UDACITY

‹ Return to "Computer Vision Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Facial Keypoint Detection

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

You've done a fantastic job completing the Facial Keypoints Detection Project! 😄 💯
It is evident that you've given enough respect & time to the project which is wonderful! ✨
Awesome job!! 🎉
Keep up the amazing work & All the very best for future!! 🎁

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 behaviour!
Nicely done adding a dropout layer to avoid overfitting and a pooling layer 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.

You did a 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.

Yay! Nicely done using the MSE loss function along with the Adam optimizer! 🎉
You've made sure to choose an appropriate loss function for the current regression problem. The chosen loss function does compare outputs value by value instead of looking at probability distribution, which is the case with classification problems. 😄 💯

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.

Awesome! 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!! ✨

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.

Awesome!! The model has been applied and the predicted key points are being displayed on each face in the image !! 😄 💯 🎉
Your model isn't trained properly & hence it's not showing the predicted keypoints properly on both the images. But since the aim of this project was not high accuracy but to make sure that you code neural networks

beautifully from scratch(which you've done clearly! Good job!) I'm passing you.
Also, the need for normalization i.e. multiplied and added values is because If you observe the content of the file '/data/training_frames_keypoints.csv' you will find the mean value of key points for each person is ~ 100, and its variance (sqrt of standard deviation) is ~50. Thus these values are used to normalize the keypoints in the range [-1,1].
I hope this was helpful in understanding why we are adding & subtracting values. If you've any more doubts, please do ask in the knowledge forum, I'll be happy to help! 😄

DOWNLOAD PROJECT

RETURN TO PATH