EEE4114F

Final Project

Developing a machine learning algorithm for shape classification

Jake Burditt
BRDJAK002

Samuel Pogrund
PGRSAM001

May 2022

Declaration:
1. I know that plagiarism is wrong. Plagiarism is to use anotherˆas work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this project. from the work(s) of other people has been attributed, and has been cited and referenced. Any section taken from an internet source has been referenced to that source.
3. This project is my own work, and is in my own words (except where I have attributed it to others).
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing
it off as his or her own work.
5. I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and
declare that this is my own work.
Signature S.Pogrund, J.Burditt

Introduction

Shape Classification

The aim of this project was to develop a machine learning algorithm to classify hand drawn shapes into the category of shape in which they belongs.
Shape classification plays a very important role in computer vision, such as being used in object recognition and image retrieval.
We looked at using a forest tree classifier which involves using multiple decision trees to achieve accurate classification.

Problem development

Neural Networks

Humans are able to identify shapes from a very young age, even when the shapes are not exactly conforming to what is expected. The question we are investigating is how can we train a neural network to accurately predict shapes with a good confidence level.

Forest tree classifier

Another good option for this approach is a random forest classifier. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Literature Review

When looking at research done on shape classification using machine learning, Zhang et al, in SCN: A Novel Shape Classification Algorithm Based on Convolutional Neural Network, states that shape classification works best when image pixels are converted to binary values. This is because using RGB values gives the network too many network parameters and will take much more computational power. They also state that the best results are achieved when a convolutional network is used.
An approach taken by babu et al, in Book cover Machine Learning Based Shape Classification Using Tactile Sensor Array, is to use a voting based classifier which uses a c4.5 algorithm and a naive Bayes classifier and uses both of the results to achieve a more accurate classifier.
In an article, Tony Yiu suggests that random forest classifiers are best for using data that can be represented in binary. Drawing from the papers looked at previously, the first step in shape classification is converting the pixels to binary values, thus, it would make the most sense to use a decision tree and therefore by extension a random forest classifier to quickly and effectively classify shapes.

Problem Statement

After looking at previous research, the majority of the research was done on well structured shapes. Therefore shape classification on hand drawn shapes, being very unstructured(depending on the drawer), can be very useful in image processing.

Method

Framework
The framework we decided to use for this project is Jupyter notebook with an intro-ml kernel provided by our lecturer Jarryd Son. This is because we have had the most training on this platform through the lectures and labs and there are many online resources and lots of online support to help us on the chosen platform.

Packages
One of the main packages used when developing the classifier was PIL, python imaging library. This package helps to add image processing capabilities to the project.
Another important package used was the sklearn package, which provides tools for predictive data analysis and machine learning. The specific package used from here was the random forest classifier package which builds decision trees which classify the data.

Data used
Shape set
The data set, downloaded from kaggle.com, consisted of over ten thousand PNG images of squares, circles, stars and triangles. The images consisted of black and white pixels only and were 200 x 200 pixels wide each. The images contained the same shapes in slightly different orientations but essentially the same size. The first classifier was trained on this original data set, however the lack of range in sizes of the shapes set led to the classifier overfitting and performing badly when tested on images with larger or smaller shapes. Thus, it was clear further data would be needed to create an accurate model.

Reworking data
A random subset of the original data set was selected to be resized to a variety of different scales. This was done by importing the image, using the Python Imaging Library, scaling the image using the zoom_at function and resaving the new image, again using the Imaging Library. These new images were used to train a second model.

Model
Random forest algorithms make use of several learning methods in order to solve, most commonly, classification and regression tasks. This is achieved by constructing a ‘forest’ of decision trees while being trained.

Decision trees use the attributes of the items in a data set to draw conclusions about that item. To continue the analogy, the item's attributes, and their relation to each other are represented as branches and the class labels reached are the leaves of that tree. Decision trees tend to overfit their data and the deeper the tree grows the more they tend to do so, thus they often have a low bias at the expense of a high variance. To correct this, random forests use a number of decision trees, each trained on a different part of the data set, to reduce this variance and produce a far more accurate model than each tree. For classification tasks the forest algorithm uses the mode of the outcomes of the trees to classify the item in question.

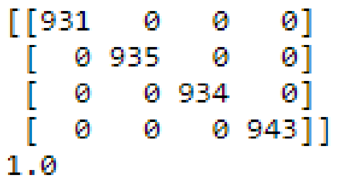
In order to construct the forest, the algorithm needs to partition the training set into a number of subsets, this is done using bootstrap aggregation. The process of bootstrap aggregation repeatedly selects, with replacement, samples from the data to construct decision trees. The samples are selected at random.

Furthermore, the training set can be further split by the algorithm by selecting a random subset of each sample's features when bootstrapping. This is referred to as feature bagging. This can be done to find features that are stronger predictors of the outcome and thus these features will be used by more of the trees in the forest.

Training, testing and validation

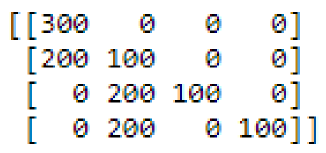
Results

Classifier 1



```
[[931   0   0   0]
 [  0 935   0   0]
 [  0   0 934   0]
 [  0   0   0 943]]
1.0
```

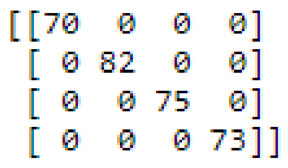
Confusion matrix of first classifier on original dataset



```
[[300   0   0   0]
 [200 100   0   0]
 [  0 200 100   0]
 [  0 200   0 100]]
```

Confusion matrix of first classifier on modified dataset

Classifier 2



```
[[70  0  0  0]
 [ 0 82  0  0]
 [ 0  0 75  0]
 [ 0  0  0 73]]
```

Confusion matrix of second classifier on modified dataset

```
[[ 44 465 422   0]
 [  0 212 569 154]
 [  0   0 831 103]
 [  0  13 574 356]]
```

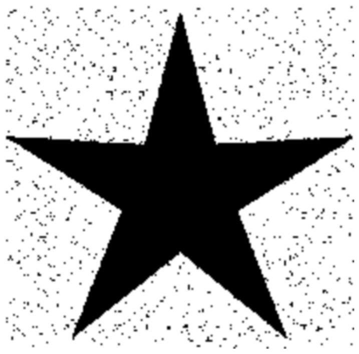
Confusion matrix of second classifier on original dataset

Classifier 3

```
[[1001    0    0    0]
 [   0 1017    0    0]
 [   0    0 1009    0]
 [   0    0    0 1016]]
```

Confusion matrix of second classifier on both original and modified dataset

Individual Image Tests:

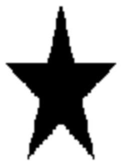


```
Out[28]: array(['star'], dtype='<U8')
```

Test 1

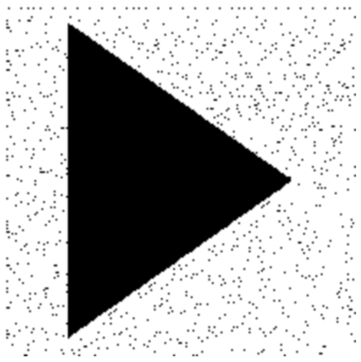
```
Out[23]: array(['star'], dtype='<U8')
```

Test 2



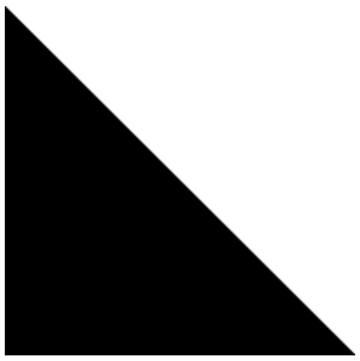
```
Out[24]: array(['star'], dtype='<U8')
```

Test 3



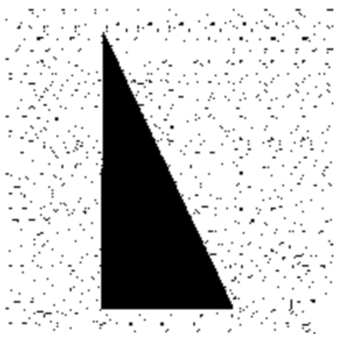
```
Out[27]: array(['triangle'], dtype='<U8')
```

Test 4



```
Out[26]: array(['triangle'], dtype='<U8')
```

Test 5



```
Out[25]: array(['triangle'], dtype='<U8')
```

Test 6

```
Out[33]: array(['square'], dtype='<U8')
```

Test 7

```
Out[30]: array(['square'], dtype='<U8')
```

Test 8

```
Out[32]: array(['circle'], dtype='<U8')
```

Test 9

```
Out[31]: array(['circle'], dtype='<U8')
```

Test 10

Analysis and discussion

Based on the results above, the first classifier performed well when tested against data of the same size, yet performed very badly when tested against the data set including images of shapes of varying sizes. The original model was overfit to its data set and appears to be classifying the second set only based on the size of the black pixels in the image.

As can be seen in the confusion matrices, second classifier appeared to have the same issue, of overfitting, as the first classifier and thus performed well on similar data but poorly on the original set. This low bias but high variance is common for random forest classifiers and would be expected for this sort of problem.

The third classifier appears to overfit less than the previous two as it correctly classifies the entire test set, which of course contains both modified and the original data. The overfitting problem appears to have been improved. This can be further seen in its classification of the images gathered randomly from the internet.

Conclusion

Although the random tree classifiers tend to overfit to their data, with retraining this overfitting can be reduced. Thus, given the simplicity of their implementation, they are a useful solution to the classification of basic images.

References

Zhang, C., Zheng, Y., Guo, B., Li, C. and Liao, N., 2021. SCN: A Novel Shape Classification Algorithm Based on Convolutional Neural Network. Symmetry, 13(3), p.499.\\

Babu,D. Bhattacharjee,S. Bandyopadhyaya,I. and Roychowdhury,J. 2021. Machine Learning Based Shape Classification Using Tactile Sensor Array. Smart innovation,Systems and technologies. volume 27.

Yiu, T., 2022. *Understanding Random Forest*. [online] Medium. Available at: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2> [Accessed 20 May 2022].

Four Shapes Dataset
https://www.kaggle.com/datasets/smeschke/four-shapes
[Accessed 18 May 2022]