1. Introduction section describing the project in general and why is it useful to do the comparison.

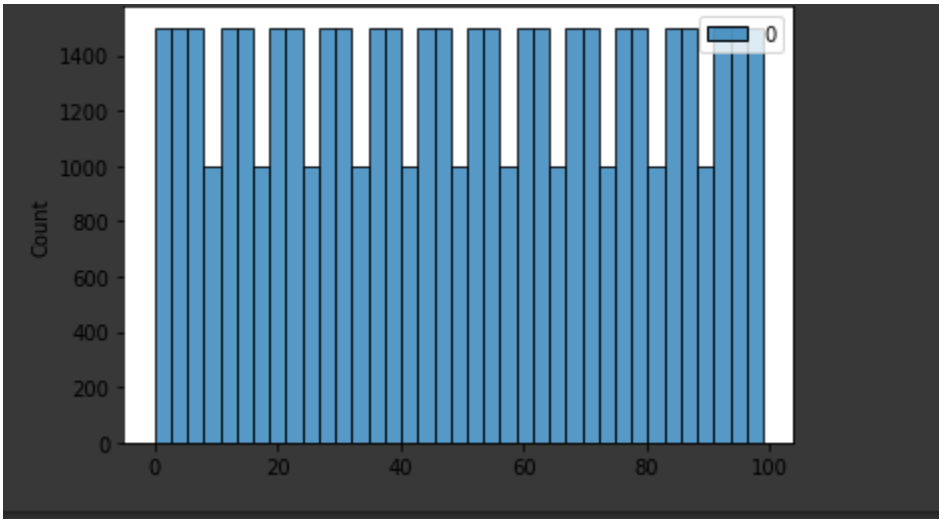
The project -1 requires us to run 3 different deep models namely:

1. VGG-16
2. RESNET and
3. Inception-Net

There are many reasons for comparison of these three models. This project will help us better understand some of the best deep learning architecture models present currently. Also, this project helps us to have a vague idea regarding how different models respond to different optimization or regularization parameters. Also, this project gives some idea regarding which model should be used for a particular type of data. The architecture of these models is some of the best in world and many new models are being developed based on them. So, it will serve us as a basic for future model development and manipulation. Also this project helps us to explore the math's and other fundamentals behind different optimization and regularization method and by the end of project, we have some idea of how to tune the parameters in order to improve the performance of model. Finally this project gears us up in keras, tensorflow pytorch and other machine learning libraries which will prove quite useful in our upcoming masters project.

The data set provided is CIFAR-100 which in no way is a skew dataset. It has 100 different classes and every class has almost equal number of datasets. In these cases, accuracy can be considered as a sufficient performance matrix. However, we will be considering recall and precisions too.

Below is the image of the counts of each class in CIFAR-100:



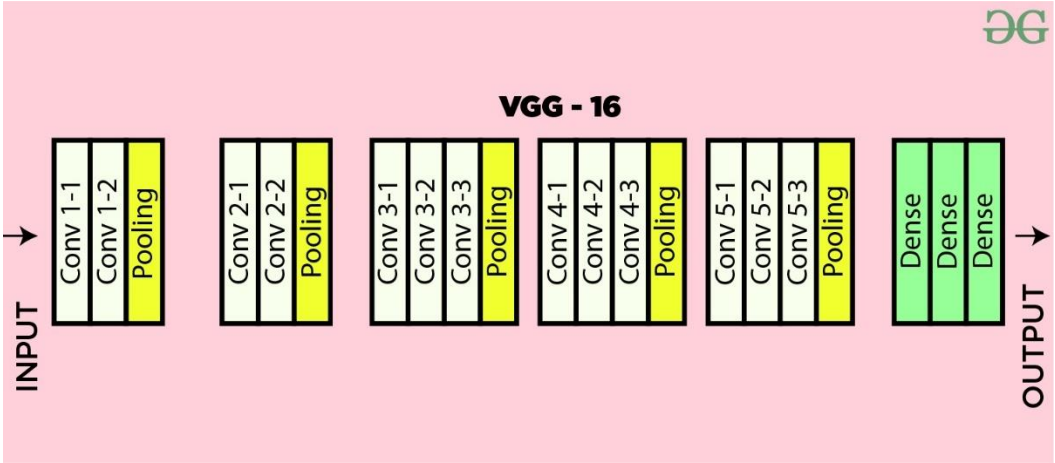
The above graph shows the count of each class in YTrain set. We can thus conclude that there are enough data points for representing each class.

2. **Describe implementation of the architectures, the training as well as the specifics of the training procedure used**

1. **VGG-16:**

   As per google, **VGG-16** is a convolutional neural network that is **16** layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. VGG16 is a convolution neural net (CNN) architecture which was used to win ILSVR(ImageNet) competition in 2014. It is considered to be one of the excellent vision model architecture till date. Its very uniform.

The architecture of VGG-16 is shown below:



I have started with initializing the model by specifying that the model is a sequential model.

As we can see above, VGG-16 is a seems a simpler of three layers. It's a combination of Pooling, Convolutional and Dense layers. However, its simplicity comes at the cost of computational cost.

In fact, after evaluation, we found that this is the slowest model, and generates the largest number of weights ie 59,068,836.

This is like nearly 9 times more than the weights generated by inception layers.

Implementation:

As the name suggests, VGG-16 consists of 16 layers and they are implemented as follows:

2 x convolution layer of 64 channel of 3x3 kernel and same padding

→ 1 x max-pool layer of 2x2 pool size and stride 2x2

→ 2 x convolution layer of 128 channel of 3x3 kernel and same padding

→ 1 x max-pool layer of 2x2 pool size and stride 2x2

→ 3 x convolution layer of 256 channel of 3x3 kernel and same padding

→ 1 x max-pool layer of 2x2 pool size and stride 2x2

→ 3 x convolution layer of 512 channel of 3x3 kernel and same padding

→ 1 x max-pool layer of 2x2 pool size and stride 2x2

→ 3 x convolution layer of 512 channel of 3x3 kernel and same padding

→ 1 x max-pool layer of 2x2 pool size and stride 2x2

The input and output dimension of every layer/set of layers of VGG16 is shown below :

No	Convolution	Output Dimension	Pooling	Output Dimension
layer 1&2	convolution layer of 64 channel of 3x3 kernel with padding 1, stride 1	224x224x64	Max pool stride=2, size 2x2	112x112x64
layer3&4	convolution layer of 128 channel of 3x3 kernel	112x112x128	Max pool stride=2, size 2x2	56x56x128
layer5,6,7	convolution layer of 256 channel of 3x3 kernel	56x56x256	Max pool stride=2, size 2x2	28x28x256
layer8,9,10	Convolution layer of 512 channel of 3x3 kernel	28x28x512	Max pool stride=2, size 2x2	14x14x512
layer11,12,13	Convolution layer of 512 channel of 3x3 kernel	14x14x512	Max pool stride=2, size 2x2	7x7x512

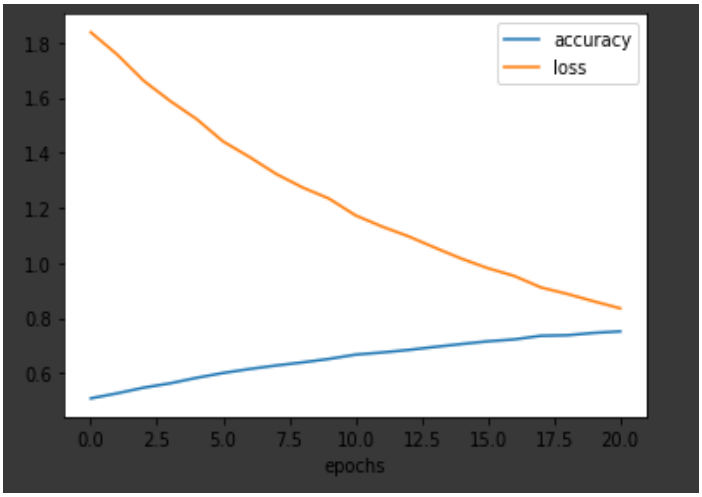
After this, we have 3 dense layers

However, input size of our model is 32*32*3. This model has many advantages. I found it quite easy to understand and analyses how it works. It's a sequential model and although theory suggests that the larger a model, the better performance it will have, we tend to forget that we even need a bigger data set too, in order to train that model.

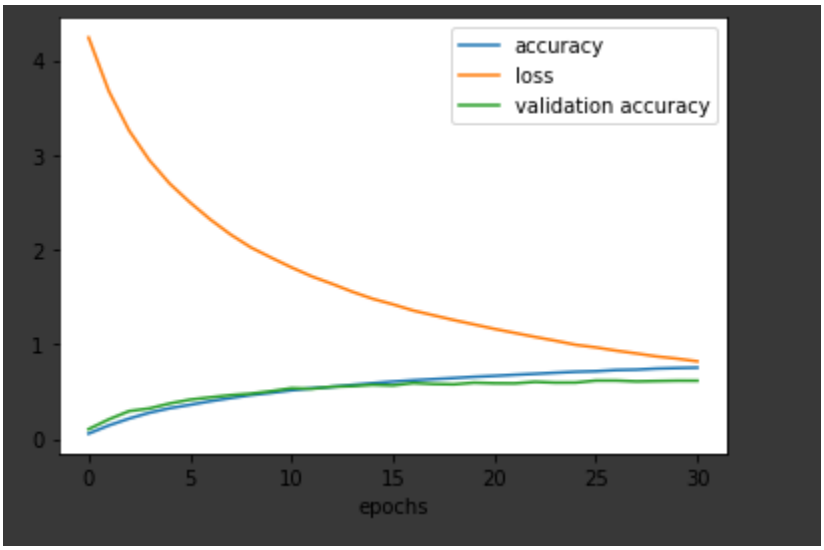
The training procedure

As google said VGG-16 was actually designed for larger data sets, it was actually designed for very large image sets**.** *However, our data set, ie CIFAR-100 is not that big data. So, the chances of over fitting are higher in this condition. Thus, I am removing one of the dense layers (precisely, the layer just before the output layer) This seems to work quite well as we can see that without any regularization, the accuracy for SGD optimizer goes to a whopping 64.68% and 62.41% for adam optimize*

The following graph show the loss and accuracy graphs for non-regularized SGD for VGG-16 model



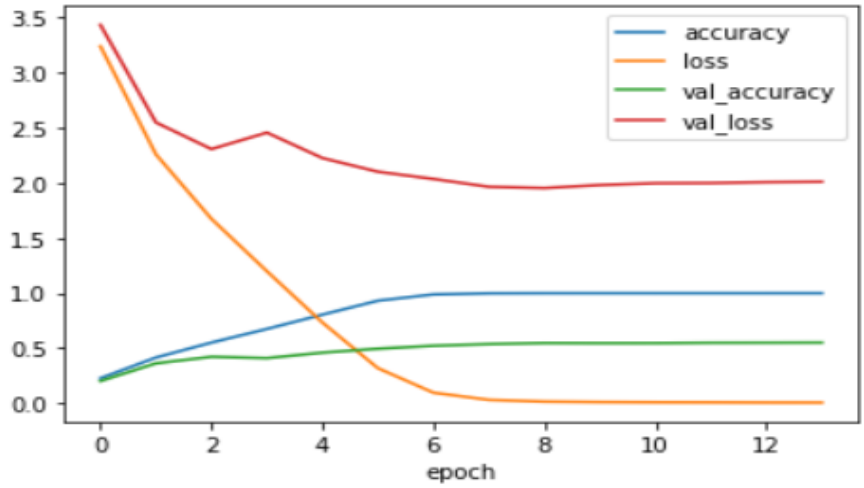
The following graph show the loss and accuracy graphs for non-regularized Adam for VGG-16 model:



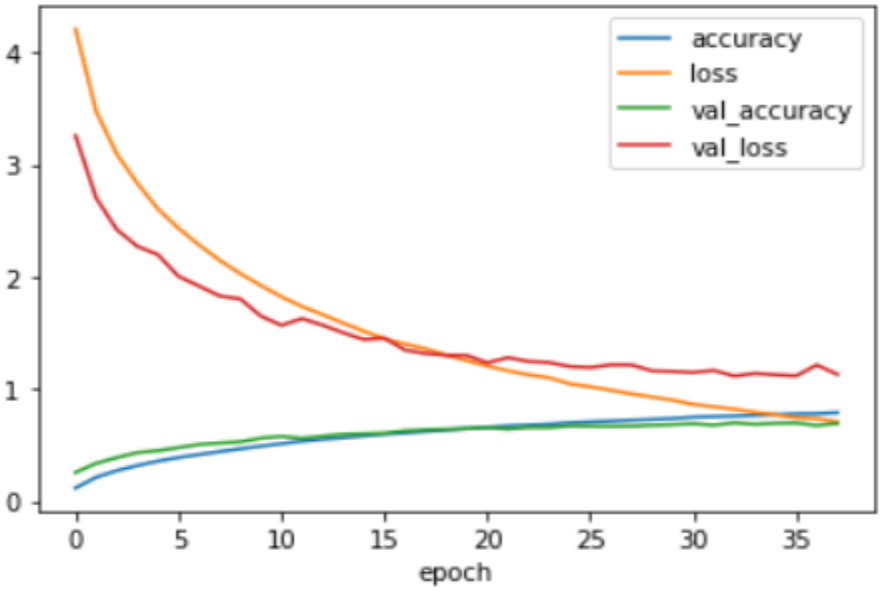
Regularization of VGG-16 with Batch-Normalization

Ones I apply batch-norm to both Adam-optimized and SGD-optimized models, there accuracy increases further to 69.4% and, which is the best accuracy I received for this data set. In fact, they even lowered the over fitting as the training accuracy and test accuracy seems to be coming nearer. This shows that this model along with the specified parameters for optimizer is best fitting the given data set.

The following graph shows the performance of model when regulation of Batch norm and optimization of SGD was applied:



The following graph shows the performance of model when regulation of Batch norm and optimization of Adam was applied:

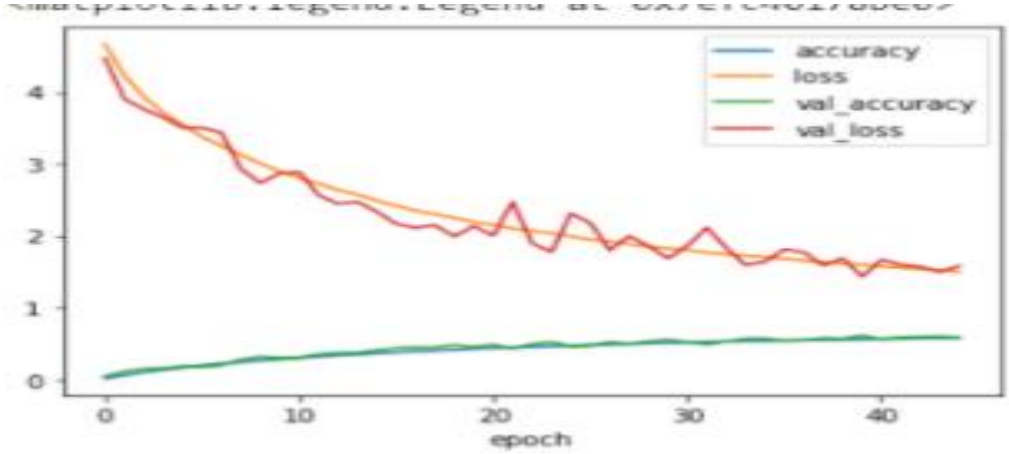


Regularization of VGG-16 with Drop out-Normalization

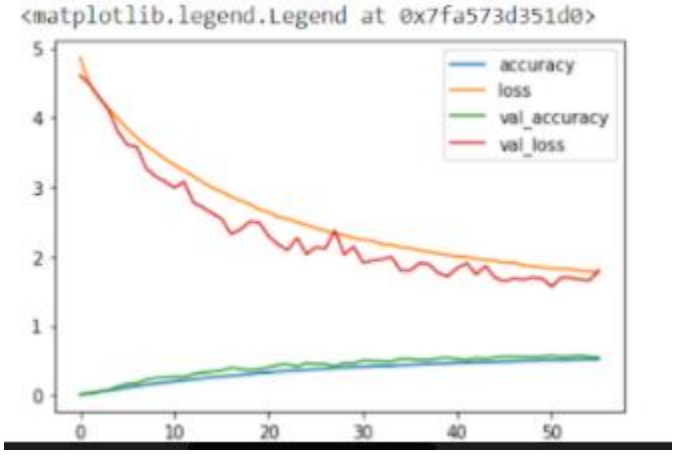
Dropouts doesn't seem to be helping that much, however they are giving a more stable model. In fact, when I compared the test accuracy of SGD optimized, Dropout regulated model with its training accuracy, it was 49.5% to 52.7%. This shows that even though the model is not giving good accuracy on the test data set, its providing a balanced model, ie has a balanced bias and variance which shows it's a good model.

However, even though we are seeing really high accuracies and a seemingly well-balanced bias-variance, model this comes on the cost of high computational timing and resources. Thus, as per my analysis, though VGG-16 is a really good and easy to understand model, however it can't be used as frequently.

The following graph represents the performance of model with Adam optimizer and Drop out regularization:



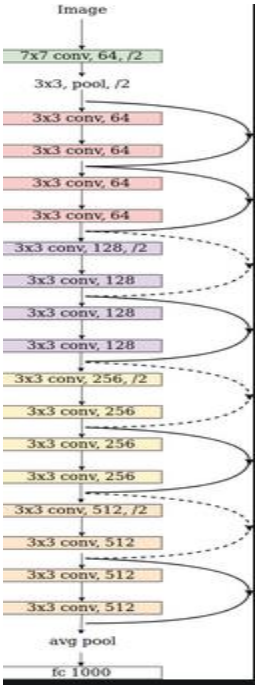
The following graph represents the performance of model with SGD optimizer and Drop out regularization:



In the next section ,we will be discussing about Resnet model

2. Resnet-18 Model:

The architecture design of ResNet-18 can be seen in below diagram



The motivation and working principal of Resnet:

Generally, neural networks are referred as great functions of approximators, thus they should be able to learn identity, f(x)=x quite easily, ie if we attach a network that predicts f(x)=x to any existing network, then by logic, must output the same thing. In other words, we can say that if we keep adding more and more identity layers, we are never changing the answers and thus deep networks should perform at least as better as a smaller network.

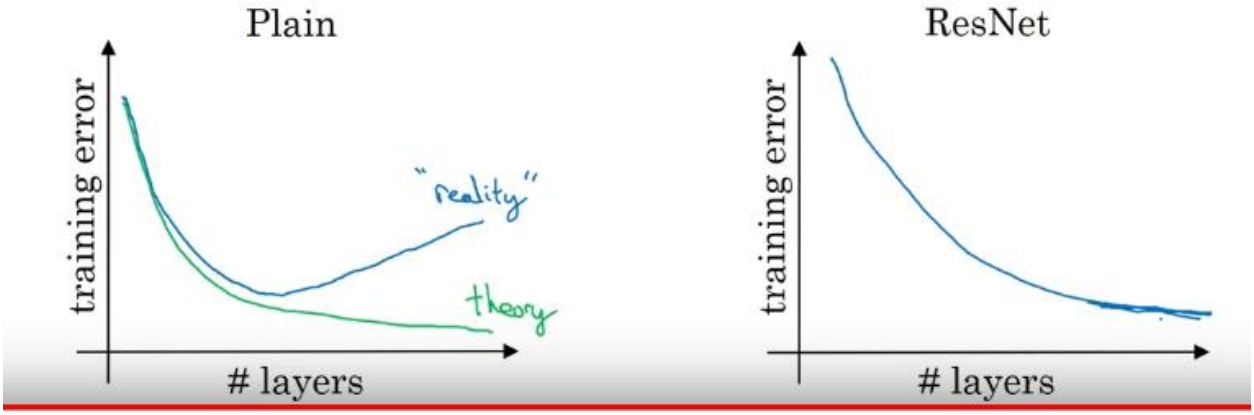
However this is not possible is real life, because in real life, the network has to train all the weights simultaneity, so learning the identity becomes really tough.

Now suppose the neural network isn't able to learn identity function, then the best possible way is to directly send the input to it. In this way, even if the function it got gives zero, we got the input to convert that output equal to input. However, if it wants to learn something new, its always welcome.

In more general terms :

As we start training very-very deep learning models, the chances of facing of vanishing and exploding gradient issues become high. Resnet works on the principal of skip connections, which takes activation from one layer and feed them to another layer which may be quite deep inside the model. In Reset, the output from the previous layer, called residual, is added to the output of the current layer. The above picture visualizes this operation

In theory, we say that the deeper the model becomes, the lower training error it will have. But in practical scenarios, it's a bit different. When we initially start deepening the model, training error reduces, however upon further deepening, it starts rising again. It can be represented in the graph below:



Resnet on the other-hand, allows the model to get even deeper. Due to its skip connection technique, it makes sure that the output of a very deep layer, if not useful, doesn't become harmful to the model and this is achieved by adding the activation of a back layer to the linear product of the input and corresponding weights of the deep layer.

Though unlike VGG-16, we can make our Resnet model really deep, it has one major drawback ie, its deeper layers take quite long time to train and in order to sort it , we need to drop few random layers.

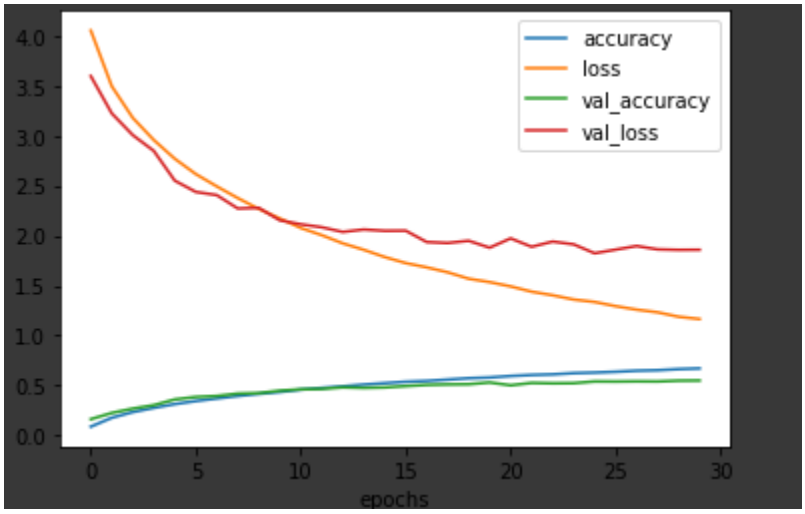
Training the model:

As we training the model in VGG-16, we are following the same pattern, however this time, we are reducing the size of kernels and this seems to be fitting the model to given data set quite well. Please check the code to find the reduced kernel sizes. This time, we are not removing any layers.

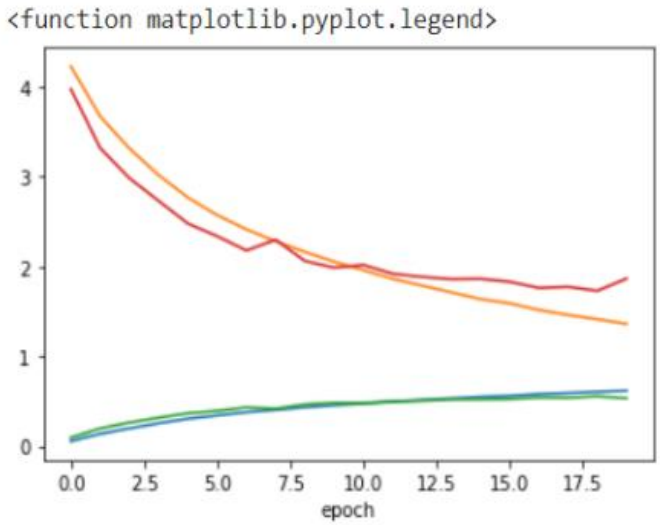
Non-regularized SGD and ADAM optimizer performance on RESNET-18 model

The accuracy of Res-Net model without any regularization is around 57% when the optimizer is SGD and 52.2% when the optimizer is Adam. This may not be as good as that of VGG-16, however the computational cost and time gone certainly down.

Below graph represent the performance of ADAM based model without any regularization:



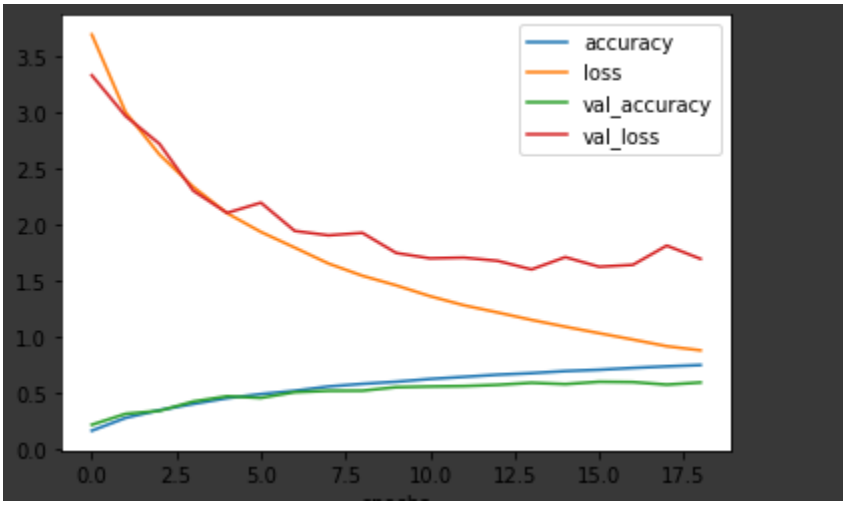
Below graph represent the performance of SGD based model without any regularization:



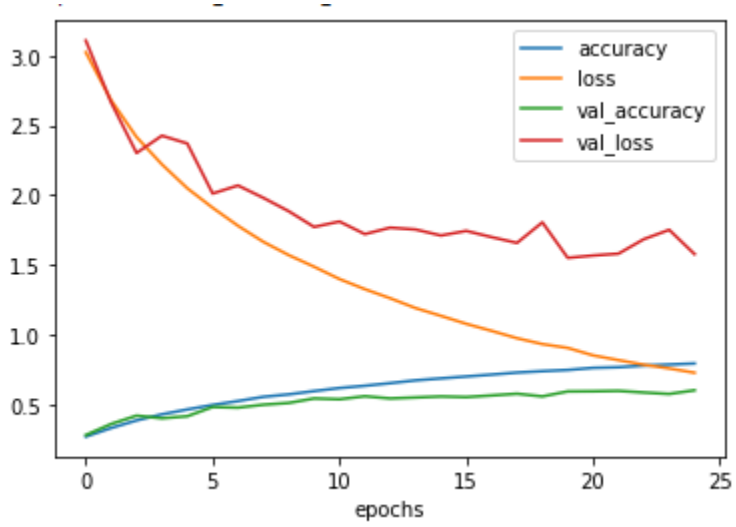
Regularized SGD and ADAM optimizer performance on RESNET-18 model using Batch-Norm

Upon regularization (using Batch norm), the accuracy of both models increases slightly, however, again the gap between the training and test accuracies seems to be coming down, so the model is regularizing well. However, I still feel that the better can perform better. The hyper-parameters of regularization and optimization method have to be studied further and more experimentations needs to be carried on.

Following graph represent the performance of SGD RESNET model using batch-norm:



Following graph represent the performance of ADAM RESNET model using batch-norm:

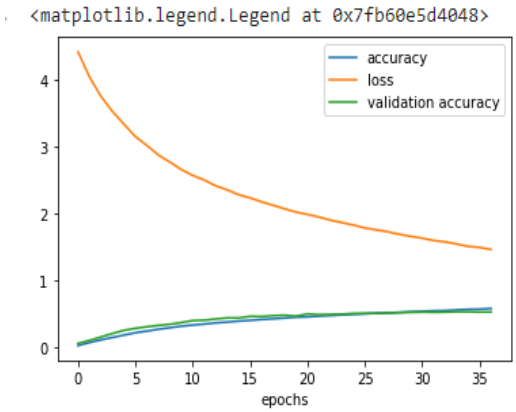


Regularized SGD and ADAM optimizer performance on RESNET-18 model using Dropout

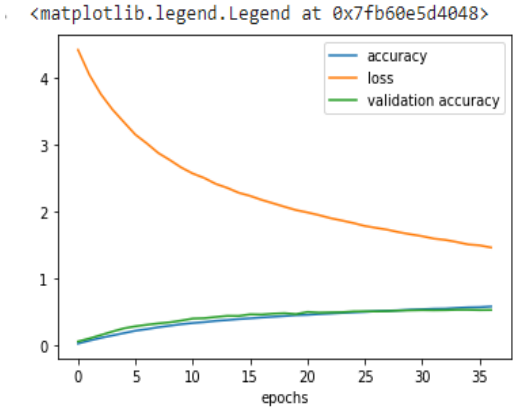
Dropout was not very successful on my model. Even after tuning the dropout percentage and other hyperparameters of optimizers, I didn't see any significant increase in the performance of my model. Though again, the training accuracy and test accuracy didn't vary too much, the accuracy of Adam's optimized model was around 57% whereas that of SGD's was 47%.

Thus, as per my analysis, Batch-norm seems to be working better on this model compared to Dropout.

Below is the graphical presentation of performance of model with Dropout regularization and SGD optimizer:



Please check the graphical presentation of performance of model with Dropout regularization and Adam optimizer:



3. **Inception V-2:**

Inception model is a game changing model in the filed of deep learning. Prior to it, all other models just tried to make the models deeper and deeper by stacking in convolutional layers.

However, Inception layer is kinda of tough to understand. It uses the concept of factorization to bring down the size of convolutional layer and I feel its very sensitive to any change. However, its really fast, got less parameters and thus is far better than Resnet or VGG in term of computational cost or time. In fact, the Inception architecture of GoogLeNet was also designed to perform well even under strict constraints on memory and computational budget. For example, GoogleNet employed only 5 million parameters, which represented a 12 times reduction with respect to its predecessor AlexNet, which used 60 million parameters. Furthermore, VGGNet employed about 3x more parameters than AlexNet.

Architecture of Inception V2:

Its hard to show the complete architecture of this model in one diagram, so I will be showing them in parts.

The main architecture of this model is:

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Here, figure 5 is the following figure:

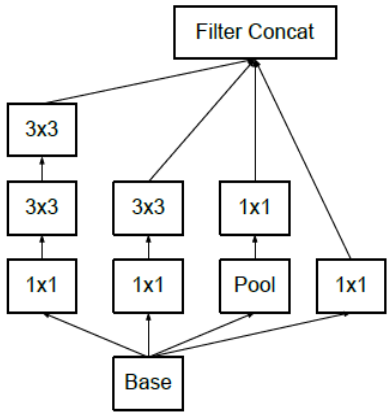


Figure 6 is represented by the following diagram:

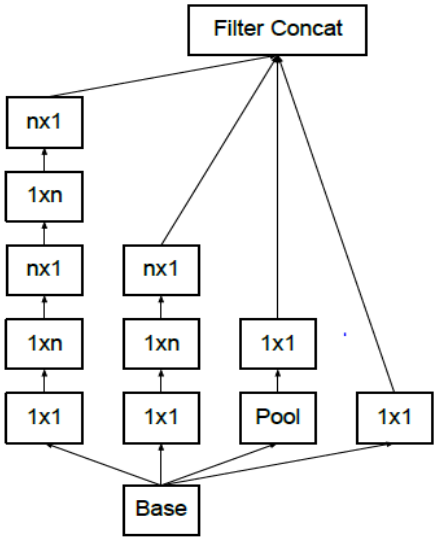
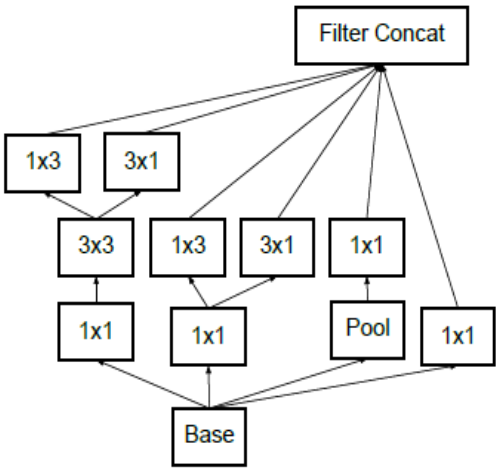


Figure 7 is represented by the following image:



Working of Inception model:

Dimensionality reduction and factorization are main principals behind working of Inception model. Convolutions with larger filters are more prone to expensive computation. For example, a 5 X5 convolutions with n filters over a grid with m filters is 25/9 = 2.78 times more computationally expensive than 3 X 3 convolutions with the same number of filters. Of course, a 5*5 filter can capture dependencies between signals between activations of units further away in the earlier layers, so a reduction of the geometric size of the filters comes at a large cost of expressiveness. However, it can be shown that convolutions with higher spatial filters could be replaced by a multi-layer network with less parameters with the same input size and output depth. And this is the reason why we have several small valued convolutions in this model. In other words, factorize 5x5 convolution to two 3x3 convolution operations to improve computational speed. Although this may seem counterintuitive, a 5x5 convolution is 2.78 times more expensive than a 3x3 convolution. So, stacking two 3x3 convolutions in-fact leads to a boost in performance.

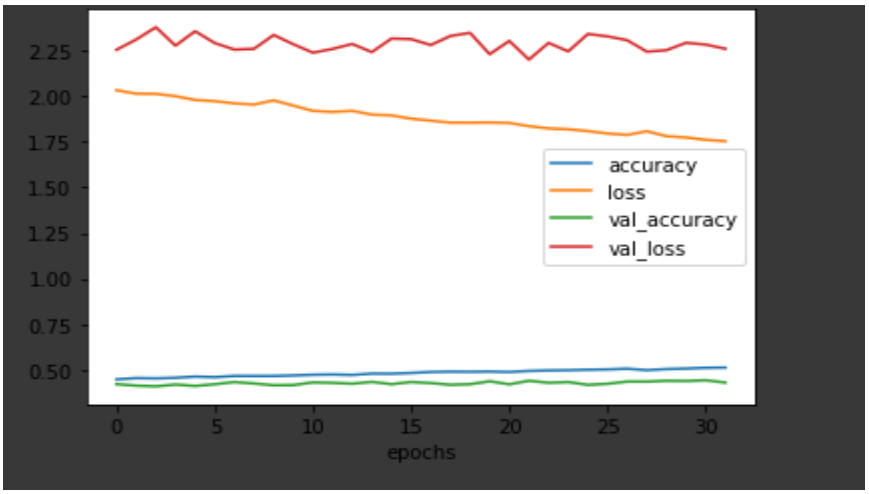
Moreover, convolutions of filter size nxn are further factorized to a combination of 1xn and nx1 convolutions.

Finally, the filter banks in the module were expanded (made wider instead of deeper) to remove the representational bottleneck.

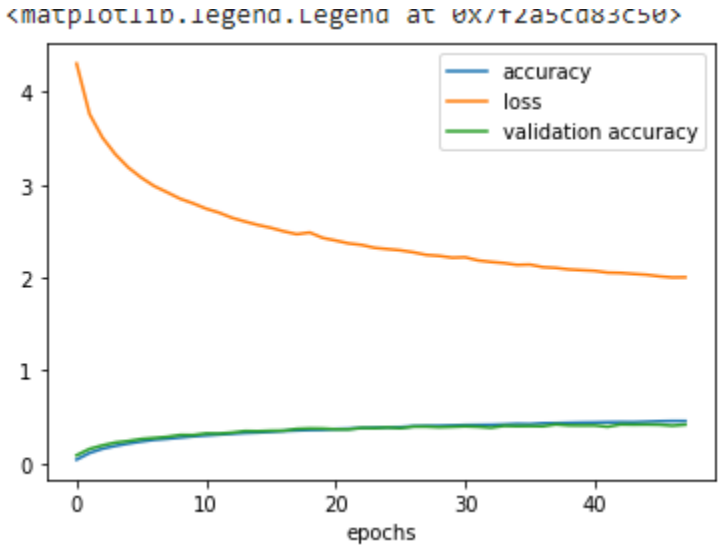
Non-regularized SGD and ADAM optimizer performance on INCEPTION-V2 model:

Even for this model, I followed the previous mechanism. As our data set is relatively small and chances of over fitting are high, I changed the number of kernels. However, upon careful analysis, I feel that the model got underfit. The accuracy with other any normalization for SGD optimizer is around 44% whereas that of Adam is around 44.16

Please check the below graphical presentation of model's performance when there is no regularization and optimizer is Adam.



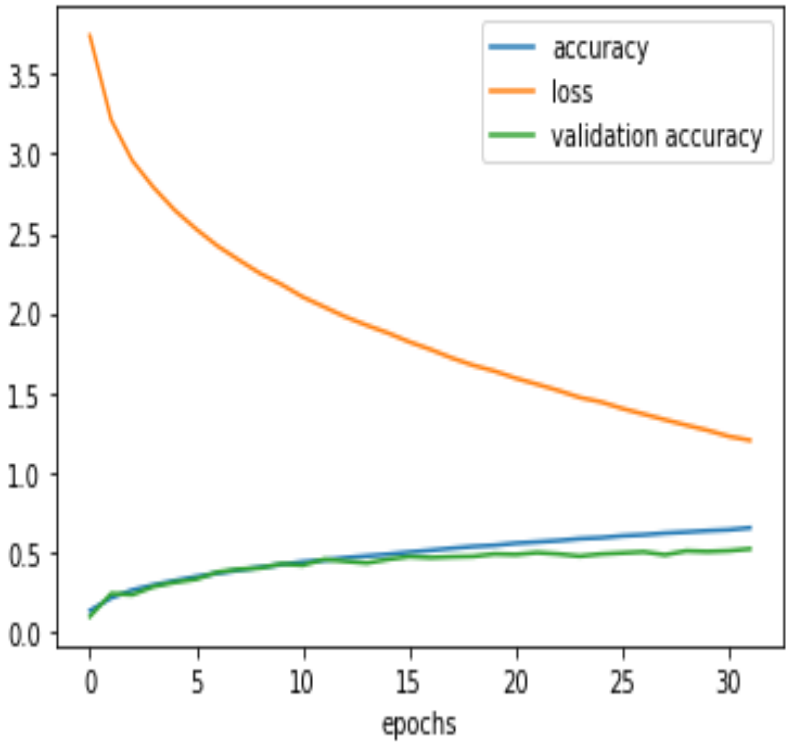
Please check the below graphical presentation of model's performance when there is no regularization and optimizer is SGD.



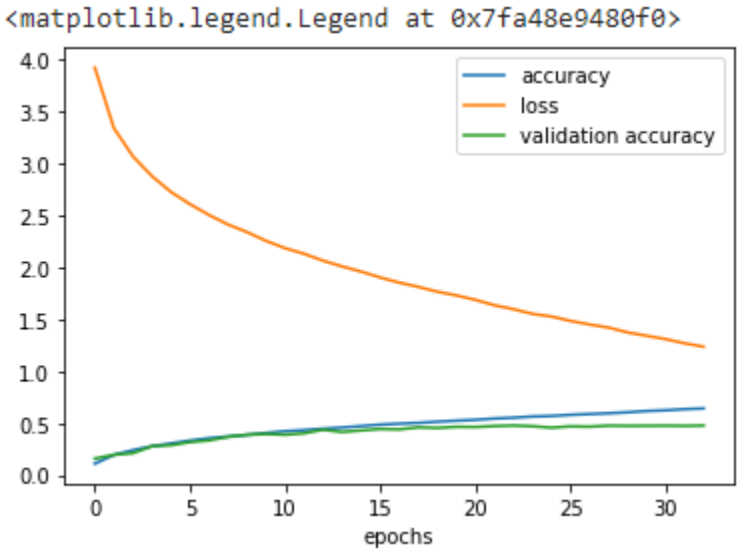
Regularized SGD and ADAM optimizer performance on INCEPTION-V2 model using Batch-Norm

Upon applying the batch regularization, the performance of model goes slightly up, however, even the training accuracy is not too high, so we can term it as under fitting model. Thus, the model seems to have higher bias and its not able to even fit on training data. The accuracy of model with SGD and Batch norm is around 48% where as that with Adam and Batch norm is about 53%.

Please check the below graphical presentation of model's performance for batch-norm regularization and Adam optimizer



Please check the below graphical presentation of model's performance for batch-norm regularization and SGD optimizer



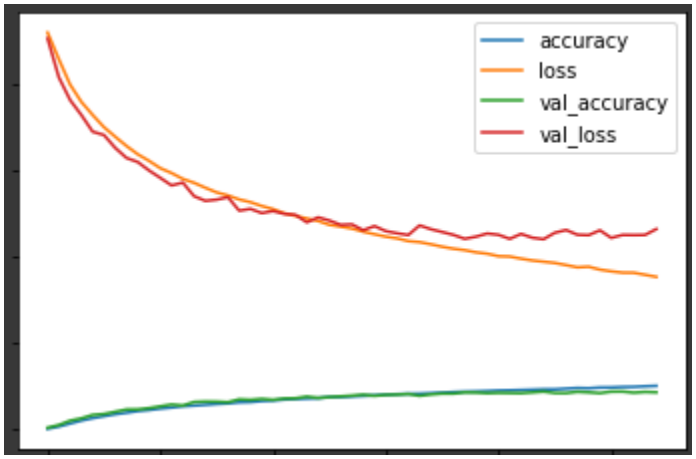
Regularized SGD and ADAM optimizer performance on INCEPTION-V2 model using Drop Down

The value of drop-down percentage is kept quite low as the model has high bias. Had the variance been high, we would have gone for higher percentage values.

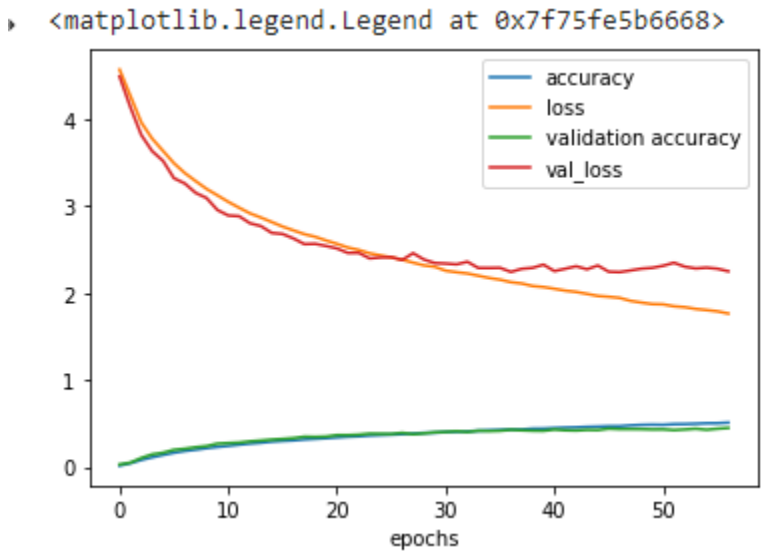
For SGD, the value for drop down percentage is kept as 0.01, whereas for Adam its kept at 0.05. This was because, any value like even 0.1 was making the model to under fit.

However, this seems an interesting point and I will continue to work on this model. I will try to increase the model length and width and check its effect on performance.

Please check the below graph for performance evaluation of model with Adam optimizer and Drop out regularization:



Please check the below graph for performance evaluation of model with SGD optimizer and Drop out regularization:



Please note that for most of the cases, the learning rate for Adam optimizer is kept at 0.0001 and for SGD, learning rate is 0.01 with momentum around 0.9.

Please check the performance matrices for every model:

	ARCH	VGG16			RESNET 18			INCEPTION V2		
	SCORE									
OPTIMIZER	SETTING	PRECISION	RECALL	ACCURACY	PRECISION	RECALL	ACCURACY	PRECISION	RECALL	ACCURACY
	WITH BATCHNORM	54.83	54.4	54.83	59.54	57.06	57.06	54	52.17	52.17
	WITH DROPOUT	50.8	49.5	49.5	58.23	56.96	56.96	46.48	44.5	44.5
SGD	NO REGULARIZATION	64.68	62.97	62.97	59.45	57.03	57.03	45.8	43.99	43.99
	WITH BATCHNORM	70	69.4	69.4	61.3	58.3	58.3	54	52.17	52.17
	WITH DROPOUT	62.28	59.58	59.28	47.8	46.78	46.78	44.2	42.08	42.08
ADAM	NO REGULARIZATION	62.88	62.41	62.41	53.7	52.2	52.2	45.15	44.16	44.16

References :

1. Rethinking the Inception Architecture for Computer Vision
2. https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202
3. **https://www.youtube.com/watch?v=0tBPSxioIZE**
4. **https://www.youtube.com/results?search_query=resnet+architecture**
5. **https://towardsdatascience.com/implementing-a-resnet-model-from-scratch-971be7193718**