CNN(Convolutional Neuraul Network) Code review

1. First get data from kaggle(or UCI).
2. Get Dogs and cats images, and our goal is to train a model and distinguish whether an image is a dog or a cat.
3. First, we use feature detector to do Convolution, some information can refer to

[A Beginner's Guide To Understanding Convolutional Neural Networks](https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/)

1. Max Pooling,which preserve main feature, reduce size and prevent overfitting.
2. Adding a second convolutional layer.
3. Flatten outputs of convolutional layer in order to feed into full connection layer.
4. Fully connected layer are applied, and the number of outputs depend on how many predictions would be.(here we just have a output with sigmoid, which 0 represent cat and 1 represent dog.)
5. Rescale the images to small size in order to speed up the training process.
6. Training model (Not only the weights are trained, but also the feature detectors are trained).
7. Test single cat or dog image to see the result.

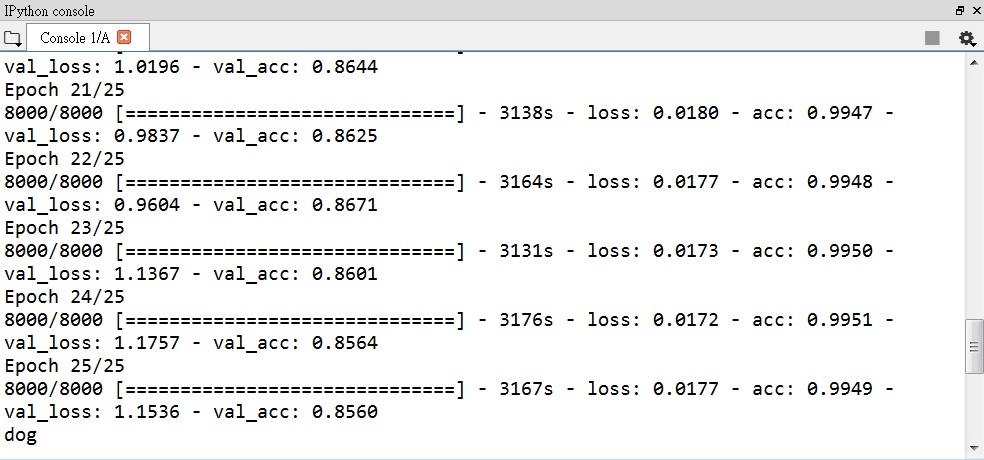
Data:

Lots of dogs and cats images from kaggle.

Code:

|  |
| --- |
| # Part 1 - Building the CNN  # Importing the Keras libraries and packages  **from** keras.models **import** Sequential  **from** keras.layers **import** Conv2D  **from** keras.layers **import** MaxPooling2D  **from** keras.layers **import** Flatten  **from** keras.layers **import** Dense  # Initialising the CNN  classifier = Sequential()  # Step 1 - Convolution  classifier.add(Conv2D(32, (3, 3), input\_shape = (64, 64, 3), activation = 'relu'))  # Step 2 - Pooling  classifier.add(MaxPooling2D(pool\_size = (2, 2)))  # Adding a second convolutional layer  classifier.add(Conv2D(64, (3, 3), activation = 'relu'))  classifier.add(MaxPooling2D(pool\_size = (2, 2)))  classifier.add(Conv2D(64, (3, 3), activation = 'relu'))  classifier.add(MaxPooling2D(pool\_size = (2, 2)))  # Step 3 - Flattening  classifier.add(Flatten())  # Step 4 - Full connection  classifier.add(Dense(units = 128, activation = 'relu'))  classifier.add(Dense(units = 1, activation = 'sigmoid'))  # Compiling the CNN  classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])  # Part 2 - Fitting the CNN to the images  **from** keras.preprocessing.image **import** ImageDataGenerator  train\_datagen = ImageDataGenerator(rescale = 1./255,  shear\_range = 0.2,  zoom\_range = 0.2,  horizontal\_flip = True)  test\_datagen = ImageDataGenerator(rescale = 1./255)  training\_set = train\_datagen.flow\_from\_directory('dataset/training\_set',  target\_size = (64, 64),#input size  batch\_size = 32, #batch size for updating weights every epoch  class\_mode = 'binary')#how many classes  test\_set = test\_datagen.flow\_from\_directory('dataset/test\_set',  target\_size = (64, 64),  batch\_size = 32,  class\_mode = 'binary')  classifier.fit\_generator(training\_set,  steps\_per\_epoch = 8000,  epochs = 25,  validation\_data = test\_set,  validation\_steps = 2000)  **import** numpy **as** np  **from** keras.preprocessing **import** image  test\_image = image.load\_img('dataset/single\_prediction/cat\_or\_dog\_6.jpg', target\_size = (64, 64))  test\_image = image.img\_to\_array(test\_image)  test\_image = np.expand\_dims(test\_image, axis = 0)  result = classifier.predict(test\_image)  training\_set.class\_indices  **if** result[0][0] == 0:  **print**('cat');  **else**:  **print**('dog'); |

Result:



Summary:

The average accuracy is about 86%. In the end, we use single image to test whether this model can tell it is a dog or a cat. Boom!! Correct! To adjust the parameter, we can use other optimizer which can refer to [Deep Learning optimizers](https://medium.com/towards-data-science/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f), and also we can change batch size, epoch, rescale size or convolutional layer…etc, and then use cross validation to find better training model for our test set data.

