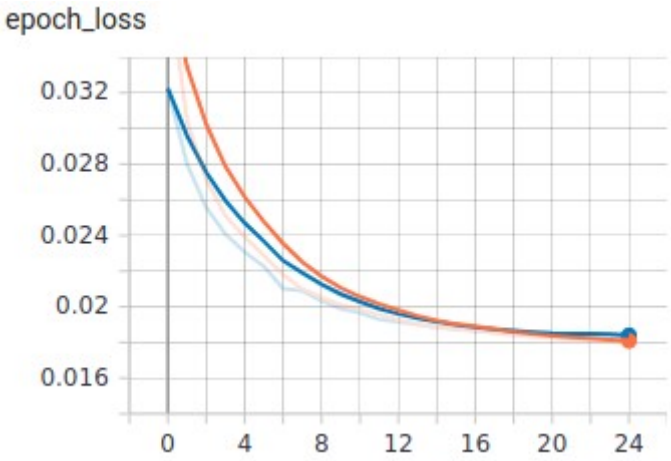
Image Retrieval

1. Imports
2. Loading Data
3. Creating Convolutional Autoencoder
4. Training the model
5. Saving the model
6. Visualizing the output of the auto encoder
7. Clustering and optimizing for K
8. Training Kmeans with 6 clusters
9. Predictions

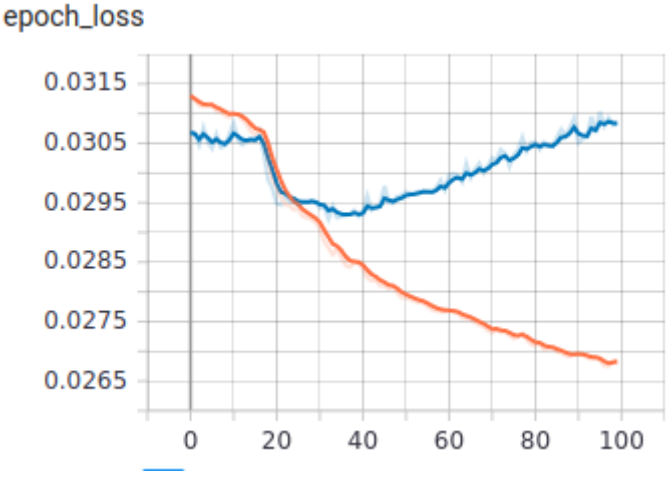
First we extract the images and '`dataset_path_list`' is a list with all the paths of images. First 4000 images are set for training and rest are for validation.

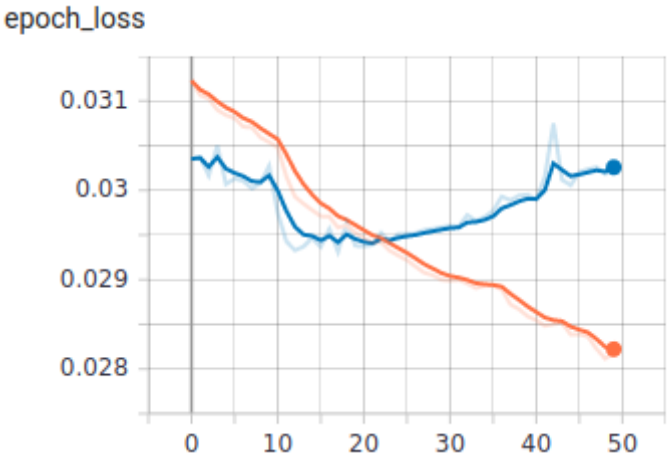
Then we define a dataloader '' which returs batchs of images which are filped horizontally with 50% probability. Then crate a train and validation data.

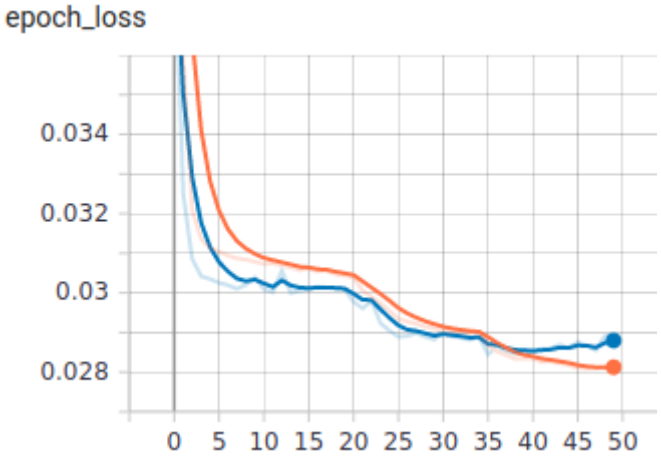
Then we define an hour glass model Conv_Autoencoder. I tried with few different values, 32,16,8,8 with 512 latent spaces is the final one.



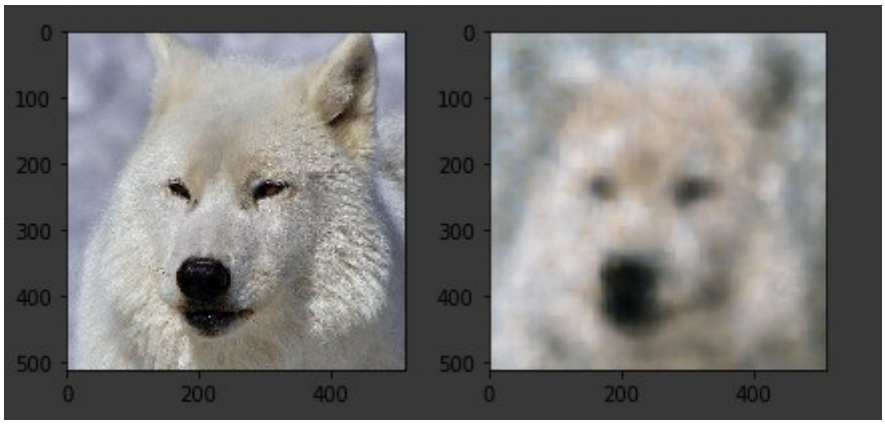
Few others graphs are as follows:



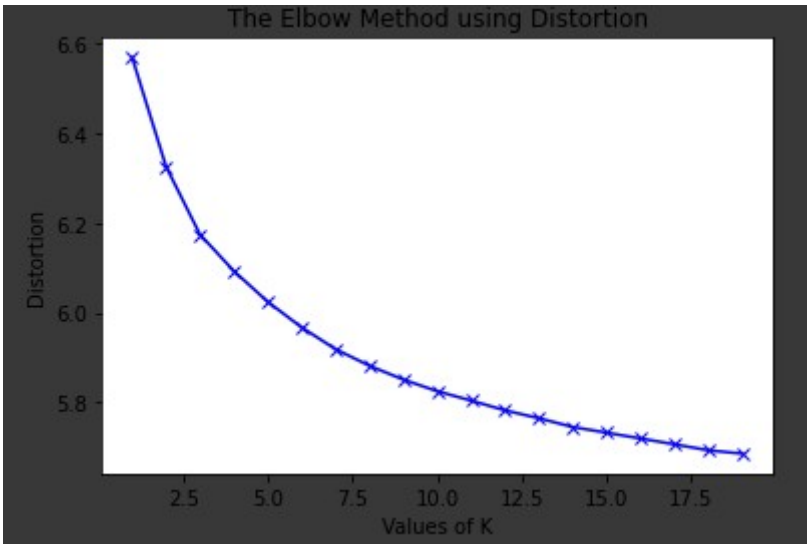




Visuvalizing the output



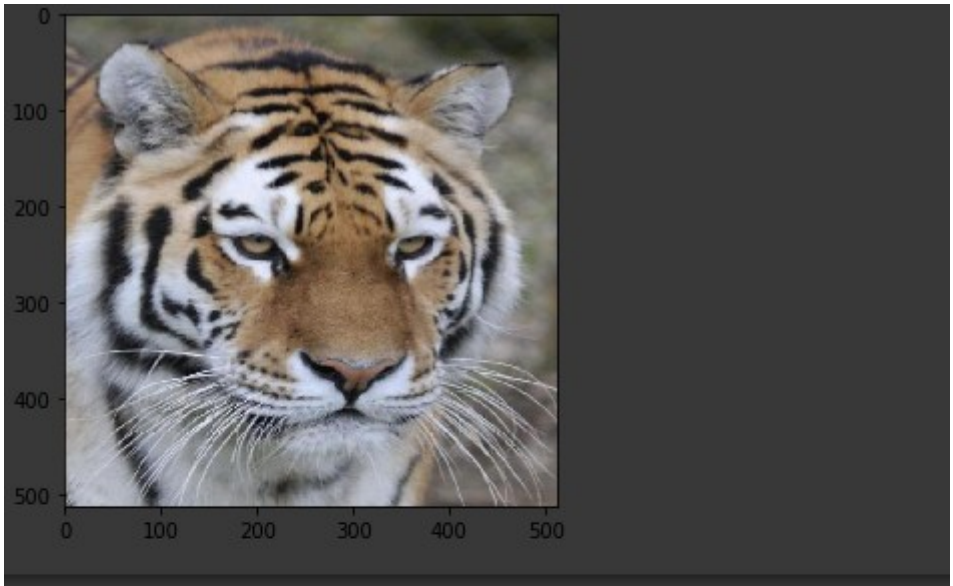
then we make a numpy.array of the 512 dim vectors on train dataset, and test for elbow method for number of clusters. (picking 6)



finally we make a dict containing the keys as clusters and the encoder output as the value.

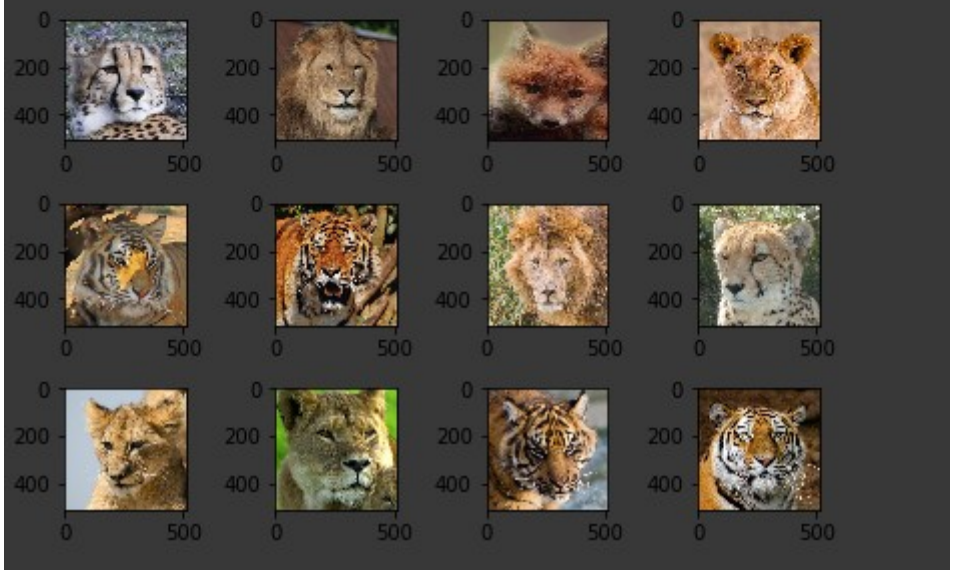
For an input we first normalize it ,pass it though the encoder and get the 512 dim vector.
Then we pas this into the Kmeans clustering model and get the cluster number.

Then we qurey for similarty of the image within this cluster




```
plt.figure()
for idx , result in enumerate(results):
  image = imread(train_path_list[results[idx][1]])
  plt.subplot(3,4,idx+1)
  io.imshow(image)

plt.show()
```



dataset:
https://drive.google.com/file/d/12m1t5UaQTViKCJfnIlBfQXuo8HTfTwM0/view?usp=sharing

trained model:
https://drive.google.com/file/d/1svYwPTcbHTLCKq6Mam-NmeFl5lYETz1Z/view?usp=sharing

Kmeans:
https://drive.google.com/file/d/11XwV20L6nGsQUefnBC5VO_kUA5e8yShA/view?usp=sharing