International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552 Volume- 8, Issue- 6, November 2020
https://doi.org/10.21276/ijircst.2020.8.6.8
www.ijircst.org

Resizing Tiny Imagenet: An Iterative Approach Towards Image Classification

Praveen Kumar, Rishabh Singh, Nilesh Kumar Singh, Hitesh Agarwal

ABSTRACT- Deep neural networks have attained almost human-level performance over several Image classification and object detection problems, deploying robust and powerful state-of-the-art networks. Stanford's Tiny ImageNet dataset has been around for a while and neural networks have struggled to classify them. It is generally considered one of the harder datasets in the domain of image classification. The validation accuracy of the existing systems maxes out at 61-62% with a select few shooting beyond 68-69%. These approaches are often plagued by problems of overfitting and vanishing gradients. In this paper, we present a new method to get above average validation accuracy while circumventing these problems. We use the resizing image technique which trains multiple model times over different image sizes. This approach enables the model to derive more relevant and precise features as we move towards the original image size. It also makes the training process adaptable to available hardware resources. After reaching the image size, we hyper-tune other parameters such as Learning Rate, Optimizers, etc. to increase the overall validation accuracy. The final validation accuracy of the model using resizing and hyper tuning is 62.57%.

KEYWORDS- Tiny ImageNet, Residual Networks, Classification, ILSVRC, Deep Architectures.

Manuscript Received November 18, 2020

Praveen Kumar, Student, Department of Computer Science & Engineering, MVJ College of Engineering, Whitefield, Bangalore, India (email: inboxpraveenkumar17@gmail.com)

Rishabh Singh, Student, Department of Computer Science & Engineering, MVJ College of Engineering, Whitefield, Bangalore, India

Nilesh Kumar Singh, Student, Department of Computer Science & Engineering, MVJ College of Engineering, Whitefield, Bangalore, India

Hitesh Agarwal, Student, Department of Computer Science & Engineering, MVJ College of Engineering, Whitefield, Bangalore, India

I. INTRODUCTION

Neural networks have existed since the late 1950s. Ever since then, it has been continuously evolving, taking great strides in surpassing the traditional methods. With the breakthrough of Multiple GPU training in 2012 and with the introduction of various advanced hyper tuning and training techniques, neural networks have been constantly over-performing their counter-parts. ImageNet Large Scale Visual Recognition Challenge (ILSVRC), has been hosting the image classification and object detection challenge since 2010 [1]. Tiny ImageNet is derived from ImageNet and is the default dataset for Stanford's cs231n. Tiny ImageNet has 200 classes, where each class has 500 training images and 50 validation images. We are provided with class labels and bounding boxes as annotations text files. The training is evaluated based on the classification of the model over the test dataset. The recent improvement in GPU's computational power and the advent of robust convolutional networks (CNN) offers a new perspective of looking at these challenges. CNN combines feature extraction and training into a single coherent structure and provides end to end solutions from raw pixels to class scores. It makes explicit assumptions about input being an image and starts with a base model with randomly initialized weights and random neuron values, and then iteratively improves over this model. Specifically, neurons are connected to localized neurons, also known as, local receptive connection. It simply passes updated weights and updates neurons based on the loss function. Most importantly, it uses a backpropagation algorithm to take the updated weights back to the head of the model.

II. DATASET

Tiny ImageNet is a subset of ImageNet. It contains 10,000 training images and 1000 validation images [10]. It has 200 classes, where each class has 500 training images, 50 validation images, and 50 test images. All the image sizes are 64x64. Labels and bounding boxes are present only for training and not for test images. All the bounding box labels are stored in the 'annotations' text file. For the validation set as well, the labels are stored in text files. All the original class labels are stored in a separate file named 'words.txt', which helps in mapping the predicted class labels to words. Every image has 3 channels, namely, Red, Green, and Blue. Some images are very hard to predict because the bounding boxes on

several images are beyond the scope of the image, hence making it tougher to predict. For instance, in the following image (figure 2) of a wooden spoon, the receptive field is much higher than the resolution of the image. Hence the whole of the object lies beyond the resolution of 32x32 input image and it's very challenging to predict the ground truth in such cases.

Fig. 1: Wooden Spoon

Fig. 2: Volleyball

Considering a separate scenario where the object itself is not visible or recognizable by the human eye as in figure 2, where the system is supposed to predict a volleyball that is present at the middle of the image but is not visible [10].

III. EXISTING MODELS

Several authors have tried and tested a myriad of neural architectures. Some architectures are deep from the perspective of layers [1], while others are wide in terms of the number of channels [2]. Some authors have overtrained the model to achieve high accuracy [2], while some have controlled the training accuracy [1]. Almost all the models which take on the Tiny ImageNet challenge suffer any of the following 3 problems:

1. Gradient explosion or Vanishing gradients
2. Overfitting
3. Saturating validation accuracy.

Both exploding and vanishing gradients problem occurs when the neural network is trained on gradient-based methods. Vanishing gradient is the decay of information through time. During backpropagation, the weights on neurons are updated and these values tend to degrade as the model trains. Hence, the finer the neurons become, the more information we lose which in turn impacts the accuracy of the model [3]. Exploding gradients occur when large error gradients accumulate and result in abrupt updates during backpropagation. When we have large gradients, the networks tend to become more unstable. Activation functions don't work in the way they are supposed to and we start losing relevant information while the error gradient accumulates, which leads to poor predictions. Methods such as gradients clipping and regularization are used to avoid exploding gradient issues [4]. The issue of Overfitting has been for long enough since 2012. Overfitting is caused when the model can achieve very high training accuracy but very less validation or test accuracy. It suggests a large gap between the training and validation accuracy curves. It tends to degrade the efficiency of the model in unseen conditions and hence leads to unreliable network performance. One way to reduce the accuracy gap between training and validation is to use dropouts [11], which is essentially random dropping of a fixed percentage of neurons at every epoch. The dropping of neurons is completely random and the dropped neurons do not play any role in prediction. We have the dropout phase, only during training and not during validation.

IV. PROPOSED MODEL

The proposed model suggests the following remedies to the discussed pitfalls.

1) Cyclic Learning Rate (Figure 3) is effective in situations where the gradient is stuck at local minima or dangling at a plateau. Cyclic learning rate helps the network by iterating between minimum and maximum learning rate range based on the step size [8].

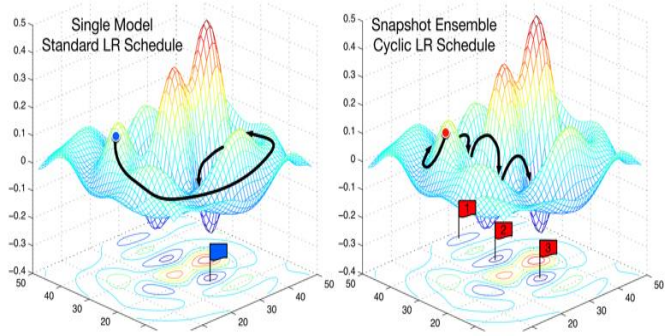

Fig. 3: CLR

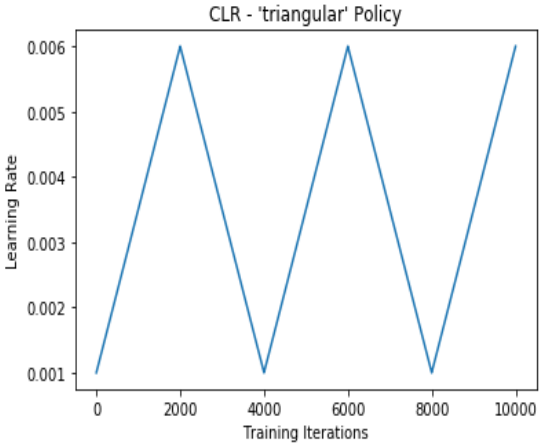

Fig. 4: CLR Triangular policy

The length of a cycle and the input parameter step size can be easily computed from the number of iterations in an epoch. An epoch is calculated by dividing the number of training images by the batch size used [8]. Saddle points have small gradients that slow the learning process. However, increasing the learning rate allows for more rapid traversal of saddle point plateaus [8].

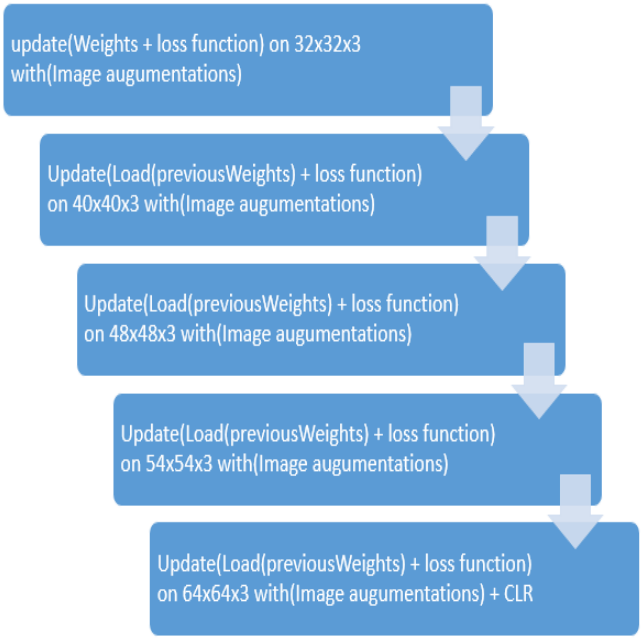

Fig. 5: Proposed Model Architecture

2) Instead of training the whole network on Residual Nets in an end to end approach on the images of size 64x64, we train 5 separate models with variant image sizes. The weights of the 1st model are used for initializing the weights of the next model. This process continues in a chained manner until we reach the desired size of 64X64. The five models are similar but only separable based on input image size and image augmentation techniques. The proposed 5 models follow the structure depicted in Fig. 5.

Using the proposed model, we can train the model on variable dimensions of the object, and hence the model is more flexible and can adapt to multiple receptive fields based on the new input sizes. The weights of the initially trained model are then utilized to initialize a new model. The new model is further trained on new input image size with other hyperparameters such as the number of layers, learning rate, optimizers, loss, and accuracy metrics are kept constant.

The proposed model tackles the problems present in the existing models in the following manner:

1) Saturating accuracy is resolved using Reduce Learning rate on plateau [5].
2) To avoid the problem of vanishing gradient [6], we monitor the loss function vs the number of epochs. As google TensorFlow Playground [7] suggests, the loss should decrease with increasing epochs.
3) Overfitting is tackled by carefully choosing the steps per epoch and deploying Dropouts at a later stage in the training process [11].

The complete training process follows the following hyper parameters and transitions.

The 1st model is trained on an image size of 32X32. The batch size during training remains constant at 128. SGD optimizer [12] is used with a learning rate of 1.0 and a momentum of 0.9. The image augmentation strategies used is:

- rotation=35%
- width shift range=0.2
- height shift range=0.2
- fill mode=nearest
- feature-wise standard normalization=True

We train the model on given configurations for 145 epochs. The weights from the previous model are used to initialize weights of a new model which is trained on images of size 40X40.

The 2nd model has a learning rate of 0.01 and a batch size of 70. The model is trained for 100 epochs and the best weights are saved. The 3rd, 4th, and 5th models work on the same configuration except for the learning rate which is switched to 0.1 and image size. The 3r d model trains with an image size of 48X48, 4th on 54X54, and 5th on 64X64 with the number of epochs being 100, 65, and 50 respectively. The triangular cyclic learning rate is introduced at 64x64.

Cyclic learning rate tuned based on SGD scheduler [12] with scale function value 1. The minimum learning rate is set to 0.001 and a maximum of 0.1 with a step size of 782, which is 2 times the length of the image generator. The model is compiled over SGD without focal loss function [12].

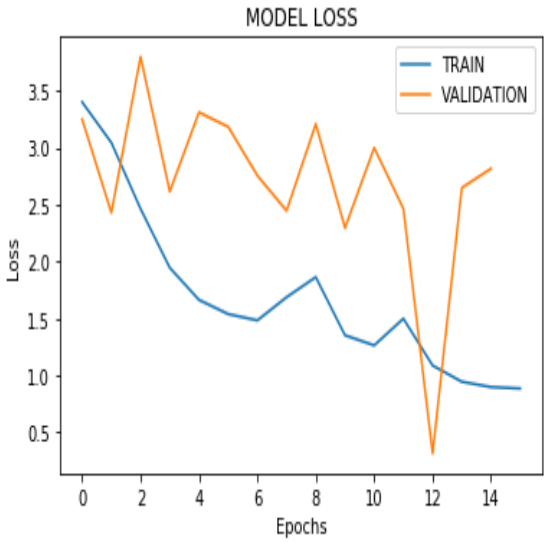

Fig. 6: Final validation accuracy

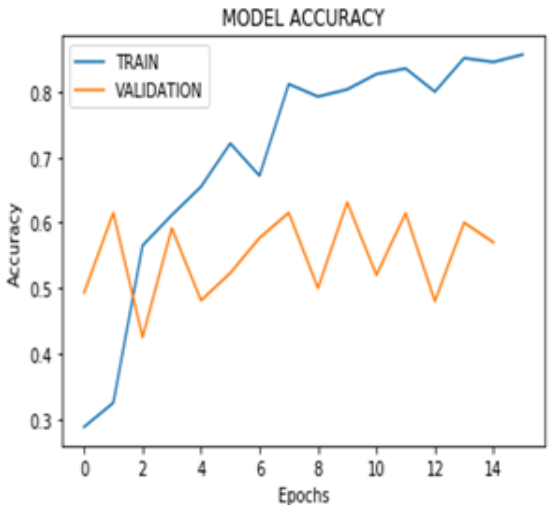

Fig. 7 Final loss

V. RESULT AND DISCUSSION

The highest accuracy that the final model could achieve was 62.57% after deploying CLR. Image size, number of parameters, and accuracy for each model is tabulated in table 1

Table 1: Model wise description of the training process

Image size	Number of parameters	Number of epochs	Validation accuracy
32x32	11,282,248	145	43.91%
40x40	11,282,248	100	48.58%
48x48	11,282,248	100	54.03%
54x54	11,282,248	65	56.81%
62x62	11,282,248	50	59.6%
62x62 with CLR	11,282,248	16	62.57%

It is evident from the table that the accuracy gradually increased with increasing image sizes. There seems to be very little evidence of overfitting because the accuracy gap of training accuracy and validation accuracy is very less. This implies that any of the trained models could be Fine-Tuned and be trained to work for other similar datasets thereby saving a lot of time and resources. The proposed model provides a solution for training variable images without overfitting and explores new possibilities of training with different aspects of input data. Hard images are the images that are difficult to classify, for example, depicted in Fig 2. Such images can be trained multiple times compared to other easy examples. We did not perform OHEM [9] (Online hard example mining) which is also used in state-of-the-art mAP on PASCAL VOC 2007 and 2012 respectively. However, OHEM is a very GPU extensive process and requires multiple high-end Tesla/NVIDIA GPU hours to train the model. As proposed in [9], we are confident that the model's accuracy and results will increase with further methodologies of training hard examples.

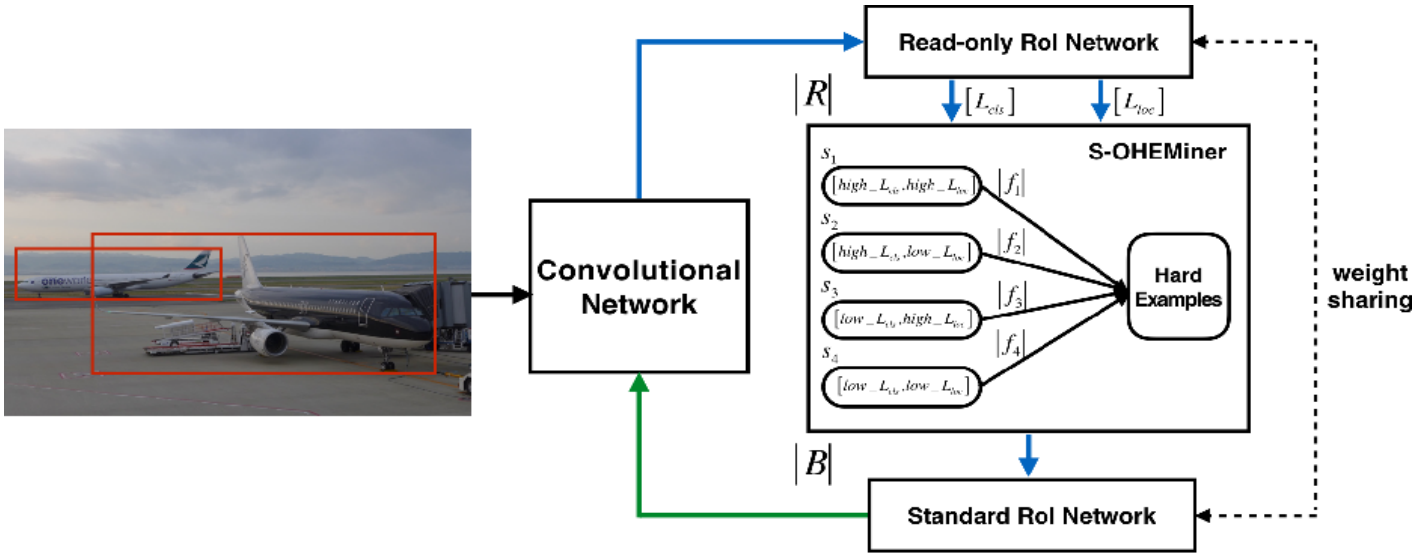

Fig. 8: OHEM

VI. CONCLUSION

It is evident from the findings that the iterative approach used to train the custom Residual Network helps to achieve higher validation accuracy. The five-model approach working in sync with CLR (Cyclic Learning Rate) ensures that there is very little to no overfitting. The approach also seems to take care of the vanishing gradient problem as the loss function is monitored against the number of epochs. These findings can be utilized by researchers and engineers to adopt similar approaches toward solving a wide array of image classification problems.

REFERENCES

[1] Z. Abai, N. Rajmalwar, et al., "DenseNet Models for Tiny ImageNet Classification," arXiv:1904.10429 [cs.CV].

[2] S. F. L. Shi, "Kaminet - a convolutional neural network for tiny imagenet challenge," CS, vol. 231, 2015.

[3] A. S. Walia, "The Vanishing Gradient Problem." [Online]. Available: https://medium.com/

[4] "What is the Exploding Gradient Problem?" [Online]. Avail- able: https://deepai.org/machine-learning-glossary-and-terms/exploding- gradient-problem

[5] L. N. Smith, "Nicholay Topin. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates," arXiv:1708.07120 [cs.LG].

[6] S. Hochreiter, " The vanishing gradient problem during learning recurrent neural nets and problem solutions," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 2, no. 107–116, 1998. [Online]. Available: https://www.bioinf.jku.at/publications/older/2304.pdf

[7] Google, "TensorFlow playground ." [Online]. Available: https:// playground.tensorflow.org/

[8] L N. Smith, "Cyclic Learning rates for training neural networks", arXiv:1506.01186[cs.CV]

[9] Abinav Shrivastava, Abhinav Gupta, Ross Girshick," Training region-based object detectors with online hard example mining", arXiv:1604.03540 [sc.CV]

[10] ILSVRC, "https://tiny-imagenet.herokuapp.com/"

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov," Dropout: a simple way to prevent neural networks from overfitting", journal of machine learning research 15 (2014)

[12]Jarek Duda, "SGD momentum optimizer with step estimation by online parabola model", arXiv:1907.07063 [cs.LG]

