Machine Learning Engineer Nanodegree

Capstone Project

Luke Morson
3 November, 2018

Proposal

Domain Background

Optical character recognition (OCR) deals with the conversion of images into machine encoded text. There has been a resurgence in interest in OCR in recent years, mainly for digitizing documents to increase the re-usability of data.

There are many applications of OCR and under certain conditions, such as with cleanly scanned book images, OCR is close to being a solved problem. OCR engines like Tesseract, which is now open source, have been in development since 1985. There are however a number of interesting challenges where there is still active research such as in character recognition in natural images[1] and handwriting detection.

Problem Statement

Character recognition in scene images is potentially a harder task compared to more classical OCR problems due to the many possible variations in background, texture, font and resolution.[2] Text in natural scenes contains a lot of valuable information, such as street signs or shop names, and an algorithm that is able to do text detection and recognition has a lot of use cases.

A complete text recognition system includes two general tasks: text localisation and word recognition. The text localisation detects the text locations from the image while the word recognition identifies the characters and then the words.[3] This project will focus on the recognition of single characters in scene images.

Datasets and Inputs

The Chars74K dataset contains symbols that can be used for character recognition in natural images. It contains symbols from English with 62 classes with the possible characters of 'A-Z', 'a-z', and '0-9'. [4]

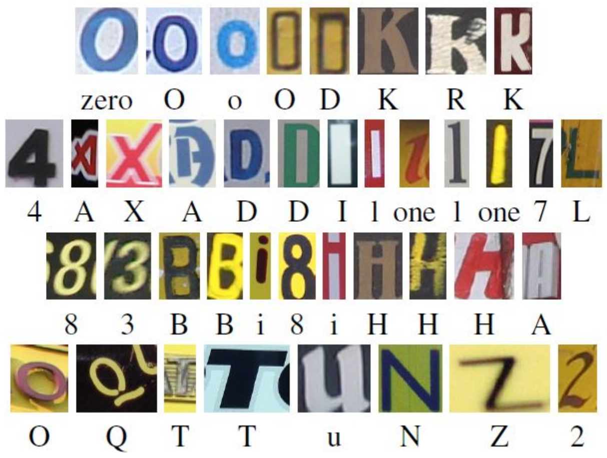

Fig1. Shows an example of the images from the Chars74K dataset.

The download *EnglishImg.tgz* contains a folder *GoodImg* with 7705 images of characters in natural scenes. The class distribution has been briefly explored [here](). There is an imbalanced distribution of the 62 classes as shown below in *fig2*.

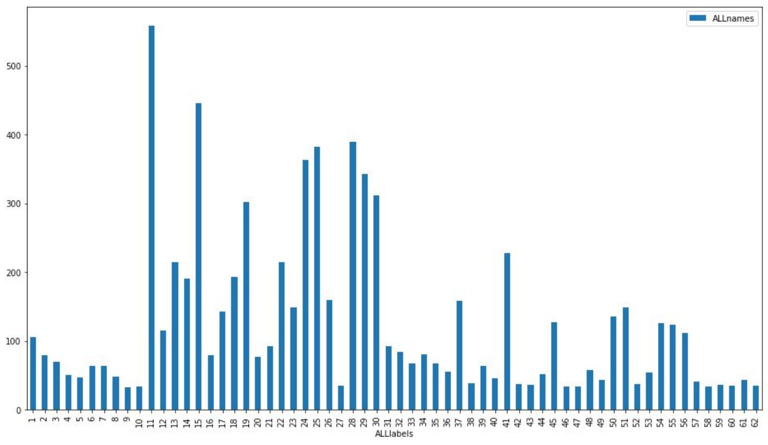

Fig2. Shows the class distribution of the 7705 images of characters to be used in the project

The images have different shapes and sizes so will need to be normalised to be used for the model. I will experiment with converting to different image sizes to determine the impact on training time and accuracy.

Solution Statement

A convolutional neural network (CNN) will be used as it known to perform well on this sort of problem. CNN are a good solution because they can be applied to problems where you want to find patterns within images. The solution will involve number pre-processing and data augmentation steps that will optimise the data for the CNN.

Benchmark Model

This model achieved an accuracy of around 86% using a CNN approach on the Chars74K dataset.[6] The conclusion of this model was that there was little room for improving the accuracy without the risk of overfitting. The main source of errors such as a difference between 0 and O would be difficult for a human to determine without more context, which isn't available in this single character dataset. The main area for improvement is identified as improving the classification of rotated characters.

Evaluation Metrics

As there is a class imbalance in the dataset the predictions will be evaluated using the F1 score:

F1 = 2 * (precision * recall) / (precision + recall)[5]

The aim of the project is to correctly classify labelled characters therefore the accuracy will reflect how well the algorithm is able to correctly classify the characters. The accuracy on the training and testing set can be plotted over different training iterations steps.[6]

Project Design

The project design will follow the following steps with exploration at each stage therefore this is an initial outline of the design. As the project develops, the design is likely to expand and change based on the results.

Preprocessing

The images from the Chars74K subset *EnglishImg.tgz* will be used along with the data structure definitions in *ListsTXT.tgz.*

The images in the dataset are mostly in colour but will can be converted to grayscale. It is computationally less expensive to work with a single-channel image and should have no negative impact on the accuracy as the text largely have a high contrast with their background. [6]

All the images are different sizes so will require some normalisation. I will experiment with converting to different image sizes and with the aspect ratio to determine the impact on training time and accuracy. This source used 32x32 images because of the perceived trade-off between speed and accuracy but I can also explore other dimensions such as 64x64.[6]

The dataset can will be randomly split into 60% training set, 20% validation set and 20% test set.

Data augmentation

Neural networks require large amount of labelled data to train. One method to increase the amount when we have a small dataset is to use data augmentation. This project states that data augmentation could help towards better generalisation.[7] Although it's not clear if this will be included in the scope of this project I can explore how augmentations like rotation, stretching, translation and noise-adding affect the overall accuracy.[8]

CNN architecture

The typical CNN architecture follows the pattern of an Input Layer, feature-extraction layers and finally a classification layer. For image classification a typical CNN architecture has multiple convolutional layers followed by pooling layers.

The initial CNN architecture to be explored will have the following layers. This will serve as a starting point to experiment from:

- Conv2D: 3x3 kernels | 32 maps | relu | input shape = (1,32,32)
- MaxPooling2D | 2x2 pool size | 2x2 stride
- Conv2D: 3x3 kernels | 64 maps | relu
- MaxPooling2D | 2x2 pool size | 2x2 stride
- Conv2D: 3x3 kernels | 128 maps | relu
- MaxPooling2D | 2x2 pool size | 2x2 stride
- Fully Connected | 512 units | rule
- Dropout (0.3)
- Fully Connected | 62 units | softmax

Other considerations:

- Loss function using categorical cross entropy
- Explore the best optimization solution with stochastic gradient descent, ADAM, ADADELTA etc.
- Use Keras with a Tensorflow backend on AWS

References

1. http://personal.ee.surrey.ac.uk/Personal/T.Decampos/papers/decampos_etal_visapp2009.pdf
2. https://ai.stanford.edu/~ang/papers/icdar01-TextRecognitionUnsupervisedFeatureLearning.pdf
3. http://www.mirlab.org/conference_papers/international_conference/ICASSP%202016/pdfs/0001322.pdf
4. http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/#download
5. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
6. http://ankivil.com/kaggle-first-steps-with-julia-chars74k-first-place-using-convolutional-neural-networks/
7. https://github.com/fhennecker/chars74k
8. http://www.diva-portal.org/smash/get/diva2:1108415/FULLTEXT01.pdf