UDACITY

‹ Return to Classroom

Landmark Classification & Tagging for Social Media

REVIEW

HISTORY

Requires Changes

6 specifications require changes

Hi,

Almost there, you really got this going, and it almost was a perfect submission! It showed that you understood each concept in detail and you precisely answered each question at desired location, very good! ⚡

Now the test accuracy in part I must be increased by at least 1% more, since the minimal requirements is 20%, further cells in part II must be run and suggestion for model improvement must be stated, please see rubrics below.

To continue learning I recommend to work through chapters in the deeplearningbook.

Good luck for your resubmission you will make it shortly!

Files Submitted

The submission includes the required notebook file and HTML file. When the HTML file is created, all the code cells in the notebook need to have been run so that reviewers can see the final implementation and

output.

The submission includes the notebook and the report in form of a .html file, awesome 👍

See here for further elaborations on why this is important for cross-platform specific readability/functionality of the project.

Step 1: Create a CNN to Classify Landmarks (from Scratch)

The submission randomly splits the images at `landmark_images/train` into train and validation sets. The submission then creates a data loader for the created train set, a data loader for the created validation set, and a data loader for the images at `landmark_images/test`.

You have performed a three-way data-split randomly in accordance with the script, making use of SubsetRandomSampler(), this is great work! 👏

Now SubsetRandomSampler() is the default method to perform the data-split and always applicable, a brand new tailormade pytorch alternative, when working with the newest versions of torch and torchvision in an environment, was torch.utils.data.random_split(), which you can consider utilise in future projects!

Alternatively an external library splitfolders can be used , which is great, as like so you could easily augment the training data exclusively! 💡

The default three-way split is 60-20-20 (train-valid-test), when the dataset is small the test-data can be reduced and added to the train-data, but the valid-data should be 20% or more at best ✅

Answer describes each step of the image preprocessing and augmentation. Augmentation (cropping, rotating, etc.) is not a requirement.

Excellent 👏, you have precisely described the steps you have taken to load the data, including resizing the images to speed up training, you could also add augmentations on the training set to enlarge the data, to further explore this venue you can have a look at this repository for what is further possible in regards to augmentations, and see here torchvision native augmentations which can freely be chosen from!

The submission displays at least 5 images from the train data loader, and labels each image with its class name (e.g., "Golden Gate Bridge").

Well done, you have displayed a reasonable number of images like a good citizen, meeting requirements here 💯

You can further costumize this by changing e.g. the figure size if the image labels overlap, great that you left the class-number in the label, for orientation!

The submission chooses appropriate loss and optimization functions for this classification task.

For the CNN network you have perfectly chosen SGD as the optimiser, which is expected to decrease the losses a bit better when it comes to a larger number of epochs, again it should be experimented with the learning rate here, to see how training progresses!

Great work choosing CrossEntropyLoss() as the loss function, this is 2 steps in 1: NLLLoss() + LogSoftmax() 👏

The submission specifies a CNN architecture.

The CNN architecture is built neatly with a good number convolutional layers, including dropout and the needed flattening step in the forward pass.

As a sidenote, newer papers have shown to achieve better results and more effective training with smaller kernels (kernel_size=3), as you did, and using a smaller kernel may allow better feature detection 🔎

Further the relatively large images are better accomodated with 4-6 convolutional layers, increasing the size of the network like this will significantly increase the accuracy 👌

Answer describes the reasoning behind the selection of layer types.

Well done, you have precisely described the steps you have taken to built the model, choosing relu() activation as the default, in all except the last layer, which is activated by LogSoftmax()(inside CrossEntropy()), since we are performing a **multiclass classification**. ✨

The submission implements an algorithm to train a model for a number of epochs and save the "best" result.

Great job, training algorithm correctly uses train and validation data, and the epochs which improved the validation loss are saved as the model, kudos to you! 👌

Also great work with this function and formatting, feel free to explore the new improved f-string formatting functionalities in python!

The submission implements a custom weight initialization function that modifies all the weights of the

model. The submission does not cause the training loss or validation loss to explode to `nan`.

A general rule normal distribution was chosen and the models weights did not exploded during the run, no `nan` losses occurred ✅

The trained model attains at least 20% accuracy on the test set.

❗ **[Required]** Almost there, you achieved 19% accuracy, however the rubric asks to achieve at least 20% accuracy, since. few rubrics below must be changed as well, please retrain for resubmission.

This really is not too hard, please optimise by tweaking hyperparameters, such as:

- optimizer
- hidden nodes and number of hidden layers
- learning rate
- epochs

Step 2: Create a CNN to Classify Landmarks (using Transfer Learning)

The submission specifies a model architecture that uses part of a pre-trained model.

Well done, vgg16 was chosen, and it was opted for completely freezing the parameters, due the relatively small size of our dataset. This way we fully profit of the trained features and maps of the pretrained models weights and biases trained on the ImageNet Dataset! 💎

The submission details why the chosen architecture is suitable for this classification task.

You are meeting requirements here ✔ another reason for why vgg was suitable would be that it consolidated at a standard as being on of the winners of ImageNet competition winners.

The submission uses model checkpointing to train the model and saves the model weights with the best validation loss.

Great work with this function and formatting, feel free to explore the new improved f-string formatting functionalities in python!

Accuracy on the test set is 60% or greater.

❗ **[Required]** Code cell that tests model on testdata is not run, please run and achieve at least 60%

Step 3: Write Your Landmark Prediction Algorithm

The submission implements functionality to use the transfer learned CNN from Step 2 to predict top k landmarks. The returned predictions are the names of the landmarks (e.g., "Golden Gate Bridge").

❗ **[Required]** Please run and show functionality of this cell.

❕ **[Suggestion]** As a recommendation: In your predict functiony you have unsqueezed the 1st dimension, since you only loaded a single image, and the model expects a 4-dimensional tensor with the batch size being the first dimension, great job! 👏

While just applying the transforms is ok, since this is the most concise way of doing this to solve the problem, the better way is to use PIL functionalities.
The reason why manual transformations are more effective code and the light weight solution in this instance, is that we only want to prepare 1 image at a time for a prediction, when using pytorch´s generic dataloader you essentially built a file structure around this image, which needs more computing, hence the manual transformations are the more effective solution, in regards to computational cost: BigO

The submission displays a given image and uses the functionality in "Write Your Algorithm, Part 1" to predict the top 3 landmarks.

❗ [Required] Please run and show functionality of this cell.

The submission tests at least 4 images.

❗ [Required] Please run and show functionality of this cell.

Submission provides at least three possible points of improvement for the classification algorithm.

❗ **[Required]** At least three suggestions must be given for model optimisation which can data-, model-, and training-centric, further it should be considered to tweak hyperparameters, which are to be distinguished as 🔍 :

1. Optimizer hyperparameters: learning rate, batch size, epochs as mentioned

2. Model hyperparameters: hidden units and layers, model specific parameters
3. Datacentric hyperparameters: Data label quality, augmentations, batch_size

Please add these three points of improvement.

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review

START