Deep Learning for Computer Vision (CS776)
Indian Institute of Technology Kanpur
Assignment 1

Name: Rahul Kumar
Email: rahulkumar21@iitk.ac.in
Roll Number: 21111069
Date: February 3, 2022

---

1 Overview

Our goal is to implement and train a multi-layer perceptron (MLP) from scratch on the CIFAR10 (1) original dataset and also with some augmentation. I tried to implement 3 layer MLP model including input layer, one hidden layer and final output layer. Before feeding into network, I also did data augmentation using 4 different image transformation methods i.e random rotation, random cutout, random crop and changing contrast of the image and standardized all the augmented images. Then evaluate the performance on both augmented and the original datasets and reported both the results in later sections.

2 Data Augmentation

There were 4 image transformation methods I used for data augmentation 1. random rotation between $[-180^0, 180^0]$ 2. random cutout of having block size ranges from 0 to 16 pixels in width and height of image 3. random crop 4. contrast and horizontal flipping with prob. 0.5. Fig. 1 shows the example after apply all this transformation.

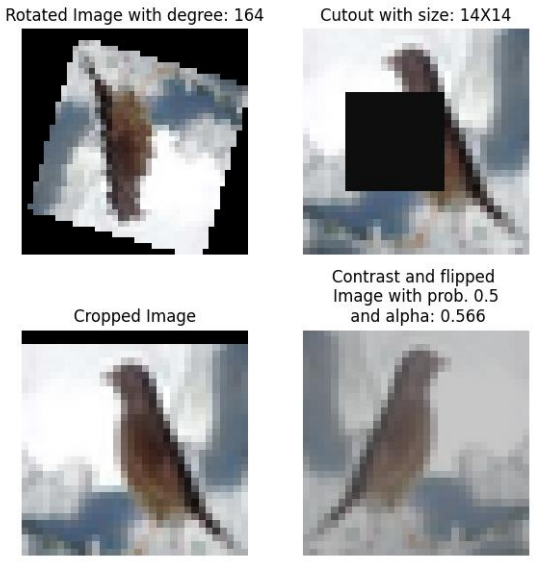


Figure 1: Data Augmentation

3 Model Architecture

An overview of the model architecture is shown in the Fig.2. This is composed of 4 main sections i.e ResNet model(already given with this assignment) and 3 other layers consists of an input layer, one hidden layer and one output layer. I first given images of size (224, 224, 3) as an input into the ResNset model and that gives me an feature vector of size 512 for each images. These features vectors are then passed as an input to the Input layer and then multiply this inputs with weight matrix to get output which then passed into hidden layer and finally I processed this hidden layer output at the final output layer to classify the image into 10 different categories of images given in CIFAR-10 (1). All 3 layers are discussed in detail in the below subsections. (2) (3)

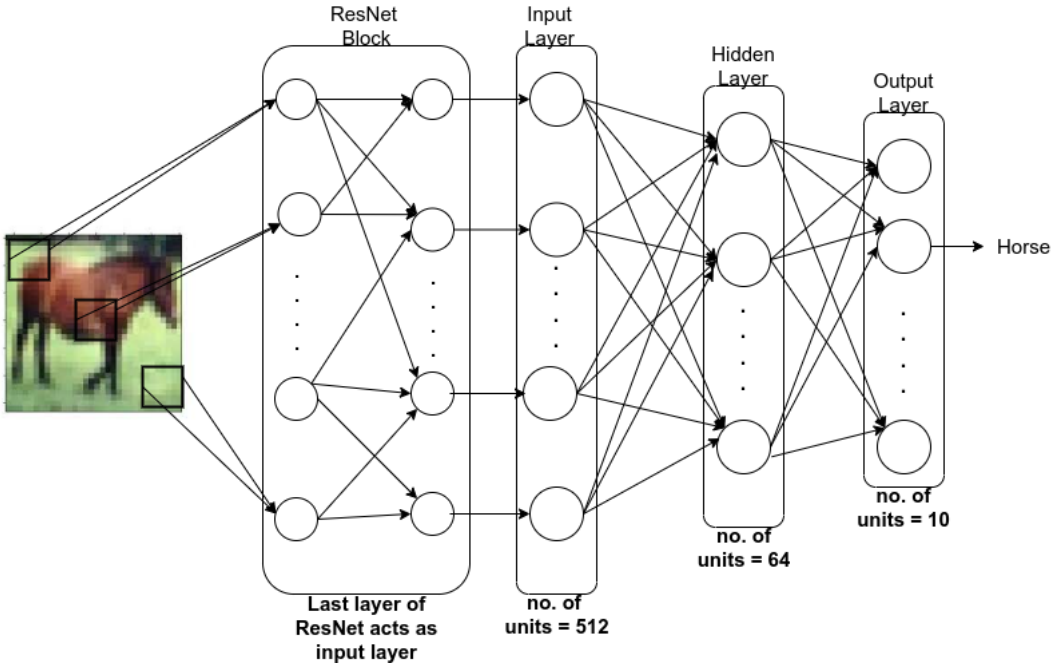


Figure 2: Model Architecture

3.1 Initialization

Each feature vector of image is of size (512, 1). So, we need to use 512 neurons as in input layer where each neuron represent one feature in the space. Now the number of neurons are decided then we need to initialize weights and bias vectors in order to train the model. Since there are 2 layers (i.e hidden and output) other than input layer so we need two sets of weight matrices and bias vectors. It is given that 64 neurons are to be used in hidden layer. So, first weight matrix is of size (64 x 512) i.e (no. of hidden layer neurons x input feature) and bias is of size (64, 1) i.e (no. of hidden layer neurons). Similarly for next output layer, weight matrix is of size (10, 64) and bias is of size (10, 1) as there are 10 categories in the datasets so no. of neurons for output layer is 10. I initialized both weight matrices as gaussian distribution bounded around 0. But the problem is that it will give some values more than 1 that may explode while training so we have to multiply by some value to keep it less than 1 i.e say 0.1 just to make sure all values lies in between 0 and 1. Another initialization method I also tried was Xavier initialization (4) which works better with ReLU activation function. Bias vectors can be initialized by zeros.

3.2 Feedforward calculations

Consider one image having feature vector X = $[X_1, X_2, ..., X_{512}]$. Here X is an input feature vectors of shape (512, 1) is getting multiplied with first weight matrices of shape (64, 512) and get added with bias of shape (64, 1) that gives Z of shape (64, 1). Then we will apply relu activation function on Z i.e a = max(0, Z). It basically replaces all negative value with 0. After that output a of shape (64, 1) is then multiplied with another matrices of shape (10, 64) plus bias of shape (10, 1) produces Z_2 is of shape (10, 1). To know about the probabilities of each category in the class labels we need to pass output Z_2 through softmax function. It will generate a vector of shape (10, 1) and each item contains the probabilities of each class.

$$ReLU(z) = max(0, z)$$

$$softmax(z) = \frac{e^{z_i}}{\sum_{i=1}^{10} e^{z_i}}$$

$$Z_1 = \begin{bmatrix} b_1^1 \\ b_2^1 \\ \vdots \\ b_{64}^1 \end{bmatrix}_{(64\ X\ 1)} + \begin{bmatrix} W_{11}^1 & W_{12}^1 & ... & W_{1512}^1 \\ W_{21}^1 & W_{22}^1 & ... & W_{2512}^1 \\ W_{31}^1 & W_{32}^1 & ... & W_{3512}^1 \\ \vdots & \vdots & \ddots & \vdots \\ W_{641}^1 & W_{5122}^1 & ... & W_{64512}^1 \end{bmatrix}_{(64\ X\ 512)} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_{512} \end{bmatrix}_{(512\ X\ 1)}$$

$$a_1 = ReLU(Z_1)$$

$$Z_2 = \begin{bmatrix} b_1^2 \\ b_2^2 \\ \vdots \\ b_{10}^2 \end{bmatrix}_{(10\ X\ 1)} + \begin{bmatrix} W_{11}^2 & W_{12}^2 & ... & W_{164}^2 \\ W_{21}^2 & W_{22}^2 & ... & W_{264}^2 \\ W_{31}^2 & W_{32}^2 & ... & W_{364}^2 \\ \vdots & \vdots & \ddots & \vdots \\ W_{101}^2 & W_{102}^2 & ... & W_{1064}^2 \end{bmatrix}_{(10\ X\ 64)} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{64} \end{bmatrix}_{(64\ X\ 1)}$$

$$\hat{y} = softmax(Z_2)$$

3.3 Backpropagation

Using backpropagation, we're going to update our weight and bias matrices. For that we first calculate loss function. Since this is a multi-class classification problem so, we'll use cross-entropy ($\mathcal{L}$) to define loss function shown in below eq.1

$$\mathcal{L} = -\sum_i y_i \log \hat{y}_i \tag{1}$$

where $\hat{y}$ is the predicted output, y is the true label and N is total number of samples. In order to update weights we'll use following eq.2

$$W_{new}^1 = W_{old}^1 - \frac{\alpha}{N}\left(\frac{\partial \mathcal{L}}{\partial W^1}\right); \quad b_{new}^1 = b_{old}^1 - \frac{\alpha}{N}\left(\frac{\partial \mathcal{L}}{\partial b^1}\right) \tag{2}$$

$$W_{new}^2 = W_{old}^2 - \frac{\alpha}{N}\left(\frac{\partial \mathcal{L}}{\partial W^2}\right); \quad b_{new}^2 = b_{old}^2 - \frac{\alpha}{N}\left(\frac{\partial \mathcal{L}}{\partial b^2}\right) \tag{3}$$

where W^1 and b^1 are weight and bias associated with layer 1, W^2 and b^2 are associated with layer 2 and α is the learning rate

Firstly re-writing all required equations used in the calculations for backpropagation and then calculate gradients moving from last layer to input layer.

$$Z_1 = b^1 + W^1.X \tag{4}$$
$$a_1 = max(0, Z_1) \tag{5}$$
$$Z_2 = b^2 + W^2.a_1 \tag{6}$$
$$\hat{y} = \frac{e^{Z_{2_i}}}{\sum_k e^{Z_{2_k}}} \tag{7}$$

For W^2 and b^2, we can write it as

$$\frac{\partial \mathcal{L}}{\partial W^2} = \frac{\partial \mathcal{L}}{\partial \hat{y}}.\frac{\partial \hat{y}}{\partial Z_2}.\frac{\partial Z_2}{\partial W^2} \tag{8}$$
$$\frac{\partial \mathcal{L}}{\partial b^2} = \frac{\partial \mathcal{L}}{\partial \hat{y}}.\frac{\partial \hat{y}}{\partial Z_2}.\frac{\partial Z_2}{\partial b^2} \tag{9}$$
$$\tag{10}$$

For W^1 and b^1, we can write it as

$$\frac{\partial \mathcal{L}}{\partial W^1} = \frac{\partial \mathcal{L}}{\partial \hat{y}}.\frac{\partial \hat{y}}{\partial Z_2}.\frac{\partial Z_2}{\partial a^1}.\frac{\partial a^1}{\partial Z_1}.\frac{\partial Z_1}{\partial W^1} \tag{11}$$
$$\frac{\partial \mathcal{L}}{\partial b^1} = \frac{\partial \mathcal{L}}{\partial \hat{y}}.\frac{\partial \hat{y}}{\partial Z_2}.\frac{\partial Z_2}{\partial a^1}.\frac{\partial a^1}{\partial Z_1}.\frac{\partial Z_1}{\partial b^1} \tag{12}$$

So, now we need to calculate each term separately and then will multiply all those items. For $\frac{\partial \mathcal{L}}{\partial W^2}$,

$$\mathcal{L} = -\sum_k y_k \log \hat{y}_k \tag{13}$$
$$\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = -\sum_k y_k \frac{\partial \log \hat{y}_k}{\partial \hat{y}_i} \tag{14}$$
$$\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = -\sum_k y_k \frac{1}{\hat{y}_k} \frac{\partial \hat{y}_k}{\partial \hat{y}_i} \tag{15}$$

Here are two conditions to consider when i=k and when $i \neq k$

$$
\begin{aligned}
& & \hat{y_k} &= \frac{e^{Z_{2_k}}}{\sum_k e^{Z_{2_k}}} \\
Case1 : i = k & & \frac{\partial \hat{y_i}}{\partial Z_{2_i}} &= \frac{\sum_k e^{Z_{2_k}}.e^{Z_{2_i}} - e^{Z_{2_i}}.e^{Z_{2_i}}}{(\sum_k e^{Z_{2_k}})^2} \\
& & &= \frac{e^{Z_{2_i}}}{\sum_k e^{Z_{2_k}}}.\left(\frac{\sum_k e^{Z_{2_k}}}{\sum_k e^{Z_{2_k}}} - \frac{e^{Z_{2_i}}}{\sum_k e^{Z_{2_k}}}\right) \\
& & &= \hat{y}_i.(1 - \hat{y}_i) \\
Case2 : i \neq k & & \frac{\partial \hat{y_k}}{\partial Z_{2_k}} &= \frac{0 - e^{Z_{2_i}}.e^{Z_{2_k}}}{(\sum_k e^{Z_{2_k}})^2} \\
& & &= -\frac{e^{Z_{2_i}}}{\sum_k e^{Z_{2_k}}}.\frac{e^{Z_{2_k}}}{\sum_k e^{Z_{2_k}}} \\
& & &= -\hat{y}_i.\hat{y}_k
\end{aligned}
$$

$$
\frac{\partial \hat{y_k}}{\partial Z_{2_i}} = \begin{cases} \hat{y}_i.(1 - \hat{y}_i) & if\ i = k \\ -\hat{y}_i.\hat{y}_k & if\ i \neq k \end{cases}
$$

Putting the above equations in eq.15,

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \hat{y}_i} &= -\sum_k y_k \frac{1}{\hat{y_k}} \frac{\partial \hat{y_k}}{\partial \hat{y}_i} \\
&= -\sum_{i=k} y_k \frac{1}{\hat{y_k}} \frac{\partial \hat{y_i}}{\partial \hat{y}_i} - \sum_{i \neq k} y_k \frac{1}{\hat{y_k}} \frac{\partial \hat{y_k}}{\partial \hat{y}_i}
\end{aligned}
$$

$$
\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = \begin{cases} -\frac{y_i}{\hat{y}_i} & if\ i = k \\ -\sum_k y_k \frac{1}{\hat{y_k}} \frac{\partial \hat{y_k}}{\partial \hat{y}_i} & if\ i \neq k \end{cases}
$$

And the last term to calculate for W^2 is $\frac{\partial Z_2}{\partial W^2}$,

$$
\begin{aligned}
Z_2 &= b^2 + W^2.a_1 \\
\frac{\partial Z_2}{\partial W^2} &= a_1
\end{aligned}
$$

For $\frac{\partial \mathcal{L}}{\partial W^2}$,

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial W^2} &= \frac{\partial \mathcal{L}}{\partial \hat{y}}.\frac{\partial \hat{y}}{\partial Z_2}.\frac{\partial Z_2}{\partial W^2} \\
&= \left(-y_i.(1-\hat{y}_i) - (\sum_k y_k \frac{1}{\hat{y}_k}.(-\hat{y}_i.\hat{y}_k)) \right).a_1 \\
&= \left(-y_i.(1-\hat{y}_i) + \sum_k y_k \hat{y}_i \right).a_1 \\
&= \left(-y_i + y_i\hat{y}_i + \sum_k y_k \hat{y}_i \right).a_1 \\
&= \left(-y_i + \hat{y}_i(y_i + \sum_k y_k) \right).a_1 \\
&= (\hat{y}_i - y_i).a_1
\end{aligned}
$$

Since y is an one hot-vector so the sum across all dimensions is 1 i.e $y_i + \sum_k y_k = 1$. Similarly for $\frac{\partial \mathcal{L}}{\partial b^2}$,

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial b^2} &= \frac{\partial \mathcal{L}}{\partial \hat{y}}.\frac{\partial \hat{y}}{\partial Z_2}.\frac{\partial Z_2}{\partial b^2} \\
&= \left(-y_i.(1-\hat{y}_i) - (\sum_k y_k \frac{1}{\hat{y}_k}.(-\hat{y}_i.\hat{y}_k)) \right).1 \\
&= \left(-y_i.(1-\hat{y}_i) + \sum_k y_k \hat{y}_i \right) \\
&= \left(-y_i + y_i\hat{y}_i + \sum_k y_k \hat{y}_i \right) \\
&= \left(-y_i + \hat{y}_i(y_i + \sum_k y_k) \right) \\
&= (\hat{y}_i - y_i)
\end{aligned}
$$

For W^1 and b^1,

$$a_1 = max(0, Z_1) \tag{16}$$

$$\frac{\partial a_1}{\partial Z_1} = \begin{cases} 0 & if Z_1 <= 0 \\ 1 & if Z_1 > 0 \end{cases} \tag{17}$$

$$\frac{\partial \mathcal{L}}{\partial W^1} = \frac{\partial \mathcal{L}}{\partial \hat{y}}.\frac{\partial \hat{y}}{\partial Z_2}.\frac{\partial Z_2}{\partial a^1}.\frac{\partial a^1}{\partial Z_1}.\frac{\partial Z_1}{\partial W^1} \tag{18}$$

$$= (\hat{y}_i - y_i).(W^2).(1).(X) \tag{19}$$

$$= (\hat{y}_i - y_i).W^2.X \tag{20}$$

$$\frac{\partial \mathcal{L}}{\partial b^1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial Z_2} . \frac{\partial Z_2}{\partial a^1} . \frac{\partial a^1}{\partial Z_1} . \frac{\partial Z_1}{\partial b^1} \tag{21}$$

$$= (\hat{y}_i - y_i).W^2 \tag{22}$$

Note- W^2 is not a square on W that just representing weights of second layer. So, $\frac{\partial \mathcal{L}}{\partial W^1} = 0$ and $\frac{\partial \mathcal{L}}{\partial b^1} = 0$ when $Z_1 <= 0$

Now we'll put all the derivatives in eq.2 and eq.3, we get

$$W^2_{new} = W^2_{old} - \frac{\alpha}{N}\left(\frac{\partial \mathcal{L}}{\partial W^2}\right); \quad b^2_{new} = b^2_{old} - \frac{\alpha}{N}\left(\frac{\partial \mathcal{L}}{\partial b^2}\right)$$

$$\boxed{W^2_{new} = W^2_{old} - \frac{\alpha}{N}\left((\hat{y}_i - y_i).a_1\right)} \tag{23}$$

$$\boxed{b^2_{new} = b^2_{old} - \frac{\alpha}{N}\left((\hat{y}_i - y_i)\right)} \tag{24}$$

$$W^1_{new} = W^1_{old} - \frac{\alpha}{N}\left(\frac{\partial \mathcal{L}}{\partial W^1}\right); \quad b^1_{new} = b^1_{old} - \frac{\alpha}{N}\left(\frac{\partial \mathcal{L}}{\partial b^1}\right)$$

$$\boxed{W^1_{new} = W^1_{old} - \frac{\alpha}{N}\left((\hat{y}_i - y_i).W^2.X\right)} \tag{25}$$

$$\boxed{b^2_{new} = b^2_{old} - \frac{\alpha}{N}\left((\hat{y}_i - y_i).W^2\right)} \tag{26}$$

So, now, we can update the weights and biases corresponding to both layers using eq.23, 24, 25 and 26.

3.4 Model Training

For model training, I did grid search on different hyperparameters and analyzed the results. While training the model, I updated the weights and bias for those losses only which are minimum than global minimum loss. And at each epoch I shuffled all the training datasets and evaluated its performance on testing datasets. Following hyperparameters were used to perform a grid search to check the performance of the model shown in Table 1. Fig.3 shows the variations of training accuracy vs loss graph.

Hyperparameters	
Epochs	500, 1000
Batch Sizes	32, 64, 128, 256
Learning rates	0.1, 0.01, 0.001
Loss	Categorical Crossentropy
Activation Functions	ReLU & Softmax

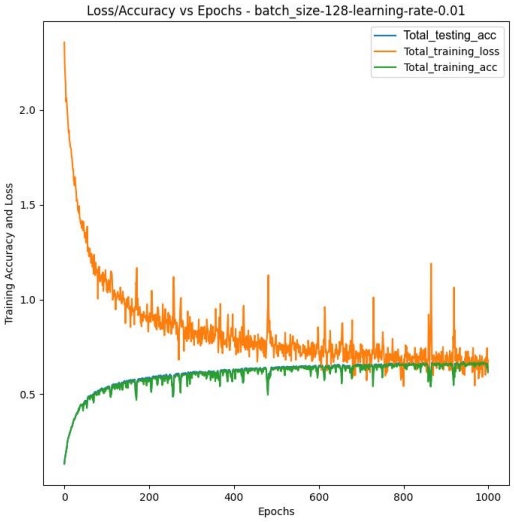
Table 1: Hyperparameters

3.5 Model Evaluation

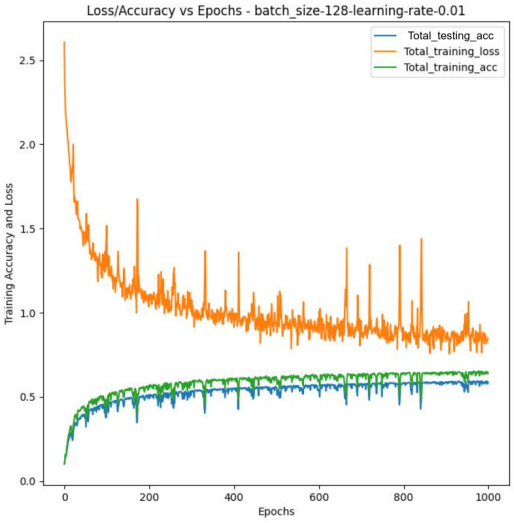
After performing grid search, I got an accuracy of **61.48**% on test dataset with batch size **128** and learning rate **0.01**. This accuracy is based on when model is trained on unaugmented datasets. While I got an accuracy of **64.35**% when model is trained on augmented datasets. Evaluation is measured on how many categories are exactly matched with ground truth divided by total no. of examples. Other accuracy and loss based on other hyperparameters are available in the logs provided with the code.

3.6 Conclusions

Ideally model gives good performance with data augmentation. And as we can see here the performance increases when tested on model trained on augmented datasets. The reason why accuracy is more because when model is trained on augmented datasets is that it contains more datasets including with some noisy data that helps model to generalize the datasets and perform well. We can improve further the model accuracy by increasing hidden layers to learn the parameters well for generalization purpose.



(a)



(b)

Figure 3: Training and Testing accuracy, loss vs Epochs (a) unaugmented datasets (b) augmented datasets

References

[1] https://www.cs.toronto.edu/~kriz/cifar.html.

[2] https://www.youtube.com/watch?v=Wo5dMEP_BbI.

[3] Mohamed Elegandy. Deep learning for vision systems. In *Deep Learning for Vision Systems*, 1989.

[4] https://www.deeplearning.ai/ai-notes/initialization/.