Report

David Saunders (910995)

April 2020

Abstract

Write abstract here. 4 page report!

Contents

1 Introduction

Contextualise the machine-learning problem and introduce the task and the hypothesis. Make sure to include a few references to previous work. You should demonstrate an awareness of the research-area.

Image detection is a classic machine learning problem, with hundreds of papers on the topic. History of image detection is relatively recent. Maybe started with the MNIST handwritten digit classification problem. Is regarded by many as the 'Hello world' of machine learning/CNNs [1]. One guy on kaggle used a cnn for the first time to great effect. Other more advanced problems such as image recognition, gesture , emotion detection now possible thanks to deeper networks, and convolution's layers.

Maybe touch on the problem some people have of using too advanced method for the task? Especially with stuff like pretrained CNN specialised for facial recognition used to detect a black or white dot for example.

2 Methodology

The model(s) you trained to undertake the task. Any decisions on hyperparameters must be stated here, including motivation for your choices where applicable. If the basis of your decision is experimentation with a number of parameters, then state this.

Break up the projected images into windows of size 51x51, which is the same as the samples in the training data. THen the x,y,z, coordinates are found by measuring the distance to each sphere. This is done by taking the depth of the middle pixel and moving to the closest one. Decide to look, rotate by pi/6 if nothing is found.

2.1 Classifier 1: Convolutional Neural Network

Decided to use a Convolutionial Neual Network, due to their tried and trusted effectiveness at image detection. Detecting the red sphere is a trivial use of the technology so accuracy is expected to be very high. Problem will be with the training data not relating to the real life data where the spheres will be obscured and different sizes from the training. Problem of training/archived data not matching production is a well known issue (with prediction and stuff) TODO: ref.

All of the architecture of the CNN will be explained in this paragraph and all of the hyperparameters explained. (Maybe picture of model summary?, or look at a tool online to visualise) The CNN used in this task is similar to other networks such as Imagenet where there are a series of convolutional and pooling layers, followed by a fully connected layer [2].

First layer is $layers.Conv2D(32,(3,3),input_shape=(51,51,3)$ which inputs the training image and uses a kernel size of 32 pixel. Activation of each internal layers use relu as that is default for whatever reason. Alternative is to use something like sigmoid, but relu is now generally used because xyz.

Followed by a max-pooling layer to ..... do what they do. Why max-pooling is used and not min pooling, or average pooling. Pooling size is 2, 2 WHY and what exactly does it mean?

Second layer is the same but with a different input shape for obvious reasons. Followed by another activation and pooling.

Third layer is the same but with an initial kernel size of 64. Bigger kernel size is choice as the hope is that the lower levels will get smaller or more localised features but with a larger kernel we will hope it pieces the smaller features together. However as image recognition problems go this one is straightforward so it really shouldnt need to piece together layers of features. Regardless when a shallower network was tried it didn't work so here we are.

Then we flatten the layers so that we can feed the outputs from the convolutional layers into a more traditional dense network of nodes. This is fed into 64 neurons, which are then fed into a single neuron which will fire if a sphere is detected. Between the layers there is a dropout amount specified at 50%, this means that half of these neurons randomly wont fire. This method has

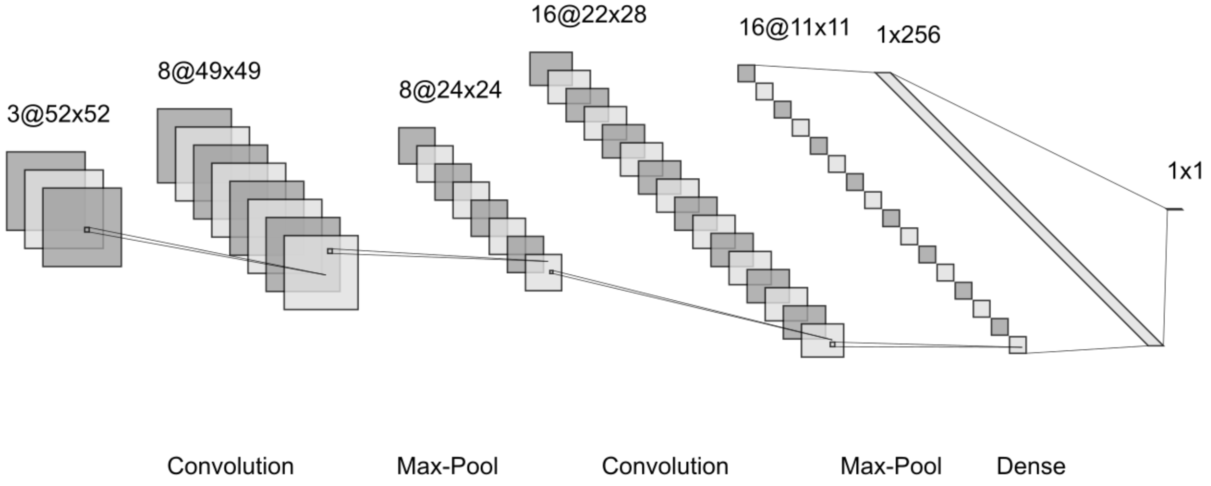

Figure 1: A diagram of the network showing the layers [3]

been shown to reduce overfitting as each hidden neuron cannot rely on another neuron firing (weak explanation) [4].

Loss function is chosen to be $binary/_crossentropy$ as it is the best choice for binary classification. Why did I choose rmsprop as an optimizer?

Data currently trained with 5 epochs. Obviously an epoch is a full pass through the dataset and we're only given a small dataset to work with. 5 epochs might be too much, the accuracy doesn't improve after the first epoch so delete the rest and see what accuracy is like then? TODO: Reference of course. `machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/` This article list loads of nice academic sources about how to avoid overfitting.

The model is then saved to disk. This is so the network doesn't have to be retrained all the time and to help reproducibility. (Get reference to why you should always save a model) Since networks are non-deterministic they may not always have the same result (ref).

2.2 Classifier 2: Dummy classifier

CNN is good and all but is it too complicated for the job?

while messing around it was found that multiple layers were needed to properly classify the spheres. Could manual feature engineering help? (Feature engineering is a good word just TODO: find reference.) In the training data the red sphere is always in the centre, with the center pixel having a very high red intensity, with a low intensity for the green and blue channels. Idea was that by using human knowledge to extract features using domain knowledge could prove just as effective as state of the art machine learning. Perhaps there'd even be benefits of a computational speedup when a more simple method was used.

3 Results

Describe, compare and contrast the results you obtained on your model(s). Any relationships in the data should be outlined and pointed out here. Only the most important conclusions should be mentioned in the text. By using tables and confusion-matrices to support the section, you can avoid describing the results fully.

The methods of completing the PacMan game as discussed above were successful when the CNN architecture was used. The main for loop of the program ran for a time of 12 minutes, and every sphere was detected and moved to flawlessly. The time is high however that is not because of the classifier used but rather the function of projecting the point cloud to the 5 different maps that bottlenecked the system. Neural networks are even known for their speed of classification once they've been trained.

Confusion matrix of CNN: (Only text or with a nice graph)

As discussed NNs are fast at classification, but how does that compare to an even simpler method of classification. To complete the game using the dummy classifier took only 9 minutes. Despite the classifier not being the bottleneck, there was a notable speed increase. One possible explanation for the speedup is that the dummy, classifier is less robust and adaptable to channge than the CNN. Sphere would have to be directly in centre of screen, meaning less spheres were identified and so less things to process? TODO: Look at results to see if this theory holds any weight at all.

TODO: time both classifiers on the same starting test image, and then sliding windows classifier measured to compare time differences.

Confusion matrix of dummy: (Only text or with a nice graph)

4 Discussion and Conclusion

Restate the task and hypothesis/-ses concisely. Reiterate the methods used. Describe the outcome of the experiment and the conclusion that you can draw from these results in respect of the hypothesis/-ses.

Task at hand was a simple binary image classifier that was to be used in conjunction with other methods to play a game moving thorough 3D space. A CNN was used as they are the de facto, tried and tested method of dealing with this problem. Despite this a simple dummy classifier was created using manual feature extraction which performed just as well. This shows how there is not always a need for the most advance options and that given some knowledge of the problem, manual feature engineering can be the most effective.

References

[1] TensorFlow Tutorials. *Basic classification Classify images of clothing.* Apr. 2020. URL: `https://www.tensorflow.org/tutorials/keras/classification` (visited on 19/04/2020).

[2] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems.* 2012, pp. 1097–1105.

[3] Alexander LeNail. "NN-SVG: Publication-Ready Neural Network Architecture Schematics". In: *Journal of Open Source Software* 4.33 (2019), p. 747. DOI: `10.21105/joss.00747`. URL: `https://doi.org/10.21105/joss.00747`.

[4] Geoffrey E Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).