Cognitive Robotics CW - Deep Learning and Robotics

Kieran Shave
University Of Manchester

1 State of the Art

Robotics encompasses the study of robots and their autonomous and purposeful sensing and acting in the physical world[7]. Cognitive robotics can be defined as "the field that combines insights and methods from AI, as well as cognitive and biological sciences, to robotics" [2]. A large wave of research into the application of deep learning to robotics has occurred over the past decade, as the unique challenges that robots face are usually not always addressed by the fields of computer vision and machine learning[8]. The main factor for the application of deep learning to robotics is that it is more general then other available learning algorithms, the ability for neural networks to perform high level abstraction make them ideal tools for robots acting in unregulated environments[8].

Currently, the most natural application of deep learning to robotics is in the field of robotic vision[2]. Convolutional Neural Networks, or CNNs, are currently the most widely used deep learning architectures for computer vision problems. CNNs have demonstrated exceptional performance in object recognition tasks, methods like pre-processsing, dropout, batch normalisation and ensemble learning have contributed to the performance increase seen by CNNs[6]. EnsNet is an example of one of the currently highest performing CNNs at the task of image classification, achieving some of the lowest error rates among other state of the art models[6].

Deep learning models in robotics are often used as preprocessing units for local images and measurements, where raw sensor data can be converted into a feature space with fewer dimensions able to be used for robot control[8]. Current challenges for deep learning in robotic vision stem from the idea that a robot must consider that its immediate outputs will result in actions in the real world[10]. This paradigm is not present in computer vision and has motivated research into new challenges which are split into three conceptional domains: learning, embodiment and reasoning[10].

Another robotic function that deep learning has recently been applied to is tactile sensing, for tasks such as object recognition, tactile properties recognition and grasping[2]. Prior to the use of deep learning, tactile sensor data was used only with handcrafted features or to trigger specific actions. These methods would not scale well with advances in tactile technology where larger amounts of data are available but can be handled by deep learning methods[2].

Deep learning methods can also be applied to other robotic tasks such as speech recognition and mobile robot navigation. Communication between a worker and robot sharing a common workspace is key to ensure the safety of the worker[4]. An example research area is the use of deep learning cloud based speech processing systems to facilitate this communication, as explored in Deuerlein et al[4]. In the context of robot navigation, the powerful representation capabilities of deep learning technology has suggested the use of systems which directly learn navigation strategies from unaltered sensor inputs[11]. This approach avoids the computational errors generated by the combination of aspects of the traditional navigation framework, which relied on a noise-sensitive high precision global map which limited its ability in unexpected environments[11].

The development of deep learning methods over the previous decade have provided new solutions for a range of robotics domains, and continues to be a focus of research for the development of robot capabilities.

2 DNN Classification Task Introduction

In this coursework I have explored the image classification performance effects of changing three hyper parameter values for a convolutional neural network. Image classification is the task of assigning a label or class to an entire image. Image classification models take an image as input and output a prediction about which class this image belongs to[1]. These models are trained by using example images who's image classes are known.

I choose to use the iCubWorld Transformations dataset to train and test the DNN. This dataset contains a large volume of images captured by an iCub robot for over 200 different objects split into 20 categories. Each object has also been captured undergoing 4 viewpoint transformations, these are 2D rotation, 3D rotation, scaling and background change. Another set of images, named MIX, was captured with a human moving an object in front of the robot, in this set all image viewpoint transformations could occur and I decided to use these images as my dataset. The variation in image viewpoint provides a realistic view of how a robot would capture an object in a real-world scenario, and so the MIX images are an interesting dataset to use to train the CNN. I also decided to only use 10 of the available object categories as this would decrease the size of my training set, which would decrease the time taken to train the network. My dataset therefore was made up of 53103 images split over 10 object categories.

Our DNN topology is outlined in Figure 1. This network was taken from the Lab2b notebook where it demonstrated image classification capabilities on the CIFAR-10 dataset. I wanted to explore how this network would perform on a much larger dataset.

The hyperparameters I decided to explore were kernel size, dropout rate and batch size. I chose to modify kernel size and dropout rate to explore the robustness of the network, a similar approach is taken in Chen et al[3] and this was inspiration for our choice. I changed the kernel size of the first 3 Conv2D layers and the dropout rates of the first 2 Dropout layers shown in Figure 1. Batch size was chosen to explore how its modification would change the performance of our network. The kernel sizes we chose were 3x3, 5x5 and 7x7, dropout rates were 0.25, 0.35 and 0.45 and batch sizes were 32, 64 and 128.

3 Results and Analysis

Each testing simulation trained the model for 3 epochs resulting in around 7.5 hours of total training time. Using a larger number of epochs may have resulted in more accurate testing results but would have required significantly longer to train. My train-val-test split was 0.8, 0.1, 0.1 resulting in 42478, 5306, and 5319 images being used for training, validation and testing respectively. The object categories that I chose to use from the iCub World dataset were the MIX images of books, cellphones, hairbrushes, hairclips, mouses, pencilcases, perfumes, ringbinders, sunglasses and wallets. The network was trained to classify

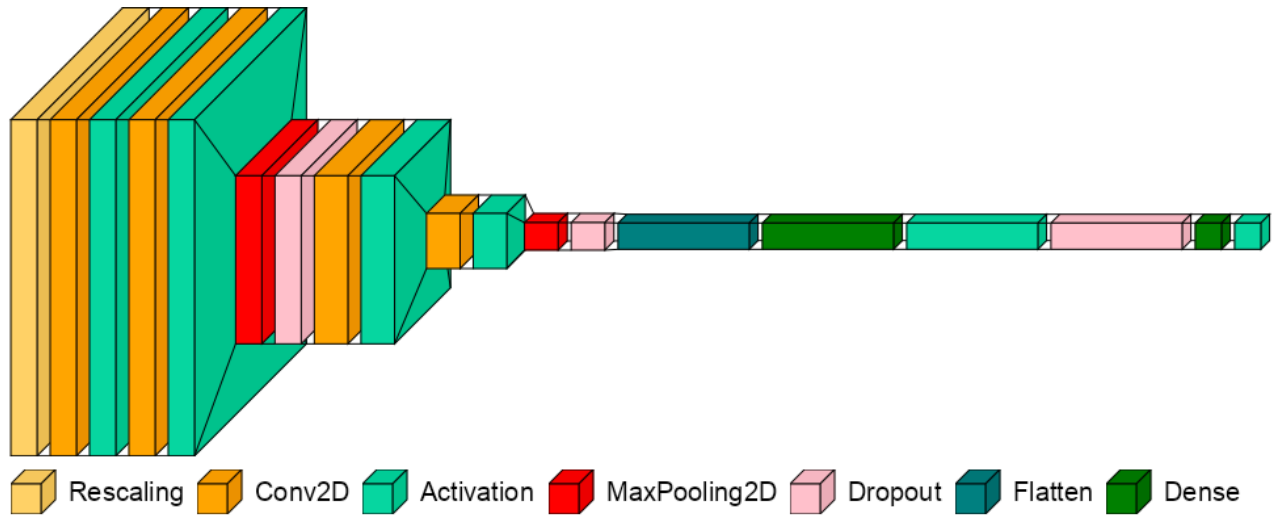

Figure 1: Topology of the DNN chosen. Image generated using the visualkeras library[5].

each of the input images into one of these categories. The results of the 27 hyperparameter testing simulations I ran are displayed Table 1, the training and validation values were taken from the final training epoch for each experiment. Figure 2 displays the changes in testing accuracy for different kernel sizes and dropout rates for each batch size.

The best performing hyperparameter combination was a kernel size of 3, dropout rate of 35% and batch size of 128, however the 25% dropout rate with the same kernel size and batch size achieved nearly the exact same testing accuracy. For all batch and kernel sizes, a dropout rate of 25% was the most consistent at obtaining the highest or near highest testing accuracies, except for a kernel size of 7 and batch size of 64. From these results, I suggest that a 25% dropout rate is the best hyperparameter setting out of the ones I have tested. However, our results are varied and more experimentation would be needed to strengthen this claim.

In our experimentation we only trained and tested the network with each hyperparameter combination once, taking this obtained value as our final result. This has reduced the quality of our results, as any analogous testing accuracies cannot be detected. If we had more time or access to more powerful machines, we would take the average of several runs of each hyperparameter combination to increase the precision of our results. This would much improve the quality of our current results by smoothing the effects of analogous testing accuracies, and perhaps introduce an easier to view trend between combinations of hyperparameters.

An interesting observation was that the networks final epoch validation accuracy was higher than its training accuracy for all hyperparameter combinations. This perhaps can be explained by the use of dropout layers which, during training, disable a % of units to reduce overfitting and improve overall performance of the network[9]. During testing and validation, these dropout layers are disabled to allow the network to access all units, which improves the accuracy of the network during testing and validation, as seen in my results.

By using the iCubRobot transformations dataset, we have access to a large volume of excellent quality training data which includes large viewpoint variation between images, especially when using the MIX set of images which include all possible viewpoint transformations of objects. Therefore, one perhaps should not be as worried about overfitting since our model will be exposed to a large range of object transformations which could allow the model to be robust to variations in newly observed images. With this in mind,

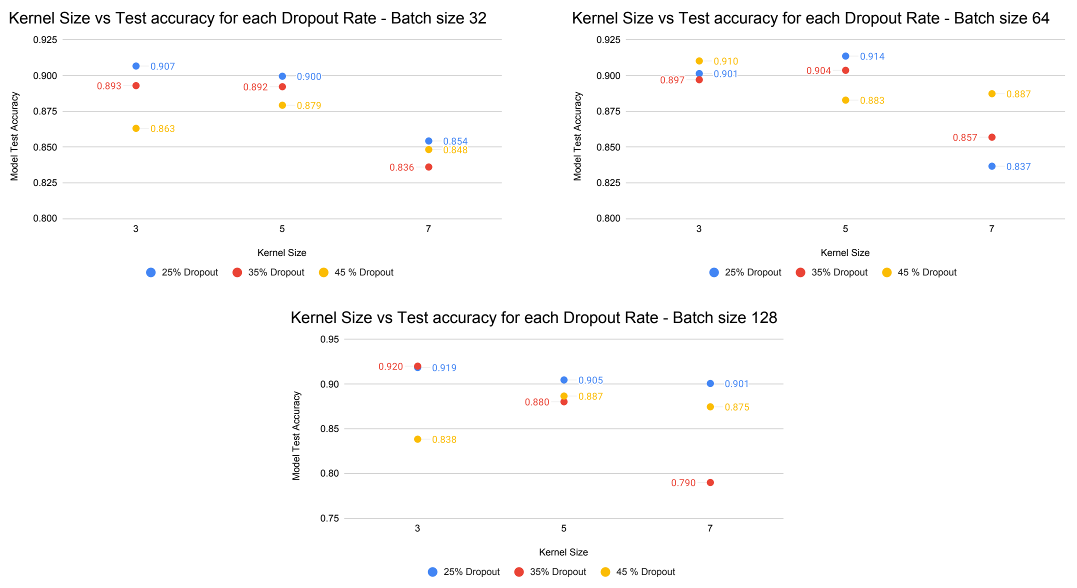

Figure 2: Kernel Size vs Test Accuracy for each hidden layer dropout rate using each batch size.

the use of smaller dropout rates could allow the model to better fit to the training data, which could improve testing accuracy due to the large variation in the training set. A future extension to these experiments would be to try smaller dropout rates, perhaps 5%, 10% and 15%, to explore and evaluate the resulting effect on performance of the network.

Experiment Number	Kernel Size	Dropout Rate	Batch Size	Test Accuracy	Test Loss	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss
1	3	0.25	32	0.907	0.284	0.866	0.900	0.388	0.305
2	5	0.25	32	0.900	0.296	0.862	0.893	0.403	0.317
3	7	0.25	32	0.854	0.451	0.791	0.848	0.592	0.456
4	3	0.35	32	0.893	0.325	0.829	0.896	0.488	0.337
5	5	0.35	32	0.892	0.338	0.820	0.882	0.516	0.361
6	7	0.35	32	0.836	0.490	0.784	0.836	0.629	0.494
7	3	0.45	32	0.863	0.423	0.782	0.858	0.631	0.438
8	5	0.45	32	0.879	0.372	0.789	0.876	0.609	0.386
9	7	0.45	32	0.848	0.459	0.776	0.841	0.647	0.486
10	3	0.25	64	0.901	0.308	0.841	0.898	0.457	0.319
11	5	0.25	64	0.914	0.268	0.879	0.912	0.352	0.271
12	7	0.25	64	0.837	0.504	0.775	0.836	0.641	0.513
13	3	0.35	64	0.897	0.335	0.831	0.892	0.496	0.346
14	5	0.35	64	0.904	0.297	0.847	0.903	0.442	0.302
15	7	0.35	64	0.857	0.448	0.779	0.849	0.637	0.460
16	3	0.45	64	0.910	0.270	0.848	0.909	0.442	0.281
17	5	0.45	64	0.883	0.350	0.836	0.881	0.471	0.362
18	7	0.45	64	0.887	0.360	0.788	0.875	0.608	0.381
19	3	0.25	128	0.919	0.253	0.886	0.917	0.330	0.259
20	5	0.25	128	0.905	0.296	0.855	0.903	0.415	0.306
21	7	0.25	128	0.901	0.311	0.825	0.897	0.503	0.316
22	3	0.35	128	0.920	0.237	0.874	0.917	0.361	0.245
23	5	0.35	128	0.880	0.367	0.827	0.877	0.502	0.372
24	7	0.35	128	0.790	0.593	0.728	0.794	0.759	0.605
25	3	0.45	128	0.838	0.490	0.748	0.836	0.716	0.499
26	5	0.45	128	0.887	0.358	0.797	0.885	0.579	0.368
27	7	0.45	128	0.875	0.380	0.800	0.869	0.572	0.395

Table 1: Results from our 27 hyper-parameter combinations. All values rounded to 3 significant figures.

References

[1] What is image classification? https://huggingface.co/tasks/image-classification. Online; accessed 25/04/23.

[2] Angelo (editor) Cangelosi and Minoru (editor) Asada. *Cognitive Robotics*. The MIT Press, 2022.

[3] Rui Chen, Meiling Wang, and Yi Lai. Analysis of the role and robustness of artificial intelligence in commodity image recognition under deep learning neural network. *PLoS One*, 15(7):e0235783, July 2020.

[4] Christian Deuerlein, Moritz Langer, Julian Seßner, Peter Heß, and Jörg Franke. Human-robot-interaction using cloud-based speech recognition systems. *Procedia CIRP*, 97:130–135, 2021. 8th CIRP Conference of Assembly Technology and Systems.

[5] Paul Gavrikov. visualkeras. https://github.com/paulgavrikov/visualkeras, 2020.

[6] Daiki Hirata and Norikazu Takahashi. Ensemble learning in CNN augmented with fully connected subnetworks. *CoRR*, abs/2003.08562, 2020.

[7] Maja J. Mataric. *The Robotics Primer (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2007.

[8] Radouan Ait Mouha. Deep learning for robotics. *Journal of Data Analysis and Information Processing*, 09No.02:14, 2021.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[10] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, and Peter Corke. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[11] Kai Zhu and Tao Zhang. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, 26(5):674–691, 2021.