Landmark Classification & Tagging for Social Media

REVIEW | CODE REVIEW | HISTORY

Meets Specifications

Great job! A really well-rounded submission as you fixed the things which needed a bit of rework.

- Do think of adding more features beyond this model like converting the text you get as a prediction to geocode and then geotagging the image which would make for a cool map visualization of user-uploaded images if you ever think of building a social media platform :D
- Also there was a Kaggle challenge as well by Google which was done for Landmark classification in 2020 so do check it out if you want to play with bigger dataset. (~100GB)
  https://www.kaggle.com/c/landmark-recognition-2020

All the best for your AI journey!

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.

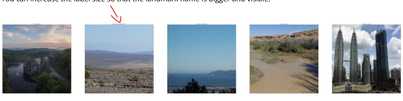
Step 1: Create a CNN to Classify Landmarks (from Scratch)

✓ 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`.

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

✓ 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").

- You had to display at least 5 images so done really well.
- You can play around with figsize to have larger/smaller images if you want and have a bit less spacing and having subplots balanced.
- You can increase the label size so that the landmark name is bigger and visible.



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

Tip - You can also try Adabound which is a blend of Adam and SGD
https://medium.com/syncedreview/iclr-2019-fast-as-adam-good-as-sgd-new-optimizer-has-both-78e37e8f9a34

✓ The submission specifies a CNN architecture.

✓ Answer describes the reasoning behind the selection of layer types.

- You can also try and imitate smaller architectures like LeNET like structures for training scratch models as well.
- Including batch norm can also be done in general during these tasks but if you check and find that the dataset is imbalanced then only it gives good results. Here you can probably avoid using batch norm as well because it's a balanced dataset.
- You can visualize each layer and filter with what shape they are learning at each layer to make it more intuitive to builds CNNs from scratch.
  https://cs231n.github.io/understanding-cnn/
- You can reduce/remove dropout while training from scratch as the model would learn slowly because of this for FC layers are the ones that tune the weights to give to features obtained from CNN layers to align to classes so having this lesser or no dropout is suggested initially so do keep that in mind. If you see drastic overfitting then try to first check your data and architecture while training from scratch.

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

- You can also use lr_scheduler which automatically decays your learning rate if when you want to train for 50+ epochs.

  https://stackoverflow.com/questions/59017023/pytorch-learning-rate-scheduler
- You can always plot the losses against epochs to see if the training set goes into overfitting/underfitting mode or not so do make decisions from it initially by just training it for 5-10 epochs.
- Also, we can implement early stopping as well if you feel some epsilon-delta (1e^-04) change is not happening consistently for n epochs in validation loss. https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/

✓ 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`.

- It has been seen with certain techniques like xavier and kaiming significant improvements are gained in results so you can train for 10 epoch and see which one is working well and if the loss is coming as Nan values then you need to debug your training loop above.
- https://paperswithcode.com/method/he-initialization
- https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79

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

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.

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

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

✓ Accuracy on the test set is 60% or greater.

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").

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

✓ The submission tests at least 4 images.

- Tested well on images.
- Do try to test more monuments given from the test dataset and also other ones outside the dataset to see how close they get to predict. As it only contains 50- classes, there would be interesting outputs for monuments/locations which are not in the given dataset.
- You can also try and check for non-dataset monument images online of the classes which we have trained and see different predictions which the model makes to similar looking landmarks.

```
suggest_locations('images/prag.jpg')
```

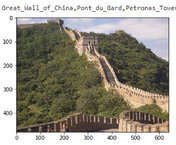
```
Is this is of the

Prague_Astronomical_Clock,Petronas_Towers,Ljubljana_Castle
```

```
suggest_locations('images/china.jpg')
```

```
Is this is of the

Great_Wall_of_China,Pont_du_Gard,Petronas_Towers
```



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

```
I can increase the epochs (30-40.but not too much)
...increase the conv layer not three maybe 6 or 7.
...increase size 224, and can add more transforming line (flipping,cropping part).
```

You can definitely tinker around the params, augmentations that come under the model-centric approach.

Just to add some points related to data ->

- There maybe class imbalance as well so check if that is the case. If yes then maybe using SMOTE to oversample minority classes might help
- More data is needed as you rightly mentioned for different classes as there are only 50 classes and the number of samples are also low so always try to think also in terms of a data-centric approach while starting out the project as fewer data will take you to a certain accuracy only but working on data might help you push boundaries even more.
- If you are interested, there was a data centric competition a few months back so you can read about the solutions and approaches and understand while keeping the model fixed you can work more on data.

DOWNLOAD PROJECT

RETURN TO PATH