Machine Learning Engineer Nanodegree

Capstone Project

Amr Elbeleidy

December 04, 2017

I. Definition

Project Overview

This project explores unsupervised learning as applied to images.

Images to us, humans, are naturally processed through our vision and brain systems. We learn to recognise what things are without being explicitly told how to do so. To a computer, images are a series of numbers representing the brightness of different colours in particular parts of the image.

In traditional classification problems, we present the computer with images and labels describing what the images represent. We then ask it to learn how to classify similar images by looking at the ones we gave it. Using deep convolutional networks the computer detects patterns in the images and relates them to the classification labels using fully connected neural networks.

In unsupervised learning, the computer is asked to infer a pattern, structure, or relationship in the data that we cannot see. One method of applying this is clustering where we ask the computer to mark similar data points with the same label. Unsupervised clustering has been used for image classification before.

In their paper[1] Seldem et al used image segmentation for unsupervised clustering of nature views and paintings by painting style. Yang et al[2] also used a convolutional neural network along an unsupervised clustering algorithm, but trained them both together.

While in some aspects of machine learning we rely on computers doing things much faster and better than we can do, in other aspects we are aiming to automate what we can already do, even if it currently does that at worse performance. The latter is the case in this project.

We explore the notion that we can combine the powers of convolutional networks to represent images in a more meaningful way for computers with the ability of unsupervised clustering algorithms to group data points that are similar to each other in order to separate images of different cat and dog breeds.

For this project we will use the Oxford IIIT Pet Dataset, available at http://www.robots.ox.ac.uk/~vgg/data/pets/ at the time of writing. The dataset contains images of 37 different cat and dog breeds, with about 200 images for each category. The images have different poses, scales and light levels, making our task of separating them non-trivial.

1. http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/attachments/Seldin_Stark_Werman_SCTV03_%5b0%5d.pdf
2. https://arxiv.org/abs/1604.03628

Problem Statement

Given a number of images of two or more breeds of pets from the Oxford IIT Pet Dataset, label them such that all images of the same breed have the same label, and images of different breeds have different labels.

To do this, I propose to use convolutional neural networks to represent the images, before passing on these representations to an unsupervised clustering algorithm. We can then examine the clusters to check if the computer has correctly separated the images of different pet breeds.

Metrics

At its heart, this problem is a classification problem. Although in traditional classification problems we provide names for the labels, in this case here, the computer will provide arbitrary label names (Cluster 1, 2, etc…) and assign them to the images of different breeds.

In the usual cases, we can measure the accuracy of the classification algorithm by using accuracy, precision, recall, and F1 score. In non-binary classification, where we have more than two labels, a global F1 score can then be calculated by different ways of weighing it out between the different class pairs.

In our case here, we can use the same classical classification metrics, once we've taken a look at the computer-named labels and assigned these to ones that make more sense to us.

We chose the F1 score as our metric of choice for evaluation of model quality and performance. The F1 score is useful when the label distribution is not equal. Although in our dataset, most

breeds have more or less the same number of images, the F1 would be also useful if the model was tested with a different label distribution. It also takes into account the balance between precision and recall.

The formula for calculating an F1 score is as below.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- Source: https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/
- https://en.wikipedia.org/wiki/Precision_and_recall

II. Analysis

Data Exploration

The general notion of representing images by using convolutional networks and then separating them using unsupervised clustering algorithms is applicable to many datasets. In our case here, we will be using the Oxford IIIT Pet dataset, comprising of images of 37 categories of pet breeds, 12 of cats, and 25 of dogs. The images are of different scales, poses and light levels. This is in contrast to some other datasets which will have a more uniform set of images.

It is not known to what lengths the authors of the dataset have verified that the individual cats in the images are in fact of the particular breed they have been labelled by, or how they defined a cat to be a purebred that has not had genes from other breeds in its lineage.

Upon downloading of the data, images were checked manually to see if there are any that are corrupt. Corrupt images will not read into our program, and will raise errors. We cannot proceed without weeding them out. The dataset was downloaded multiple times to ensure that these images were not corrupted during data transfer from Oxford University servers to my computer.

8 images were removed from the Egyptian Mau cat images, however this category was not actually used during our testing and therefore did not affect the results.

The images were also manually moved from being in a single folder, to placing images of the different breeds in their own subfolders. Images were already labelled in their filenames with which breed they were of, and so we do not expect this process to have misplaced any breed images in the wrong folder. This was done to simplify the process of loading the images into memory for the algorithms to work on them.

Exploratory Visualization

To start with, we do a count and a list of the number of breed images available. This count was done by code written as part of this project and gives the table result below. The codes were assigned by me, in order to facilitate referencing the different breeds, and were consequently used as labels when measuring algorithm performance. The choice of labels does not affect algorithm performance, as the separation algorithm is never exposed to them, and they are only used after it has produced its results.

The dataset has 7380 images, roughly distributed at around 200 images for each breed, with some breeds having less images from source, or because the images were corrupt.

Code	Folder name	Image count
C1	C1-Abyssinian	198
C10	C10-Russian_blue	200
C11	C11-Siamese	200
C12	C12-Sphynx	200
C2	C2-Bengal	200
C3	C3-Birman	200
C4	C4-Bombay	200
C5	C5-British_Shorthair	200
C6	C6-Egyptian_Mau	192
C7	C7-Maine_coon	200
C8	C8-Persian	200
C9	C9-Ragdoll	200
D1	D1-American_bulldog	200
D10	D10-Great_pyrenees	200
D11	D11-Havanese	200
D12	D12-Japanese_chin	200
D13	D13-Keeshond	200
D14	D14-Leonberger	200
D15	D15-Miniature_pincher	200
D16	D16-Newfoundland	200
D17	D17-Pomeranian	200
D18	D18-Pug	200
D19	D19-Saint_bernard	200
D2	D2-American_pit_bull_terrier	200
D20	D20-Samoyed	200
D21	D21-Scottish_terrier	199
D22	D22-Shiba_inu	200
D23	D23-Staffordshire_bull_terrier	191
D24	D24-Wheaten_terrier	200
D25	D25-Yorkshire_terrier	200
D3	D3-Basset_hound	200
D4	D4-Beagle	200
D5	D5-Boxer	200
D6	D6-Chiuahua	200
D7	D7-English_Cocker_Spaniel	200
D8	D8-English_setter	200
D9	D9-German_shorthaired	200

While it is not practical and unnecessary to include all these images as part of this report, the images below were chosen to highlight the variation in images that was present in the dataset. They have been selected from images of breeds that were used during this project.

Sample image of C1-Abyssinian at actual image size of 990 x 962 pixels

The images below are of the same C1-Abyssinian breed, at their original aspect ratio, but resized for inclusion in the report.

Original size: 150 x 116 px

Original size: 500 x 375 px

Original size: 313 x 310 px

We can see here that the images have different original sizes, different individual cats, striking different poses in different lighting conditions. In addition, sometimes other items are present in the images, such as plants and carrier bags.

Below are images from more breeds, including dog breeds, that were used in this project.

C8 - Persian

D19 - Saint Bernard

D25 - Yorkshire Terrier

These images show us more variation in what is present in the dataset, with multiple pets of the same breed in the same image, and a drawing of a human face visible in another one. We can also see that pets may have accessories on their bodies.

We expect this variation to make it more difficult for the algorithm to separate the images than if photos were uniform.

Algorithms and Techniques

A conda python environment was set up with a number of libraries including scikit-learn, tensorflow and keras. Scikit-learn is a machine learning library, tensorflow is used for, among other things, modelling neural networks while keras provides a high level interface with a number of other libraries including tensorflow. OpenCV, an open-source computer vision library was also used for image pre-processing and manipulation.

The following algorithms were used:

- OpenCV: Image reading and resizing.
- SciKit-Learn: Data shuffling, Kmeans and Gaussian Mixture clustering, Principal Component Analysis, and performance metrics.
- Keras wth tensorflow backend: VGG16, VGG19 and ResNet50 convolution network models with ImageNet weights.

Convolutional neural networks are neural networks whose architecture takes advantage of knowing that the input is an image in order to improve classification ability and reduce the number of parameters. Their layers are arranged in 3 dimensions, and each neuron is only connected to a small part of the layer before it, not to all nodes as we would expect in a fully connected layer.

Convolutional networks also use pooling layers to reduce the spatial size of the data. Pooling layers downsize the data by taking a number of features and transforming them into a single feature by applying a chosen function to them. A common function is the Max function where the new feature has the same value as the largest of the old features.

By using convolutional layers, and pooling layers to downsize we can reduce the number of features of our images far below the number of pixels they have.

ConvNets, as they are called, like all networks need to be trained in order to be able to classify images, and in order to learn which patterns in images are of interest in our scenario.

VGG16, VGG19 and ResNet50 are all convnet architectures that have proven themselves highly accurate in image classification tasks. ImageNet is a dataset of over 14 million labelled

images. When the convnets are trained on a dataset of very general images, they learn to recognise patterns in these images. Transfer learning is when we teach a neural net labels in one domain, but then use what it has learnet in another domain.

In the case here, instead of building our own conv net and training it on the images available, we use transfer learning and the weights that resulted of training these convnets on the ImageNet dataset. They are therefore generalised pattern recognition networks and would lend themselves to being used on many different data sets, not just on pet images.

Although the convnets would reduce the number of features in our images, and give us more useful ones, they could still be too many for our clustering algorithm. Enter principal component analysis.

Principle component analysis is a technique that brings out the variations in the dataset. It reduces the number of features of a data set such that the features it provides produce maximum variance between the points. This allows us to represent the differences between the different points in less features, and makes it easier for our clustering algorithms to operate.

We therefore use PCA to further reduce the output of the pooling layer of the convnet before passing on the data to our clustering algorithm.

KMeans clustering is a method that clusters data points together based on feature similarity. It starts by randomly assigning data points to clusters and then works on minimising the eucledian distances of points from their cluster centroids.

In Gaussian Mixture Model (GMM) clustering, the algorithm assumes that each cluster is in a gaussian distribution and assigns it to the cluster it has the largest probability of belonging to. GMM clustering has an advantage of not requiring clusters to be spherical, i.e. the cluster boundary does not have to be equidistant from its centroid.

Sources:

- http://cs231n.github.io/convolutional-networks/
- http://ruder.io/transfer-learning/
- http://www.robots.ox.ac.uk/~vgg/research/very_deep/
- https://github.com/KaimingHe/deep-residual-networks
- http://image-net.org/index
- http://setosa.io/ev/principal-component-analysis/
- https://www.datascience.com/blog/k-means-clustering
- https://www.cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_mixture.pdf

Benchmark

As this a classification problem, the benchmark chosen is perfect clustering of the images, with all images of the same breed exclusively in the same cluster.

The F1 score was used to measure model quality and compare performance. F1 scores are an industry standard metric for assessing classification model performance.

III. Methodology

Data Preprocessing

1. The files of the images in the dataset were manually moved into subfolders for each breed.
2. Images were read into arrays, and resized to 224 x 224 pixels. This size was chosen because it is the input size for the VGG16, VGG19 and ResNet50 networks.
3. Since OpenCV reads images into BGR arrays by default, the images were then translated by an OpenCV algorithm to RGB channels, so that they can be displayed on screen using the Matplot library.
4. Labels were generated for the images read, with a breed code assigned to each image.
5. The image arrays were normalised by dividing the value of each element by 255. Convolutional neural networks are generally expected to perform better with normalised images.

Implementation

Now that we have prepared our images and transformed them into 224 x 224 normalised arrays, they are ready to be passed through the convolution networks.

Before we do that, we use the test_train_split function from scikit-learn with a test ration of 0. This effectively shuffles the order of the images and the labels we have provided it without splitting the data. We are using here all the data available, since the clustering model will always train on any new data it is given, and the convnet is pre-trained in a different domain. This way our image order would be randomised and should not bias the results.

Keras was used to load built-in models of VGG16, VGG19 and Resnet50 with their tops off and using weights that were a result of training them on the Imagenet dataset. A convolutional network with its top off is a network that does not have the final fully connected, or dense, layers.

The shuffled image arrays are then passed to the convolutional network models. The output of these networks is a 3d tensor which we then flatten by reshaping the array.

Once we've flattened the image arrays, we can see that in the case of the VGG networks, each image has more than 25,000 features. In ResNet's case, we are looking at 2048 features. We then use scikit-learn to perform principal component analysis (PCA) on the data, to reduce its dimensionality to something that can be more easily managed by the clustering algorithms.

The number of components the PCA produces will depend on which breeds we have chosen to separate, as it is restricted by the number of samples we supply to it. We use all the components that the PCA produces and transform the image arrays.

We use two clustering algorithms for comparison, kmeans and gaussian mixture model (GMM) clustering. The number of clusters or components is defined to be the number of breeds we have selected. We set GMM co-variance to full.

GMM, within the available computing capabilities, was not able to execute given the data without PCA, and so we only pass it the data with reduced dimensionality. KMeans, at scikit-learn default settings, was passed both the data with dimensionality reduced and without.

We then implemented a python algorithm to count and display a table of which cluster contained how many of each breed's photos for each combination of full data / data with dimentionality reduced and KMeans / GMM clustering.

The counting tables were manually looked at to be able to assign a breed code to each arbitrarily named cluster. Since the clustering algorithms have never been exposed to the breed labels, they have assigned different sets of images a cluster number.

If a counting table shows no meaningful separation of breeds, then this combination of data representation / algorithm was not pursued further. The main criteria for a meaningful separation is that each cluster contains a majority of images pertaining to one breed, and that each breed has a cluster with a larger number of its images than other clusters. The clusters for the different breeds must be unique.

One thing to notice is that we have never supplied the clustering algorithms with a count of each breed's images or that we expect any particular image distribution between the clusters. This means that the algorithm could end up placing a majority of all images in one cluster, and a small number of images in another. It is cases like these, where we can see the algorithm has failed to separate the breed images and we pursue it no further.

Once we've assigned breed codes to the clusters, the counting tables are now effectively a confusion matrix. We have the original image labels as the true label, and the clustering

predictions as the predicted labels and we pass these onto a scikit learn F1 and Accuracy score calculators to compare model performance.

Refinement

Initially the image arrays were passed to the clustering algorithms without PCA. KMeans was able to handle the large number of features that came from the VGG convolutional networks, however GMM was not and consistently ran out of memory, even when given images from only two breeds. The algorithm was running on a computer with 16GB RAM and with a swap of another 16GB, however these were not enough.

PCA was used, and explained variance was plotted to check how many components were needed to accurately represent the data. The decision was taken to use all available components since they are much reduced than the over 25,000 featured VGGNets produced. The algorithm was then able to cope.

For comparison, when KMeans was given all the components from the PCA-transformed data, and the raw data from the convolutional networks, it produced exactly the same results. This was an indication that PCA data was accurately describing the raw data.

Although refinement using PCA did not give any difference in performance of KMeans clustering, it allowed us to check GMM model performance which would not otherwise run with the larger feature set on available computing power. It also improved the model training time, while giving the same results.

IV. Results

Model Evaluation and Validation

The main aim of this project is to show that this methodology of transforming images through convolutional networks and then passing them onto clustering algorithms is a feasible one for image separation. In that respect, different models get a Pass / Fail metric as to whether they perform better than randomly assigning images to clusters. If a model fails, there is no need to evaluate it further and pursuing it stops.

It is also important to note that the specific F1 and accuracy scores are going to be different for each breed combination. Breeds that look similar to each other, are likely to be more difficult to separate than those who look completely different.

The model was ran on these different combinations of breeds (C for cat and D for dog):

1. Two cats and two dogs
    a. C1 - Abyssinian
    b. C8 - Persian
    c. D19 - Saint Bernard
    d. D25 - Yorkshire Terrier
2. Three dogs
    a. D9 - German Shorthaired
    b. D21 - Scottish Terrier
    c. D11 - Havanese
3. Two cats
    a. C12 - Sphynx
    b. C8 - Persian
4. One dog and one cat
    a. D1 - American Bulldog
    b. C7 - Maine Coon

We have two formats for the data, raw and PCA-transformed, given to 3 different convolutional networks and passed to two different clustering algorithms. The Pass/Fail results were consistent for all models except the GMM / VGG ones, in all breed combinations attempted and are summarised in the table below:

	VGG16	VGG19	ResNet50
KMeans (raw data)	PASS	PASS	FAIL
KMeans (PCA)	PASS	PASS	FAIL
GMM (raw data)	N/A	N/A	N/A
GMM (PCA)*	PASS/FAIL	PASS/FAIL	FAIL

* The GMM PCA model worked for some combinations and not for others. When it worked, it always performed worse than the KMeans model. Since it is unreliable and has never shown better results than KMeans, it was not pursued and all focus was on Kmeans with VGG16 and VGG19.

Keeping in mind that the results for PCA and non-PCA transformed data were exactly the same, the following F1 scores were observed for each breed combination:

Breed Combination	KMeans / VGG16	KMeans / VGG19
C1 - C8 - D19 - D25	0.77355392	0.54872423
D9 - D21 - D11	0.81818354	0.54700167
C12 - C8	0.975	0.93998650
D1 - C7	0.97749986	0.85454638
Benchmark	1.0	1.0

Justification

The results show what we set out to achieve, that is possible for the computer to separate unlabelled images by passing them through a convolutional network, followed by a clustering algorithm. The final model chosen performs significantly better than random, but is not perfect. How well the model performs is dependant on the images it is given.

The KMeans / VGG16 combination is a suitable model for our dataset. It has solved our problem, albeit to a different extent depending on the breed combinations.

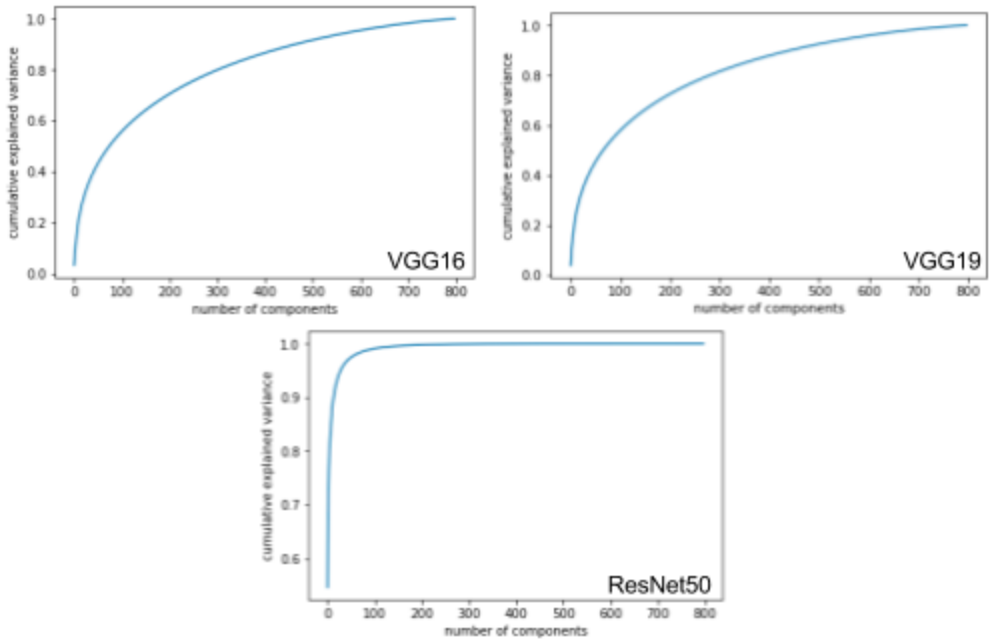
V. Conclusion

Free-Form Visualization

When the GMM algorithm was unable to process the large featureset of the images, on the available computing capabilities, after they've been passed through the convolutional networks, PCA was identified as a dimensionality reduction method.

Traditionally when PCA is used, we chose the number of component we want to use based on how much explained variance we need. There is a compromise between the reduction in dimensionality possible and the amount of information conserved.

One good way of visualising this are charts of explained variance vs number of components. We can see below such charts for the three neural networks used when given images of the first breed combination.



Although in this case we opted for all the PCA components and thus maintaining the variance information, it is possible to reduce the dimensionality even more at a trade off that we can now see.

The shape of the graph for ResNet50 is notable. In about 100 features, it is possible to summarise all variance between the produced images. This seems to imply that the rest of the components don't carry much extra information. Could this mean that the ResNet50 network does not give very descriptive information about our images? It is possible that this is why it consistently failed to provide the clustering algorithms with information that they would be able to use to separate the images.

Reflection

In this project, we took a dataset of pet images, of different sizes, breeds, poses and light levels and aimed to separate the images of different breeds without giving the computer any information about the images. We passed the images through pre-trained convolutional networks, flattened the output and passed it to clustering algorithms, which separated them.

It is difficult to compare the results of a model run on one combination of breeds with another. Intuitively, we expect the model to perform better when the different pet breeds look very

different to one another. We also expect it to perform better when images of the same breed look closer to one another. But how can we define these aspects quantitatively? If we can calculate numbers to describe these aspects of the dataset, we can then use these numbers and find which models work best for particular types of datasets.

This would not require any new information to be input. During pre-processing these aspects can be calculated and a more suitable model chosen on that basis.

Aspect ratio is another issue. The images supplied were all made smaller or larger to a uniform size of 224 x 224. A convolutional neural network needs all inputs to be the same size. In our case we used the available resizing algorithm and placed no restriction in input image size. Could it be that when the original aspect ratio is lost, some information is lost in that transformation? Probably. Would this information affect model performance? Possibly.

Would it be preferable to implement an algorithm in the pre-processing stage that detects the pet's face and then crops a specific image size around its centroid? What would then happen if the photo was of the pet's back, or if the pose does not allow the algorithm to correctly place where the pet's face is? What about the case where there were two individual pets of the same breed in the same photo?

If the aim is to use photos that people take of their pets in everyday scenarios, then we would not be able to impose such restrictions on the input data. We would have to work around the large variations in the photos.

The model shown here, to an extent works. But is it the definitive solution to a pet breed image separator? I have my doubts.

Ideally this solution would be a helper to humans working on the matter. Once the separation has been done, it is then easier for humans to go through the results and improve them, with much of the work already done for them.

Improvement

The low hanging fruit for improving this algorithm lies in the pre-processing of the data. There are better solutions than simple resizing of the images and one of them as discussed above is cropping the relevant part of the image.

There are dog and cat face detector libraries openly available. These could be used to hone in on the salient features of the individual pet and give the model a higher ratio of useful information.

It might even be possible with large improvements to reach a level where photos of individual pets are separated even within the same breed.