UDACITY

‹ Return to Classroom

Train a Character Recognition System with MNIST

REVIEW

HISTORY

Meets Specifications

Great work! You have worked end to end on a DL model in PyTorch, now you would have good knowledge of data loading, model creation, hyperparameter tuning, training, and testing. These steps are critical going forward in Deep Learning!

Practice with more datasets and different models to increase your confidence for the model!

All the best!

Section 1: Data Loading and Exploration

Data is preprocessed and converted to a tensor, either using the .ToTensor() transform from `torchvision.transforms` or simply manually with torch.Tensor.

Awesome work!
The validation set is crucial in ensuring that we understand that our model is not overfitting the training data, as the validation is separate from the training data this will give us an estimation of whether we need to tune our model, hyperparameters or make any additional data processing.

A `DataLoader` object for both train and test sets has been created using the train and test sets loaded from `torchvision`.

Great Job!
You have worked thoroughly on the dataset and made the necessary preprocessing steps! Way to go!

- Creating validation split was optional so do look at that directly with dataloaders. There are many ways in which you can specify the split by using different methods for eg. using random_split is a good choice. What you are using here as validation set is actually the test set.
- Subsetrandomsampler and using 0.x multiplied by data length gives you more control when you decide to split by % (80-20 split) instead of hardcoding values. The validation set is crucial in ensuring that we understand that our model is not overfitting the training data, as the validation is separate from the training data this will give us an estimation of whether we need to tune our model, hyperparameters or make any additional data processing.
https://stackoverflow.com/questions/50544730/how-do-i-split-a-custom-dataset-into-training-and-test-datasets

Notebook contains code which shows the size and shape of the training and test data.

The provided function or some other method (e.g. plt.imshow) is used to print one or more images from the dataset

The submission contains a justification of necessary preprocessing steps (e.g. flattening, converting to tensor, normalization) in a comment or (preferably) a markdown block.

Good Work! Some more references for you to refer to.
https://www.datanami.com/2019/11/08/why-you-need-data-transformation-in-machine-learning/
https://stats.stackexchange.com/questions/211436/why-normalize-images-by-subtracting-datasets-image-mean-instead-of-the-current

Section 2: Model Design and Training

A `Model` or `Sequential` class is created with at least two hidden layers and implements a `forward` method that outputs a prediction probability for each of the 10 classes using softmax.

Simple yet effective model!

A loss function that works for classification tasks is specified.

Great Work!

Any optimizer from `torch.optim` is used to minimize the loss function.

Good choice using Adam optimiser, it could further be experimented with setting additional parameters, as a suggestion, beta2 is supposed to be set close to 1 for most effective training in computer vision problems, see further elaboration. 💡

Section 3: Model Testing and Evaluation

The test `DataLoader` is used to get predictions from the neural network and compare the predictions to the true labels.

Good Work!

Hyperparameters are modified to attempt to improve accuracy and the model achieves at least 90% classification accuracy.

Wonderful job getting over 95% accuracy here!

The `torch.save()` function is used to save the weights of the trained model.

Good Work saving both the models!

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START