IMAGE CLASSIFICATION USING DEEP NEURAL NETWORKS
Final Project Report (17 July 2020)

Harish Bharadwaj S

1. Abstract

This report analyzes the problem of image classification with different deep neural networks apt for this particular problem.Since the introduction of Alex net, deep neural networks especially CNN (convolutional neural networks) have shown a promising way for image classification. Being one of the classic problem in Computer Vision, there have been many research works in this field and new models and techniques have been invented. But not all are suitable for our problem. In this paper we will experiment with the given dataset using some CNN models and modify, optimize them to better fit the given task

2. Introduction

Convolutional Neural Networks formed as the base for Computer Vision.These networks are capable enough to extract features from an image. For example like the color, shape and symmetry of an object. This paved way for object recognition in an image.
CNNs generally consists of a series of kernel multiplication, pooling, activation function. Many models apply this in a repeated fashion with slight changes. After Alex net, VGG network had a breakthrough as it out preformed its predecessors.The hidden layers in the network help in extracting smaller details like edges, corners,curves and surfaces.More hidden layers helped classifying the image better. But as the neural network got deeper and deeper, it became hard update the weights.

Nowadays CNN models apply blocks which are made up of these layers. Generally there are two blocks: Identity block and Convolutional block. This kind of architecture was implemented in later architectures like Resnet. Resnet made a huge break through and are among the first to use batch normalization.This model constitutes of blocks that help in building a deep network quickly and also reduce diminishing gradients.

The recent networks include inception net and xception network which are further development of previous models. These models are designed with deeper layer but also not compromising the power of generalization.These new models are among the first to implement asymmetric convolution and batch normalization to auxiliary layers.

3. Related Works

The state of the art models have been referred in this report. A brief study of each model will follow in this section. We are going to experiment on the following models by altering the parameters, adding/removing layers and reshaping the filters and regularization. The experimentation part will be discussed in the coming sections.

VGG16 and 19:

The CNN architecture started getting deeper. This was due to the fact that the performance of the CNN improved by increasing the size.VGG16 was built by Visual Geometry Group and it had 13 convolutional networks and 3 fully connected layers (thus VGG16) along with ReLU activation similar to alex net.This particular network has around 138 million parameters.The neural network consists of five convolutional layers, some of which are followed by max-pooling layers, and two globally connected layers with a final way softmax.VGG19 is a variant of VGG model which in short consists of 19 layers (16 convolution layers, 3 Fully connected layer, 5 MaxPool layers and 1 SoftMax layer).

IMAGE CLASSIFICATION USING DEEP NEURAL NETWORKS
Final Project Report (17 July 2020)

Harish Bharadwaj S

ResNet-50:
CNNs have seen an increase in number of layers for better performance but with the network depth increasing, accuracy gets saturated and then degrades rapidly. This problem was rectified with ResNets by a group of Microsoft researchers and it had around 26 Million parameters.. ResNets overcome this problem by using skip connections called residuals/ short connections at the same time building the models more deeper. ResNet is among the first to implement batch normalization.The basic building block for ResNets is the Convolutional and Identity block.
.Popularized skip connection.
. Designing even deeper CNNs (up to 152 layers) without compromising model's generalization power
. Among the first to use batch normalization.

Inception V3:
Inception-V3 is a successor to Inception-v1, with 24M parameters.The idealogy for Inception-v3 is to avoid representation bottlenecks ie to reduce the input dimensions of the next layer and have more efficient computations by using factorization methods. Inception-v3 is the network that adds tweaks to the optimizer, loss function and adding batch normalization to the auxiliary layers in the auxiliary network.
. Among the first designers to use batch normalization
. Factorizing n×n convolutions into asymmetric convolutions: 1×n and n×1 convolutions
. Factorize 5×5 convolution to two 3×3 convolution operations
. Replace 7×7 to a series of 3×3 convolutions

Xception Model:
Xception Model is proposed by Francois Chollet. Xception is an extension of the inception Architecture which replaces the standard Inception modules with depthwise Separable Convolutions.

4. **Methods**
The models discussed in the above section have been altered and optimized for better results.I have implemented VGG16, Resnet50 and VGG19. The inception and xception nets are very deep networks, so training them from scratch would be difficult with the small dataset. Transfer learning also difficult for the above two models as the minimum image size should be 75. Resizing the image is an option but this could lead to distortions in the image while upscaling.

a. **VGG16**:
It is one of the simple but powerful CNN model. It has very large number of parameters and is not that deep in architecture. It is a series of convolutional and maxpool layers. The image matrix is reduced to smaller ones using a set of filters. The kernel size is set to a smaller value since the size of the input image is small. The padding is kept same in all convolutional layers.After the series of convolutional layers and maxpool layers we add flatten to make the output one dimensional. Then we add 2 fully connected dense layer of same size with relu.Dropout layers is optional after each dense layers.Finally a dense layer with softmax.
This model is great getting started with to understand the dataset.Based on this model we can make further decisions whether the model is underfitting and requires more data or else overfitting and requires normalization.

b. **Resnet50**:

This model unlike VGG is more deep, It uses a series of blocks which is comprised of convolutional and activation layers.The model has 5 stages. It is one among the few models that were first to implement batch normalization.After stage 5 there is an average pooling layer followed by flatten and dense layer with softmax activation.
This model can help if previous model underfits as its has more layers to extract more features. This model can also reduce over training by implementing batch normalization.

c. **VGG19**:

This model is very similar to VGG16 but it has 19 layers in total.All other methods of the model are same as VGG16.
It is an improved model and does better for larger classification.

5. Dataset And Features

The train dataset has 200 classes with class_id as name of the folder.Each class consists of images folder and annotation.txt. Each images folder have 450 images. The task in Data Preprocessing is to resize each image according to the bounding box provided in its respective class's annotation.txt file. But resizing doesn't do well and the images are not changed. Instead these images are converted to numpy arrays for easy calculations.

The validation dataset consists of images folder and an annotations.txt file. The image folder has about 10000 images.The annotations.txt file specify the class to which each image in images folder belongs to and the bounding box positions. The task in DataPreprocessing here is to create 200 folders for the 200 class ids.Then for each image in images folder it is resized based on the bounding box location and saved in class id folder to which it belongs. Resizing is later ommited due to poor performance. After executing above process, each class folder in validation set has 50 images.

There is no changes to the test folder. The test folder has the images folder. The images folder has 10000 images, all 200 classes of images, 50 in each class in random order.

Some of the images in the above folders are black and white i.e they have only one channel. But all our models require 3 channels in input. So to tackle this problem I copied the first channel and added the copied as the other 2 channels. As we know that if all RGB have same value then we get a shade of gray.Since color is one of the important feature, it is important to maintain 3 channel, to define boundaries of objects. The rest of the features are also given equal weightage and can be achieved through weight normalization.

Data Augmentation is another way to increase the dataset and generalize the features of a class. It can help when model is underfitting. The images are flipped, zoomed and rotated. This also helps in generalizing the object. This is because the boundary edges and curves of the object are moved when the image is transformed. Thus this helps the model to understand the object and learn its feature rather than the whole image's feature.

IMAGE CLASSIFICATION USING DEEP NEURAL NETWORKS
Final Project Report (17 July 2020)

Harish Bharadwaj S

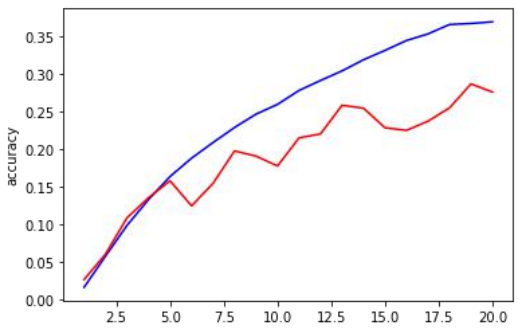
6. **Experiments**

The task is to predict the class of each image in test dataset.The output is a csv file. Each line in the file contains the image id and its predicted class Id separated by a comma.

Experiment 1:
VGG16 (No dropouts):

This is the first model I implemented for this problem. There were not too many layers. Using data generator I loaded the images into the model. Since there were 200 classes with 450 images in each. Training the network and learning the features were cumbersome. As I suspected the model suffered from underfitting. Though this model has many parameters but it is not deep enough to extract features easily.

The solution to this problem would be to add more data. This could be easily tackled with using data augmentation. We can also add batch normalization after convolutional layers to normalize the weights. We can also add regularization like L2 in the final dense layers. But still I believed that a deeper architecture would do much better. That is the reason to change to a much more deeper architecture.



In the above graph the blue and red lines are training and validation accuracy plotted against number of epochs.
We can clearly see that the training accuracy and validation accuracy are pretty low.This is a case of underfitting. The training accuracy is saturating around **37%(Loss falls to 2.5195)** and validation accuracy starts to drop after reaching a value of **29%(Loss falls to 3.2940)**.
The F1 score on test data is **21.431**

Experiment 2:
Resnet50(No Data Augmentation):

The images from now on are loaded as numpy arrays as they are easy to handle and modify.The model is deeper and has skip connections. There are 90000 images in training data equally divided into 200 classes. The training and validation performed better than the one's of VGG16 model but still there was room for improvement. The model still had underfitting as training and validation accuracy were still low at saturation.

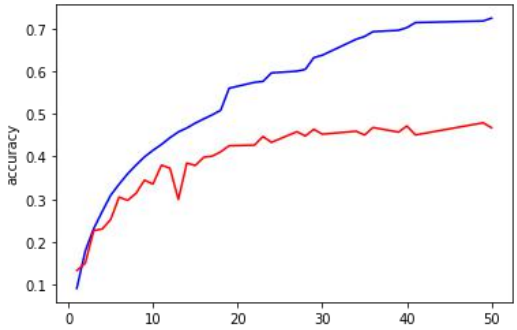
Resnet50(Data Augmentation):

This time using data generator object we transform images and expand the dataset. There is an improvement on training as well as on validation accuracy. Initially the validation accuracy increases along with the training accuracy. Later validation accuracy saturates and training accuracy keeps increasing. The model performs better than the previous experiment. But still we can optimize this case of overfitting.

Resnet50(Data Augmentation +Pixel Normalization):

To improve the accuracy after training we apply pixel normalization before training to the training and

validation dataset.We divide the pixels by 255 to normalize the range Then we subtract the mean pixel value of the training set from all of the training set and validation pixels.



In the above graph the blue and red lines are training and validation accuracy plotted against number of epochs.

Here we can see that the training and validation accuracy have saturated. But there is a huge gap between the saturated accuracy values of training and validation dataset. This is a case of over fitting. The training accuracy saturates around **72%(Loss falls to 1.4506)** and the validation accuracy saturates around **47%(Loss falls to 3.0811)**.
The F1 score on test dataset is **39.542**

Experiment 3:
VGG19(Transfer Learning and Dropouts):

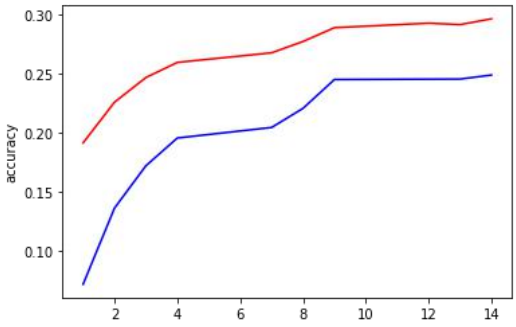
As discussed VGG19 is an improvement over VGG16 model. Transfer learning is a technique where the model has predefined weights.The model has been trained over with other images. So with these generalized weights we can increase the accuracy while training.Afterwards we can add more layers and train them to classify this problem. This method also overcomes overfitting with dropouts optimization after dense layers.

VGG19(Transfer Learning and Batch Normalization and Regularization):
After flattening the output of the convolutional layers we apply batch normalization. This helps the model to regularize weights and study all the features rather than keep on training of a particular feature that was frequent in that batch. Thus it reduces overfitting.At the dense layers we can apply regularization(usually L2) to tackle this problem.

VGG19(Transfer Learning with freezing and unfreezing the bottom layers):

First we freeze the bottom layers then add more layers to the model that are trainable. We train this model for few epochs before it reaches saturation. Then we unfreeze the bottom layers and train the model till saturation. This model gives the best results compared to all the previous models. Thus this model is suited for the given task.

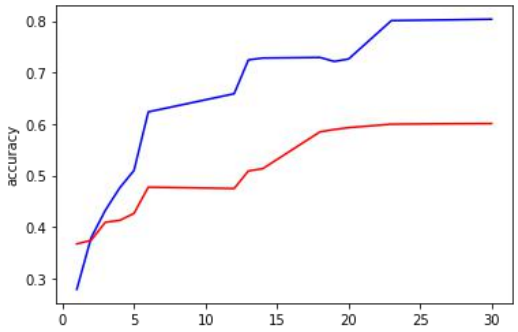
The following two graphs belong to the same experiment. The first one trains on 14 epochs with base model weights frozen.



In the above graph the blue and red lines are training and validation accuracy plotted against number of epochs.

This is the training done with base model weight updation frozen. Here we can study that due to predefined

normalized weights the validation accuracy is greater than the training accuracy.
After training of 14 epochs we unfreeze the base model and again train on 30 more epochs.



In the above graph the blue and red lines are training and validation accuracy plotted against number of epochs.

After unfreezing the bottom layers the training accuracy increases faster than validation accuracy. The training accuracy saturates at around **81%(Loss falls to 0.896)** and the validation accuracy saturates around a value of **60%(Loss falls to 1.713)**.
The F1 score on test dataset is **50.628**

7. Conclusion

As we can see that VGG16 model when trained suffered from underfitting and the Resnet50 model suffered from overfitting. In the final experiment we were able to draw the conclusion that transfer learning along with some kind of normalization can help rectify both underfitting and overfitting problems.

VGG16 model could have been optimized further by introducing data augmentation but it is difficult to say whether it can out perform the succeeding experiments results.

The Resnet50 model could have been further developed by adding dropouts and L2 regularization at Dense layers and train till saturation, but it is difficult to say whether it can out perform the succeeding experiment result.

Finally we have our VGG19 model. This model perform well on our dataset compared to the rest of the models in other experiments.

For further improvement instead of unfreezing all the bottom layers while second training, we can unfreeze the top layers of the base model and keeping the initial few layers frozen. This method will retain the weights for basic classification. Through optimization we can find the from which layer to unfreeze the model.

8. References

Classifying images with CNN
https://towardsdatascience.com/an-image-classifier-with-deep-learning-7284af97b36a
Dogs vs Cat classification
https://www.kaggle.com/c/dogs-vs-cats/
Very Deep Convolutional Networks for Large-Scale Image Recognition
Karen Simonyan, Andrew Zisserman
https://arxiv.org/abs/1409.1556
Deep Residual Learning for Image Recognition
Kaiming He, Xiangyu Zhang, Shaoqing Ren,
Jian Sun
https://arxiv.org/abs/1512.03385
Rethinking the Inception Architecture for
Computer Vision
Christian Szegedy, Vincent Vanhoucke, Sergey
Ioffe, Jonathon Shlens, Zbigniew Wojna
https://arxiv.org/abs/1512.00567
Towards Data Science
Wikipedia