Report

David Saunders (910995)

April 2020

1 Introduction

Image detection is a one of the most classic machine learning problem, with hundreds of papers on the topic. Arguably the most influential paper on the topic "Imagenet classification with deep convolutional neural networks" showed how deep convolutional neural networks are considerably better at image classification than other methods. The task of automatically playing a 3D game of PacMan poses a novel use of the technology. From a starting position the points must be projected from 3D to 2D and scanned using image recognition to detect a series of 11 spheres located in a pointcloud. Detection of pointclouds using deep learning methods is still a developing field so for this task the 2D image will be used [Guo+19]. The spacial coordinates of the sphere will be computed, and the position will be updated to that of the sphere. That sphere will then be marked as collected, this process will then be repeated until all 11 spheres are collected.

With machine learning there is a tendency to always use the newest and most complicated model when a simpler model may perform comparably [Sok20]. This bias-variance tradeoff can be used to find a balance between the complexity of a model and its effectiveness, where more complicated models are not always more effective [Nea+18]. Large complex neural networks can suffer from the problem of explainability. It is common regard the inner workings as an unexplained "black box" [Zed19]. Alternatively simple dummy classifiers are often used as a baseline to compare to other more advanced models [sci19].

This report will design a relatively advanced CNN and a simpler dummy classifier. Their effectiveness in completing the task will be compared.

2 Methodology

The images obtained from the pointcloud have shape 160x240x3, where as the given training images have a different shape of 51x51x3. This means that images from the pointcloud cannot directly be input into a classifier created from the given images. Instead a sliding window approach will be used, to split larger image into windows the same size as the training data. The windows will overlap strongly so that the set of windows will contain multiple partial spheres [FP02].

We will attempt to train the classifiers so that it only recognises a sphere if it is in the centre of an image, therefore the centre pixel of the image will always be the depth. The camera will then move to the sphere with the lowest depth, which will be the closest sphere. The X,Y,Z coordinates of the sphere will be found from the middle pixel value for the projected mappings. If no spheres are detected in any window then the viewpoint will rotate $-\frac{\pi}{6}$ radians in the Y axis until a sphere is detected. This process will repeat until all spheres have been collected.

2.1 Classifier 1: Convolutional Neural Network

It was decided to use a Convolutional Neural Network (CNN) as a sphere classifier 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. A benefit of using a CNN is that we do not have to worry about feature extraction, the training data can be input directly into the network.

All of the architecture of the CNN will be explained in this section and all of the hyperparameters explained. The CNN used in this task is similar to previous Imagenet classification CNN's where there are a series of convolutional and pooling layers , followed by a fully connected layer [KSH12]. The architecture used in this CNN is heavily inspired by the LeNet-5 Convolutional neural network, originaly used for handwritten digits recognition. This architecture was chosen as it is one of the first CNN's to be proposed and it is relatively shallow compared to newer deep learning methods, so training time and memory usage will be less intensive [LeC+98].

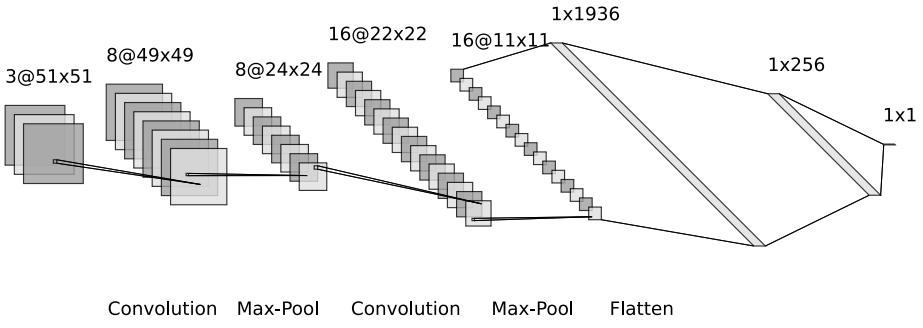

Figure 1: A diagram of the network showing the layers of the network designed for this task [LeN19].

Figure 1 shows the network consists of two convolutional layers , which are then flattened and fed into a dense fully connected layer , which leads to a

single output neuron. Each convolutional layer is followed by a maxpooling layer, which is used to downsample the image. It is hoped that max pooling will pick up any feature found by the earlier layers.

All internal nodes of the network use the standard relu as the activation function, other than the single output neuron which uses the probability based sigmoid function. The kernel size of the convolutional kernels were chosen arbitrarily as 3x3, and the pooling layers used are 2x2. Due to the nature of kernel convolutions the input size of each layer will change slightly. This will be detailed below.

The first convolutional layer takes in a 51x51x3 image, and outputs 8 filters with a reduced size of width and height of 49x49. The filters are then max pooled to give a new shape of 24x24x8. The second convolutional layer inputs the downsampled results and outputs a higher number of 16 filters. The number of filters is increased in an attempt to group together any lower level features that have been identified. Compared to other image classification tasks this one is trivial so there is no need for more layers to detect low and high level features. Regardless when a shallower network was tried it didn't work so here we are. Pooling after this layer results in an output of shape 11x11x16. After the convolutional layers we then flatten the output into a vector of length 1936 so that it can be input into a more traditional dense network of nodes. A layer of 256 neurons is used as the hidden layer, which connects to a single neuron which predicts what class the input image belongs to. This is given as a probability from 0 (no sphere present) to 1 (sphere present). Between the hidden and output layers there is a dropout amount specified at 50%, this means that half of these neurons randomly wont fire. This method has been shown to reduce overfitting as each hidden neuron cannot rely on another neuron firing [Hin+12].

When training the network the loss function is chosen to be binary crossentropy the CNN is predicting binary data. The adaptive learning rate method Adam was chosen as optimizer as it has low memory requirements and works with little tuning of the hyperparameters [Che18]. The data will be trained with 5 epochs, as this should maximise accuracy but prevent overfitting which is an issue with such a small dataset.

The CNN will be used to classify a sphere only if there is a confidence of at least 99.9% to avoid false positives.

2.2 Classifier 2: Dummy classifier

A CNN should be able to detect a sphere, but could a simpler method complete the task just as effectively? Feature engineering is the process of selecting features for a model using domain knowledge and some claim it is the most important factor to the success of a machine learning project [Dom12]. From the training data we can see that the centre pixel of a positive sample contains a high intensity for the red channel and low intensities for the green and blue channels. Without using any machine learning methods a classifier was created that looks at the intensity of a single pixel and predicts if an image contains a sphere. A dummy classifier can be defined as "a classifier that makes pre-

dictions using simple rules" so that is what we wil call this classifier [sci19]. If the classifier is successful then it will show using knowledge to extract features could prove just as effective as state of the art machine learning. There may even be computational speedup benefits when a more simple method was used.

3 Results

The PacMan game was successfully completed when both methods of classification were used. The main for loop of the program took approximately 16 minutes to complete, regardless as to whether the CNN or dummy classifier was used.

The accuracy of the CNN was 99.7% (3.s.f) as expected from a powerful detection network. Only 2 of the test images were falsely identified as negatives and there were no false positives. The dummy classifier was surprisingly effective for such a simple classifier, with a comparable accuracy of 99.0% (3.s.f). Similarly there were no false positives but a higher 6 false negatives. The impact of false negatives in this task are mitigated by the sliding window algorithm. Since a sphere appears in multiple windows the classifier has multiple chances for detection.

The biggest bottleneck to the system is not the chosen classifier, but rather the function of projecting the point cloud to the 5 different maps which is very time intensive. Any difference in time between the classifiers is unimportant to this task as the classifier does not significantly impact the time of the task.

4 Discussion and Conclusion

The task performed in this report was to create simple binary image classifier that was to be used in conjunction with other methods to play a game similar to PacMan, moving thorough 3D space. The classifier was used in conjunction to a sliding window algorithm to to detect the presence of a red sphere in an image. A CNN was used as they are known to be effective at image classification. A simple dummy classifier was also created using manual feature engineering, which was found to perform the task just a well just as well as the more adveanced classifier. 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 just as effective.

References

[Guo+19] Yulan Guo et al. "Deep Learning for 3D Point Clouds: A Survey". In: *arXiv preprint arXiv:1912.12033* (2019).

[Sok20] James Sokolowski. *How To Keep Our AI Obsession From Overcomplicating Data Projects*. Apr. 2020. URL: https://www.forbes.com/sites/forbestechcouncil/2020/04/20/how-to-keep-our-ai-obsession-from-overcomplicating-data-projects/#7b6659b55236 (visited on 21/04/2020).

[Nea+18] Brady Neal et al. "A modern take on the bias-variance tradeoff in neural networks". In: *arXiv preprint arXiv:1810.08591* (2018).

[Zed19] Carlos Zednik. "Solving the Black Box Problem: A Normative Framework for Explainable Artificial Intelligence". In: *Philosophy & Technology* (2019), pp. 1–24.

[sci19] scikit-learn. *sklearn.dummy.DummyClassifier*. 2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html (visited on 20/04/2020).

[FP02] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002, pp. 519–539.

[KSH12] 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.

[LeC+98] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[LeN19] 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.

[Hin+12] Geoffrey E Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[Che18] Chengwei. *Quick Notes on How to choose Optimizer In Keras*. Feb. 2018. URL: https://www.dlology.com/blog/quick-notes-on-how-to-choose-optimizer-in-keras/ (visited on 19/04/2020).

[Dom12] Pedro Domingos. "A few useful things to know about machine learning". In: *Communications of the ACM* 55.10 (2012), pp. 78–87.