DEEP LEARNING FOR COMPUTER VISION

ASSIGNMENT NO 1

Member Names: Anshul Sharma, Lt Cdr Vijay Singh Sisodiya
Member Emails: anshulsh22@iitk.ac.in, vijayss22@iitk.ac.in
Member Roll Numbers: 22111009, 22111089
Date: February 3, 2023

1 Problem Statement

Objective is to implement and train a multi-layer perceptron (MLP) on the CIFAR- 10 dataset with data augmentation

1.1 About the CIFAR-10 Dataset

In the provided CIFAR-10 dataset, collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, there are 60,000 32x32 coloured (RGB) images divided into 10 different classes, such as vehicles, animals, and birds. A test batch and five training batches, each containing 1000 photos, have been created from the dataset. The remaining photographs are included in the training batches in a random order, however some training batches may have a greater proportion of images from one class than another. Exactly 5000 photos from each class are included in each of the training batches combined.

2 Problem solving Process-flow

2.1 Image transformation techniques

In order to augment dataset, we have utilised four image transformation techniques which are covered in subsequent paragraphs.

2.1.1 Image Enhancement

In the images obtained from CIFAR-10 dataset, image enhancement has been carried out at pixel levels using the formula

$$\frac{(i - min)}{(max - min)} * 255$$

where the i is pixel in the image, min and max are the minimum and maximum pixel values in that particular image.

Figure 1: Image enhancement performed on data set samples

2.1.2 Image Posterisation

In posterisation process, the number of colors in the image are reduced to obtain a visual art. As per the instructions, following operations have been performed on each of the images:
(a) Minimum and maximum pixel values, say *min* and *max* resp, fixed in the range of [0-255].
(b) For each pixel i in image, *Range* using Range= (Max-min)
(c) Calculated *Divider* using Divider= 255/range
(d) Calculate level of colors using i= i/ Divider
(e) Applying the color palette on pixel using i=i+min

Figure 2: Image Posterisation performed on data set samples

2.1.3 Image Rotation

Rotation of images is performed randomly at pixel levels [x,y] using function [-180,180]

Figure 3: Random rotation operation on dataset images

2.1.4 Contrast and horizontal flipping

In this process, we changed the contrast of the image with random factor α, where α lies in range (0.5, 2.0). Further, the image was flipped horizontally with a probability of 0.5.

Figure 4: Changing contrast of sample images

2.2 Augmentation of Dataset

By creating fresh and distinctive instances for training datasets, data augmentation helps machine learning models perform better and produce better results. A machine learning model performs better and is more accurate if the dataset is large and sufficient. By producing variables that the model could encounter in the real world, data augmentation approaches allow machine learning models to be more resilient. Therefore, in the given problem set, we have applied data

augmentation technique to obtain a total of 50,000 augmented images. This is achieved by applying data augmentation randomly on any one of the four transformed images obtained previously. Thus, we gathered a total of 1,00,000 samples (50,000 dataset samples + 50,000 transformed dataset samples)

2.3 Feature Extraction

Every computer vision problem is involved with with feature extraction. The process of identifying an image's distinctive qualities is crucial for analysis and classification. Therefore, for this problem statement, we have done the feature extraction by undermentioned steps:

(a) We have used the provided feature_extractor.py file on both unaugmented as well as augmented dataset i.e. 1,00,000 samples in order to get one-dimensional input vectors.

(b) Since, the feature_extractor.py accepts images of size (3x244x244), however, the images contained in CIFAR images were of dimensions (3x32x32). Therefore, we have utilized CV2 library to convert the same into (3x244x244) dimensions.

(c) Subsequently, the resized images were passed to feature extraction function of BBResNet18 class in order to generate feature vectors

2.4 MLP Network Architecture

After passing the image through the feature extractor function of BBResNet18 class, it gives a 512-dimensional feature vector. Therefore the number of nodes in the input layers will be 512. Two hidden layers having 64 neurons each and ReLu activation function have been incorporated in the network. Also, we use the CIFAR10 dataset with ten different classes, so the number of nodes in the output layer will be 10 and *Softmax* classifier is used to classify them.

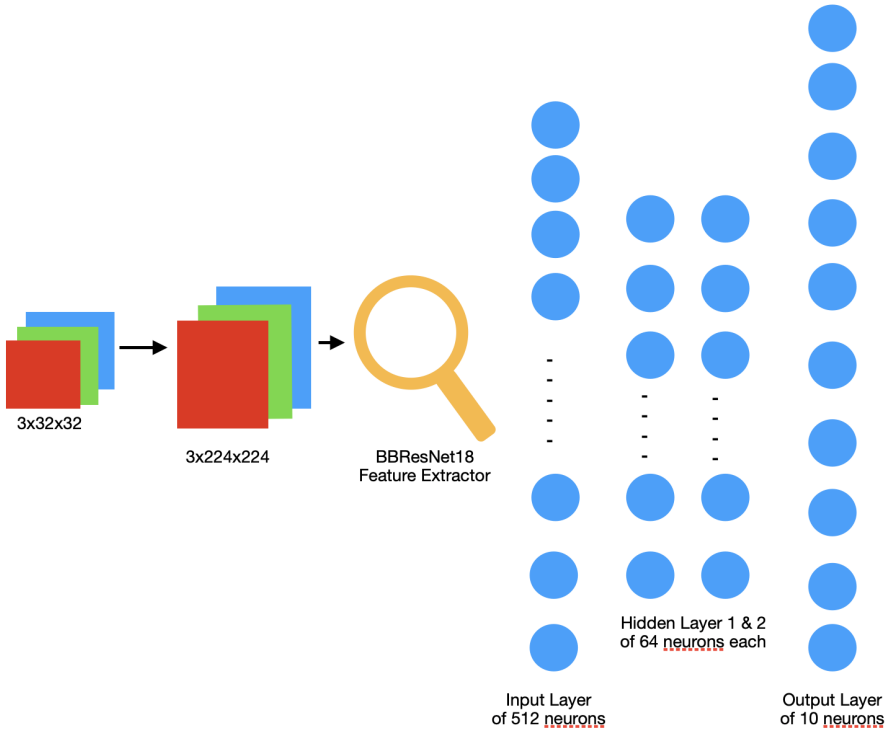

Figure 5: Network Architecture

2.4.1 Weight dimension and initialization

By using np.random.randn and dividing by 1000, weights are initialised at random and with very small numbers, while biases are initialised with zeros. The following are the dimensions of weights and biases:

1. W1 = $(512, 64)$
2. B1 = $(1, 64)$
3. W2 = $(64, 64)$
4. B2 = $(1, 64)$
5. W3 = $(64, 10)$
6. B3 = $(1, 10)$

2.4.2 Forward Propagation

Data input is transferred from the network's first layer to its bottom layer using forward propagation. Each layer's node calculates the weighted total of their

input before sending it to the activation layer, which adds nonlinearity to the network.

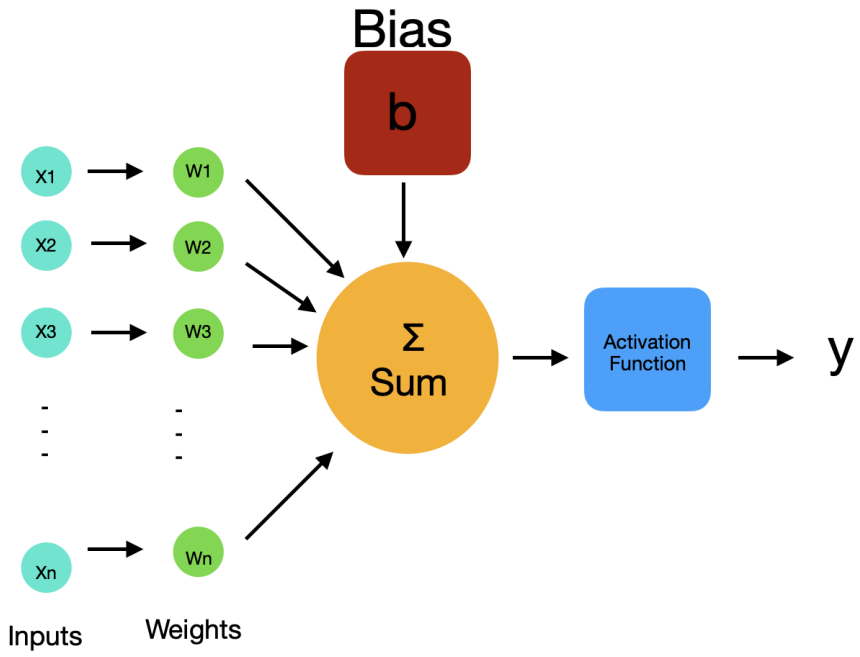

Figure 6: Working of single node in MLP

Let X_0 be the input to the network and Z_i be the output of i^{th} layer and A_i be the output of i^{th} activation layer.

So the forward propagation for model looks like this:

$$Z_1 = X_0 * W_1 + B_1$$
$$A_1 = \text{Re}\,LU\,(Z_1)$$
$$Z_2 = A_1 * W_2 + B_2$$
$$A_2 = \text{Re}\,LU\,(Z_2)$$
$$Z_3 = A_2 * W_3 + B_3$$
$$A_3 = \text{Softmax}\,(Z_3)$$

2.4.3 Activation Function- ReLu

In order for the neural network to learn more complicated data properties, activation functions are essential since they add non-linearity into the model. Determine the activation function ReLU activation is used for the hidden layer, softmax activation is used for the output layers. When the input is zero, it behaves as a dead neuron; otherwise, it produces the same result. In order to

achieve better results and address the issue of disappearing gradients, it is also utilised in hidden layers.

$$\text{relu}(x) = \max(0, x)$$

2.4.4 Classifier Function- Softmax

When dealing with multiclass classification issues, the last layer of the network employs the softmax activation function. It changes every value between 0 and 1, which is equivalent to a probability for that class.

$$\text{softam}\, x\,(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

2.5 Loss Function

How far an estimated value is from its real value is determined by the loss function. It is a technique for assessing how well our model performed. In our model, we employ cross-entropy loss. It evaluates the effectiveness of categorization models, the results of which are probabilities.

$$\text{Loss}_{CE} = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log\left(\hat{y}_i\right)$$

3 Implementation of Back Propagation Algorithm

The artificial neural network uses backpropagation, a learning method, to increase prediction accuracy. It is a method used in mathematics to determine the gradient of the loss in relation to weights and biases.

Since the model's prediction is the third layer's output, ie., $A_3 = \hat{Y}$ Therefore

$$\text{Loss}_{CE} = -\frac{1}{m}\sum_{i=1}^{m} y^i \cdot \log\left(A_3^i\right)$$

The gradient of Softmax activation is given as

$$\frac{\partial a^i}{\partial z^j} = \begin{cases} -a^i a^j & \text{if } i \neq j \\ a^i\left(1 - a^i\right) & \text{if } i = j \end{cases}$$

3.1 Derivation of Gradient

$$
\begin{aligned}
\frac{dL}{dZ_3^i} &= \frac{d}{dZ_3^i}\left[-\sum_{k=1}^{c} Y^k \log\left(A_3^k\right)\right] = -\sum_{k=1}^{C} Y^k \frac{d\left(\log\left(A_3^k\right)\right)}{dZ_3^i} \\
&= -\sum_{k=1}^{C} Y^k \frac{d\left(\log\left(A_3^k\right)\right)}{dA_3^k} \cdot \frac{dA_3^k}{dZ_3^i} \\
&= -\sum_{k=1}^{C} \frac{Y^k}{A_3^k} \cdot \frac{dA_3^k}{dZ_3^i} \\
&= -\left[\frac{Y^i}{A_3^i} \cdot \frac{dA_3^i}{dZ_3^i} + \sum_{k=1,k\neq i}^{C} \frac{Y^k}{A_3^k}\frac{dA_3^k}{dZ_3^i}\right] \\
&= -\frac{Y^i}{A_3^i} \cdot A_3^i\left(1 - A_3^i\right) - \sum_{k=1,k\neq i}^{C} \frac{Y^k}{A_3^k} \cdot \left(A_3^k A_3^i\right) \\
&= -Y^i + Y^i A_3^i + \sum_{k=1,k\neq i}^{C} Y^k A_3^i \\
&= A_3^i\left(Y^i + \sum_{k=1,k\neq i}^{C} Y^k\right) - Y^i = A_3^i \cdot \sum_{k=1}^{C} Y^k - Y^i \\
&= A_3^i \cdot 1 - Y^i, \text{since} \sum_{k=1}^{C} Y^k = 1 \\
&= A_3^i - Y^i = A_3 - Y
\end{aligned}
$$

Using the above derivation we can find the gradient of loss with respect to weight and biases as follows:

$$dZ_3 = \frac{\partial L}{\partial Z_3} = dZ_3 = A_3 - Y$$
$$dW_3 = \frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial Z_3}\frac{\partial Z_3}{\partial W_3} = \frac{1}{m}A_2^T dZ_3$$
$$dB_3 = \frac{\partial L}{\partial B_3} = \frac{\partial L}{\partial Z_3}\frac{\partial Z_3}{\partial B_3} = \frac{1}{m}np \cdot \text{sum}\,(dZ_3,\ \text{axis}\ = 0)$$
$$g_1' = \frac{\partial A_2}{\partial Z_2} = 1 \text{ if } A_2^i > 0 \text{ otherwise } 0$$

$$dZ_2 = \frac{\partial L}{\partial Z_2} = \frac{\partial L}{\partial Z_3}\frac{\partial Z_3}{\partial A_2}\frac{\partial A_2}{\partial Z_2} = dZ_3 W_3^T * g_1'$$
$$dW_2 = \frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial Z_3}\frac{\partial Z_3}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial W_2} = \frac{1}{m}A_1^T dZ_2$$
$$dB_2 = \frac{\partial L}{\partial B_2} = \frac{\partial L}{\partial Z_3}\frac{\partial Z_3}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial B_2} = \frac{1}{m}\,\text{np. sum}\,(dZ_2, axis = 0)$$
$$g_0' = \frac{\partial A_1}{\partial Z_1} = 1 \text{ if } A_1^i > 0 \text{ otherwise } 0$$

$$dZ_1 = \frac{\partial L}{\partial Z_1} = \frac{\partial L}{\partial Z_3}\frac{\partial Z_3}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1} = dZ_2 W_2^T * g_0$$
$$dW_1 = \frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial Z_3}\frac{\partial Z_3}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1}\frac{\partial Z_1}{\partial W_1} = \frac{1}{m}x_0^T dZ_1$$
$$dB_1 = \frac{\partial L}{\partial B_1} = \frac{\partial L}{\partial Z_3}\frac{\partial Z_3}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1}\frac{\partial Z_1}{\partial B_1} = \frac{1}{m}\,\text{np. sum}\,(dZ_1, axis = 0)$$

4 Gradient Descent

Gradient descent is an iterative optimization algorithm for finding the local minimum of a function.66 After performing the back-propagation, it gives the gradients of loss with respect to all weights and biases. For updating weights and biases, we are using mini batch gradient descent.

$$w3 = w3 - \ \text{learningrate}\ * dw3$$
$$b3 = b3 - \ \text{learningrate}\ * db3$$
$$w2 = w2 - \ \text{learningrate}\ * dw2$$
$$b2 = b2 - \ \text{learningrate}\ * db2$$
$$w1 = w1 - \ \text{learningrate}\ * dw1$$
$$b1 = b1 - \ \text{learningrate}\ * db1$$

5 Model Training and Hyper-parameters

Model training is the phase in the machine learning development lifecycle where we try to fit the best combination of weights and bias to a deep learning algorithm to minimize a loss function over the prediction range 7 .

For this assignment, two models are trained, one on the original dataset and one on the augmented dataset. Hyper-parameter for both the model training are mentioned as follows:

1. All the weights are initialized with numpy seed = 0 so that result can be produced again.
2. Models are trained on original(Unaugmented) data and Augmented data for 100 and 40 epochs respectively.
3. Batch size = 32.
4. Mini-batch gradient descent is used for updating the weights and bias.
5. Initial Learning rate = 0.01 and it performed better on this selected learning rate.
6. Cross entropy loss is used for the calculation of loss.

6 Result and Conclusion

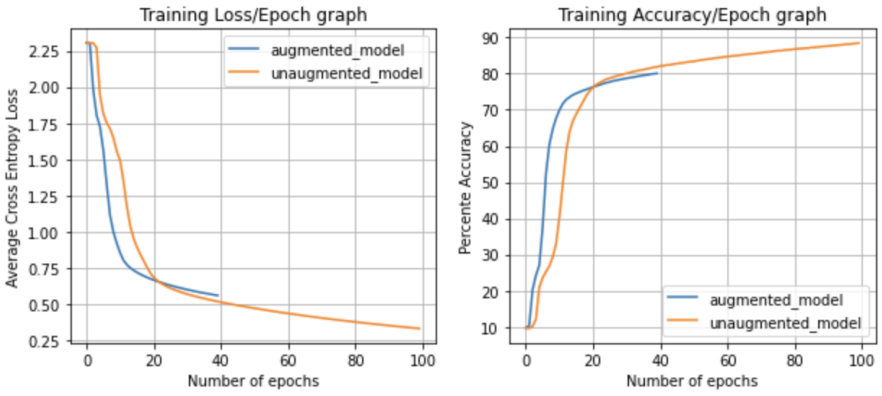

Figure 7: (a) Training Loss and Accuracy for augmented model and original model. (b) Test Loss and Accuracy for the augmented and original model

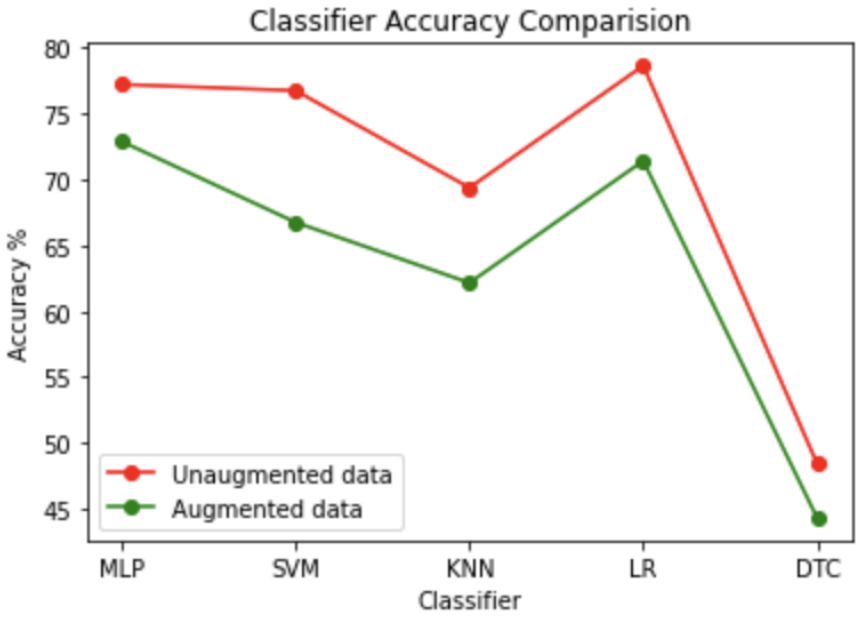

Figure 8: Classifier Performance Comparison on both augmented test set and original test set.

Model	Training Accuracy	Training Loss	Test Accuracy	Test Loss
Original Dataset	88.474%	0.333	77.220%	0.818
Augmented Dataset	80.076%	0.561	72.780%	0.804

Table 1: Comparative Study of MLP on both Augmented and Unaugmented Data

Classifier	Accuracy on Aug. Data	Accuracy on Org. Data
$MLPClassifier$	72.780%	77.220%
$SVMClassifier$	66.75%	76.75%
$DecisionTreeClassifier$	44.34%	48.46%
$LogisticRegressionClassifier$	71.42%	78.64%
$KNNClassifier$	62.16%	69.38%

Table 2: Comparative Study of all classifiers

7 Conclusion

1. The model trained with the original dataset converges faster than that trained with the augmented dataset.
2. After 100 epochs, the training accuracy of the unaugmented model is 88.474%, whereas the accuracy of the augmented model after 40 epochs is more than 80.076%.
3. The model that used the Unaugmented dataset gives better test set accuracy than the model trained with the augmented dataset.

8 References

[1] Building a Neural Network with a Single Hidden Layer using Numpy https://towardsdatascience.com/building-a-neural-network-with-a-single-hidden-layer-using-numpy

[2] Understanding and implementing Neural Network with SoftMax in Python from scratch https://www.adeveloperdiary.com/data-science/deep-learning/ neural-network-with-softmax-in-python/

[3] Rotating Image By Any Angle(Shear Transformation) Using Only NumPy https://gautamnagrawal.medium.com/ rotating-image-by-any-angle-shear-transformation-using-only-numpy-d28d16eb5076

[4] The CIFAR-10 dataset https://www.cs.toronto.edu/ kriz/cifar.html

[5] What is Data Augmentation? Techniques, Benefit Examples https://research.aimultiple.com/data-augmentation/amp/

[6] How Does the Gradient Descent Algorithm Work in Machine Learning? https://www.analyticsvidhya.com/blog/2020/10/ how-does-the-gradient-descent-algorithm-work-in-machine-learning/

[7] Model Training https://c3.ai/glossary/data-science/model-training/#: :text= What %20 is %20 Model%20Training%3F, function%20over %20 the %20 prediction%20range.

[8] Neural Networks and Deep Learning https://www.coursera.org/learn/neural-networks-deep-learning?specialization=deep-learning

[9] Training with Image Data Augmentation in Keras https://stepup.ai/train_data_augmentation_keras/