Comparison of Multi-Layer Perceptron Performance Against Convolutional Neural Network for Image Classification

Sultan AlRashed

May 2022

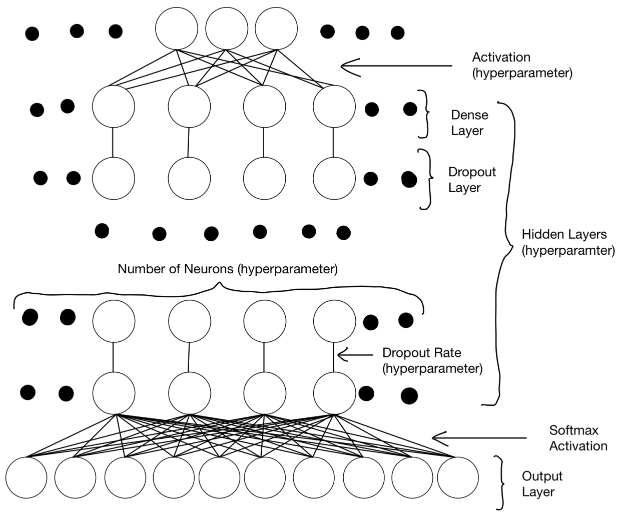
1 Introduction

This report explores a comparison of multi-layer perceptrons against convolutional neural networks in the CIFAR-10 image recognition database. It is a commonly held understanding that convolutional neural networks tend to edge out multi-layer perceptrons due to the filters incorporated, which resembles how the visual cortex of a human brain works[Kuz+18].

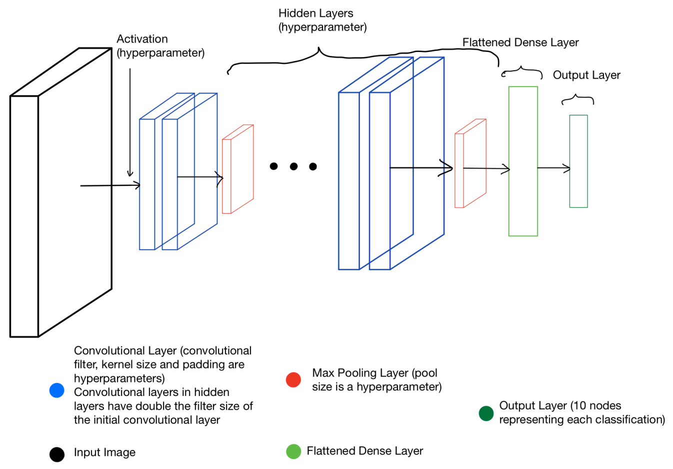
The methodology is as follows: first a general and easily extendable neural architecture is written for both the multi-layer perceptron and the convolution neural network. Then a hyperparameter tuning platform, in this case Ax, is used to test for multiple different hyperparameters in an effort to select optimal hyperparameters. By writing easily extendable architectures the hyperparameters can be chosen effortlessly thus allowing us to make the network topology, for the most part, itself a hyperparameter.

2 Network Topology

Making the network topology mostly a hyperparameter allows for the selection of an optimal topology. Not all aspects of each network's topology are hyperparameters, the topologies can be seen in the following figures:


Activation (hyperparameter)
Dense Layer
Dropout Layer
Hidden Layers (hyperparamter)
Number of Neurons (hyperparameter)
Dropout Rate (hyperparameter)
Softmax Activation
Output Layer

(a) Multi-layer perceptron general network architecture.


Activation (hyperparameter)
Hidden Layers (hyperparameter)
Flattened Dense Layer
Output Layer
Convolutional Layer (convolutional filter, kernel size and padding are hyperparameters) Convolutional layers in hidden layers have double the filter size of the initial convolutional layer
Max Pooling Layer (pool size is a hyperparameter)
Output Layer (10 nodes representing each classification)
Input Image
Flattened Dense Layer

(b) Convolutional neural network general network architecture.

The implementation for the multi-layer perceptron is as follows:

1. A method is created containing all hyperparameters as arguments.
2. Sequential model is initialized. The input dimensions are normalized to a single dimension vector (from 32x32x3 to 3072).
3. For each hidden layer, a dense layer followed by a dropout layer is used. (The number of hidden layers, number of neurons in the dense layer, dropout rate, and activation function for each layer are all hyperparameters.)
4. A final dense layer containing 10 neurons, representing each classification, is created using a softmax activation function.

5. The model is compiled using an optimizer and learning rate, which are both hyperparameters.

The implementation for the convolutional neural network is as follows:

1. A method is created containing all hyperparameters as arguments.
2. Sequential model is initialized. The input dimensions are that of the original image.
3. For each hidden layer, two 2D convolutional layers are added with an activation function for each, followed by a max pooling layer, and finally ending with a dropout layer. (The number of hidden layers, convolutional filter, kernel size, activation function, and pool size are all hyperparameters.)
4. Every hidden layer after the first has double the convolutional filters in each convolutional layer.
5. A flattened dense layer is included with an activation function, and a dropout layer with double the dropout rate.
6. A final dense layer containing 10 neurons, representing each classification, is created using a softmax activation function.
7. The model is compiled using an optimizer and learning rate, which are both hyperparameters.

2.1 Justification of Topology

For both the multi-layer perceptron and the convolutional neural network, the final output layer uses the softmax function. The softmax function is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one, which is ideal for multi-class classification problems[LBH15][Bis+95].

For the multi-layer perceptron all other aspects of topology are defined as a hyperparameter, therefore a near optimal set of parameters is generated through experimentation. However the model is not scaled with either depth or width, since scaling showed no major signs of improvement over testing so scaling was not used.

In the case of the convolutional neural network most aspects of topology are hyperparameters, the minor exceptions are two factors:

1. Every convolutional filter after the first has double the convolutional filters.
2. Final dense layer consists of 512 neurons.

The reason for doubling the convolutional filter lies in the fact that convolutional neural networks show reliable improvements when width is scaled according to depth[Law+97][Neb98][TL19]. Doubling convolutional layers' filters after the first offers a computationally simple way to scale width with depth, but an ideal approach would be one that consists of using a compound scaling method[TL19].

In terms of the final dense layer of 512 neurons, it was found to be the most ubiquitously selected number of neurons for the last flattened layer[JK21][JDC14]. The layer being flattened must occur to conform to the output dimensions.

3 Related Works

Deep learning is a constantly evolving field, with new technologies on the rise regularly. These technologies also assist in many aspects of robotics, especially for tasks such as image recognition. Robotic vision is only one part of a large moving system, since the immediate outputs ultimately result in a final action while leads to a much more complex system[Sün+18][Mou+21].

There are three essential challenges to overcome for robotic vision: reasoning (addresses separate and joint reasoning about the geometry and semantics of a scene and its objects), embodiment (which separates computer vision from robotic vision), and learning (problems arising from deployment in open-set conditions). Reasoning encompasses joint reasoning, geometry, and semantics while embodiment encompasses temporal embodiment, spatial embodiment, active vision, and manipulation for perception and finally learning encompasses uncertainty estimation, identifying unknowns, incremental learning, class-incremental learning, and active learning[Sün+18].

Deep learning is shown to be both highly robust and much more computationally efficient compared to model-based solutions in robotic vision. However deep learning cannot be as broadly applicable as model-based solution, deep learning also has a much lower data efficiency since training requires significant data collection[Sün+18][RLS18][Mou+21].

One can take inspiration from deep learning methods used in other instances. Deep reinforcement learning is a flourishing field, with companies such as DeepMind presenting deep-q network. Deep-q network was shown to perform exceptionally in Atari games, performing better than the best human players[Mni+13]. Combining deep-q networks with convolutional neural networks has been shown to increase performance of image recognition[OGU18][Qia+18].

4 Hyperparameters

Hyperparameter selection is accomplished by using Ax, where we can select hyperparameters to change and select a range for them to test for. Two hundred trials are done to test for differing hyperparameters, using fifty epochs for each trial to allow for a fair amount of time for each trial.

Validation loss was chosen as the metric to minimize for the hyperparameter tuning platform, since minimizing loss or maximizing accuracy would both lead to over-fitting to the training data. By minimizing validation loss, a focus is placed on reducing error to new data as much as possible thereby hopefully resulting in a good model.

For context VL represents validation loss, DR represents dropout rate, HL represents hidden layers and LR represents learning rate.

Trial Index	VL	LR	DR	HL	Neurons	Activation	Optimizer
0	2.30152	0.00024	0.02896	7	204	relu	sgd
1	1.59887	0.00371	0.10184	2	155	sigmoid	adam
2	1.45078	0.00052	0.33212	3	221	relu	rms
3	2.27373	0.00377	0.02215	7	159	relu	sgd
...	...	...	...	...	...	...	...
196	1.36596	0.0004	0.03249	2	278	relu	adam
197	1.44582	0.00036	0.06488	3	289	relu	rms
198	1.54541	0.00071	0.12112	5	203	tanh	adam
199	1.37218	0.00045	0.03905	2	280	relu	adam

Table 1: Hyperparameter testing table results for the multi-layer perceptron.

Trial Index	VL	LR	DR	HL	Kernel Size	Conv Filter	Pool Size	Activation	Optimizer
0	2.52523	0.00197	0.07381	2	2	61	2	tanh	rms
1	33.23375	0.05517	0.2634	2	4	28	2	tanh	rms
2	2.31851	0.16305	0.28861	2	3	35	1	relu	rms
3	2.89156	0.00409	0.25265	3	2	19	1	tanh	rms
...	...	...	...	...	...	...	...	...	...
196	1.12549	0.00116	0.17983	3	6	46	2	relu	rms
197	0.58829	0.00052	0.207	4	3	25	2	relu	adam
198	2.76188	0.00049	0.09797	4	2	55	1	relu	rms
199	1.81293	0.00011	0.142	1	4	62	1	tanh	sgd

Table 2: Hyperparameter testing table results for the convolutional neural network.

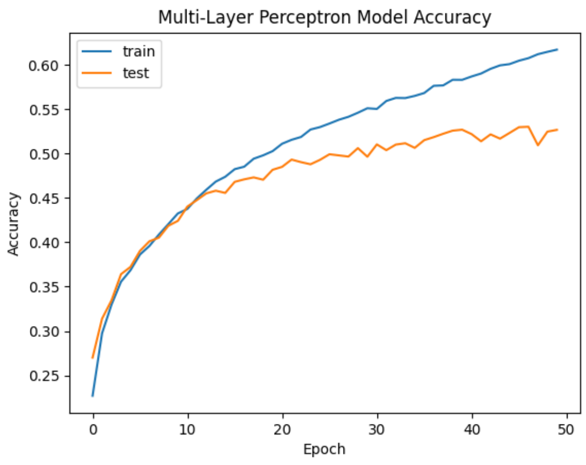
LR	DR	HL	Neurons	Activation	Optimizer
0.00052	0.03759	2	276	relu	adam

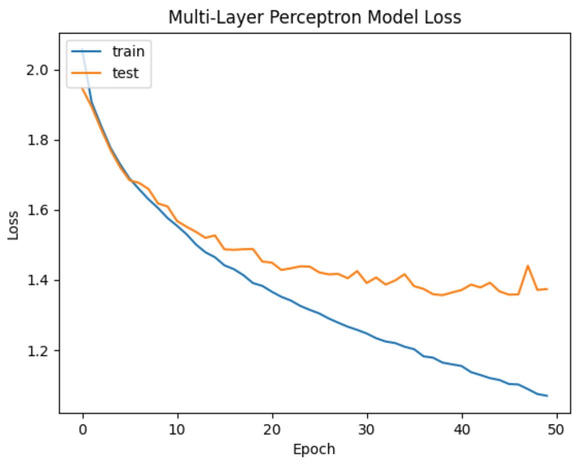
Table 3: Result with lowest validation loss for multi-layer perceptron.

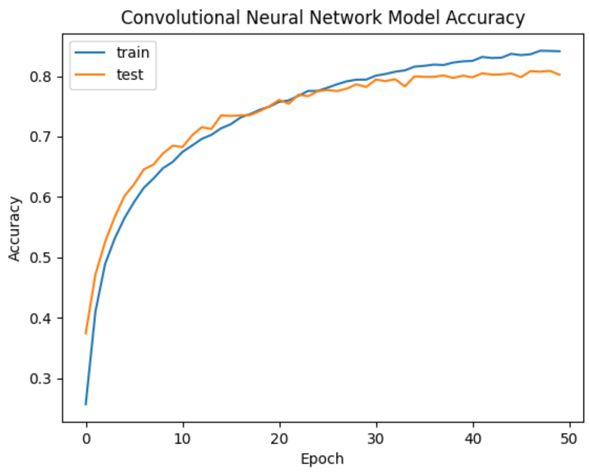
LR	DR	HL	Kernel Size	Conv Filter	Pool Size	Activation	Optimizer
0.00047	0.20524	4	3	25	2	relu	adam

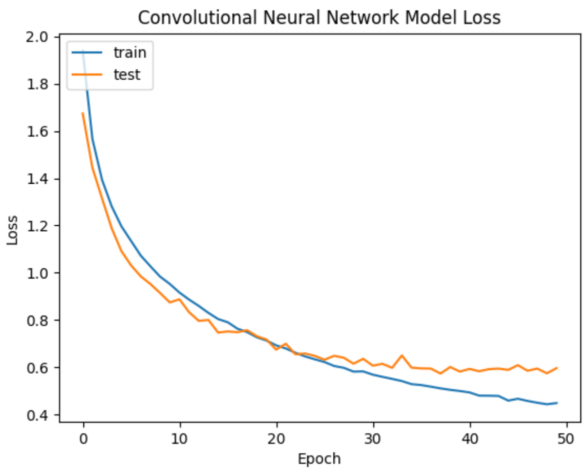
Table 4: Result with lowest validation loss for convolutional neural network.

5 Experimental Testing Results









Architecture	Testing Accuracy	Testing Loss
Multi-Layer Perceptron	0.5328	1.3361
Convolutional Neural Network	0.8023	0.5992

Table 5: Final testing results.

6 Discussion of Results and Conclusion

Clearly the convolutional neural network performs better than the multi-layer perceptron, the reason being is that the convolutional neural networks are inspired by biological processes[Lin21][Kuz+18][FM82]. Convolutional neural networks mimic how the visual cortex works by applying layers of filtering to an image, individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. All the receptive fields together cover the entire image[HW68].

Biomimetics is the process of taking concepts from biology and emulating them in science and technology[Vin+06]. Due to evolution and natural selection biology has refined certain processes over eons, such as flight. Mimicking natural processes of flight lead to huge advancements in aircraft performance[**quest**]. This process of mimicking biology is partly the reason that mimicking the visual cortex for image detection (convolutional neural network) performs better than the general multi-layer perceptron[HW68].

In fact, the convolutional neural network presents a 50.58% increase in accuracy compared to the multi-layer perceptron, as can be seen from the results above. This is an expected result, since convolutional neural networks have been shown to repeatedly outperform multi-layer perceptrons[PMG15]. Two major limitations however are the computational cost of training the network and the computational effort required to implement it, which are much higher than multi-layer perceptrons. Each epoch took ≈ 45 seconds for training the convolutional neural network while it took ≈ 3 seconds for the multi-layer perceptron.

Some major improvements can be done to improve both the results and the architectures implemented. In terms of improving results, one can employ deep reinforcement learning algorithms such as deep-q networks. Deep-q networks have been proven to achieve great results in image recognition when used in conjunction with convolutional neural networks, as discussed in the related works section. To improve the architectures implemented more fine-tuned control over each layer that scales width and depth as a hyperparameter will most likely perform better[TL19].

References

[Bis+95] Christopher M Bishop et al. *Neural networks for pattern recognition.* Oxford university press, 1995.

[FM82] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets.* Springer, 1982, pp. 267–285.

[HW68] David H Hubel and Torsten N Wiesel. "Receptive fields and functional architecture of monkey striate cortex". In: *The Journal of physiology* 195.1 (1968), pp. 215–243.

[JDC14] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. "Flattened convolutional neural networks for feedforward acceleration". In: *arXiv preprint arXiv:1412.5474* (2014).

[JK21] Ernest Jeczmionek and Piotr A Kowalski. "Flattening Layer Pruning in Convolutional Neural Networks". In: *Symmetry* 13.7 (2021), p. 1147.

[Kuz+18] Ilya Kuzovkin et al. "Activations of deep convolutional neural networks are aligned with gamma band activity of human visual cortex". In: *Communications biology* 1.1 (2018), pp. 1–12.

[Law+97] Steve Lawrence et al. "Face recognition: A convolutional neural-network approach". In: *IEEE transactions on neural networks* 8.1 (1997), pp. 98–113.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[Lin21] Grace W Lindsay. "Convolutional neural networks as a model of the visual system: Past, present, and future". In: *Journal of cognitive neuroscience* 33.10 (2021), pp. 2017–2031.

[Mni+13] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[Mou+21] Radouan Ait Mouha et al. "Deep Learning for Robotics". In: *Journal of Data Analysis and Information Processing* 9.02 (2021), p. 63.

[Neb98] Claus Nebauer. "Evaluation of convolutional neural networks for visual recognition". In: *IEEE transactions on neural networks* 9.4 (1998), pp. 685–696.

[OGU18] Takafumi Okuyama, Tad Gonsalves, and Jaychand Upadhay. "Autonomous driving system based on deep q learnig". In: *2018 International Conference on Intelligent Autonomous Systems (ICoIAS).* IEEE. 2018, pp. 201–205.

[PMG15] Clément Peyrard, Franck Mamalet, and Christophe Garcia. "A Comparison between Multi-Layer Perceptrons and Convolutional Neural Networks for Text Image Super-Resolution." In: *VISAPP (1).* 2015, pp. 84–91.

[Qia+18] Junfei Qiao et al. "An adaptive deep Q-learning strategy for handwritten digit recognition". In: *Neural Networks* 107 (2018), pp. 61–71.

[RLS18] Javier Ruiz-del-Solar, Patricio Loncomilla, and Naiomi Soto. "A survey on deep learning methods for robot vision". In: *arXiv preprint arXiv:1803.10862* (2018).

[Sün+18] Niko Sünderhauf et al. "The limits and potentials of deep learning for robotics". In: *The International journal of robotics research* 37.4-5 (2018), pp. 405–420.

[TL19] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning.* PMLR. 2019, pp. 6105–6114.

[Vin+06] Julian FV Vincent et al. "Biomimetics: its practice and theory". In: *Journal of the Royal Society Interface* 3.9 (2006), pp. 471–482.