UDACITY

‹ Return to Classroom

Landmark Classification & Tagging for Social Media

REVIEW

HISTORY

Meets Specifications

Awesome job on the project! Great work getting high accuracies on your from-scratch and transfer learning model! To make the net run faster, we could possibly use some compression. Another similar way of reducing the net's size is a new version of pruning weights is described in this MIT paper described and linked here.

Something else interesting is looking at how you can reduce memory usage in Pytorch. Basically by using auto-mixed precision and checkpointing, we can reduce the memory footprint by 5x in some cases.

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.

Yes, the correct files are there.

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`.

Nice work! You can normalize the images with the imagenet normalization factors. This typically improves performance of neural nets.

You can also normalize images by calculating the means dynamically with something like `np.mean(train_set.train_data, axis=(0,1,2))/255)` : https://discuss.pytorch.org/t/normalization-in-the-mnist-example/457/12

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").

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

Good work! Many optimizers work well, but with proper hyperparameter tuning, the `adam` optimizer can work even better: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
However, in this case, it seems SGD with momentum works well, and adam with the default hyperparameters doesn't work so well.
Here is a comparison of some different optimizers: http://ruder.io/optimizing-gradient-descent/

The submission specifies a CNN architecture.

Great model! Usually, I use batch normalization because it trains faster and performs better (higher accuracy). I also tend to use ELU instead of ReLU since it can achieve better accuracy and shorter training times (more info here), but there are also other, possibly better and newer activations like swish. You can also use a Bayesian hyperparameter optimizer like BoTorch to create a better net.

Answer describes the reasoning behind the selection of layer types.

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

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`.

Nice work using Kaiming initialization! You might try another initialization method like Xavier, which can improve performance.

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

27% is good!

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.

These nets can work well because they were trained on classifying many images; for example, this is a list of classes for VGG-16
http://image-net.org/challenges/LSVRC/2014/browse-synsets

Also, the deeper layers find more complex patterns: https://youtu.be/McgxRxi2Jqo?t=96 (around 8 minutes in the video is where he directly shows this.)

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.

71% is good!

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.

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

DOWNLOAD PROJECT

RETURN TO PATH