Final Project: Image Classification System

Universidad Carlos III de Madrid, Multimedia, Year 2018-2019

Autors
Mateusz Kobierski - Federico Goldfryd Sprukt

Assigned Class
TIE Interceptor
Chosen Classification Model
K-Nearest Neighbours / Support Vector Machine
Upgrades Implemented
Upgrade 1 and 2

Development

Preprocessing

P1. Is the segmentation correctly performed in all cases? In case of a negative answer, why do you think Otsu's method fails to discriminate object from background?

The segmentation is performed correctly in many cases, but not in all of them. That happens since Otsu's method need the image to have its colours distributed in a bimodal way, having two significant peaks. When that does not happen, this method is not so good for the segmentation.

P2. If the current scenario required multiple object identification, ¿which technique could be employed for such a task? Explain briefly that technique.

In order to detect and separate multiple objects the picture (input data) should be divided into smaller regions containing one object each. This technique is called segmentation – it can be achieved i.e. by thresholding, colour-detection or clustering. It differentiates object with the difference between object's pixels and pixels surrounding the object.

P3. What was the main benefit of the opening operation?

Opening the image allows to decrease the noise by erasing some small objects found in the background (or black part). In our tests, we also found that when the white part of the image is bigger than the black part, applying the opening operation reverses the colours.

P4. If we set the radius of the structural element too large for the opening operation, what risk do we take?

If we set the radius too big, most likely we will get important parts of the object in the image removed so we will lose the shape.

P5. Why was the closing operation useful?

The closing operation reduces the noise inside the body of the object (or the white part) by removing the small objects found there. This combined with the opening operation leads us to having an image with two really differentiated parts, background and foreground. This way, the shape of the object is more noticeable.

Feature Extraction and Normalization

P6. What advantages does the HSV colour space offer with respect to RGB?

Unlike RGB, HSV separates luma, or the saturation, from chroma or the colour information. This is helpful when extracting only the colours, not caring about the light and colour intensity.

P7. What would happen if we tried to extract this feature for the whole image, i.e., without previously applying segmentation and morphology pre-processing?

Extracting features from whole images would mean extracting features and information from parts of the picture that are irrelevant for the algorithm such as background noise, not related extra objects etc. As an effect system of recognition would examine data that is not related to our object of interest thus, it could be less accurate.

P8. What measures can be taken so that image size does not affect the shape feature?

One way is to normalize perimeter and check the rectangularity/circularity of the object. That way size will not affect the shape feature because it will be normalized.

P9. How would a bad object segmentation affect the extraction of generic features related to shape?

Bad quality of masks (the output of bad object segmentation) could affect in extracting features of areas that are not related to our object of interest, i.e. objects that are overlapping and/or objects in background. From our point of view their features are not relevant and it is crucial to “cut them off” in order to filter “information noise”.

P10. In your opinion, what is the most useful feature regarding the discrimination of the spaceship assigned to your group? Justify your answer and use figures to support your claims.

In my opinion texture is the most useful feature. It’s because texture is easy to extract from objects, it’s more objective and it’s mostly better preserved than colour or shape (due to difference in perspective). Every object differs in its texture from others and that makes it highly classifiable.

P11. Taking into account what was done in Stage 3, mention at least 3 additional features (one of each kind) that could be extracted for this system. Justify your choices.

Entropy of colour – At the moment, we are extracting the entropy of the texture of the image. By getting the entropy of colour, we can get information about the variety of colours in the image, to know if it is more or less monochromatic.

Perimeter – At the moment, we are extracting the rectangularity of the object, comparing its area with the area of the smallest rectangle that can hold it. Getting information about the perimeter would give also relevant information about the shape of the object.

Entropy of texture – In the project, we measure the entropy of the texture to know the variety that we can find in the image. Another interesting feature is the range of this texture. Two images with the same entropy, can have a different range of textures, being one more concentrated and one more spread. This information can also be useful.

Classification. Training and Evaluation of New Samples

P12. Explain the algorithm for classifying new samples followed by the K-NN method.

K-NN method classifies new samples by taking (given) K closest neighbours and assigning the class that occurs in most of them.

P13. For this particular case, what would happen if we set a value of K equal to the number of samples available for training ($K=56$)? Why?

It would mean that our algorithm, when classifying new sample, takes all the sample's neighbours into consideration. It would mean that the system would classify unknown objects the same as most objects. Taking in account that we need to detect 1 category out of 4 possible and that probably there are more or less the same number of images for each one, the system would always return false.

P14. Why are we forced to use the same normalization values we used during training?

In the end, those values are the ones used to compare the new samples and know if they fulfil our requisites or not. If the normalization value was different, that result would not have any meaning.

P15. Why can't we use the same set (e.g. our whole database) for training and testing?

The training process consists in extracting the features of a dataset in order to compare it with external data and see the level of matching with these features.
If we use for testing the same dataset than for training, we will always get an accuracy of 100% because the system is built on those specific samples.

P16. ¿What risks do we run by setting a training set too small? And by setting it too large?

If the training set is too small, the features extracted from these few samples can be too specific and therefore the model trained will be too strict. This will cause that only images really similar to the ones used for training will be detected by the model because the more general aspects of that image were not considered to train.

In the other side, if the training dataset is too large, the remaining samples used for testing would be too small to verify it correctness in a reliable way.

P17 In case K-NN was selected for classification, why do you think the model can only set fixed values of +1 and -1?

KNN is a binary algorithm. It compares the input with the K closest individuals and return the most frequent group as the output. Given that, the values -1 and +1 are the ones returned in the code because it is the simplest way to implement this in code. It could be possible to change the implementation to return other values (i.e. {0,1}) but giving a non-binary output would mean we are not using KNN anymore, but a different algorithm.

Assessment of the System

P18. In your own words, describe what each of the presented metrics mean for our case of study (spaceship discrimination system).

- p_D: It is the probability of detection. It represents the amount true positive detections divided by total positives.
- p_{FA}: It is the rate

P19. Taking into account the obtained results (specify them in your answer), do you consider the performance of the system to be acceptable? Why?

Our AUC score was 0.83
Given that we had small amount of training objects and the calculated masks were of poor quality, I'd say our score was acceptable – it means our system "guessed" much more accurately than doing so by random.

P20. Discuss the relation between the curve displayed in the figure and the values obtained for $p_D$and p_{FA}.

The curve gives us the percentage of correctness of the whole system. The values $\mathrm{p_D}$ and p_{FA} are used to find the point used to join the scenario where we only answer false and the scenario where we only answer true.

P21. If the AUC gave a value of 0.5, what would this imply?

If the AUC score was 0.5 it would mean that our classification is worthless – system guesses would be purely random.

Annex: Preliminary Report

1. Introduction

In this assignment, we are given a set of different images of 4 kinds of spaceships. Our group was assigned to recognize the first category (Imperial TIE Fighter).

Having an initial dataset and a base code, our job has been to process these data and finish the missing parts of the code taking decisions when needed in order to achieve a final system that can recognize whether an input image includes or not an Imperial TIE fighter in it.

2. Technical Solution

The algorithm execution consists of several steps, which we will explain in order:

- **Data generation.** This step basically consists in getting the images ready to be processed. We have a dataset of 20 different images from the 4 different types of spaceships. The program loads them into an array and creates another array with the value 1 or -1 depending on whether the image on that position belongs to our class or not. After that, the two arrays are split assigning 70% of them for training, and 30% of the for testing, which is a reasonable proportion.

- **Pre-processing.** In this stage, the objective is to extract our object from the whole image. To do this, we work with the image in grayscale and we calculate its *Otsu* threshold. The *Otsu* threshold is calculated in order to get the best value for splitting an image in two well defined classes given the grayscale distribution histogram. This way, we can extract the object from the background imbanarizing the image to get one in white and the other in black, with the lowest noise possible.
  Since it is impossible not to get noise, later we apply the open and close operators to respectively reduce the noise outside and inside the object.

- **Feature Extraction Stage.** After extracting pure objects, the next step is to extract useful information out of it. In this stage we take every object from the array and save features such as:

  - **Colour**
    We extract colour feature by converting RGB picture to HSV picture, then we cut out the background using masks from pre-processing stage and at the end we take median out of the colour values. We use HSV over RGB in order not to take features as luminance and saturation into account – we only want the colour feature.

  - **Texture**
    Second feature we want to extract is texture of the objects. In order to do so, we convert given pictures to greyscale pictures and then calculating its entropy. Entropy is a function describing variety/randomness of the input. We used it because given greyscale image, entropy calculates level of diversity of the data which in other words can be described as texture.

- **Shape**
  Third feature needed for classification is the shape of the object. In order to calculate it we had to take into consideration different sizes of images, so we had to come up with a way to normalize the feature. We chose to extract the rectangularity of the objects. In order to do so we divided the area of the object by the smallest rectangle the object can be inscribed in. That way the actual size of the object did not matter to the system, we only extracted it shaped (measured in this “rectangularity”).

After the extraction we saved the features corresponding to our objects in a matrix which we had to normalize for the system to work properly. Using unnormalized data in further steps could weaken the algorithms and lower the systems overall scores because features that take large values would be “seen” as more important which is unwanted in this case.

- **Training Stage.** Now we finally got to use our freshly extracted features for the training. We chose the KNN model for the classifier which is optimal for small amounts of data (we have only 56 samples). KNN doesn’t need any training, it only needs to have all the training data saved. It basically works by comparing given sample to K-number of the closest neighbours. After that it classifies new sample as that which occurs at most of them.

- **Testing Stage.** This stage consists basically in applying the model obtained to the images of the testing set that have been unused up to that moment in order to get the results of the model prediction. The testing set has to be processed as before, so we apply the same pre-processing, feature extraction and normalization.
  The difference here is that we normalized the data used the mean and standard deviation computed from the training set instead of calculating a new one. That is because those values are the ones that really tell our model if an image is in one category or another, so the result will depend on how close the characteristics of these new images are to the ones used for training.
  This stage ends with an array of predictions for each one of the testing images, saying if our model “thinks” they contain the space ship we are looking for or not.

- **Performance Assessment Stage.** At the end of the execution we assessed the correctness of our system. To do so, we compared labels created by our system with the given, true ones and counted the number of truly “recognized” and number of falsely “recognized” we called them `true_pos` and `false_pos`.
  At the end we calculated Probability of Detection and Probability of False Alarm with following equations:

$$p_D = \frac{true\_pos}{total\ positive\ samples} \qquad p_{FA} = \frac{false\_pos}{total\ negative\ samples}$$

After that we evaluated general performance score of our system (AUC) which is the area under the curve created by examination of every possible threshold in the KNN classifier.

3. Performance

To measure the performance, we can make use of three different values:

- p_D. Probability of detection. Chance of a spaceship of our class to be detected.
- p_{FA}. Probability of false alarm. Chance of a spaceship out of our class to get a positive label.
- AUC. Area Under the Curve. This is the probability of having a right detection, either positive or negative.

We run the program for ten times, measuring the values to get an idea of how good the implementation was behaving. Those are the results:

AUC	P_D	P_FA
0.89	0.83	0.06
0.75	0.57	0.058
0.77	0.71	0.17
0.87	0.86	0.12
0.86	0.73	0
0.8	0.71	0.12
0.90	0.88	0.06
0.78	0.63	0.063
0.76	0.57	0.06
0.84	0.75	0.06

The average value for each one of the parameters were p_D = 0.72, p_{FA} = 0.08 and AUC = 0.82. Those values are really good give the limited dataset used for training, so we could say our model was working well.

4. Conclusions and lines of work

By developing this project, we have got a hands-on impression of how image recognition works. It has been interesting to understand some of the mechanisms that are used in order to classify images.

Given that the model was a really basic one, there are several ways to improve it. The main one would be applying data augmentation to improve the variety in the database. Apart of that, it would be interesting to join 4 different models trained to detect each one of the categories in order to have a system than can tag each one of the space ships into one class.

5. Model Improvements

We decided to make some improvements in our system by adding 3 more features described in question 11 (entropy of colour, perimeter, entropy of texture). We implemented those features as follows:

- Entopy of colour: `entropy(relevant_pixels)` where `relevant_pixels` are just the pixels of our object

- Perimeter: `perimeter/box_perimeter`

- Entropy of texture: `entropy(entropyfilt(relevant_pixels))` where `relevant_pixels` are the greyscale pixels of our object

After adding those extra features our scores are as follows:

AUC	P_D	P_FA
0.79	0.57	0
0.81	0.63	0
0.91	0.88	0.06
0.69	0.38	0
0.78	0.67	0.11
0.67	0.57	0.24
0.81	0.67	0.06
0.8	0.67	0.07
0.81	0.63	0
0.83	0.71	0.06

Little improvement (if any) means that features, or one of the features, was chosen incorrectly, because it didn't bring any crucial information and created more complex model with higher dimensionality, hence longer execution time.

We also applied some data augmentation to our model, using a Python script to double the dataset by rotating, mirroring and/or applying noise to the images randomly. This, together with the new features extracted, increased the training of the model a lot, but it also improved the accuracy. After these upgrades, the AUC never went under 0.8, being the average around 0.9.