ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/337240963

Convolutional Neural Network for CIFAR-10 Dataset Image Classification

Technical Report · November 2019

CITATIONS
0

READS
2,557

1 author:

Akwasi Darkwah Akwaboah
Johns Hopkins University
12 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Convolutional Neural Network for CIFAR-10 Dataset Image Classification View project

Project Finite Difference Method for Simulating Tissue Wave Propagation View project

All content following this page was uploaded by Akwasi Darkwah Akwaboah on 14 November 2019.

The user has requested enhancement of the downloaded file.

Convolutional Neural Network for CIFAR-10 Dataset Image Classification

Akwasi Darkwah Akwaboah*

***Abstract*— Traditional neural networks though have achieved appreciable performance at image classification, they have been characterized by feature engineering, a tedious process that results in poor generalization to test data. In this report, I present a convolutional neural network (CNN) approach for classifying CIFAR-10 datasets. This approach has been shown in previous works to achieve improved performances without feature engineering. Learnable filters and pooling layers were used to extract underlying image features. Dropout, regularization along with variation in convolution strategies were applied to reduce overfitting while ensuring increased accuracies in validation and testing. Better test accuracy with reduced overfitting was achieved with a deeper network.**

***Index Terms*— Convolutional Neural Network, CIFAR-10, Image Classification, overfitting**

I. BACKGROUND

Convolutional Neural Networks (CNN) have recently gained popularity at image classification tasks since the initial work by Yann LeCun et al[1] (LeNet) involving the classification of handwritten digits and Alex Krizhevsky's[2] AlexNet which obtained a record performance metrics on the CIFAR dataset[3] at ImageNet classification competition in 2012. The major highlight of CNNs is their ability to extract higher level representation of image features without feature engineering, a manual and expensive process that uses domain knowledge to create features for training in machine learning algorithms. CNNs comprise several learnable filters that convolve with input images at specified strides. Another major advantage of CNNs are their ability to reduce the numbers of network parameters and consequently the computational burden, while still attaining increased performances. This report explores effects of modifying the sizes of the convolution filters as well as the overall network architecture. Changes are made to the sequential order of convolution and pooling layers. Three network architectures are presented in the next sections. The first two differ by the filter size, the number of pooling and convolution layers used. The third is an improvement on the previous two that incorporates dropout, regularization, increased number of channels and number of fully connected layers. In all three networks, a sparse categorical cross entropy loss along with the Adam[4] optimizer with a default learning rate of 0.001 was used. In training, a batch size of 128 was used over a training set of 40,000 images. Validation and test sets comprised 10,000 images each. Thus, yielding training-validation-test data split in the ratio of 4:1:1 over the total of 60,000 images in the CIFAR10 dataset.

II. NET I: CONV-POOL-CONV-POOL-CONV-POOL-FC-FC

In this architecture, 8 layers comprising alternating sequence of convolution (CONV) and pooling (POOL) layers are used. All three 2D convolution layers comprise filters of size 3×3 and stride of 2. Varying number of channels are used – 32, 64 and 64 respectively. 2×2 max pooling layers with stride of 2 are used. In all but the last fully connected layer, a reLU activation function is used. A SoftMax activation function is used for the final layer. The input size of the network was set that of CIFAR-10 RGB images, ie. (32×32×3). The number of parameters in a convolutional layer is given by equation 1 below;

$$P_n = [(m \times n \times C_i) + 1] \times C_o \quad (1)$$

where P_n is the number of parameters, $m \times n$ is the filter dimensions, C_i and $C_o$are the input and output channel (feature map) dimensions respectively for a convolutional layer. The addition of one to the inner factor accounts for biases applied to the feature maps. Also, the dimensions of the output data matrix, O_n, upon filtering or pooling is given by the equation 2 below;

$$O_n = \frac{N - F}{S} + 1 \quad (2)$$

Where N is the input dimension and F is the filter dimension similar to $m \times n : m = n = F$, and S is the filter stride length. Pooling layers do not offer any parameters to be optimized as it is a mere computation of average or the maximum value captured by the pooling filter. Using equation 1 and 2, the total number of network parameters to be optimized during training is calculated and results were consistent with output of the model.summary() module in keras shown in figure 1. The performance of the network though better than the best results for neural network without convolution (accuracies: training – 40.47%, validation – 38.79 testing – 38.18%) implemented in the previous homework [GitHub link], exhibited overfitting – after 20 epochs, training accuracy: 86.18%, validation accuracy: 72.39% and test accuracy: 71.81%

[1]Akwasi Darkwah Akwaboah is with the Electronics Engineering Department, Norfolk State University

Layer (type)	Output Shape	Param #
conv2d_33 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_21 (MaxPooling	(None, 15, 15, 32)	0
conv2d_34 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_22 (MaxPooling	(None, 6, 6, 64)	0
conv2d_35 (Conv2D)	(None, 4, 4, 64)	36928
max_pooling2d_23 (MaxPooling	(None, 2, 2, 64)	0
flatten_7 (Flatten)	(None, 256)	0
dense_17 (Dense)	(None, 512)	131584
dense_18 (Dense)	(None, 10)	5130

Total params: 193,034
Trainable params: 193,034
Non-trainable params: 0

Figure 1. NetI: number of parameters, and output sizes of the various network layers, also see the total number of parameters at the bottom of image – keras model.summary() output

. The deviation between the training and test accuracies accounting for overfitting is 14.37%. Figure 2 and 3 for Net I accuracy and loss respectively show the onset of overfitting after epoch #5. From the trend, it is likely overfitting will get worse even over more epoch, thus no need for training over large epochs. Overfitting occurs as the network is not deep enough.

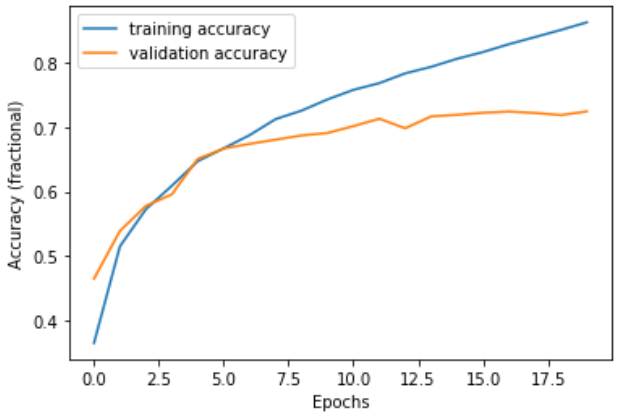

Figure 2. Net I: plot of training and validation accuracies over 20 epochs

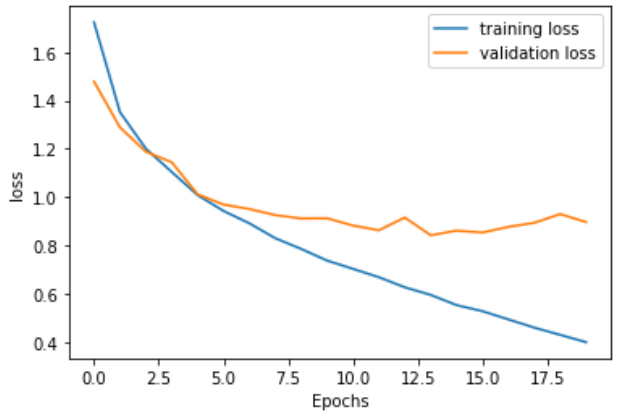

Figure 3. Net I: plot of training and validation losses over 20 epochs. Overfitting is obvious

III. NET II: CONV-CONV-POOL-CONV-POOL-FC-FC

This architecture adopts 7 layers comprising the same number (3) of convolution layers and a reduced number (2) of pooling layers. The sequential layer organization is implemented as shown in this section heading; two convolution layers followed by a max pooling layer, another convolution layer, a max pooling layer and finally two fully connected layer of size 512 and 10 respectively as classification is being done into 10 classes. In all but the last fully connected layer, a reLU activation function is used.

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 28, 28, 32)	2432
conv2d_8 (Conv2D)	(None, 24, 24, 64)	51264
max_pooling2d_4 (MaxPooling2	(None, 12, 12, 64)	0
conv2d_9 (Conv2D)	(None, 8, 8, 64)	102464
max_pooling2d_5 (MaxPooling2	(None, 4, 4, 64)	0
flatten_2 (Flatten)	(None, 1024)	0
dense_4 (Dense)	(None, 512)	524800
dense_5 (Dense)	(None, 10)	5130

Total params: 686,090
Trainable params: 686,090
Non-trainable params: 0

Figure 4. NetII: number of parameters, and output sizes of the various network layers, also see the total number of parameters at the bottom of image – keras model.summary() output

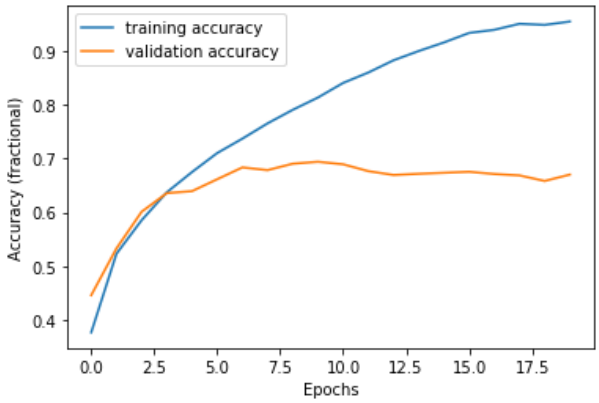

Figure 5. Net II: plot of training and validation accuracies over 20 epochs

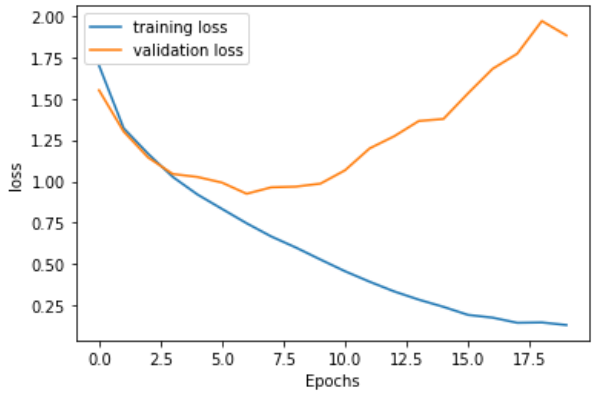

Figure 6. Net II: plot of training and validation losses over 20 epochs. A more drastic overfitting is seen.

A SoftMax activation function is used for the final layer. The network input size adopted is same as that used in NET I. Figure 4 presents the number of parameters and output size for the various networks, calculated using the equation 1 and 2 by the model.summary() keras model. Lower test accuracies and more drastic overfitting were Net II with convolution filter size of (5×5) were obtained relative to Net I, which has a smaller filter size (3×3). This is because a smaller filter/ kernel size can capture complex non-linearities in the input as a lesser number of pixels are averaged at an instance, thus variations in data are better sustained. Again, overfitting is observed for similar reasons explained for Net I. Accuracies: training – 95.37%, validation – 67.00%, testing – 67.07%. Deviation between training and test accuracies is 28.3%, which is higher than Net I, as the larger kernel size used in this case reduces the non-linearities in the input during training and therefore causing the network to generalize easily to the training data while poorly recognizing validation and test data.

IV. NET III: IMPROVEMENT ON NET I & NET II

I present a third architecture that achieves better test accuracies with reduced overfitting relative to the two earlier architectures. This comprises of six convolutional layers with channel size 96, 96, 128, 128, 128 and 128. Each of these layers, employs a filter size of 3× 3, a stride length of 1, and reLU activation functions. The choices of 96 and 128 were inspired by the AlexNet and VGGNet architectures. Three max pooling layers were also incorporated with size of 2×2 each and stride of 2. To address overfitting, a drop rate of 50% after each pooling layer and in the fully connected layers as well as L2 regularization (with hyperparameter of 0.0005) in the fully connected layers were applied. Also, three fully connected layers (instead of two as in the earlier networks) of sizes 1024, 512 and 10 with reLU activation for the first two and softmax for the final layer. The overall network sequence is CONV-CONV-POOL-CONV-POOL-CONV-CONV-CONV-POOL-FC-FC-FC.

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 30, 30, 96)	2688
conv2d_11 (Conv2D)	(None, 28, 28, 96)	83040
max_pooling2d_6 (MaxPooling2	(None, 14, 14, 96)	0
dropout_6 (Dropout)	(None, 14, 14, 96)	0
conv2d_12 (Conv2D)	(None, 12, 12, 128)	110720
max_pooling2d_7 (MaxPooling2	(None, 6, 6, 128)	0
dropout_7 (Dropout)	(None, 6, 6, 128)	0
conv2d_13 (Conv2D)	(None, 5, 5, 128)	65664
conv2d_14 (Conv2D)	(None, 4, 4, 128)	65664
conv2d_15 (Conv2D)	(None, 3, 3, 128)	65664
max_pooling2d_8 (MaxPooling2	(None, 1, 1, 128)	0
dropout_8 (Dropout)	(None, 1, 1, 128)	0
flatten_3 (Flatten)	(None, 128)	0
dense_6 (Dense)	(None, 1024)	132096
dropout_9 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 512)	524800
dropout_10 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 10)	5130

Total params: 1,055,466
Trainable params: 1,055,466
Non-trainable params: 0

Figure 7. Net III: number of parameters, and output sizes of the various network layers, also see the total number of parameters at the bottom of image – keras model.summary() output

The network was trained over 40 epochs as preliminary results for 20 epochs (accuracies: training – 68.03%, validation – 71.37%) gave an indication of potential increase in performance over more epochs. The total number of parameters and the various layer output sizes are presented in figure 7 below. Figure 8 shows the accuracies for training and validation, while figure 9 shows the training and validation loss. Accuracies after 40 epochs: training – 73.21%, validation -76.60% , testing – 75.43%. There was no overfitting and validation accuracy was higher than the training accuracies in all 40 epochs. It is apparent that the accuracy could improve over more epochs. However, a larger computational burden accompanies this.

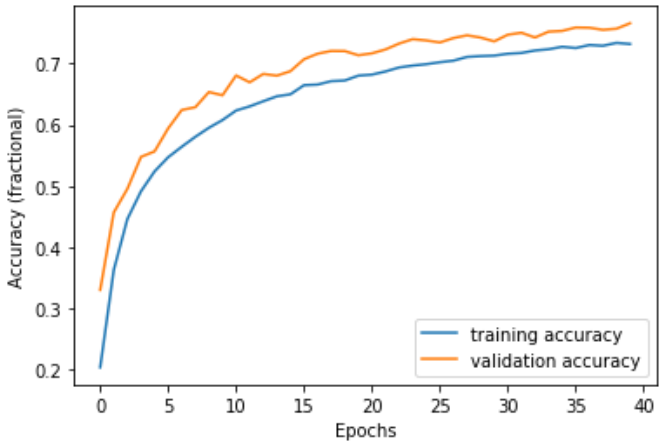

Figure 8. Net III: plot of training and validation accuracies over 20 epochs. Improved validation response devoid of overfitting

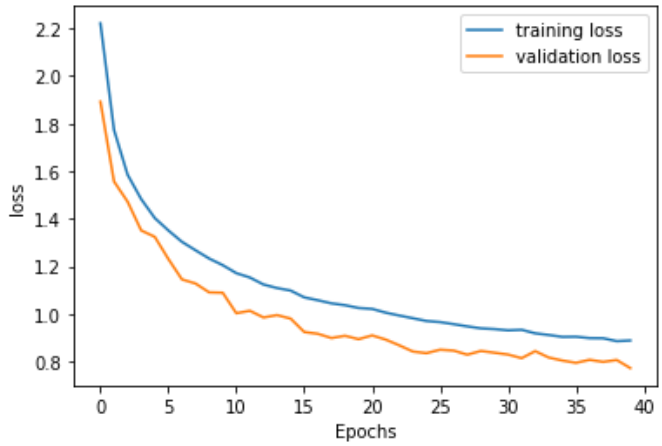

Figure 9. Net III: plot of training and validation losses over 20 epochs. No overfitting present.

V. CONCLUSION

In this work, I present three different convolutional neural network architectures for classifying images in the CIFAR10 dataset. The first two networks (8 and 7 layers respectively) though achieve appreciable training and test accuracies, possess drastic overfitting which is addressed in the third network which employs dropout and L2 regularization. More so, the third network achieves improved validation accuracy over the same number of epochs (20) as the first two. The third network was trained over more epochs as preliminary results for 20 epochs showed promise of better test validation accuracies

without overfitting. Table 1 below summarizes the accuracies achieved by the three networks. The code for this work can access via this GitHub link.

TABLE 1.SUMMARY: PERFORMANCES FOR THE 3 ARCHITECTURES

Archi-tecture	Training accuracy (%)	Validation Accuracy (%)	Test accuracy (%)	Epochs	#Parameters
NET I	86.18	72.39	71.81	20	193,034
NET II	95.37	67.00	67.07	20	686,090
NET III	68.03	71.37	-	20	1,055,466
NET III	73.21	76.60	75.43	40	1,055,466

VI. REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks."

[3] "CIFAR-10 and CIFAR-100 datasets." [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed: 20-Oct-2019].

[4] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014.

View publication stats