34212-Lab-S-Report

Horia Radu
k55592hr

October 9, 2022

1 Introduction

1.1 Problem and Dataset

Deep Learning and Neural Networks present a paramount improvement in robotics design.

Cognitive robotics is a subfield of robotics that is concerned with robots that are able to learn and reason like humans. They should be able to make decisions based on their observations and adapt to changes in their environment.

Image classification is the process of taking an image as input and outputting a class or label for that image. In cognitive robotics, image classification is used to identify objects, people, or scenes in images. This is very helpful, and possibly the first step, in allowing a robot to interpret a scenario and make decisions based on it.

For this, a Deep Learning model can be implemented into the robots.

I have decided on extending the Lab2b CNN, using TensorFlow and Keras, for image classification through a CNN model, to demonstrate the capabilities of NN in cognitive robotics.

For my experiment, I have used and tested on the more complex CIFAR100 dataset [KH+09]. The CIFAR100 dataset consists of, 60000 32×32 colour images in 100 classes, with 600 images per class.

There are 50000 training images and 10000 test images. The 100 classes in the CIFAR100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

It has 100 different fine classes, compared to CIFAR10, with only 10 labels for 60000 images.

I chose this dataset because of the extended number of "fine" labels it has, and the much higher probability of a False Positive prediction. This would build upon the dataset in the Lab and explore Neural Networks in more depth. A problem with this dataset is the small image size it has, 32×32, and only 600 images per label, considering the huge amount of labels. So, it is much more complicated for a CNN to perform the image classification.

I decided on first trying out the Deeper CNN architecture that was given to us in the lab.

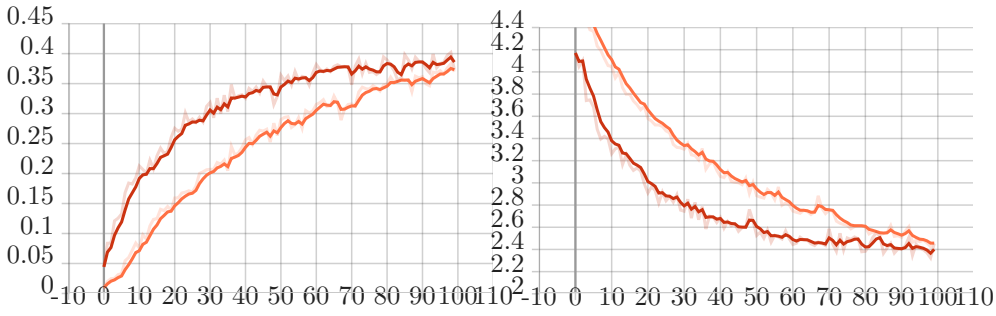


Figure 1: Epoch Accuracy (left) and Epoch Loss (right) for Deeper CNN from the labs with SGD (orange) and RMS (red).

The same network achieved a test accuracy of 72% on CIFAR10, while achieving only 38% on CIFAR100.

1.2 Hyperparameter Tuning and Optimization

I wanted to explore more possibilities to approach this dataset, so I have used Keras Tuner to find the most suitable setup and hyperparameters [OBL+19].

Keras Tuner is a hyperparameter optimization framework that makes it easy to implement a wide range of optimization algorithms. Keras Tuner provides an easy way to perform hyperparameter optimization for Keras models with Bayesian optimization, evolutionary algorithms, and other optimization methods.

I have tried optimization on the evaluation accuracy for the following parameters:

	Learning Rate	Weight Decay	Dropout Rate	Nr. of Hidden Layers	Optimizers
Min	1e-4	1e-5	0.0	2	SGD
Max	1e-2	1e-2	0.50	6	AdamW

Table 1: Values for optimization were selected between Min and Max, and two optimizers were evaluated: Stochastic Gradient Descent (SGD) and Adam Optimizer with Weight Decay (AdamW)

From the optimization, by running a simulation of 200 trials, 50 epochs each, I have got the following results:

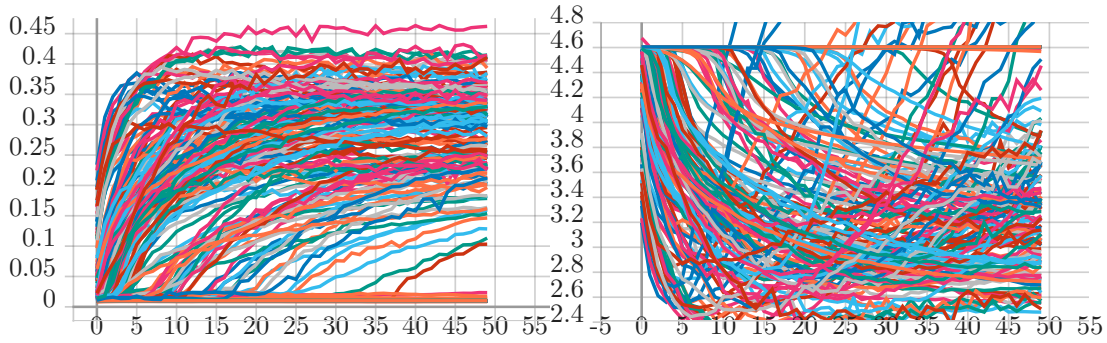
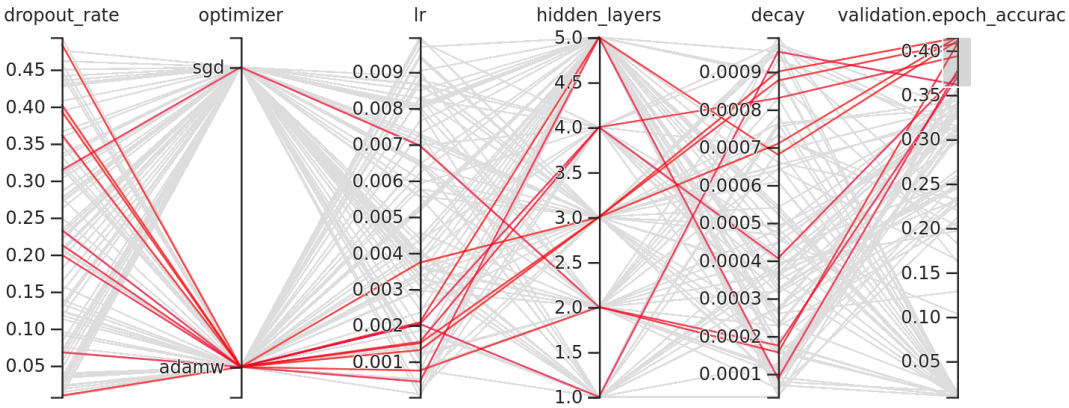


Figure 2: Epoch Accuracy (left) and Epoch Loss (right) for all my 200 trials.

For the batch size, I decided on using 512 as it was the largest I managed to handle, and proved the fastest way to process all those trials.

And the hyperparameters for each based on the evaluation accuracy:



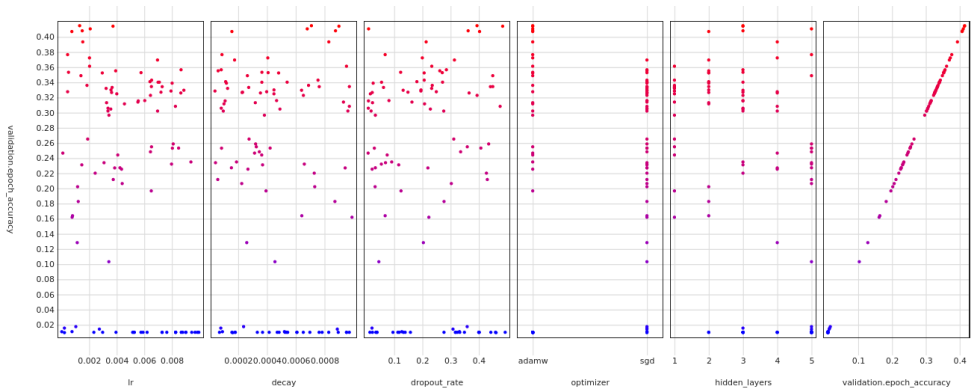


Figure 3: Evaluation accuracy compared to the hyperparameters used for each model.

To make sure that those hyperparameters are indeed appropriate, I have made another simulation of multiple runs, this time 100 epochs each, using a smaller, more precise, selection of hyperparameters from the above optimization:

	Learning Rate	Weight Decay	Dropout Rate	Nr. of Hidden Layers	Optimizers
Min	1e-4	1e-4	0.25	4	AdamW
Max	5e-3	2e-3	0.50	7	AdamW

Table 2: Those are the best/optimal hyperparameters ranges evaluated from the optimization process. AdamW outperformed SGD, so I used only AdamW

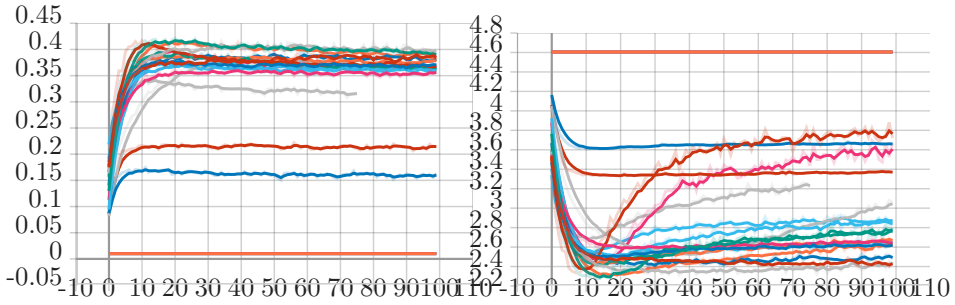


Figure 4: Epoch Accuracy (left) and Epoch Loss (right) for my further Keras Tuning with the optimal hyperparameters.

This graph confirms that the selective constriction and optimization of the initial parameters was successful. From it, I have selected a few hyperparameters to further use in my exploration.

2 Methodology and Network Topology

The test performance of the initial Deeper CNN from Lab2b only reached 38%, and the models from the Keras Tuner plateaued at rougly 40%, so I decided to continue looking for getter results.

I implemented my own Residual Convolutional Neural Network for Image Classification [HZRS16].

This includes additional downsampling steps and shortcut/skip connections in the architecture, apart from the 2D Convolutional and Max Pooling Layers, all of them using the ReLU activation function [HZRS15]. Also, it has Batch normalization layers in every block of the architecture.

The Flatten and Dense layers are the same, with Softmax activation function for output.

The specific Network that I used is inspired from the ResNet [HZRS16], but at a lower scale, with only 2.5 million trainable parameters, compared to the smallest ResNet18 with 11 million. I was not able to test on a much larger architecture due to the hardware requirements, available processing power and time.

I have also introduced skip connections, similar to a ResNet architecture, to get rid of the vanishing gradient problem and mitigate accuracy saturation.

For my custom ResNet, I was able to use a **Batch size** of 256 and 512, due to the small size of the images, 32x32. This made the training much faster, and no major performance difference was noticed between the different batch sizes.

The number of **Hidden layers** I have used is 18, but with smaller convolutional filters, compared to ResNet18. I have the following filters number: $32 \rightarrow 64 \rightarrow 128 \rightarrow 256$. Additionally, I have used downsampling to reduce the amount of information that goes in the model and ensure higher computation speed. It also makes the Max Pooling layers tolerant to small changes in the input.

The best **Optimizer** in my scenario was AdamW, as resulted from my Keras Tuner. Articles and papers roughly generalize that SGD possibly generalizes better than AdamW, but it has slower convergence than AdamW [WRS+17].

The **Dropout rate** was performing decently between 0.3 and 0.5, so I decided on using it at 0.5, due to the more complex custom ResNet I have made.

The **Learning rate** I have used to get the best value was 1e-3, somewhere in the middle from my optimization experiment, as was the **Weight decay** regularization, 1e-4.

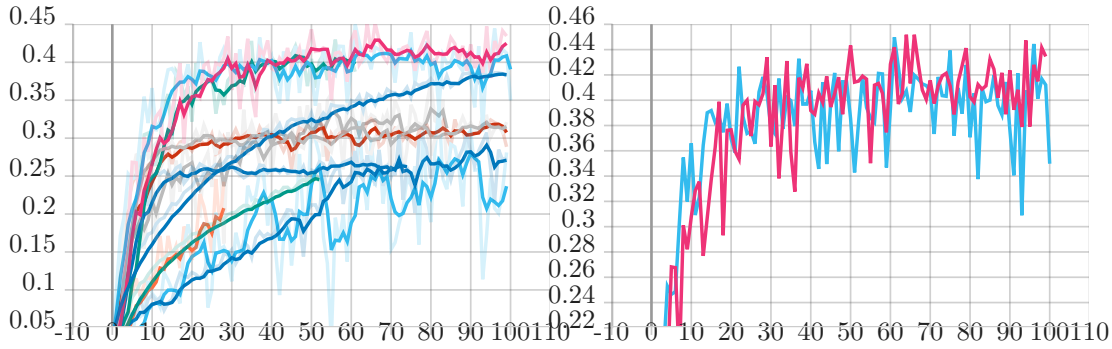


Figure 5: Accuracies by Epochs for my custom model: all experiments with smoothing applied (left) and best models using the above hyperparameters (right) with batch sizes 256 (blue) and 512 (red). Best performance was 45.18%

I have noticed that the two most important parameters were the Learning rate and the Weight Decay. Wrong variations of those could result in degrading models, as can be seen in the figure above on the left. Dropout and Optimizers also helped, but not significantly.

And, finally, I have observed, that the deeper the network, the more tedious, and possibly slower, my model was to train, but it did not plateau at a certain accuracy, as can be seen in the smaller models from the tuning phase. The smaller networks trained much faster, but did not manage to surpass 40% accuracy.

Overall, my model would have largely benefited from a bigger size, more hidden layers and trainable parameters, but it would have taken a lot more processing power and time to train to an adequate accuracy.

3 State of the art and Deep Learning in Cognitive Robotics

The current state of the art for image classification on the CIFAR100, as state on the website [sot22], is a CNN called EffNet-L2, with 96% testing accuracy. This is a huge step from my resulted accuracy, training a small and simple CNN similar to ResNet. But, in comparison, EffNet-L2 has 480 million trainable parameters and is much bigger than my own basic model. With that said, it uses extra training data, meaning that this Model achieved this performance using Transfer Learning on CIFAR100.

To achieve much better performance, I could pre-train a model on a bigger image classification dataset, and then Fine-tune it for CIFAR100, achieving much better results.

More complex datasets represent the state of the art in image classification, object detection and image semantic segmentation. In my case, I have only explored image classification, for which ImageNet

would have been a much better benchmark, with many more higher resolution images, roughly 14 million. For ImageNet, the state of the art, without extra training data, just the data from ImageNet [DDS+09], is a Visual Transformer called PeCo (Perceptual Codebook), of a significantly larger size [DBZ+21].

COCO would have been a great dataset to use, but with the 328K images of a high resolution, I was not able to utilize it. The COCO dataset is very good for the large-scale object detection capabilities, its image captioning, and per-pixel segmentation masks for image segmentation. For this dataset, the object detection SOTA is the SwinV2-G Transformer with 53.7% [LHL+], and the SOTA for segmentation is a Detection Transformer (DETR) called DINO with 63.2% [ZLL+22].

Those 3 tasks, image classification, object detection and image segmentation are 3 very important roles of a robot that have been approached through Deep Learning. At the moment, the methods that are doing the best and achieve state of the art are CNNs and Transformers, plus variations of them.

There is no doubt that Deep Learning has played a significant role in robotics. Deep Learning enabled robots to become more accurate and efficient. Neural networks are a powerful tool for learning from data, and they have been successfully applied to a variety of tasks in robotics, including object detection and recognition plus their movement and control.

There are a few potential limitations to Deep Learning in robotics: computationally expensive models, which can limit their to real-time capabilities and large amounts of data needed to train them.

Overall, Deep Learning is a key tool for the advancements in the robotics field. While there are some potential limitations, Deep Learning is expected to continue playing a major role in the development of cognitive robotic systems.

References

[DBZ+21] Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, and Nenghai Yu. Peco: Perceptual codebook for bert pre-training of vision transformers. *arXiv preprint arXiv:2111.12710*, 2021.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[LHL+] Z Liu, H Hu, Y Lin, Z Yao, Z Xie, Y Wei, J Ning, Y Cao, Z Zhang, L Dong, et al. Swin transformer v2: Scaling up capacity and resolution. arxiv 2021. *arXiv preprint arXiv:2111.09883*.

[OBL+19] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner. https://github.com/keras-team/keras-tuner, 2019.

[sot22] State of the art from papers with codehttps://paperswithcode.com/sota/image-classification-on-cifar-100, Accessed 01 May 2022.

[WRS+17] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

[ZLL+22] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.