ISSN 1870-4069

Analysis of Performance for a Chairs Classifier through Deep Learning

Javier Maldonado Romo, Mauricio Olguín-Carbajal, Israel Rivera-Zárate, Raul Galvan

Instituto Politécnico Nacional, Mexico City, Mexico

javier.mr.21@gmail.com, molguinc@ipn.mx, irivera@ipn.mx, raulgalvan92@outlook.com

Abstract. Deep learning is a branch of machine learning and this technique allows us to create classifiers. We must find the best dataset size for a classifier process to permit using less time and give good accuracy. In this paper we will propose models with different deep layers and size dimensions for detecting the best model to solve a task that needs quick time processing.

Keywords: Artificial intelligence, deep learning, convolutional neural network, classifier.

1 Introduction

In this topic of investigation about artificial intelligence there are many techniques for processing and classifying the information. Each technique performs in different situations. Recently, there has been an interest in topics regarding artificial intelligence that is deep learning. Deep learning is a topic that is not new but has one feature that allows it to be a good technique to detect patrons in photos, audios and linguistics. Deep learning is a popular technique, but it still needs graphic card units to improve performance. GPU helps to improve processing times but it is still necessary to use deep learning because the GPU has many computer process units that work in parallel to solve the problem faster.

There are web sites which contains several datasets concerning different subjects. One dataset that is famous from deep learning is the CIFAR-10 dataset. This dataset is found on the Kaggle web site. It is a competition where there are ten classes that contain one thousand images by class; the images are in red, blue and green channel colors. Also it has the dimensions of thirty two in width and height. In 2009 one research team achieved an accuracy of 92% using deep learning. Many research teams around the world use deep learning for processing data using GPU. The time is less compared with CPU and cheaper than CPU cluster.

To apply deep learning we can use the following tips: the first aim is that dataset has much information, the different classes have many images that represents a split and the system could process the information because dataset has much data that helps to identify the features of each class. If dataset does not have much data that represents the class information it might not be useful to apply deep learning. This step is the most important before selecting deep learning. The second step is to select the image dimensions because Kaggle competition has 32 pixels, but it is possible to use this dimensions for all datasets. The last step is the number of deep layers; this aim is important because it depends on the performance system.

This work shows behavior in different situations. If the size dimensions are shorter it could reduce the processing time and hold the accuracy or increment size dimensions. To develop this paper we used Torch7 which is a tool for deep learning developed by Toronto University. This tool has a great performance when used in CPU and GPU. There are many papers that measure the performance between CPU and GPU. The result is that using GPU is better, and it is not necessary to check the performance with different hardware; we will only make the test on GPU. The test with CPU is not important because we know that the GPU is quicker in this case. Our classifier has four classes where three classes are different type chairs and the last class is the nothing class or anything that is not a chair.

Alex Krizhevsky has several paper which show that using GPU on deep learning achieved better performance than using CPU. In his paper called Convolutional Deep Belief Networks on CIFAR-10, he mentions how to solve the CIFAR-10 model with convolutional neural networks and how these layers help improve the accuracy with dataset that contain a large number of samples, where the CIFAR-10 model has 10 classes and each class has 1000 images for the classifier. Alex Krizhevsky proposes his architecture with CUDA in the paper Convolutional Neural Networks for Object Classification in CUDA. This contribution shows that convolutional networks benefit to obtain many features with different kernels to classify any object. The last important contribution is the development where the dataset is not only 32 pixels, it is for a data set with large dimensions using other types of layer and techniques such as dropout which helps to reduce the operations between layers.

2 Experiment Description

This experiment consists in creating a fork from model CIFAR-10; in our case we have four classes and the dataset is different. The first goal is to launch the model CIFAR-10 to verify that all the tools execute correctly. We found the CIFAR-10 model for Torch7 is in its official web site. This example is just for CPU, but if we want to achieve a better performance it is necessary to makes the changes to execute on GPU. For this experiment the GPU is Nvidia GTX860M; it has 640 CUDA cores and 2 GBytes in memory RAM. The CUDA is the technology to execute parallel tasks on Nvidia GPU. Torch7 is optimized for using CUDA. Torch7 is only available in Ubuntu 14.04 and later versions. There are no supports to operate other systems, but it is possible to execute Torch7 on MacOS which only needs to have Nvidia GPU to achieve a good performance.

Torch7 uses Lua language that is a C++ extension. Lua is released for parallel tasks such as Multicore process using OpenMP. The experiment is separated in five sections that are the following: load data, model, loss function, train and test. The load data has to read all images when the images have 32 pixel size dimensions in width and height. This section exchanges our dataset with four classes. The model file contains the number layers of the original model; it contain three layers. The first layer is spatial convolution with transfer function Tanh and Maxpooling. The second layer also is spatial convolution with same function transfer Tanh and its respective Maxpooling. The last layer is classified lineal to determine which is the best result. The following figure shows some images for each class that represents our experiment.

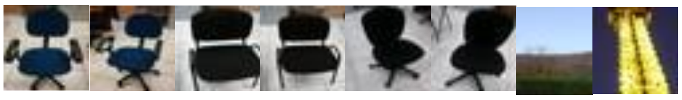

Fig. 1. Dataset for training 32 pixels in which appear two samples for each respective class.

The figure above shows the four classes: the first couple shows office chair color blue; the next couple is normal chair color black, after office chair color black and the last can be anything. Each class is represented by 180 images. The training is supervised and each image has a label indicating its class.

The most important in this experiment is measuring the different times that finished the training. We are going to change the size dimensions. The first is to execute the normal dimensions that are 32 pixels, then change dimensions to 16 pixels, after 8 pixels, and also change the size up to 64,128 and 256 pixels, remembering that the change in size is in width and height. Our target is to check which model finished first and also measure the epoch's number that is necessary to obtain a good accuracy. The epochs are the number the times it requires the training to obtain our value of accuracy.

The figure below shows the images with 16 pixels by side dimensions.

Fig. 2. Dataset for training 16 pixels showing two samples for each respective class.

Dataset 16 pixels model for us is the same image but it lost data, and the information that was lost is important when the system does not train. If the dimensions are changed in an upper value, and the dataset has more data, could we get better results? In this paper we try to prove this theory.

3 Test and Result

We have described the experiment features, now let us try the different model with respect to dataset. Before changing the dataset it is important to launch our CIFAR-10 fork model to measure the time.

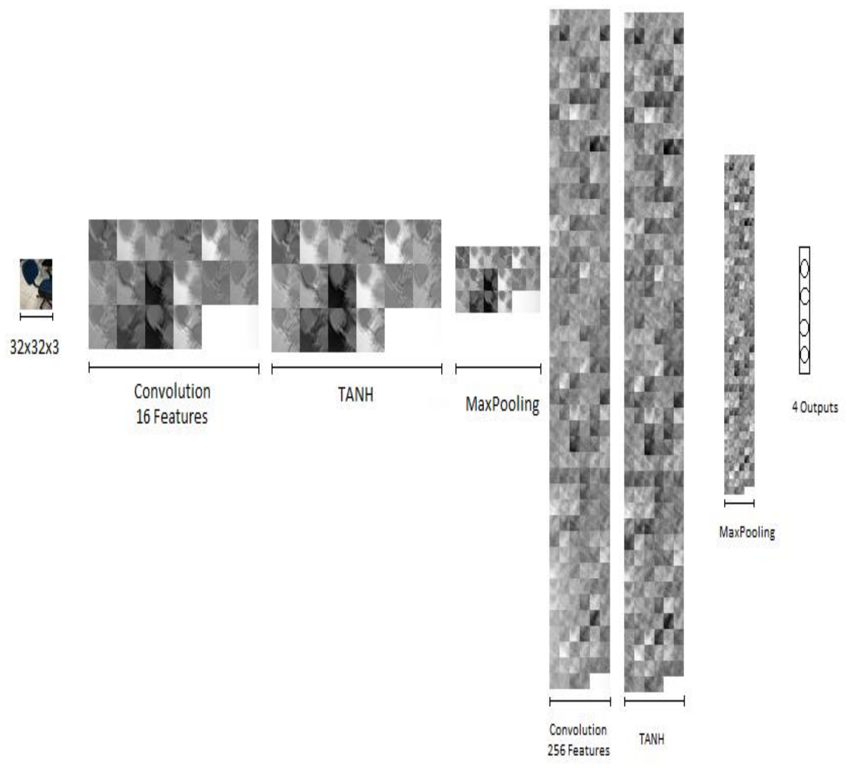

Fig. 3. Graphic description model 32, modifications in each layer is observed.

Figure 3 above shows the model behavior when processing a sample of 32 pixels. Inputs are 32 pixels side by side with red, green and blue channels; the image enters in the model and applies three layers. The first layer consists of a spatial convolution layer with 16 features of size three kernel. Figure 3 shows the image result, following the model passing into its transfer Tanh function and gives the result and last entry Maxpooling with size two kernel.

The Maxpooling result now is the input for the next layer that also has spatial convolution, but now has 256 features with size three kernel continuing with its Tanh and Maxpooling result. Before entering the last layer it is necessary to reshape the input because at this point the input is a matrix. To classify the input it needs to be a vector and reshape the matrix in 256*5*5 that is equal to 6400 samples that makes a vector. This vector is the input for the last layer that lineal classifies the output in four where each output represent a class. It is necessary to apply these steps for each model and measure the times.

The table below specifies the different features about different models. In models with 16 pixels the layers are less than with models of 32 pixels, and more layers in the 64, 128 and 256 pixels.

Table 1. Details for each model which shows their respective features.

Layer	Feature	Model 8	Model 16	Model 32	Model 64	Model 128	Model 256
1	Network	Conv.	Conv.	Conv.	Conv.	Conv.	Conv.
	Output	16 feat.	16 feat.	16 feat.	32 feat.	64 feat.	128 feat.
	Kernel / classifier	5x5	5x5	5x5	5x5	5x5	5x5
	Function	Tanh	Tanh	Tanh	Tanh	Tanh	Tanh
	Out layer	Max pool 2x2	Max Pool 2x2	Max pool 2x2	Max pool 2x2	Max pool 2x2	Max pool 2x2
2	Network	reshap e	Conv.	Conv.	Conv.	Conv.	Conv.
	Output	16x2x 2	256	256	512	1024	2048
	Kernel / classifier	Linear 128	5x5	5x5	5x5	5x5	5x5
	Function	Tanh	Tanh	Tanh	Tanh	Tanh	Tanh
	Out layer	Linear 4	Max Pool 2x2	Max Pool 2x2	Max Pool 2x2	Max Pool 2x2	Max Pool 2x2
3	Network		Reshape	Reshape	Conv.	Conv.	Conv.
	Output		256x5x5	256x5x5	256	512	1024
	Kernel / classifier		Linear 128	Linear 128	5x5	5x5	5x5
	Function		Tanh	Tanh	Tanh	Tanh	Tanh
	Out layer		Linear 4	Linear 4	Max Pool 2x2	Max Pool 2x2	Max Pool 2x2
4	Network				Reshape	Conv.	Conv.
	Output				256x4x4	256	512
	Kernel / classifier				Linear 128	5x5	5x5
	Function				Tanh	Tanh	Tanh
	Out layer				Linear 4	Max Pool 2x2	Max Pool 2x2
5	Network					Reshape	Conv.
	Output					256x5x5	256
	Kernel / classifier					Linear 128	5x5
	Function					Tanh	Tanh

	Out layer					Linear 4	Max Pool 2x2
6	Network						Reshape
	Output						256x5x5
	Kernel / classifier						Linear 128
	Function						Tanh
	Out layer						Linear 4

The following graphics show the behavior of each model until obtaining an accuracy of 100%. The 256 pixel model results are not important as 256 pixels have much data; it is slow compared with other models because it has had 46 minutes in three epochs. The result is not important because the time is longer and it is not the best performance. This dataset has much data but when is has much data the operations are bigger and require more resources and computer power. There are other techniques that reduce the data and operations, but this technique could be in another experiment. Therefore, the 256 pixels model is discarded.

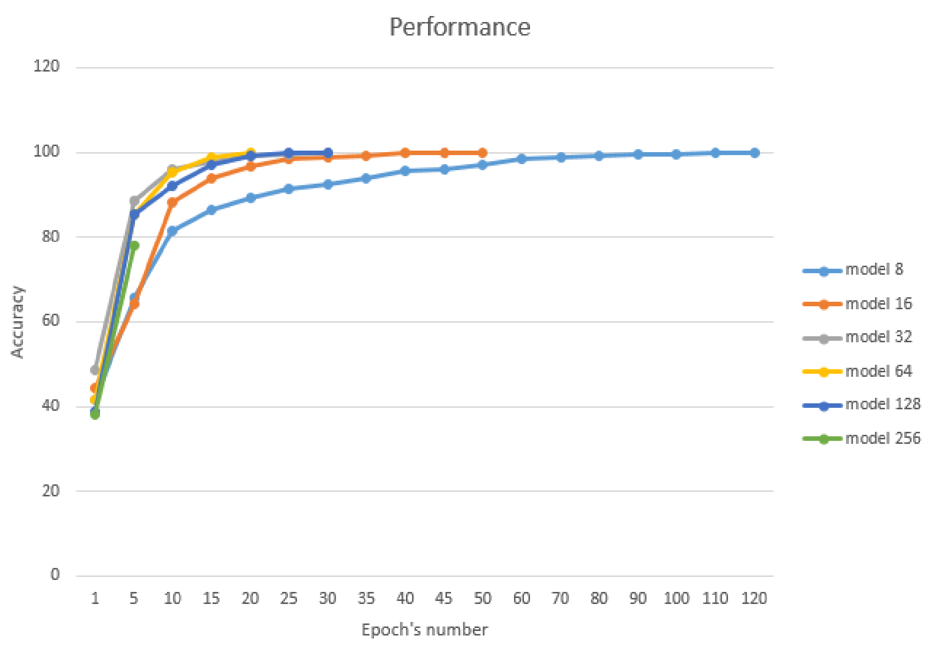

Fig. 4. Accuracy obtained in each 5 epochs.

The table below shows the principal features in our experiment. The 32 pixels model shows our threshold in parameters such as epoch number and seconds to get 100% accuracy. There are others features that demand on GPU. Other GPU's with more CUDA cores could be more powerful.

Table 2. Details about behavior and performance for every model

Feature	Model 8	Model 16	Model 32	Model 64	Model 128	Model 256
Epoch	117	41	30	17	23	3
Memory RAM GPU (MB)	440	434	445	488	651	730
GPU USES (%)	43	34	83	91	99	99
Time in Seconds	56	30	40	165	1800	2700+

The table indicating the 32 pixels model is the best option with least number of epochs and time. Another important point is the low power in GPU and fast response time. The 16 pixels model is excellent but this model needs more epochs using less time and demand in GPU. This point is relevant for embedded systems that cannot use much energy such as mobile devices. The mobile device requires Nvidia GPU and there are not many devices with these GPU on the market, or systems in real time, and it is good idea to use this model.

The model 8 requires many epochs and much time. Otherwise, it is not recommended to use very small images because there is not much information and might have a high error. When changing the dimensions upper to 64 pixels the data grows and so does the information. In the table we show that it is necessary for 17 epochs to obtain 100% accuracy, because there is more data. However, the time is not good compared with model 32 in applications where the response time and accuracy are not important. Our last observation is that models 128 and 256 are not recommendable to use this dataset with these models because the data and operations grow. This deep layer requires more power and more time; the number of epochs is not less than the 64 pixels model and the required time is longer. There are other techniques that could help to improve the performance and try to prove what happened with each model and its respective dataset.

4 Conclusions

This paper describes behavior and features of different models applying them to classify four classes. The most important are the follow two points: the first is about the model with less dimensions such as 8 and 16 pixels; these models have low information compared with 32 pixel models. The 8 pixels model is not recommendable because it has low information and could produce many errors. The 16 pixels model is excellent; both models are great for applications in real time because their response time is fast and accuracy. The second point is for the model with bigger dimensions of 64, 128 and 256 pixels. The 64 pixels model for processing information is the best because it has much information but requires 5 minutes to train. If there is an application where the time is not important this model is perfect. The 128 and 256 models are not advisable to use. Both models have much information that decreases the performance but does not reduce the epoch number. Our future work is to approve other deep layers such as the RELU and DROPOUT technique that are special for models with big images.

References

1. Alex Krizhevsky: Convolutional Deep Belief Networks on CIFAR-10 (2009)
2. Alex Krizhevsky: Convolutional Neural Networks for Object Classification in CUDA (2009)
3. Alex Krizhevsky, Ilya Sutskever: ImageNet Classification with Deep Convolutional Neural Networks (2011)
4. Torch7. www.torch.ch
5. Kaggle Competitions. www.kaggle.com/competitions
6. Cifar10 Model. www.cs.toronto.edu/~kriz/cifar.html
7. Nvidia GTX860M Specification. www.geforce.com/hardware/notebook-gpus/geforce-gtx-860m