CS 7150: Deep Learning — Summer-Full 2020 — Paul Hand

HW 1
Due: Wednesday May 27, 2020 at 11:59 PM Eastern time via Gradescope

Name: Saurabh Vaidya
Collaborators: Sumeet Gajjar

You may consult any and all resources. Note that these questions are somewhat vague by design. Part of your task is to make reasonable decisions in interpreting the questions. Your responses should convey understanding, be written with an appropriate amount of precision, and be succinct. Where possible, you should make precise statements. For questions that require coding, you may either type your results with figures into this tex file, or you may append a pdf of output of a Jupyter notebook that is organized similarly. You may use PyTorch, TensorFlow, or any other packages you like. You may use code available on the internet as a starting point.

Question 1. *Is a 1×1 convolution operation the same as scaling the input by a single scalar constant? Explain. If the answer is sometimes yes, then make sure to explain when it is and when it isn't.*

Response: Its sometimes yes.

A convolution operation for a 1x1xC filter, where C is the number of channels in the input layer, is an inner product. The output for a unit in output layer is sum of product of weight and unit's activation for a given unit position in all feature maps. A scalar multiplication would be just the product of a weight and unit's activation for a given unit in a single feature map. It will not be a linear combination of weight and unit's activation across all channels. It is shown in the figure 1, the output o_{11}, o_{12} are different for convolution and scalar multiplication for C channels in the input.

However, when the number of input channels is 1, then the output of both a convolution operation and scalar multiplication are the same. For example in case of fig 1, when C=1 in layer n the output of convolution will be
$o_{11} = w_{11} * u_{11}$
Which will be the same as a scalar multiplication as seen in figure 1.

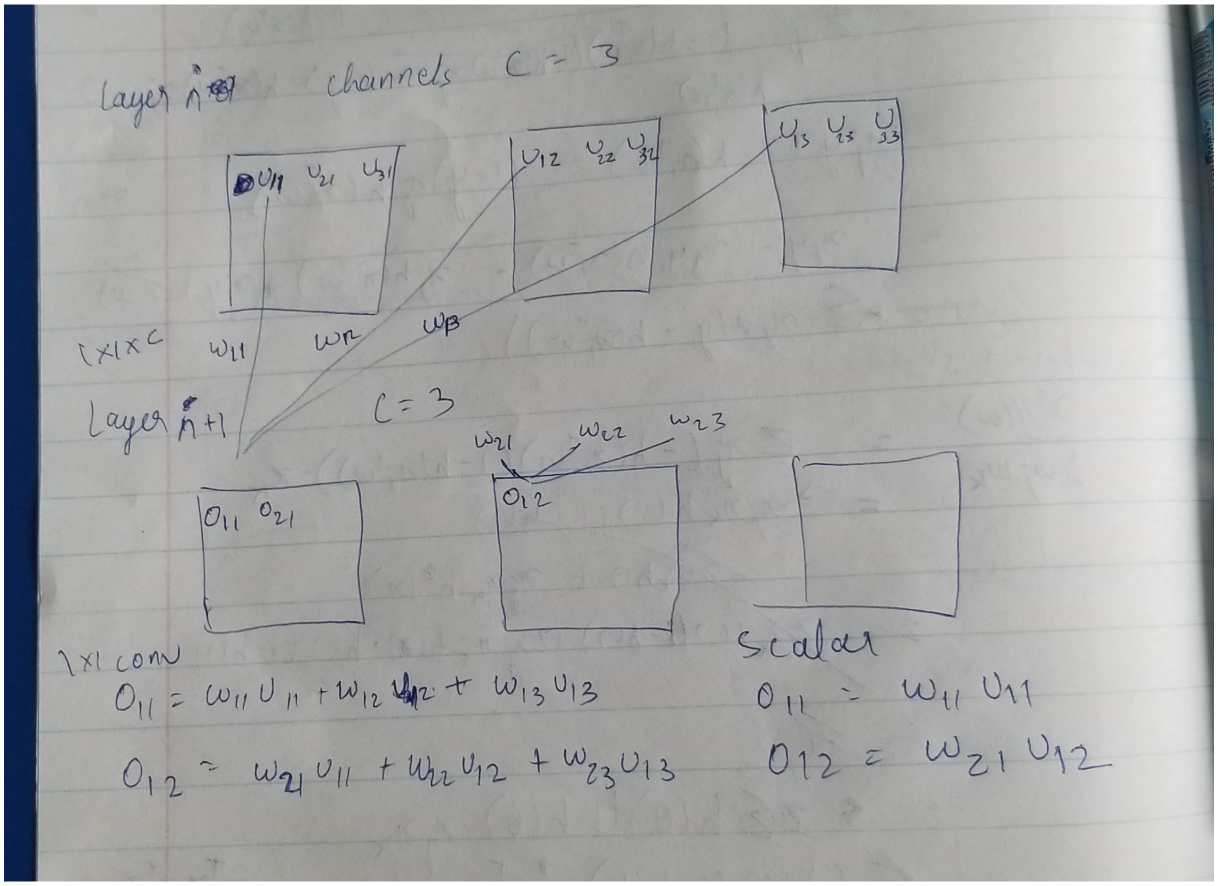


Figure 1: 1x1 convolution and scalar multiplication for input with C channels

Question 2. *Show that optimization problem resulting from logistic regression is convex in the model parameters.*

Response: For logistic regression, the maximizing log likelihood is equivalent to minimizing cross entropy loss. Thus the log likelihood can be written as

$$ll(\mathrm{w}) = -\sum_{i=1}^{n}\left[y_i \log\left(\frac{1}{1+\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}\right) + (1-y_i)\log\left(1-\frac{1}{1+\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}\right)\right] \tag{1}$$

$$\tag{2}$$

Rearranging the above equation to make differentiation easy

$$\begin{aligned}
ll(\mathrm{w}) &= -\sum_{i=1}^{n}\left[(y_i-1)\mathrm{w}^{\mathrm{T}}\mathrm{x_i} + \log\left(\frac{1}{1+\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}\right)\right] && (3)\\
\frac{\partial ll(\mathrm{w})}{\partial w_j} &= -\sum_{i=1}^{n}\left[(y_i-1)x_{ij} + \frac{\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}{1+\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}.(-x_{ij})\right] && (4)\\
&= -\sum_{i=1}^{n} x_{ij}\left(y_i - 1 + \frac{\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}{1+\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}\right) && (5)\\
&= -\sum_{i=1}^{n} x_{ij}\left(y_i - \frac{1}{1+\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}\right) && (6)\\
&= -f_j^T(\mathrm{y}-\mathrm{p}) && (7)\\
& && (8)
\end{aligned}$$

Where f_j^T is the j-th column (feature) of data matrix **X**, **y** is an n-dimensional column vector of class labels and **p** is an n-dimensional column vector of (estimated) posterior probabilities $p_i = P(Yi = 1|xi, w)$, for i = 1, ..., n. Considering partial derivatives for every component of w, we have

$$\begin{aligned}
\nabla ll(\mathrm{w}) &= -X^T(\mathrm{y}-\mathrm{p}) && (9)\\
\frac{\partial^2 ll(\mathrm{w})}{\partial w_j \partial w_k} &= -\sum_{i=1}^{n} x_{ij}.\frac{\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}{1+\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}.(-x_{ik}) && (10)\\
&= \sum_{i=1}^{n} x_{ij}.\frac{1}{1+\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}.\left(1-\frac{1}{1+\exp(-\mathrm{w}^{\mathrm{T}}\mathrm{x_i})}\right)x_{ik} && (11)\\
&= f_j^T.P(I-P).f_k && (12)\\
& && (13)
\end{aligned}$$

where **P** is an nxn diagonal matrix with $P_{ii} = p_i = P(Y_i = 1|x_i, w)$ and **I** is an nxn identity matrix. The Hessian matrix $H_{ll(\mathrm{w})}$ can now be calculated as

$$\begin{aligned}
H_{ll(\mathrm{w})} &= X^T P(I-P)X && (14)\\
\forall z : z^T H z &= z^T[X^T P(I-P)X]z && (15)\\
&= (Xz)^T P(I-P)(Xz) \geq 0 && (16)
\end{aligned}$$

Thus the Hessian is Positive semi-definite and our optimization problem is convex in model parameters.

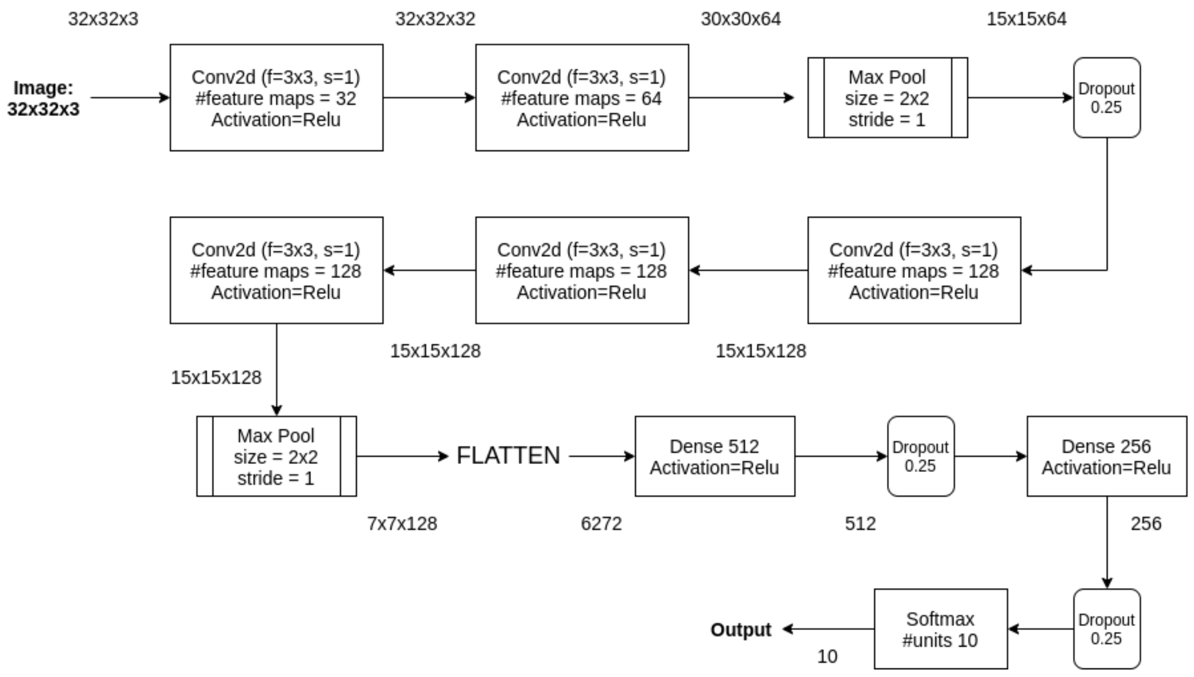

32x32x3
32x32x32
30x30x64
15x15x64
Image: 32x32x3
Conv2d (f=3x3, s=1) #feature maps = 32 Activation=Relu
Conv2d (f=3x3, s=1) #feature maps = 64 Activation=Relu
Max Pool size = 2x2 stride = 1
Dropout 0.25
Conv2d (f=3x3, s=1) #feature maps = 128 Activation=Relu
Conv2d (f=3x3, s=1) #feature maps = 128 Activation=Relu
Conv2d (f=3x3, s=1) #feature maps = 128 Activation=Relu
15x15x128
15x15x128
15x15x128
Max Pool size = 2x2 stride = 1
FLATTEN
Dense 512 Activation=Relu
Dropout 0.25
Dense 256 Activation=Relu
7x7x128
6272
512
256
Output
10
Softmax #units 10
Dropout 0.25

Figure 2: Cifar10 CNN Architecture

Question 3. *Train a convolutional neural network for classification on the CIFAR10 dataset.*

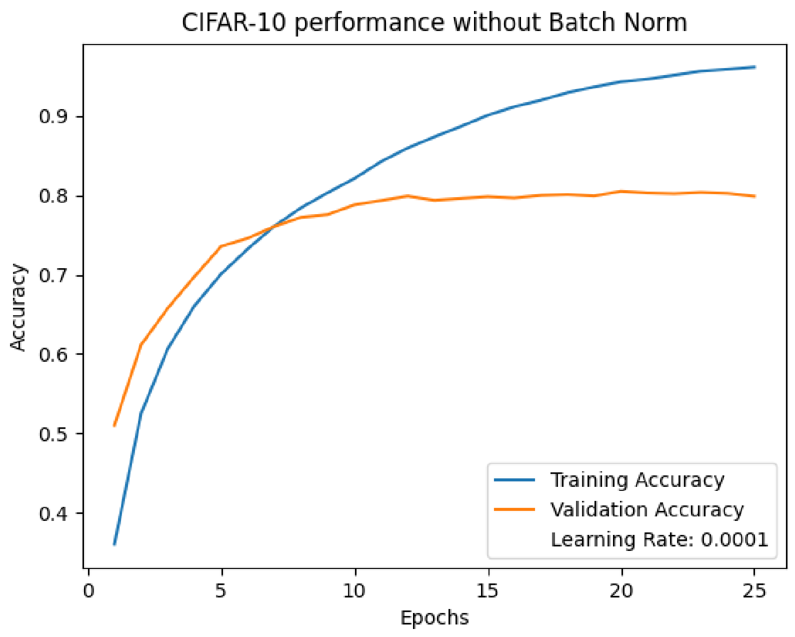
(a) Clearly convey the architecture and training procedure of your network using a figure and text.

Response: Figure 2 shows my neural network architecture.
The training procedure settings are as follows:

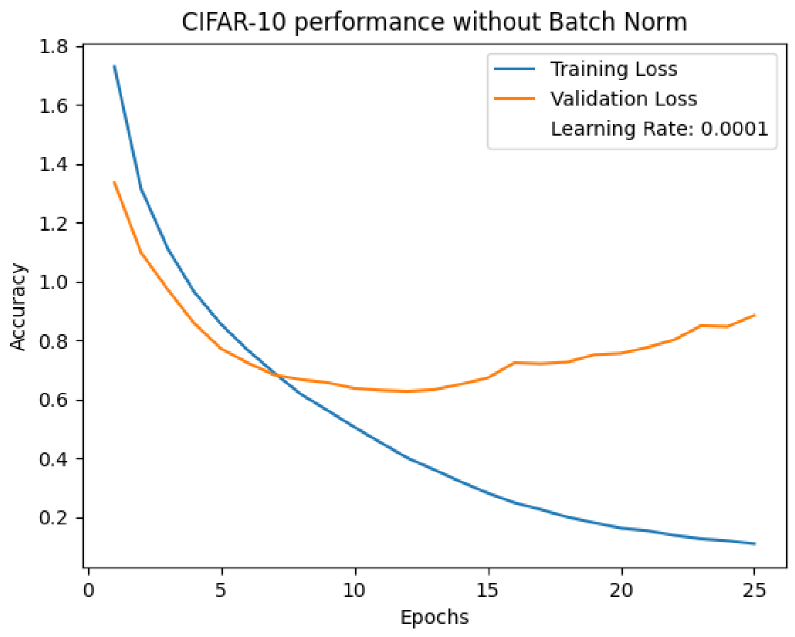
- Training is performed in batches of size 32
- The total number of epochs is 25
- Learning rate is 0.0001 with decay=1e-6
- I have used Adam optimizer and loss function is categorical cross entropy.
- Dataset split is 50000 for training and 10000 for validation. After every epoch, the validation data is used for computing validation accuracy.
- Number of classes is 10

(b) Evaluate the performance of your network.

Response: With the above training procedure settings, the training accuracy is 0.9605 and validation accuracy is 0.8012. Below is the plot for training and validation accuracy



(a) Loss



(b) Accuracy

Figure 3: Training metrics for CNN model

(c) Select three feature maps (channels) in three different layers of your network and visualize the content/structure within input image(s) that lead to large activation values of those feature maps. You may use any algorithm you like to do this. Clearly convey the algorithm you used.

Response: The algorithm I have used is called GradCAM. However it is slightly modified as would be seen next. The main idea of the algorithm is given a test image, to compute the gradient of predicted class score with feature map of layer we want to visualize. Intuitively, higher gradient corresponds to higher influence of that feature map in decision. The algorithm works as follows

- Gradients are computed for each feature maps in the given layer (whose features we want to visualize) with respect to predicted class score.
- For a given layer, we choose 3 random features. We compute gradients for only those 3 but instead of single test image, we compute them across 100 test images. The gradients thus obtained are then global average pooled to obtain the neuron important weights corresponding to the target class
- We then select gradients for 2 images out of those 100 that have the highest gradient(sorted in descending order).
- We then multiply each of the chosen feature's convolution output with their corresponding gradient to get the heatmap.
- The modification from original GradCAM is that it computes product of feature activation and pooled gradients for all features in the layer and then computes the mean of all activation maps along the channel. Our modification is to compute gradients only for our selected feature. Then choose 2 gradients with highest values and use the product of each gradient with it's corresponding feature activation. Thus for a given feature we have 2 heatmaps corresponding to two high gradient images.
- We then overlay our heatmap onto it's original image to highlight the areas in input image that were discriminatory.

The following are the visualizations from layer 1 in fig 4 and 5: Feature no 3,8,26.

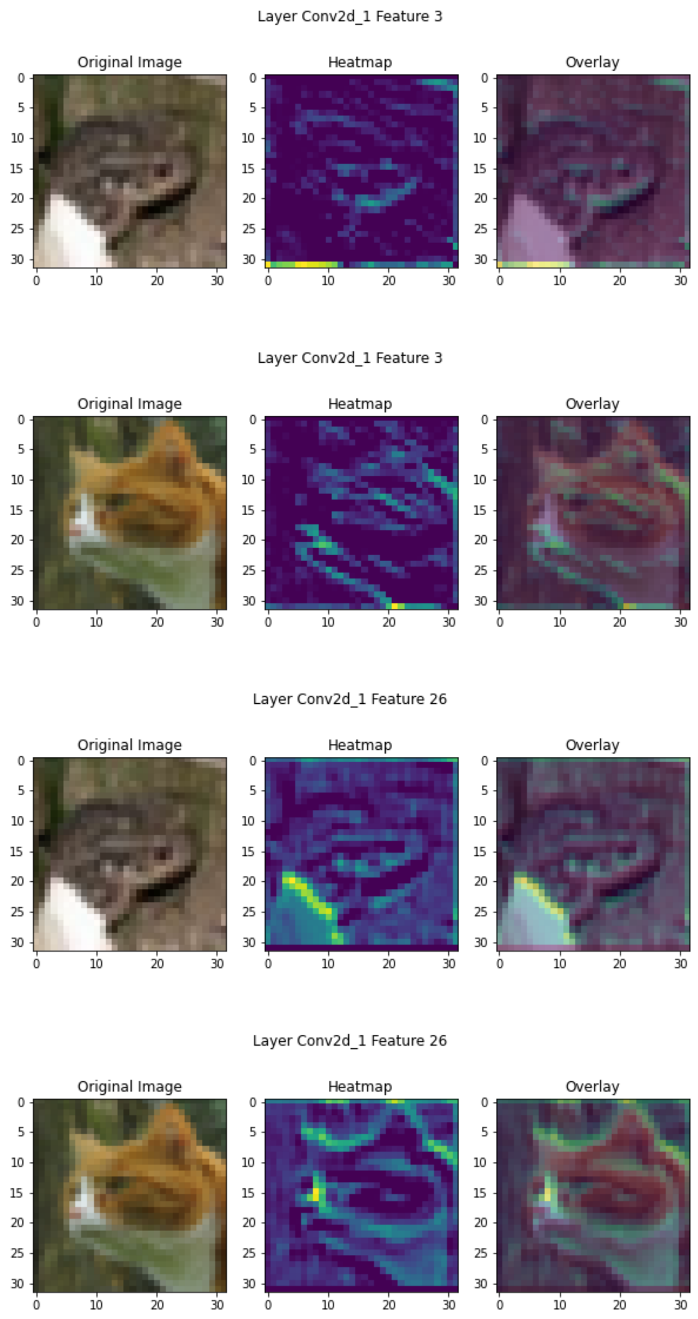
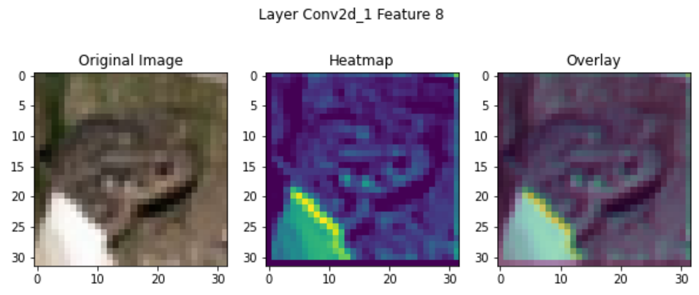


Figure 4: Layer 1 Features 3,26



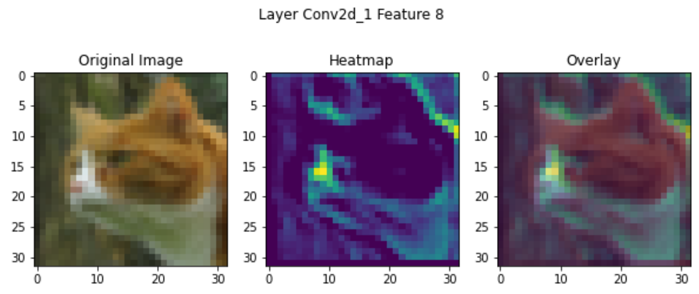


Figure 5: Layer 1 Feature 8

The following are the visualizations from layer 2 in fig 6: and 7 Feature no 3,37,49.

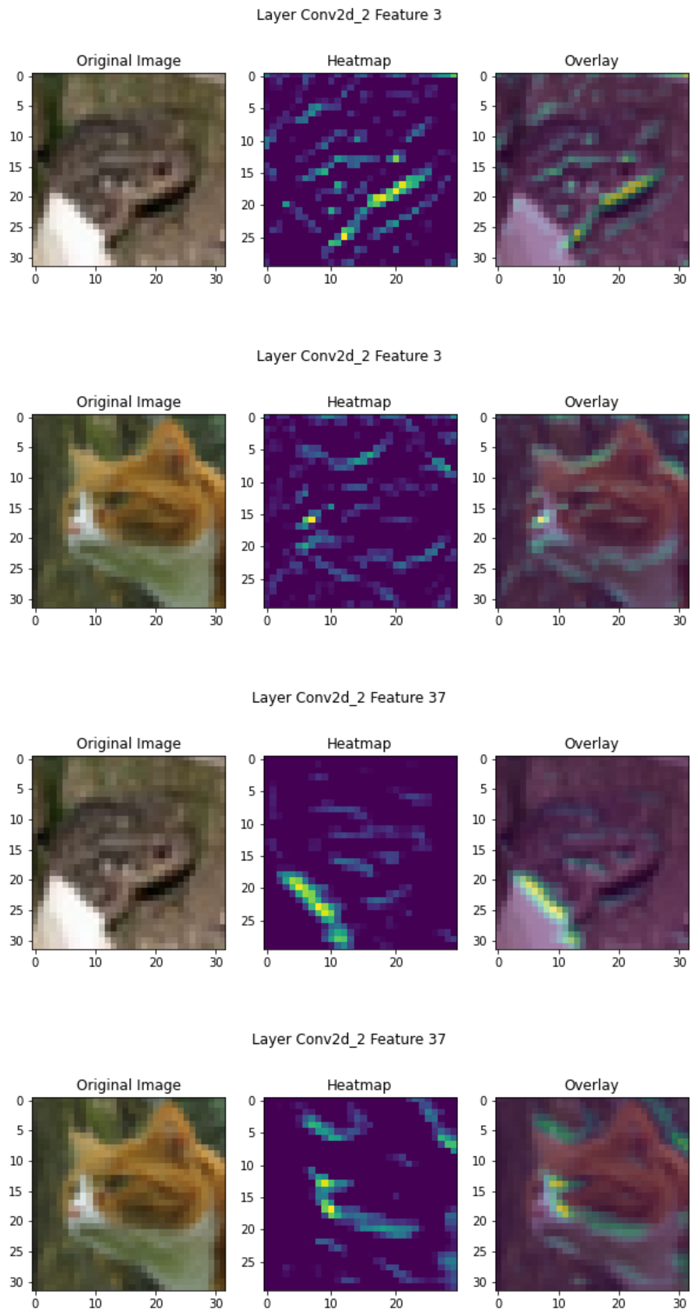


Figure 6: Layer 2 Features 3,37

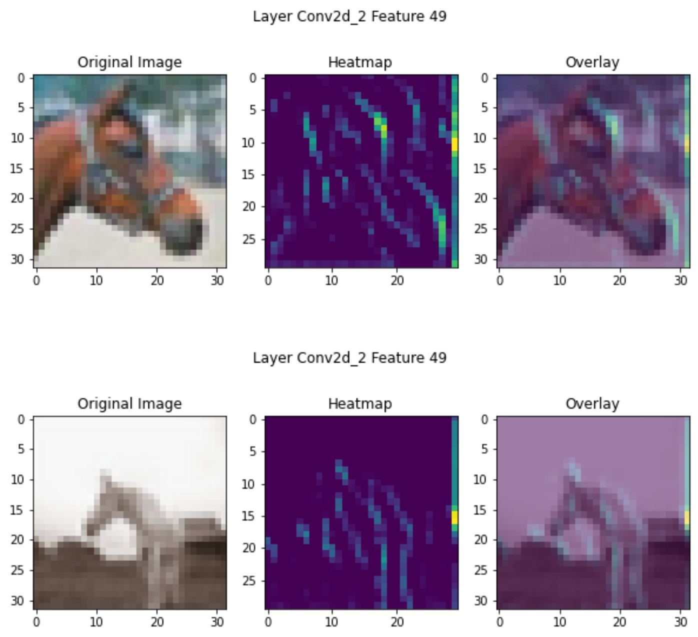


Figure 7: Layer 2 Feature 49

The following are the visualizations from layer 4 in fig 8 and 9: Feature no 25,51,55

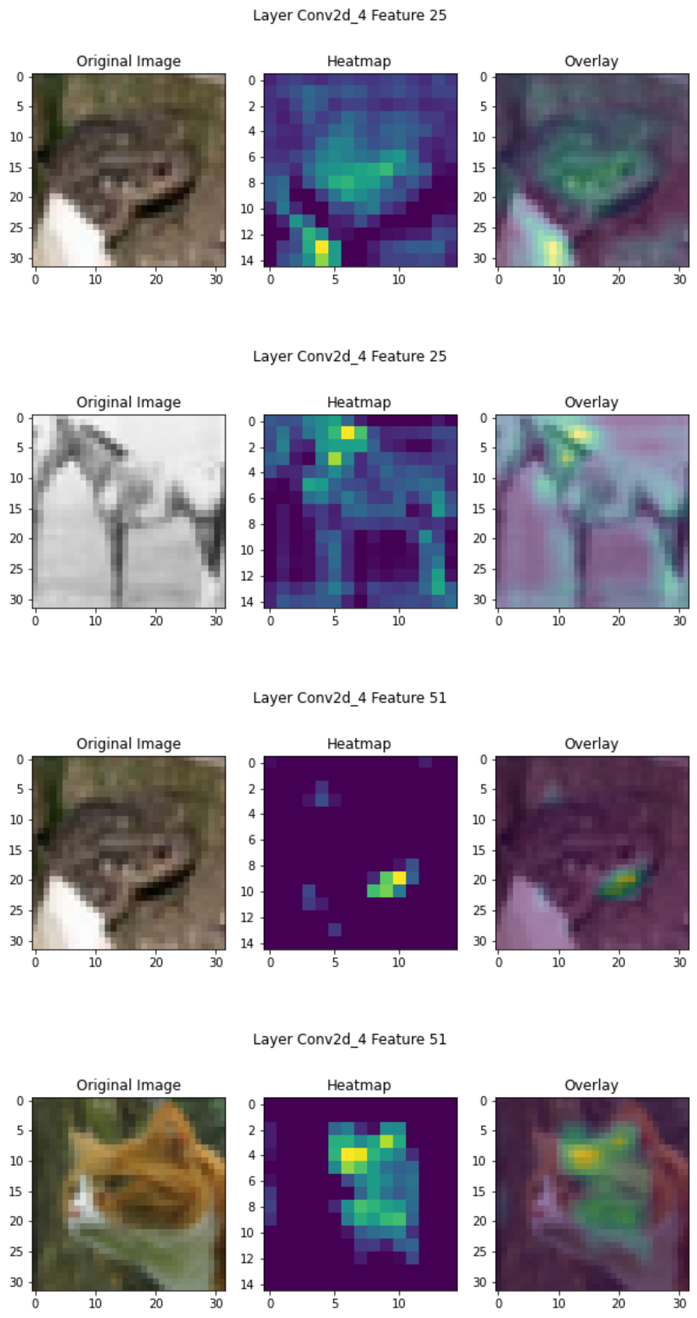


Figure 8: Layer 4 Features 25,51

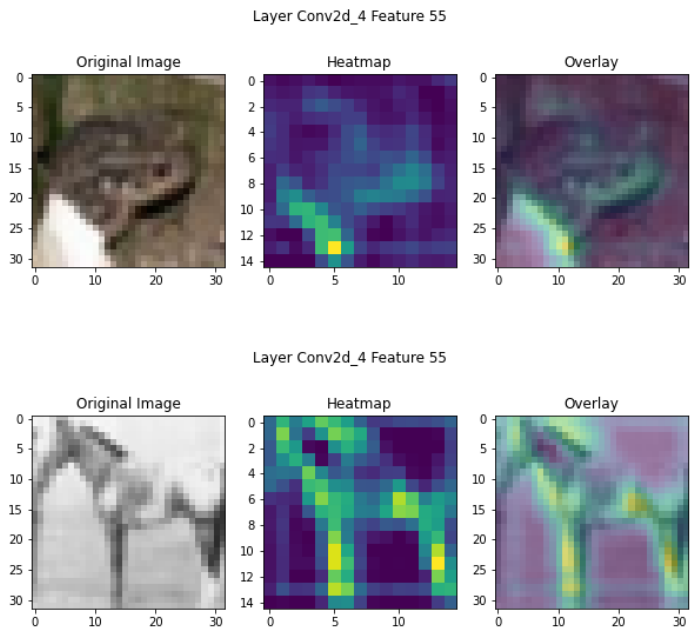


Figure 9: Layer 4 Feature 55

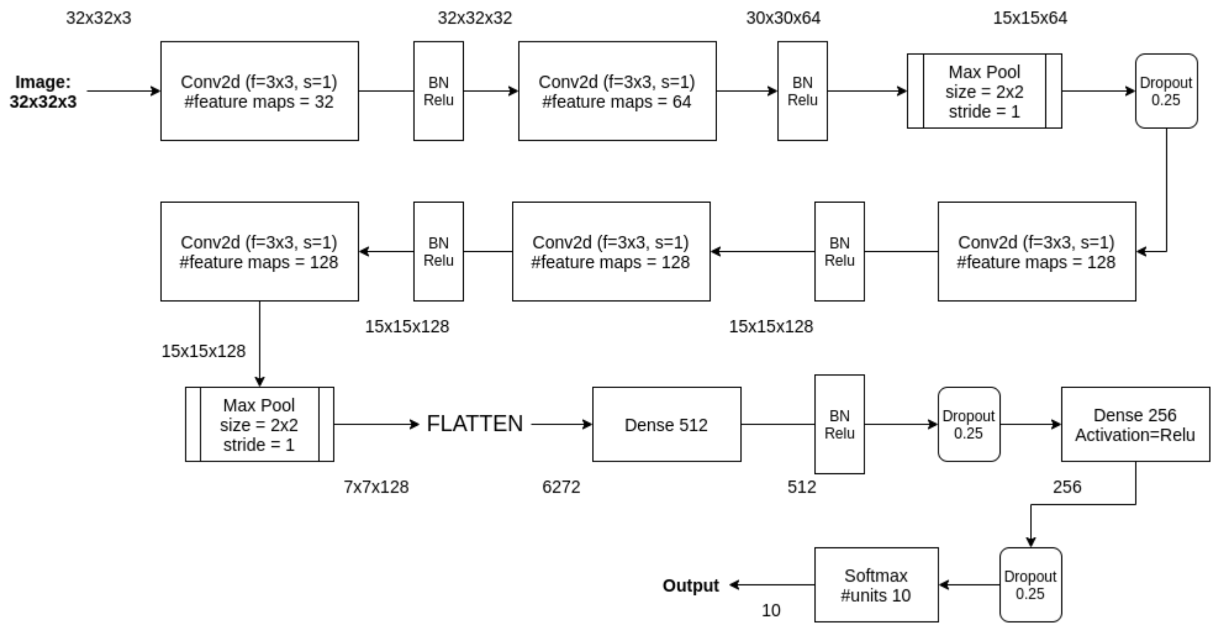

32x32x3
32x32x32
30x30x64
15x15x64
Image: 32x32x3
Conv2d (f=3x3, s=1) #feature maps = 32
BN Relu
Conv2d (f=3x3, s=1) #feature maps = 64
BN Relu
Max Pool size = 2x2 stride = 1
Dropout 0.25
Conv2d (f=3x3, s=1) #feature maps = 128
BN Relu
Conv2d (f=3x3, s=1) #feature maps = 128
BN Relu
Conv2d (f=3x3, s=1) #feature maps = 128
15x15x128
15x15x128
15x15x128
Max Pool size = 2x2 stride = 1
FLATTEN
Dense 512
BN Relu
Dropout 0.25
Dense 256 Activation=Relu
7x7x128
6272
512
256
Output
10
Softmax #units 10
Dropout 0.25

Figure 10: Batch Norm CNN architecture

Question 4. *Perform experiments that show that BatchNorm can (a) accelerate convergence of gradient based neural network training algorithms, (b) can permit larger learning rates, and (c) can lead to better neural network performance.*

Clearly explain your experimental setup, results, and interpretation.

My BatchNorm network architecture is as shown in fig 10 , where a batch normalization is applied to output of every convolution layer before relu activation.

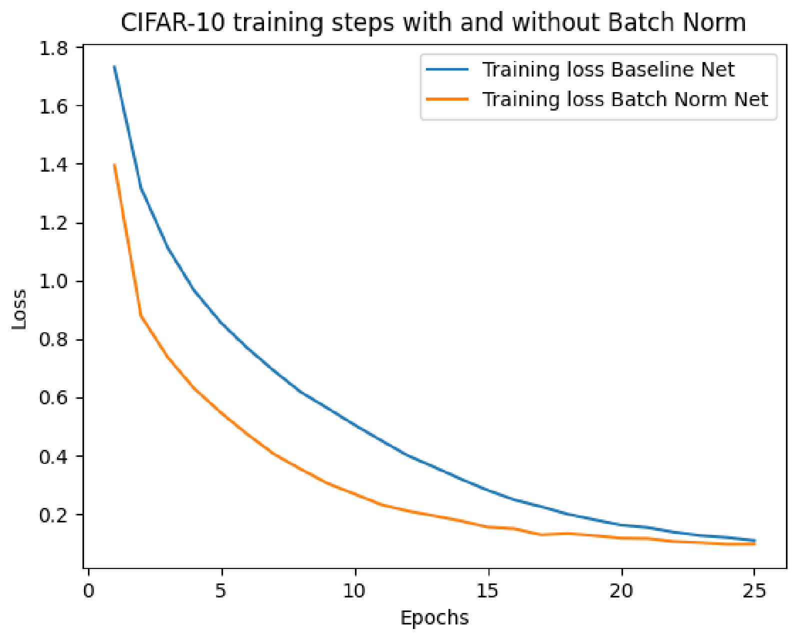
The standard network has learning rate of 0.0001. I tweak BN learning rates and present the results of my BatchNorm network with different settings. BatchNorm network achieves better accuracy than standard network(without BN) which achieved 0.8012 accuracy.

Table 1: Accuracy of different LR rates

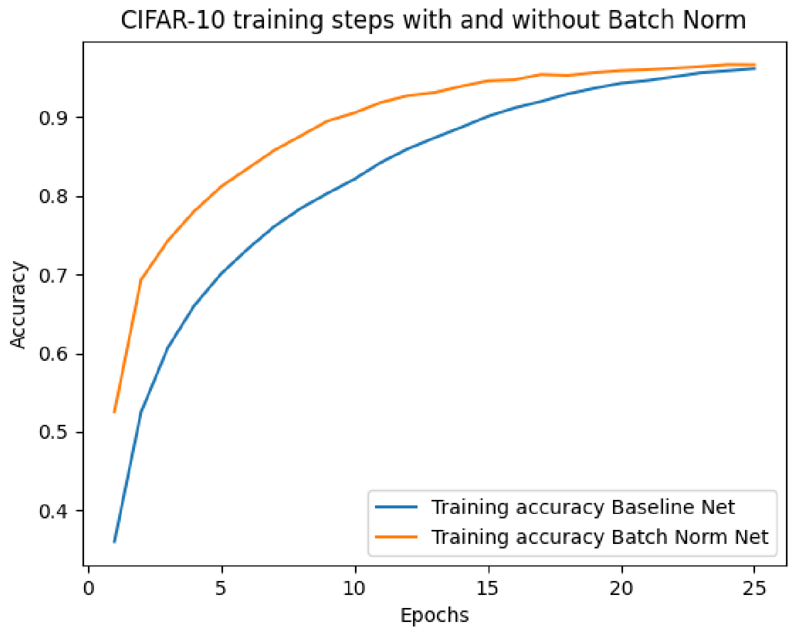
Learning Rate	**Accuracy**
BNx5 0.0005	0.8365
BNx10 0.001	0.8519
BNx30 0.003	0.8182

(a) **Response:**As seen in the figure 11, BatchNorm network has lower loss after every epoch compared to standard network. The BatchNorm network also converges around epoch 15 which is 10 epochs earlier than standard network.

The BatchNorm network also achieved the same accuracy 0.8012 of standard network 5 epochs earlier



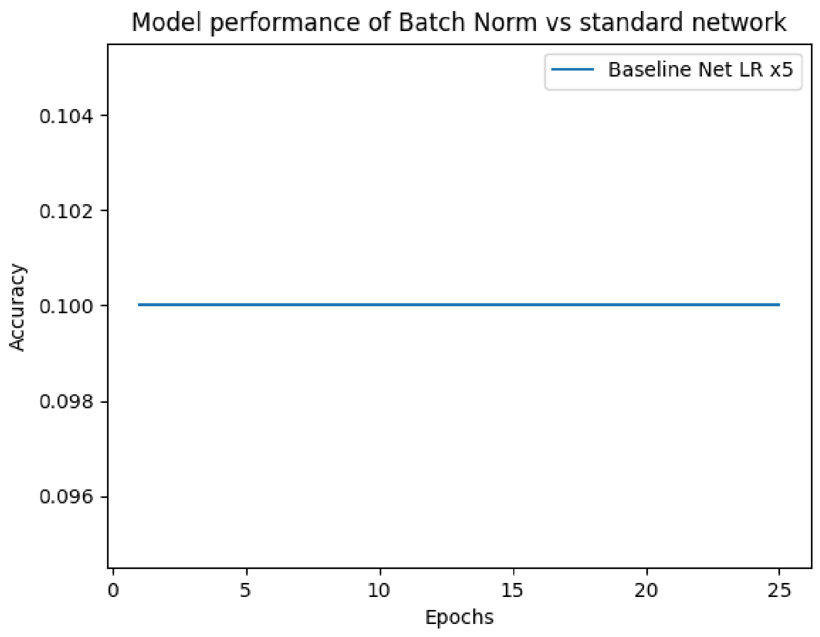
(a) Training loss



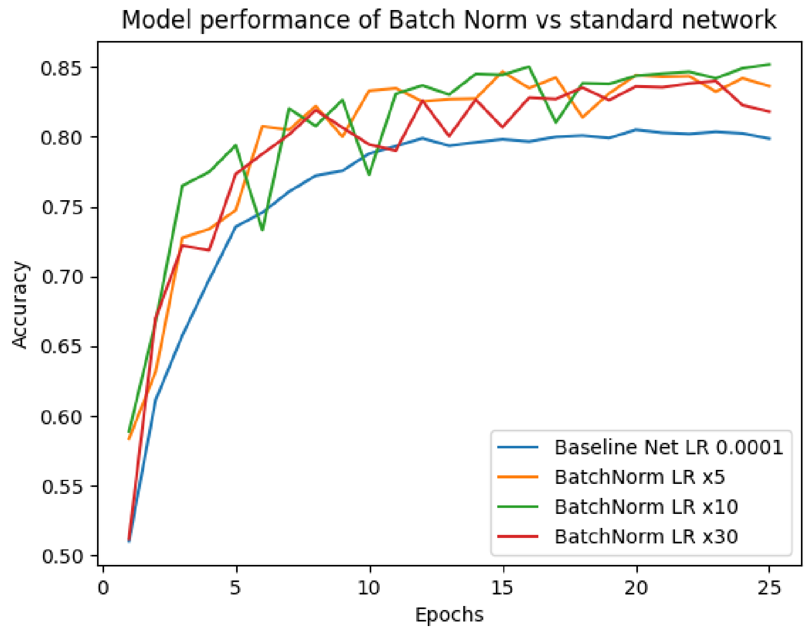
(b) Training accuracy

Figure 11: Training metrics for network with and without BatchNorm

(b) **Response:**I trained my batch normalization network with x5,x10 and x30 the learning rate and compared that with standard baseline model. For learning rate of x5 the standard model had 0.10 accuracy.



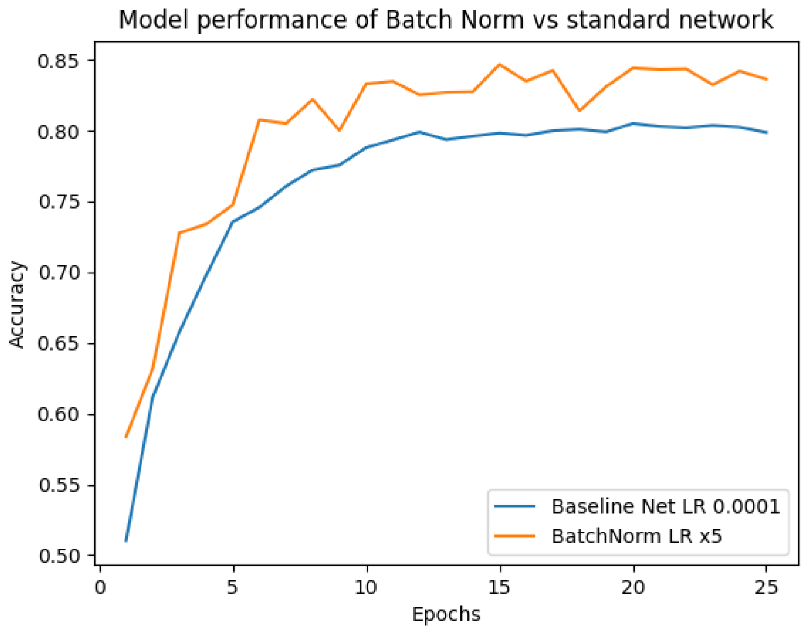
(a) Baseline model with x5 learning rate



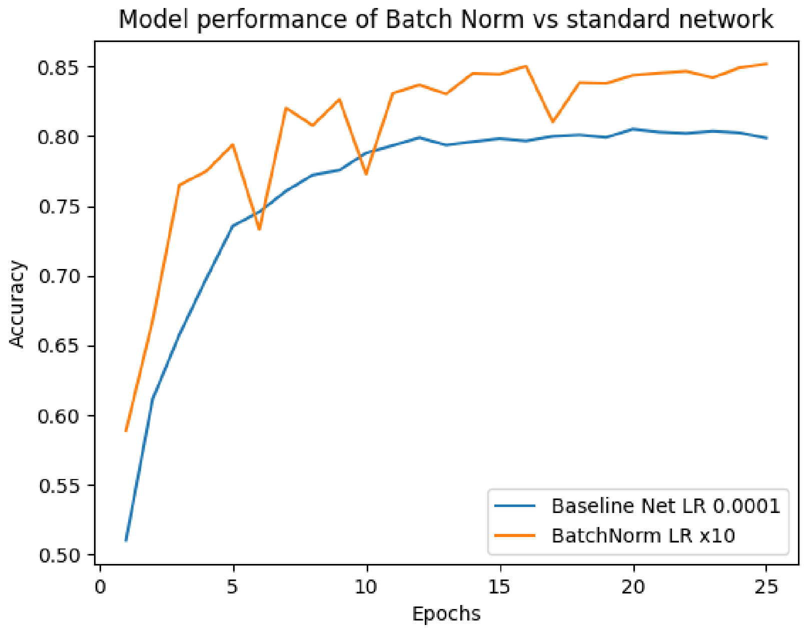
(b) BN with different learning rates vs baseline

Figure 12: Validation set accuracies for higher learning rates with BN vs baseline model

(c) **Response:**All BatchNorm networks had higher accuracies than standard model when run for same number of epochs as seen in 1. Baseline model had 0.8012 accuracy. Below are plots of validation accuracy for BNx5 vs Standard and BNx10 vs Standard



(a) BNx5 vs baseline



(b) BNx10 vs baseline

Figure 13: Validation set accuracy of BN vs baseline model