IMAGE CLASSIFICATION USING DEEP NEURAL NETWORKS

Somesh Nayak	Dhruv Tewari	Harshit Gupta	Savio Joseph	Rahul Kasliwal
nayaksomesh29@gmail.com	dtewari2001@gmail.com	msg.harshit@gmail.com	saviandro@gmail.com	kasliwalrahul05@gmail.com

Abstract

In this project, we tackled image classification on the dataset provided by the host in the Kaggle Competition by implementing and training from scratch state-of-art model structures- VGGNet, ResNet, AlexNet and other custom-made models. After fine-tuning and evaluating the performance on all models, we used our best performing model. We try to explain each of these models accurately and replicate them with our code to the best degree possible. We also process images by flipping, rotating, zca-whitening and more. Through this task, we try to correctly classify all the test images provided in the data set according to the right labels based on the training given to the model using the training and validation images.

Introduction

Over so many years of human evolution, our ability to see people, objects, and places and to understand them has helped us to develop and hone our cognitive skills. For example, when we see a car on the road, we are immediately able to identify various attributes pertaining to the car, such as its type - sports, SUV, sedan or hatchback, or its color - black, white, blue, etc. This cognitive process helps us to easily differentiate between things that may share some common characteristics, like, both a car and a mini truck have four wheels, but our past learning enables us to differentiate between a car and a mini truck based on how they look that is their body-type although they have four wheels. It only seems easy because our brain is incredibly good at understanding images. Therefore, a large amount of research work has been put in to the field of computer vision. This same process is often required to be done by a machine which is known as image

classification, a major precursor for computer vision.

Computer Vision can be defined as the idea of giving a computer the ability to see the world and identify the objects around on the basis of the input from a camera. Image classification can, therefore, be defined as the process by which the computer can recognize a new set of images and classify them into the right classes based on class labels. Convolutional Neural Networks (CNNs) have enjoyed great success in large scale image classification tasks due to availability of large image datasets and high computing power such as TPUs and GPUs. We have used ResNet model with some fine tuning of hyperparameters. We have explored and compared of our results to fine tune our model based on the dataset to obtain the best result possible.

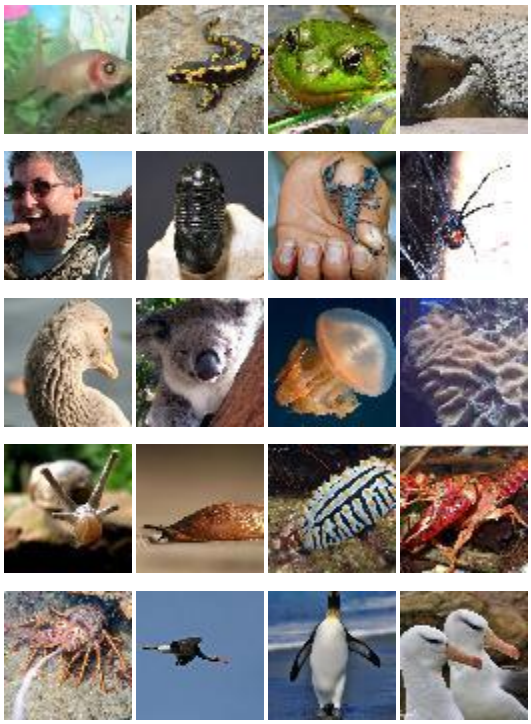
Dataset and Features

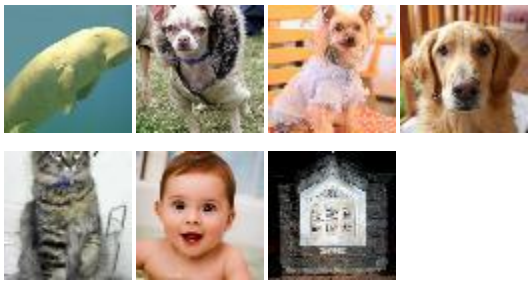
The image classification task requires us to create a model that can rightly predict the labels of each of the images provided in the test folder of the data set. The data set provided comprises of 200 different classes of image, each class containing 450 training images and their bounding box annotations.

The test and validation folders contain 10000 images. Annotations comprise of both class labels and the bounding boxes. While Test set only has images and we have to predict its label and without localizing it by bounding boxes as in training and validation sets.

The file- wnids.txt has list of all categories. The file- words.txt has words which describes images related to a particular category.

Given below are examples of some of the many classes of images from our dataset on which we have to train our model:

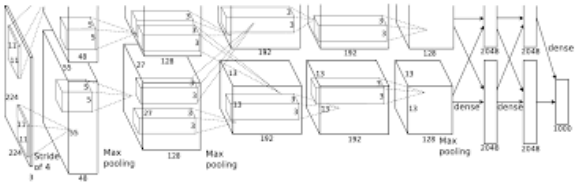




Model Architecture

This section contains model architectures and suitable parameters of different models used by us in this project.

AlexNet:



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

The above figures (from [10]) visualize the architecture of the used AlexNet model. It consists of 11×11, 5×5,3×3, convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum etc. AlexNet contains 5 convolutional layers and 3 fully connected layers. ReLU is applied after very convolutional and fully connected layer. Dropout is applied before the first and the second fully connected year. When it comes to the complexity of the model, the network has 62.3 million parameters. For the AlexNet, we use hyper parameters such as: dropout rate (0.0 or 0.5), batch-size (32 or 16), and two different optimizers (Adam and SGD).

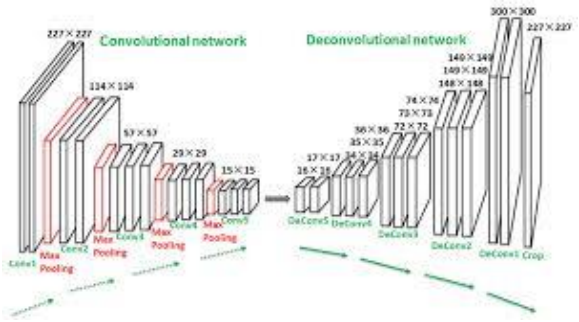
ResNet:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNet makes it possible to train up to hundreds or even thousands of layers and still achieves compelling performance. Taking advantage of its powerful representational ability, the performance of many computer vision applications other than image classification have been boosted, such as object detection and face recognition. The core idea of ResNet is introducing a so-called “identity shortcut connection” that skips one or more layers. The ResNet model tends to overfit as a result of which, the stochastic depth is

introduced. It randomly bypasses one or more ResNet blocks, thus making the network shorter instead of thinner which dropout regularisation does. This helps in increasing validation accuracy by a considerable amount.

VGGNet:



This model makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. All hidden layers are equipped with the rectification (ReLU) non-linearity.
The image is passed through a stack of convolutional layers, where the filters were used with a very small receptive field: 3×3. It also utilizes 1×1 convolution filter. The convolution stride is fixed to 1 pixel. The spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers Max-pooling is performed over a 2×2-pixel window, with stride 2.

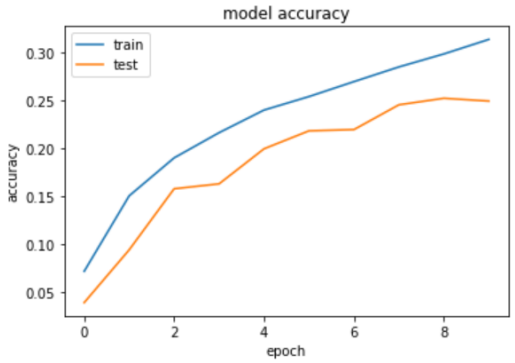
Methodology

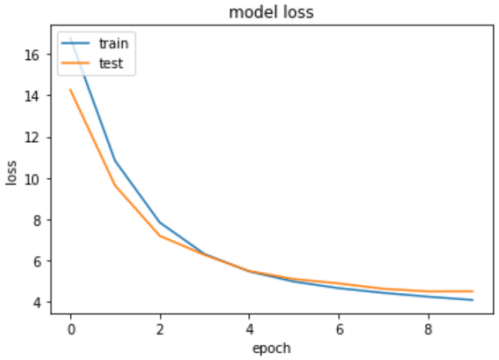
For preprocessing, we created a dictionary which had images with their respective labels. We dropped the bounding box annotations since, it was observed that not having the surrounding pixels affected the validation accuracy. We converted them to grid and each small box of grid had a pixel value. We scaled down the images to 1-256-pixel value range. We then normalized the data by subtracting the mean and dividing each value of the array by 128.
We used Resnet as our CNN Architecture. We had a low Kaggle submission score of about 0.004. This was because the test set data was being read randomly and not in the desired order. This was realized and sorted out by looping over files and not image numbers. The accuracy further improved when we shuffled the training examples. In order to introduce more variation in the images, we used the ‘Datagen’ library to rotate the images, horizontally flip them and even feature wise center them. We trained the model and managed to get 39% train accuracy and 30% validation accuracy.

However, after training for several epochs, the validation accuracy was not improving at the same rate as training accuracy, this was due to overfitting which was identified.
In order to avoid overfitting, we added L2 regularization and Dropout layers as well. It was implemented in such a way that the available amount of RAM can sustain it. We improved the weight decay and let the learning rate improve itself. We also kept switching between models like ResNet18, ResNet50 and ResNet101 till we got our best Kaggle Submission score of 0.56. Training was done in small epoch numbers of 10 and we kept saving and loading the model as well as updated weights. The accuracy further improved by roughly 5% when we increased batch size steadily from 250 to 500 and further till an OOM error occurred. By this time, the model had reached 90% train accuracy and 56% validation accuracy.

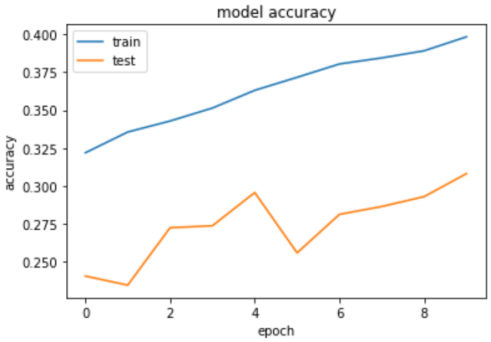
Relevant Images

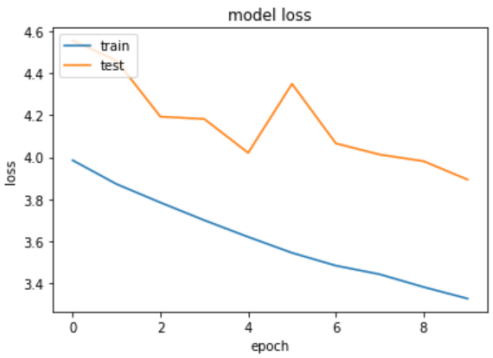
The below 2 pictures display the model accuracy and error loss after training for 10 epochs with horizontal flipping of images and batch size of 500 using ResNet model-



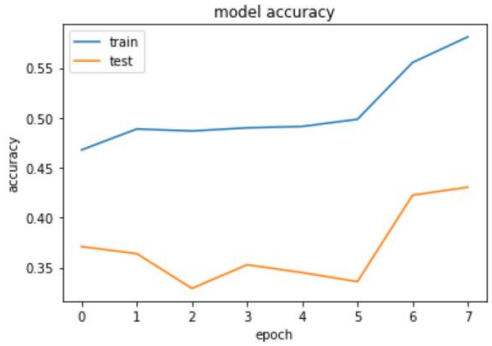


After training for another 10 epochs it is observed that the rate at which validation accuracy improves is much slower than the training accuracy. The loss graph also supports the above observation.

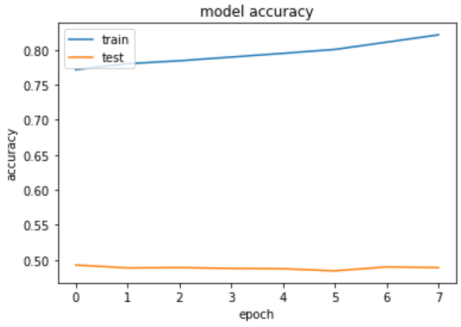




After training for several epochs, we observe that at epoch number 5 of the batch, the training and validation accuracy spikes up. This is because the learning rate has degraded and become smaller in size which in turn increases accuracy. The batch size is increased now.



After training for several epochs, the validation saturates and we used the trained model and weights to make predictions



Result and Analysis

After experimenting with all the mentioned models and trying to tune their hyper parameters, we got the best accuracy of 56.3145% from ResNet models combined with modified and processed images.
The fact that ResNet is able to outperform all of the other models even with reduced architectures than the original is understandable because it successfully eliminates the possibility of vanishing or exploding gradients and makes the optimization process easier. Therefore, ResNet was more efficient to train, and converged relatively faster than the other models. Sadly, the model is still overfitting even after tuning hyper-parameters. It is difficult to resolve this issue simply because of lack of computational power.

Conclusion and Improvements

The type of optimizer is really important for the training process. Adam outperforms SGD optimizer: Adam converges faster and reduces loss better. Hence, Adam is a good optimizer to start with and SGD can be used as an alternation.

Batch Size affects the performance at smaller learning rates of the order 1.0e-04 and smaller. It is was observed that during training when learning rate drops and batch size increase, the validation accuracy improves by 3-5% which is a considerable margin of improvement. Smaller batch sizes may require more iterations.

When we used AlexNet it performed worse than VGG16 and ResNet since it does not deal with 64x64 images which is too small for it. After adding l2-norm for the VGG-16 and ResNet models, it performed better, but the best performance was recorded by ResNet models which helped us achieve the accuracy of 56.314% which is pretty commending considering the sheer volume of data and availability of computing resources with us.

Finally, there still exists some potential ways of improving accuracy further:

Try a learning rate with a greater decay, implementing finely tuned l2-regularisation for every layer of the model, implementing batch-norm more effectively. In our baseline model, there is significant overfitting caused by insufficient ability of generalization.

References

[1] Alex Krizhevsky and Sutskever, Ilya and Hinton, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. http://papers.nips.cc/paper/4824-imagenetclassification-with-deep-convolutional-neuralnetworks.pdf

[2] Karen Simonyan, Andrew Zisserman VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGESCALE IMAGE RECOGNITION

[3] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, Laurens van der Maaten Densely Connected Convolutional Networks

[4] Hujia Yu Deep Convolutional Neural Networks for Tiny ImageNet Classification (CS-231N)

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,pages 1–9,2015.

[6] R. Srivastava, K. Greff and J. Schmidhuber. Training Very Deep Networks. arXiv preprint arXiv:1507.06228v2,2015.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. The Journal of Machine Learning Research 15(1) (2014) 1929–1958.

[8] For learning how to plot graphs and model architecture- https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/

[9] Papers for giving insight to models and parameters - https://arxiv.org/ftp/arxiv/papers/1904/1904.10429.pdf,https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf,https://papers.readthedocs.io/en/latest/imageclassif/resnet/,https://pdfs.semanticscholar.org/384c/e792cf2b2afbe001f2168bfe7d5e7804c736.pdf, ,https://www.studocu.com/en-us/document/stanford-university/convolutional-neural-networks-for-visual-recognition/other/residual-networks-for-tiny-imagenet/751997/view

[10] https://neurohive.io/en/popular-networks/ alexnet-imagenet-classification-with-deep-c onvolutional-neural-networks/