Chapter 7

Time Complexity of CNN Models

Deep CNN models take too much time to train. Sometimes it takes hours, days, or even weeks to train, depending on the hyperparameters taken. It is very crucial to estimate the amount of time it will take to run in order to train the model. This chapter work focuses on estimating the time complexity of the CNN model. The initial part focuses on finding the factors that directly affect the time complexity of the model and ends with implementation and result discussion.

7.1 INTRODUCTION

The CNN is gaining a lot of popularity in image classification problems nowadays. It has been used in many different classification problems, like medical imaging, handwritten digits, image classification, etc. It is very critical to estimate the time required by the model to achieve the desired task [23]. The proposed work involves computational studies to find the factors that affect the model's performance, the time taken by each layer to run and how it affects the model's overall performance.

7.2 FACTORS AFFECTING TIME COMPLEXITY

The success of any CNN model depends on many hyperparameters like the type of dataset used, input parameters, number of dense layers, number of convolutional layers, number of neurons in dense layers, batch size, type of pooling strategy, type of optimizer, size of filter and number of filters in each convolution layer, type of activation function, and learning rate [23]. It is very important to find the critical hyperparameters and how one parameter relates to another, affecting the model's overall success. For that, an experiment has been done on eight different CNN architectures, which vary by the number of convolutional layers, number of dense layers, filter size, number of filters used and neurons in each dense layer.

The prime objective of this work is to investigate the computational complexity of the model. As it is not feasible computationally to try every possible combination of all input parameters, crucial parameters have been chosen which might have the maximum impact on the computational complexity of the system. The chosen hyperparameters for the work are the number of convolution layers, number of dense layers, pool size, size of filters, size of neurons, number of filters and size of the convolution kernel.

Among all the layers, convolutional layers, pooling layers and dense layers are the important ones as they perform essential operations and contribute to the overall model complexity . The number of parameters at any layer is the count of "learnable" elements. The input layer provides the shape, but it has no learnable parameters. The pooling layer does not have learnable parameters but it helps in reducing the dimension of the feature map and the parameter count which results in reduced computational complexity. According to Kaiming He et al., fully connected layers and pooling layers take only 5 to 10% of the computational time and 90% of the time is taken by convolutional layers [156]. The time complexity can be reduced by wisely choosing the number of convolutions and fully connected layers. Kaiming et al. have proposed a formula for only the convolution layer which does not consider many factors like batch size and learning rate. After Extensive research and scientific methods, this research work proposed a formula to find the time complexity of a whole CNN model. Though dense layers affect only 5–10% of the complexity of the model [156], this research work has considered convolution layers and fully connected layers(dense layers) in order to find the computational complexity of the model accurately.

7.3 COMPLEXITY OF CNN

Each convolutional layer contains filters that have a depth, number of kernels and filter size and it vary for each convolutional layer. The computational complexity of a convolution layer is a multiplication of these parameters. The total computational complexity of convolution layers is obtained by doing a summation of the complexity of the individual convolution layer.

By considering the learning rate and batch size, the total time complexity of the convolution layer can be calculated as:

$$\left(\sum_{n=1}^{d} k_{n-1} \cdot s_n^2 \cdot f_n \cdot l_n^2\right) \cdot r_1 \cdot b_1 \quad (7.1)$$

Here d is the depth of the convolutional layer, l_n is the length of the output feature map, f_n is the number of filters in the n^{th} layer, S_n is the length of the filter, k_{n-1} defines the number of input channels in the l^{th} layer, *r1* is the learning rate, *b1* is the batch size.

Considering a Fully connected layer, each layer consists dimension of the input/output channel, the width of the input, the height of the input and the number of outputs. These parameters are linked to one another. It is a layer that connects higher layers with the output layer. This layer contains a number of neurons which will vary for each fully connected layer and the output size depends on these neurons. To calculate the time complexity of each fully connected layer it is require to multiply the parameters of each fully connected layer and finally add all the layer's complexity in order to find the total complexity of all fully connected layers of the model

The time complexity of the fully connected layer can be defined as:

$$\left(\Sigma_{l=1}^{f} D \cdot W \cdot H \cdot N\right) \quad (7.2)$$

Here *f* is the d epth of the fully connected layer; D, W, H and N define the Dimension of the input/output channel, the width of the input, the height of the input and the number of outputs respectively.

The total time complexity of a CNN model is summarization of eq. 7.1 and eq. 7.2 which is calculated as:

$$\left(\sum_{n=1}^{d} k_{n-1} \cdot s_n^2 \cdot f_n \cdot l_n^2\right) \cdot r_1 \cdot b_1 \quad + \quad \left(\Sigma_{l=1}^{f} D \cdot W \cdot H \cdot N\right) \quad (7.3)$$

To prove it practically, experiments has been done on eight different CNN models to examine the effect of each layer according to eq 7.1 and eq 7.2 and the overall time required to run the model according to eq. 7.3.

7.4 COMPLEXITY ANALYSIS

The eight models are implemented by varying the number of convolutional layers, the size of filters, the size of the kernel, the number of filters, the size of the neuron, and the number of dense layers. The images of size 224X224 are fed to the CNN model and the Relu activation function has been used in all layers except the output layer where SoftMax activation function has been used. As discussed in previous chapter, the learning rate of 0.0001 has been chosen. The batch size has been fixed at 16 and the Adam optimizer has been used.

The CNN models are chosen based on the systematic arrangement of increasing or decreasing the convolution and fully connected layers along with filter size, pooling and kernel size to scale the practical results with the theoretical results proposed by equations 7.1 and 7.2. The proposed equations will be the baseline for the work as the actual running time is hardware dependent.

Eight CNN models and DRCNN has been used for practical analysis of time complexity. All model's architecture along with parameters is shown in Table 7.1.

Table 7.1 CNN Architectures with Parameters

Model	Number of Convolutional Layers	Number of Filters	Pooling size	Filter size	Number of Dense layers	Neurons in each Dense layer
A	2	64,32	2X2, 1X1	*5X5, 3X3*	1	5
B	2	32,16	2X2, 1X1	*7X7,5X5*	1	5
C	2	16,8	2X2, 2X2	3X3, 3X3	1	5
D	2	16,8	2X2, 2X2	5X5, 3X3	2	64,5
E	3	64,32, 16	2X2, 2X2, 1X1	3X3,3X3,3X3	3	128,64,5
F	2	64,32	2X2, 2X2	5X5, 5X5	3	128,64,5
G	2	64,32	2X2, 1X1	3X3, 3X3	2	64,5
H	3	64,32, 16	2X2,2X2,1X1	3X3,3X3,3X3	2	128,5
DRCNN	11	16,32,64,128	2X2, 2X2, 2X2, 2X2	3X3, 5X5	1	5

The implementation has been done on Intel i7-9750H Lenovo Legion Y540 CPU @ 2.60GHz processor, which supports a multicore processor equipped with a GeForce GTX 1650 NVIDIA GPU with 8GB of memory.

Table 7.2 demonstrates the actual time (in seconds) taken by different models on the TGFD dataset. All the models run for 20,40,60,80 and 100 epochs.

Table 7.2 Time Taken by CNN Architecture on Dataset

Model	Number of Convolution & Fully Connected layers	Epochs (Time taken In Seconds)				
		20	40	60	80	100
A	2C+1F	240	490	743	962	1200
B	2C+1F	230	472	694	945	1150
C	2C+1F	220	450	670	910	1115
D	2C+2F	280	558	832	1115	1386
E	3C+3F	480	935	1438	1915	2385
F	2C+3F	220	450	650	855	1050
G	2C+2F	260	525	765	1035	1300
H	3C+2F	460	910	1400	1885	2315
DRCNN	11C+1F	740	1490	2240	2960	3700

Fig. 7.1 shows the time complexity of each of the model.

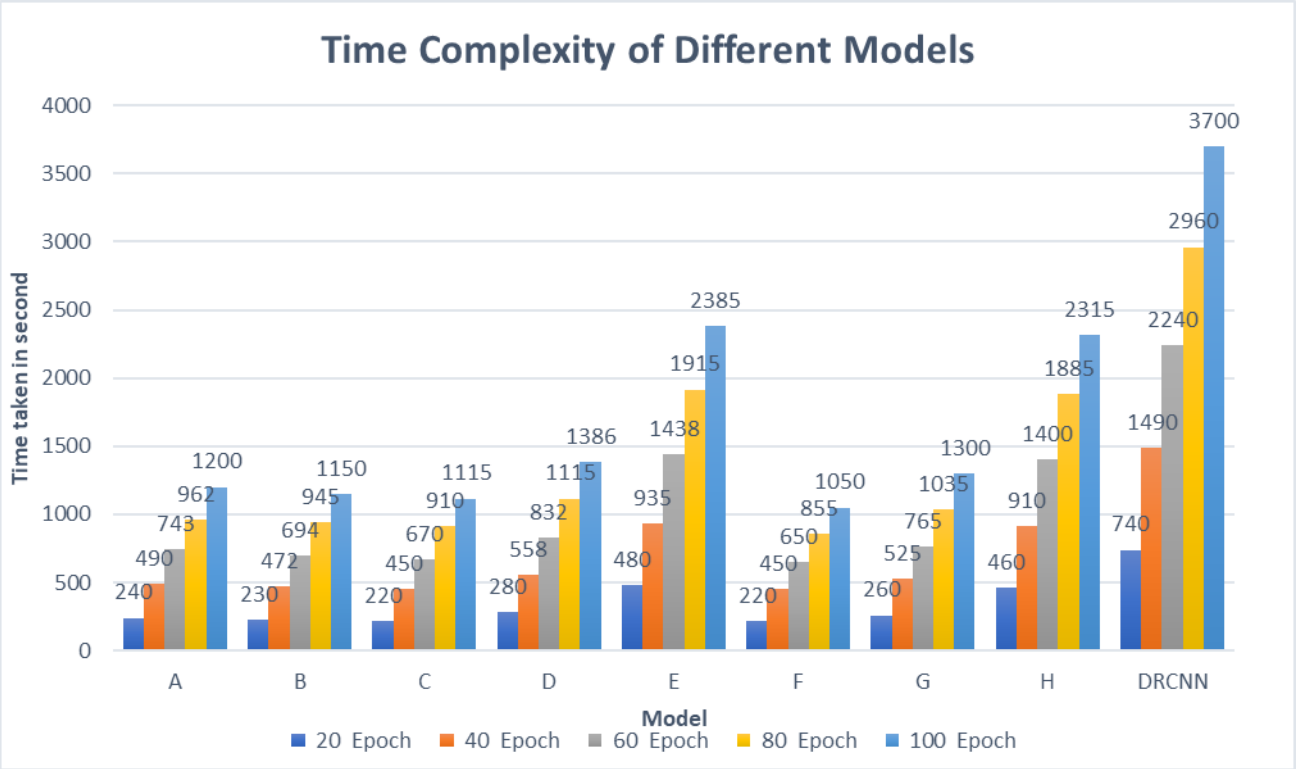

Fig. 7.1: Time Complexity of each model based on chosen parameters

After detailed analysis of the experiments and results from the Table 7.2, following are the research observation which will be helpful to a new researcher to design a CNN model.

- Models A, B and C have the same number of convolutional and fully connected layers, but they take different amounts of time to run as the parameters of the model vary with the number of filters and size of filters. Hence, it proved that the time complexity of the model also depends on the number of filters and size of filters along with the number of convolution layers.
- Model A and B have the same number of convolutional and fully connected layers, the same pooling size but they vary by the number of filters and filter size. The time taken by model A is greater than B, proving that the number of filters and size of filters has a significant effect on the time complexity of the model.
- Model E has 3 convolution and 3 fully connected layers, so it took the highest time compared to all other models.
- Model E and F have same number of dense layers but E has one more convolutional layer than F. as the convolutional layer takes 90% of the computational time there is a huge difference in the time taken by both models to run.
- Model C and D has same number of convolutional layers, but D has one more dense layer than C. As the dense layer only takes 5 to 10% of the computational time there is not much difference in the time taken by the model D.
- Models A and F have the same number of convolution layers, but F has more fully connected layers and more filter size as compared to model A. Hence model F took less time to run compared to model A. This shows that keeping the filter size higher can reduce the accuracy of the model but decrease the computational cost of the model. It concludes that the filter size is inversely proportional to the accuracy of the model.
- Models E and H have the same number of convolution layers and all other parameters except fully connected layers.

- Model H has 2 fully connected layers hence it takes less time than model E, which has 3 fully connected layers.
- DRCNN has 11 convolutional layers and 1 fully connected layer. It has highest number of convolutional layers with more number of filters and hence, it takes maximum time to run compared to all the other models.
- The number of operations for a convolution layer is much larger than the number of operations for a dense layer.
- The per epoch time for a dense layer is greater than the per epoch time of the convolution layer.
- It is also not necessary that a greater number of parameters require higher operations.
- It is not necessary that if the model has a higher number of layers, it also has a higher computational complexity.
- An optimizer, batch size, filter and neurons greatly impact the time taken by the model.
- The convolutional layers, max pool and fully connected layers directly affect the performance of the model.

These empirical findings provide evidence that the proposed formulas, namely Eq. 7.1, Eq. 7.2, and Eq. 7.3, accurately reflect the derived time complexity and exhibit a consistent correspondence with the observed practical performance of the system.

Concluding Remarks: The time complexity of any CNN model is a practical issue that all researchers find nowadays. Finding time complexity helps the researchers decide the impact of each hyperparameter they choose for building a model. This research work tried to find the time complexity of the model and decide which are the crucial parameters for determining the time complexity of the model. The research observation also helps a new researcher to design a CNN model based on these parameters.

The next chapter concludes the work and shows some directions of the future work.