UDACITY

< Return to "Computer Vision Nanodegree" in the classroom

Facial Keypoint Detection

REVIEW

CODE REVIEW 7

HISTORY

Meets Specifications

You've done a fantastic job completing the Facial Keypoints Detection Project!
The Facial Keypoints Detection is a well known machine learning challenge. You might want to check the resources below to
further improve your model:

1. Berkeley team's attempt at beating the Facial Keypoints Detection Kaggle competition
2. Facial Keypoints Detection using the Inception model

Keep up the great work and keep learning!

Files Submitted

The submission includes models.py and the following Jupyter notebooks, where all questions have been answered and training and visualization cells have been executed:
2. Define the Network Architecture.ipynb, and
3. Facial Keypoint Detection, Complete Pipeline.ipynb.
Other files may be included, but are not necessary for grading purposes. Note that all your files will be zipped and uploaded should you submit via the provided workspace.

Well done submitting all of the required files!

`models.py`

Define a convolutional neural network with at least one convolutional layer, i.e. `self.conv1 = nn.Conv2d(1, 32, 5)`. The network should take in a grayscale, square image.

Well done defining a convolutional neural network along with the feedforward behavior!
Nicely done adding batchnorm and dropout layers to avoid overfitting and pooling layers to detect complex features!

Notebook 2: Define the Network Architecture

Define a `data_transform` and apply it whenever you instantiate a DataLoader. The composed transform should include: rescaling/cropping, normalization, and turning input images into torch Tensors. The transform should turn any input image into a normalized, square, grayscale image and then a Tensor for your model to take it as input.

Great job defining the data_transform to turn an input image into a normalized, square, grayscale image in Tensor
format!

Select a loss function and optimizer for training the model. The loss and optimization functions should be appropriate for keypoint detection, which is a regression problem.

Nicely done using the SmoothL1Loss loss function along with the Adam optimizer!

Train your CNN after defining its loss and optimization functions. You are encouraged, but not required, to visualize the loss over time/epochs by printing it out occasionally and/or plotting the loss over time. Save your best trained model.

Good job training the CNN and saving the trained model in the checkpoint!

After training, all 3 questions about model architecture, choice of loss function, and choice of batch_size and epoch parameters are answered.

The questions have been answered and your reasoning is clear and sound!

Your CNN "learns" (updates the weights in its convolutional layers) to recognize features and this criteria requires that you extract at least one convolutional filter from your trained model, apply it to an image, and see what effect this filter has on an image.

The CNN does "learn" to recognize the features in the image and a convolutional filter has been extracted from the
trained model for analysis

After visualizing a feature map, answer: what do you think it detects? This answer should be informed by how a filtered image (from the criteria above) looks.

Nicely done identifying the filter as a filter that detects the edges of the map!

Notebook 3: Facial Keypoint Detection

Use a Haar cascade face detector to detect faces in a given image.

Great job using the Haar cascade face detector for detecting frontal faces in the image!

You should transform any face into a normalized, square, grayscale image and then a Tensor for your model to take in as input (similar to what the `data_transform` did in Notebook 2).

Well done transforming the face image into a normalized, grayscale image and passing it into the model as a Tensor!

After face detection with a Haar cascade and face pre-processing, apply your trained model to each detected face, and display the predicted keypoints for each face in the image.

The model has been applied and the predicted keypoints are being displayed on each face in the image

DOWNLOAD PROJECT

7 CODE REVIEW COMMENTS ›

Rate this review