribution will decide the marks split for the entire assignment and not just this question**).
- We will also check if the training and test data has been split properly and randomly. You will get 0 marks on the assignment if we find any cheating (e.g., adding test data to 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 a model pre-trained on a similar/related task/dataset. From keras, you can load any model (InceptionV3, InceptionResNetV2, ResNet50, Xception, etc) pre-trained on the ImageNet dataset. Given that ImageNet also contains many animal images, it stands to reason that using a model pre-trained on ImageNet maybe helpful for this task.

You will load a pre-trained model and then fine-tune it using the naturalist data that you used in the previous question. Simply put, instead of randomly initialising the weigths of a network you will use the weights resulting from training the model on the ImageNet data (keras directly provides these weights). Please answer the following questions:

(a) The dimensions of the images in your data may not be the same as that in the ImageNet data. How will you address this?

(b) ImageNet has 1000 classes and hence the last layer of the pre-trained model would have 1000 nodes. However, the naturalist dataset has only 10 classes. How will you address this?

Your implementation should be modular so that it allows to swap in any model (InceptionV3, InceptionResNetV2, ResNet50, Xception).

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

Question 2 (5 Marks)

You will notice that InceptionV3, InceptionResNetV2, ResNet50, Xception are very huge models as compared to the simple model that you implemented in Part A. Even fine-tuning on a small training data may be very expensive. What is a common trick used to keep the training tractable (you will have to read up a bit on this)? Try different variants of this trick and fine-tune the model using the iNaturalist dataset. For example, '___'ing all layers except the last layer, '___'ing upto k layers and '___'ing the rest. Read up on pre-training and fine-tuning to understand what exactly these terms mean.

Write down the different strategies that you tried (simple bullet points would be fine).

Question 3 (15 Marks)

Now finetune the model using different strategies that you discussed above and different hyperparameter choices. Based on these experiments write down some insightful inferences (once again you will find the sweep function to be useful to plot and compare different choices).

Here are some examples of inferences that you can draw:

- Using a huge pre-trained network works better than training a smaller network from scratch (as you did in Part A)
- InceptionV3 works better for this task than ResNet50

- Using a pre-trained model, leads to faster convergence as opposed to training a model from scratch
- ... .

(Note: I don't know if any of the above statements is true. I just wrote some random comments that came to my mind) Of course, provide evidence (in the form of plots) for each inference.

Of course, provide appropriate plots for the above inferences (mostly automatically generated by wandb). The more insightful and thorough your inferences and the better the supporting evidence (in terms of plots), the more you will score in this question.

Question 4 (10 Marks)
Paste a link to your github code for Part A

Example: https://github.com/<user-id>/cs6910_assignment2/partB

Follow the same instructions as in Question 6 of Part A.

Part C : Using a pre-trained model as it is

Question 1 (15 Marks)
Object detection is the task of identifying objects (such as cars, trees, people, animals) in images. Over the past 6 years, there has been tremendous progress in object detection with very fast and accurate models available today. In this question you will use a pre-trained YoloV3 model and use it in an application of your choice. Here is a cool demo of YoloV2 (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 (alerting lab mates of monkeys loitering outside the lab, detecting cycles in the CRC corridor, ...). More marks if you come up with an application which has social relevance.

Make a similar demo video of your application, upload it on youtube and paste a link below (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 model as it is. However, if you want to fine-tune the model on some application-specific data then 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 code. I am giving you a free hand to take existing code and tweak it for your application. Feel free to paste the link of your code here nonetheless (if you want).

Example: https://github.com/<user-id>/cs6910_assignment2/partC

Self Declaration

List down the contributions of the two team members:

For example,

CS20Mzzzz: (70% contribution)

- ...
- ...
- ...
- ...
- ...
- ...

CS20Myyyy: (30% contribution)

- ...
- ...
- ...
- ...

We, Name_XXX and Name_ZZZ, swear on our honour that the above declaration is correct.

Note: Your marks in the assignment will be in proportion to the above declaration. Honesty will be rewarded (Help is always given in CS6910 to those who deserve it!).

Created with ❤️ on Weights & Biases.

https://wandb.ai/miteshk/assignments/reports/Assignment-2--Vmlldzo0NjA1MTU
