

Machine Learning 2

by David Robin James

FILE	4073_DAVID_ROBIN_JAMES_MACHINE_LEARNING_2_316598_1335576674.DOCX (329.72K)		
TIME SUBMITTED	24-JAN-2021 05:46PM (UTC+0000)	WORD COUNT	2993
SUBMISSION ID	143210670	CHARACTER COUNT	15929

Machine Learning 2

Student Name: David James (1704050)

Module: Application of Machine Learning

Lecturer: Gordon Dickers

Table of Contents

Table of Figures..... 3

Table of Tables 3

1. Abstract..... 4

2. Introduction 4

 2.1. Convolutional Neural Networks..... 4

 2.1.1. Filters..... 4

 2.1.2. Pooling 5

 2.2. Initial Design..... 5

 2.2.1. Parameters..... 6

 2.3. Testing..... 8

 2.3.1. Model Refinement & Evaluation..... 8

3. Conclusions 10

4. References 10

Table of Figures

Figure 1 - Convolutional filtering 4

Figure 2 - Max pooling of a single feature map 5

Figure 3 - Initial CNN architecture, model summary output 6

Figure 4 - Initial CNN architecture schematic 7

Figure 5 - Revised model architecture with batch normalization and dropout 9

Table of Tables

Table 1 - Initial Test Results 7

Table 2 - Test 01 Results 8

Table 3 - Test 02 Results 9

Table 4 - Test 03 Results 10

1. Abstract

Convolutional neural networks are widely accepted as the prominent solution for efficiently solving complex image classification problems [1]. Research on convolutional neural networks is performed with the outcome of producing an efficient model architecture, justifying configuration of model parameters and hyper-parameters based on a provided image dataset.

2. Introduction

Justification on selecting a convolutional neural network as a method for classifying the images within the dataset provided is discussed. Focus on parameters, hyper-parameters, and the advantages of using a CNN for image classification is reviewed leveraging current and past research to shape the choices of the design and implementation of a simple convolutional neural network.

2.1. Convolutional Neural Networks

Research in machine learning is constantly progressing with newer and improved algorithms and methodologies pushing the technology further. With a variety of problems being applied to machine learning techniques, one area that has significantly improved within the machine learning realm is image classification [2].

Through the study and research of image classification, a powerful and performant approach expanding on the well-known tenets of *artificial neural networks (ANN)* has emerged, namely *convolutional neural networks (CNN)*. Building from previous knowledge of artificial neural networks, CNN's offer a more enhanced solution to solving image classification problems that takes advantage of available computing hardware showing incredibly conducive results relating to time and spatial complexity [3]. The main benefit of CNN's come from their ability to extract features from a dataset without supervision with remarkable effect [4]. The process of feature extraction falls towards a popular image processing operation known as *convolutional filtering* [5], where the filters are learnable parameters of the network architecture.

2.1.1. Filters

One of the fundamental building blocks of a successful CNN architecture involves the use of filters. When convolving over an image a kernel is specified which covers a partial amount of neighbouring pixels from the input. The amount of pixels that are covered is determined by the kernel size, which can be uniform or non-uniform (see Figure 1 - Convolutional filtering). A value is provided for controlling the *stride* or shift of the kernel across the 2-dimensional span of the input.

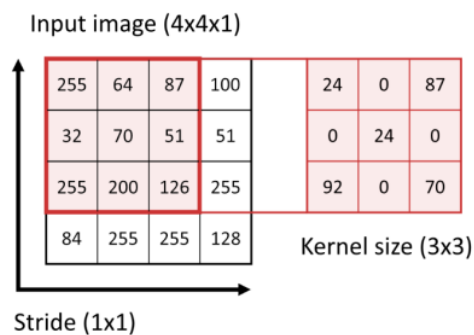


Figure 1 - Convolutional filtering

The goal of a filter is to successfully extract features from an input image, the output of a complete convolution produces a new image which represents a single feature derived from the input. Common CNN architectures employ multiple filters per-layer, where the quantity of filters increases as the network scales [6], [7], [8].

Translating this concept to that of a traditional artificial neural network, the output of the filters interprets to the weights between neurons. The feature maps then receive further processing in the form of an activation function usually specified as a *ReLU (Rectified Linear Unit)* operation, which simply returns the positive values of the feature map, ignoring any negative values [9].

2.1.2. Pooling

With feature maps being produced by the convolutional layer a pooling layer can be applied to modify the output. The purpose of a pooling layer is to reduce the spatial dimensions of the feature maps whilst preserving the critical components of the data, this is achieved by down sampling the input [10]. Although, feature maps are susceptible to *translation invariance* which can often lead to over-fitting with further issues of bad generalisation [11]. To combat over-fitting; whilst training the model various pooling methods can be applied which provides an output that summarizes the presence of features extracted from a sub-section of a feature map.

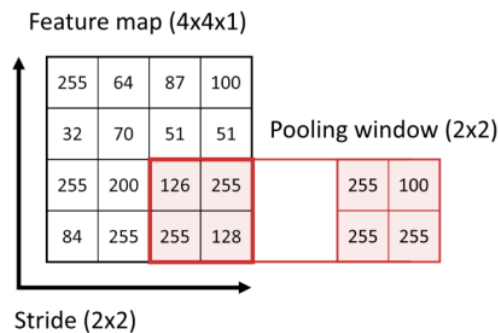


Figure 2 - Max pooling of a single feature map

Two common pooling methods resolve to returning the average or maximum values of a feature map. Modern CNN's opt for max pooling as results offer a higher degree of accuracy when training the model [12], [13]. Whilst selection of pooling methods scales to the context of the problem along with the quality of the data used for training, correct application is determined through experimentation and observation of the network's performance.

2.2. Initial Design

Reviewing papers of successful CNN's to find commonalities between network architectures provided few concepts towards constructing an initial design (see Figure 4 - Initial CNN architecture schematic). Observation of popular network structures endorsed a repeating pattern usually consisting of 1 - 2 convolutional layers (more layers can be applied before pooling if hardware constraints allow) followed by a pooling layer [4], [6]. Various factors were considered such as available computational resources, it was recognised that state-of-the-art CNN's utilized GPU hardware for faster computation, with most models being trained across multiple GPU's. Following similar state-of-the-art networks led to immediate resource limitations, a middle ground providing an acceptable model closely resembling researched architectures produced a foundation for experimentation (see Figure 3 - Initial CNN architecture).

The decision to utilize a GPU was realised during the initial design permitting quicker gains in computational performance. Project setup ensured meeting all prerequisites required for a successful development environment harnessing the computational power of a *GeForce RTX 2080Ti* with 11GB capacity. Further reading of the Tensorflow API revealed a welcomed feature allowing automatic logical selection of available devices permitting Tensorflow code to transparently run on CPU or GPU devices with no modification to Tensorflow API calls. Justification of the time spent meeting the requirements for enabling Tensorflow to run on the GPU showed promising performance gains based on findings from research which recommended training CNN models on the GPU where possible [3].

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	320
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 64)	0
flatten (Flatten)	(None, 65536)	0
dense (Dense)	(None, 512)	33554944
dense_1 (Dense)	(None, 7)	3591

```

Total params: 33,577,351
Trainable params: 33,577,351
Non-trainable params: 0

```

Figure 3 - Initial CNN architecture, model summary output

2.2.1. Parameters

Analysing common *default* parameters of successful CNN's through a variety of research papers confirmed a customary layout as a base point. Researchers followed a convention of filter quantities increasing via powers of two as traversal gets progressively deeper into the network.

Multiple sources claim insight for justifying management of filters in this way with discussions of why this convention is so popular correlating to the logical expression of what the network is doing under the hood. Diligently following the accepted convention of multiplying filters through the upper layers of the network resembles a rational understanding of how features are extracted during training. Lower convolutional layers extract primitively simple shapes such as edges, traversing to upper convolutional layers extracting increasingly abstract shapes [9].

Further research revealed acceptable values for kernel sizes and stride where the standard configuration sets filter sizes at 3x3 with a stride of 2x2 [14].

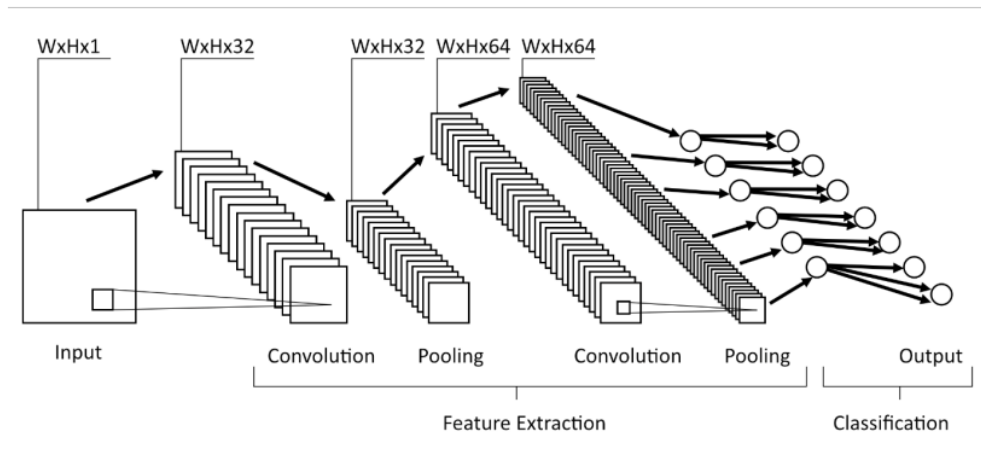


Figure 4 - Initial CNN architecture schematic

By setting the kernel size to an odd number ensured that the center value of the kernel contained an integer which correlates to a valid pixel position from the input. Zero padding added further pixels on the outer edges of the input retaining the spatial dimension after the convolution process. Padding eased any pooling operations performed as the output from a convolution was guaranteed to inherit the same spatial dimensions as the input [15].

Prior to running the model further research was made on the effects of batch sizes. A comfortable median was desired where training times didn't extend to long periods but also gave good results in loss and accuracy along with generalisation. An arbitrary selection of 10 epochs provided a base measure regarding the length of training sessions. 10 epochs were considered enough cycles to give a general observation of the current configuration of hyper-parameters along with allowing the backpropagation algorithm to have adequate effect. The initial batch size was set to a small value for observations on how long training would take and to what effect a small batch size would have on the model's performance (see Table 1 - Initial Test Results):

Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy	Elapsed Time
1	1.66	0.33	1.57	0.38	85s
2	1.47	0.42	1.47	0.42	39s
3	1.36	0.47	1.42	0.45	39s
4	1.24	0.52	1.42	0.45	39s
5	1.12	0.57	1.46	0.46	39s
6	0.99	0.63	1.64	0.45	39s
7	0.85	0.68	1.68	0.48	39s
8	0.71	0.73	1.92	0.47	39s
9	0.58	0.79	2.26	0.46	39s
10	0.46	0.83	2.58	0.45	297s

Table 1 - Initial Test Results

2.3. Testing

With a base line in place, reviewing the initial test values showed a small batch size that resulted in an approximate training session of 12 minutes. Further modifications were made to parameters and hyper-parameters of the model before further testing. Input images were scaled to a size of 16x16 to lower the amount of parameters. As seen in Figure 3 the total count of parameters sat at over 33 million. The choice of lowering the inputs spatial dimensions reflects lowering the amount of parameters to observe the effects of overfitting [16].

Other hyper-parameters such as learning rate were further tweaked. Initially a learning rate of 10^{-3} was set utilising the *Adam* optimizer which adaptively changes the learning rate throughout training [17]. Researchers favour Adam optimization with many papers discussing case studies, reviews, and improvements of the method over classical *stochastic gradient descent* [18], [19], [20].

2.3.1. Model Refinement & Evaluation

Retaining the initial architecture seen in Figure 4 along with the modifications previously discussed, a first test (see Table 2) was conducted to review any performance increases relating to accuracy, generalization, and timings.

Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy	Elapsed Time
1	1.77	0.27	1.71	0.31	11s
2	1.64	0.35	1.63	0.37	8s
3	1.59	0.39	1.60	0.39	8s
4	1.55	0.40	1.56	0.40	8s
5	1.52	0.41	1.55	0.39	8s
6	1.49	0.43	1.52	0.41	8s
7	1.46	0.44	1.52	0.41	8s
8	1.44	0.45	1.51	0.42	8s
9	1.42	0.46	1.48	0.44	8s
10	1.40	0.46	1.47	0.44	8s

Table 2 - Test 01 Results

As observed from the elapsed timings of the first test, increasing the batch size decreased training time dramatically, as well as decreasing the inputs spatial dimensions having a moderate effect. Cross referencing both validation and training accuracies from the initial test and first test also demonstrates a decrease in overfitting, this can be inferred by the decrease in parameters, as the architecture has remained constant throughout both tests.

With some minor overlap between validation and training accuracies further testing saw additional layers added to the network which applied regularization techniques utilising batch normalization and dropout, with the hope that accuracy may increase. Suspicions piqued towards the additions of batch normalization and dropout closing the gap between the overlapping accuracies, eliminating overfitting from the results completely at the cost of lower training accuracy, if applied correctly. Further research was conducted to collect information and form concepts on where to best situate batch normalization and dropout within the architecture of the network [21], [22], [23].

Reviewing the available literature surrounding the topic of batch normalization, *Ioffe et. al.* discusses batch normalization being performed before ReLU [24], interestingly most researchers advise against this [25], [26], [21]. Following the findings from literature, modifications to the existing architecture resulted in a repeating pattern of convolution, batch normalization and pooling (see Figure 5).

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	320
batch_normalization (Batch Normalization)	(None, 32, 32, 32)	128
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_1 (Conv2D)	(None, 16, 16, 64)	18496
batch_normalization_1 (Batch Normalization)	(None, 16, 16, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
flatten (Flatten)	(None, 4096)	0
dropout (Dropout)	(None, 4096)	0
dense (Dense)	(None, 512)	2097664
batch_normalization_2 (Batch Normalization)	(None, 512)	2048
dense_1 (Dense)	(None, 7)	3591

```

Total params: 2,122,503
Trainable params: 2,121,287
Non-trainable params: 1,216

```

Figure 5 - Revised model architecture with batch normalization and dropout

Increasing the inputs spatial dimensions from 16x16 to 32x32 after modifying the networks architecture to accommodate batch normalization and dropout, another test was conducted to review the effects of the revised architecture.

Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy	Elapsed Time
1	2.25	0.22	1.75	0.32	12s
2	1.97	0.28	1.68	0.36	8s
3	1.86	0.30	1.63	0.37	8s
4	1.81	0.31	1.62	0.38	8s
5	1.75	0.33	1.61	0.37	8s
6	1.73	0.33	1.57	0.39	8s
7	1.69	0.34	1.54	0.40	8s
8	1.67	0.35	1.52	0.40	8s
9	1.65	0.36	1.15	0.41	8s
10	1.63	0.37	1.50	0.42	8s

Table 3 - Test 02 Results

With the results demonstrating the welcomed lack of overfitting, further experimentation on increasing the accuracy of the model led to modifying the size of the images by a factor of 2 to increase parameters in the network. Experimenting with the affects of deepening the network with another convolution, batch normalization and pooling stack brought the following results.

Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy	Elapsed Time
1	2.26	0.25	1.77	0.34	12s
2	2.00	0.31	1.60	0.39	9s
3	1.85	0.34	1.54	0.41	9s
4	1.76	0.35	1.52	0.41	9s
5	1.67	0.37	1.46	0.44	9s
6	1.63	0.38	1.47	0.42	9s
7	1.57	0.40	1.40	0.46	9s
8	1.52	0.42	1.40	0.45	9s
9	1.50	0.42	1.37	0.47	9s
10	1.47	0.44	1.36	0.48	9s

Table 4 - Test 03 Results

3. Conclusions

With a potentially deeper network and pre-processing techniques applied to the dataset, better performance may have been gained. Further testing on data augmentation could have had a positive effect on training but attention towards balancing timing for training sessions as well as applying methods to combat overfitting were the focus during each experiment.

4. References

- [1] N. Sharma, V. Jain, and A. Mishra, "An Analysis Of Convolutional Neural Networks For Image Classification," *Procedia Comput. Sci.*, vol. 132, pp. 377–384, Jan. 2018.
- [2] M. Xin and Y. Wang, "Research on image classification model based on deep convolution neural network," *EURASIP J. Image Video Process.*, vol. 2019, no. 1, p. 40, Dec. 2019.
- [3] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, Jan. 2019.
- [4] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, Dec. 2020.
- [5] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, vol. 2018-Janua, pp. 1–6.
- [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
- [7] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2015, pp. 730–734.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, vol. 2016-Decem, pp. 770–778.
- [9] W. S. Ahmed and A. A. A. Karim, "The Impact of Filter Size and Number of Filters on Classification Accuracy in CNN," in *2020 International Conference on Computer Science and Software Engineering (CSASE)*, 2020, pp. 88–93.

- [10] L. Yao, G. Xu, and F. Zhao, "Pooling Method On PCNN in Convolutional Neural Network," *J. Phys. Conf. Ser.*, vol. 1486, no. 2, p. 022026, Apr. 2020.
- [11] H. Gholamalinezhad and H. Khosravi, "Pooling Methods in Deep Neural Networks, a Review," *arXiv*, Sep. 2020.
- [12] W. Wang and Y. Yang, "Development of convolutional neural network and its application in image classification: a survey," *Opt. Eng.*, vol. 58, no. 04, p. 1, Apr. 2019.
- [13] W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," *Neural Comput.*, vol. 29, no. 9, pp. 2352–2449, Sep. 2017.
- [14] X. Li, F. Li, X. Fern, and R. Raich, "Filter shaping for convolutional neural networks," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.
- [15] M. Hashemi, "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation," *J. Big Data*, vol. 6, no. 1, p. 98, Dec. 2019.
- [16] J. Lever, M. Krzywinski, and N. Altman, "Model selection and overfitting," *Nat. Methods*, vol. 13, no. 9, pp. 703–704, Sep. 2016.
- [17] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, Dec. 2014.
- [18] Z. Zhang, "Improved Adam Optimizer for Deep Neural Networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–2.
- [19] H. Zhong *et al.*, "Adam revisited: a weighted past gradients perspective," *Front. Comput. Sci.*, vol. 14, no. 5, p. 145309, Oct. 2020.
- [20] A. Tahmassebi, A. H. Gandomi, S. Fong, A. Meyer-Baese, and S. Y. Foo, "Multi-stage optimization of a deep model: A case study on ground motion modeling," *PLoS One*, vol. 13, no. 9, p. e0203829, Sep. 2018.
- [21] G. Chen, P. Chen, Y. Shi, C.-Y. Hsieh, B. Liao, and S. Zhang, "Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks," *arXiv*, May 2019.
- [22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [23] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimed. Tools Appl.*, vol. 79, no. 19–20, pp. 12777–12815, May 2020.
- [24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *32nd International Conference on Machine Learning, ICML 2015*, 2015, vol. 1, pp. 448–456.
- [25] J. Bjorck, C. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," *arXiv*. 2018.
- [26] M. Hasani and H. Khotanlou, "An Empirical Study on Position of the Batch Normalization Layer in Convolutional Neural Networks," in *2019 5th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, 2019, pp. 1–4.

Machine Learning 2

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

62/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4



Comment 1

give quick numerical results if appropriate

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

GRADING FORM: SMU GENERIC_GJ

DAVID ROBIN JAMES

61.50

RESEARCH-INFORMED LITERATURE

 IF LATE THEN CAPPED AT 50%. Key areas identified and refs used to support and inform arguments. Good. Review of Datasets/ biases, etc no detailed here.

7

KNOWLEDGE AND UNDERSTANDING

 A very good understanding of DNN application and design demonstrated. Topics related to dataset and DNN analysis could be more detailed.

9

ANALYSIS

 Some good work here. The dataset has some limitations and biases - these need to be analysed and detailed in your report. Evaluation fine but F1, confusion matrix etc not discussed.

19

PRACTICAL APPLICATION AND DEP.

 Some good work here. Some additional evaluation and use of reporting/evaluating results would have been even better.

23

SKILLS FOR PROFESSIONAL PRAC.



A well structured report. Good use of refs.
Diagrams included.

3.5