Deep Learning for Computer Vision (CS776A)
Indian Institute of Technology Kanpur
Assignment 1

Member Names: Himanshu Lal
Member Emails: himanshul21@iitk.ac.in
Member Roll Numbers: 21111403
Date: February 3, 2022

MLP Model for CIFAR10 Classification

1 CIFAR-10 Dataset

CIFAR-10 dataset is a subset of 80 million tiny image dataset. It contains 60,000 32x32 colored (RGB) images of 10 classes. It has 50,000 training images stored in 5 batches and 10,000 test images in one batch. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Both training and test images are drawn from the same distribution.[4]

2 Data Augmentation

Data augmentation is helpful to improve the performance and outcomes of machine learning models by forming new and different examples to train datasets. If the dataset in a machine learning model is rich and sufficient, the model performs better and is more accurate. Data augmentation techniques enable machine learning models to be more robust by creating variations that the model may see in the real world.[5]

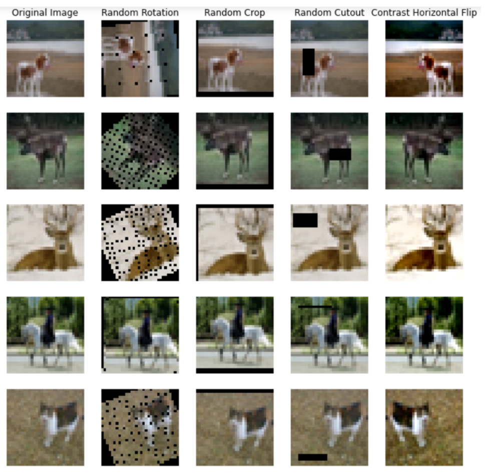

Figure 1: Different type of augmentation performed on randomly selected images

A few data Augmentation methods that we have used in this assignment are mentioned below.

1. Random Crop

2. Random Rotation
3. Random Cutout
4. Random Contrast and Horizontal Flip.

After performing data augmentation on original dataset, total number of samples in augmented dataset is 100K.

3 Feature Extraction

Feature Extraction is an essential step of any Computer Vision task. It is a process of finding the distinguishing characteristics of image that is important for analysis and classification.

In this assignment for extracting features, we use the pre-trained CNN architecture of ResNet 18 on ImageNet Dataset. It takes image input in batches of dimension (m, 3, 224, 224) and gives data features of dimension (m, 512).

4 Network Architecture

After passing the image through the ResNet18 feature extractor, it gives a 512-dimensional feature vector. Therefore the number of nodes in the input layers will be 512. Also, we use the CIFAR10 dataset with ten classes, so the number of nodes in the output will be 10. The only hidden layers have 64 nodes.

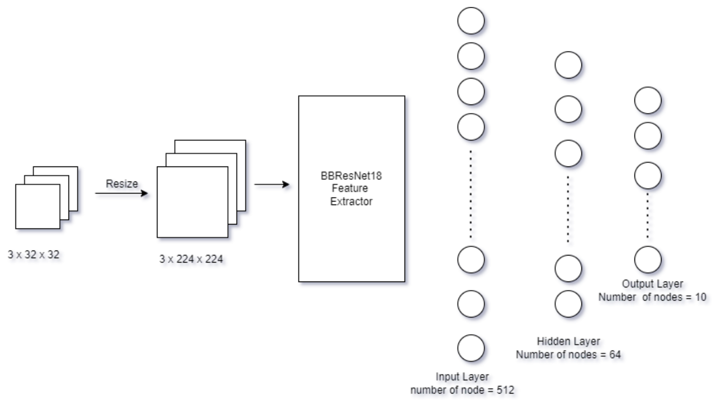
3 x 32 x 32
Resize
3 x 224 x 224
BBResNet18 Feature Extractor
Input Layer number of node = 512
Hidden Layer Number of nodes = 64
Output Layer Number of nodes = 10

Figure 2: Network Architecture

Weight dimension and initialization

Weights are initialized randomly by using np.random.randn and divided by 1000 so that all the weights are initialized with very small numbers and Biases are initialized by Zeros. The dimension of weights and biases are as follow:

1. W1 = (512, 64)
2. B1 = (1, 64)
3. W2 = (64, 10)
4. B2 = (1, 10)

5 Forward Propagation

Forward Propagation moves input data from the first layer to the last layer of the network. All the node in the layers calculate weighted sum of their input and then pass it to the activation layer for introducing non-linearity into the network.

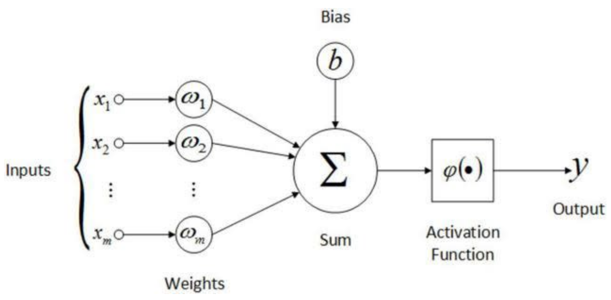

Figure 3: Working of single node in MLP.

Let A_0 be the input to the netowrk 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:

$$Z_1 = A_0 * W_1 + B_1 \tag{1}$$

$$A_1 = ReLU(Z_1) \tag{2}$$

$$Z_2 = A_1 * W_2 + B_2 \tag{3}$$

$$A_2 = Softmax(Z_2) \tag{4}$$

6 Activation Function

Activation functions are crucial for the neural network because they introduce non-linearity into the model that helps the network learn more complex data features. Activation funciton decide We are using ReLU activation for the hidden layer and softmax activation for the output layers.

6.1 ReLU - Rectified Linear Unit

ReLU is the most used activation function in deep learning. It acts as a dead neuron when the input goes to zero otherwise returns the same value. Also, it is used in hidden layers to solve the problem of vanishing gradient and for better results.

$$relu(x) = max(0, x) \tag{5}$$

6.2 Softmax

The softmax activation function is used in the network's last layer when dealing with multi-class classification problems. It transforms all the values in the range 0 and 1, which can be interpreted as a probability of that class.

$$softamx(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{6}$$

7 Loss Function

The loss function measure how far an estimated value is from its actual value. It is a method to evaluate the performance of our model. We are using Cross-Entropy Loss in our model. It measures the performance of classification model whose outputs are probability values.

$$Loss_{CE} = -\frac{1}{m}\sum_{i=1}^{m} y_i.log(\hat{y}_i) \tag{7}$$

8 Back Propagation

Backpropagation is a learning algorithm used by the artificial neural network to improve the accuracy of the prediction. It is a mathematical tool for calculating the gradient of the loss with respect to weights and biases. [2]

Since output of the second layer is the prediction of the model ie., $A_2 = \hat{Y}$ Therefore

$$Loss_{CE} = -\frac{1}{m}\sum_{i=1}^{m} y^i.log(A_2^i) \tag{8}$$

The gradient of Softmax activation is given as

$$\frac{\partial a^i}{\partial z^j} = \begin{cases} -a^i a^j & \text{if } i \neq j \\ a^i(1-a^i) & \text{if } i = j \end{cases} \tag{9}$$

Derivation of Gradient[2]

$$\frac{dL}{dZ_2^i} = \frac{d}{dZ_2^i}\left[-\sum_{k=1}^{c} Y^k log(A_2^k)\right] = -\sum_{k=1}^{C} Y^k \frac{d\big(log(A_2^k)\big)}{dZ_2^i} \tag{10}$$

$$= -\sum_{k=1}^{C} Y^k \frac{d\big(log(A_2^k)\big)}{dA_2^k}.\frac{dA_2^k}{dZ_2^i} \tag{11}$$

$$= -\sum_{k=1}^{C} \frac{Y^k}{A_2^k}.\frac{dA_2^k}{dZ_2^i} \tag{12}$$

$$= -\left[\frac{Y^i}{A_2^i}.\frac{dA_2^i}{dZ_2^i} + \sum_{k=1,k\neq i}^{C} \frac{Y^k}{A_2^k}\frac{dA_2^k}{dZ_2^i}\right] \tag{13}$$

$$= -\frac{Y^i}{A_2^i}.A_2^i(1-A_2^i) - \sum_{k=1,k\neq i}^{C} \frac{Y^k}{A_2^k}.(A_2^k A_2^i) \quad ..using\ equation\ (9) \tag{14}$$

$$= -Y^i + Y^i A_2^i + \sum_{k=1,k\neq i}^{C} Y^k A_2^i \tag{15}$$

$$= A_2^i\big(Y^i + \sum_{k=1,k\neq i}^{C} Y^k\big) - Y^i = A_2^i.\sum_{k=1}^{C} Y^k - Y^i \tag{16}$$

$$= A_2^i.1 - Y^i \text{ , since } \sum_{k=1}^{C} Y^k = 1 \tag{17}$$

$$= A_2^i - Y^i = A_2 - Y \tag{18}$$

Derivation of above gradients is inspired by [2].Using the above derivation we can find the gradient of loss with respect to weight and biases as follows:[1],[8]

$$dZ_2 = \frac{\partial L}{\partial Z_2} = dZ_2 = A_2 - Y \tag{19}$$

$$dW_2 = \frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial Z_2}\frac{\partial Z_2}{\partial W_2} = \frac{1}{m}A_1^T dZ_2 \tag{20}$$

$$dB_2 = \frac{\partial L}{\partial B_2} = \frac{\partial L}{\partial Z_2}\frac{\partial Z_2}{\partial B_2} = \frac{1}{m}np.sum(dZ_2, axis = 0) \tag{21}$$

$$g_1^{'} = \frac{\partial A_1}{\partial Z_1} = 1\ if\ A_1^i\ >\ 0\ otherwise\ 0 \tag{22}$$

$$dZ_1 = \frac{\partial L}{\partial Z_1} = \frac{\partial L}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1} = dZ_2 W_2^T * g_1^{'} \tag{23}$$

$$dW_1 = \frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1}\frac{\partial Z_1}{\partial W_1} = \frac{1}{m}A_0^T dZ_1 \tag{24}$$

$$dB_1 = \frac{\partial L}{\partial B_1} = \frac{\partial L}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1}\frac{\partial Z_1}{\partial B_1} = \frac{1}{m}np.sum(dZ_1, axis = 0) \tag{25}$$

9 Gradient Descent

Gradient descent is an iterative optimization algorithm for finding the local minimum of a function.[6] 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.

$$w2 = w2 - learningrate * dw2 \tag{26}$$

$$b2 = b2 - learningrate * db2 \tag{27}$$

$$w1 = w1 - learningrate * dw1 \tag{28}$$

$$b1 = b1 - learningrate * db1 \tag{29}$$

10 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 same and as follows:

1. All the weights are initialized with numpy seed = 0 so that result can be produced again.
2. Models are trained for 300 epochs.
3. Batch size = 64.
4. Mini-batch gradient descent is used for updating the weights and bias.
5. Initial Learning rate = 0.01 and it is reduced by the factor of 0.1 after every 100 epochs.
6. Cross entropy loss is used for calculation of loss.

11 Result and Conclusion

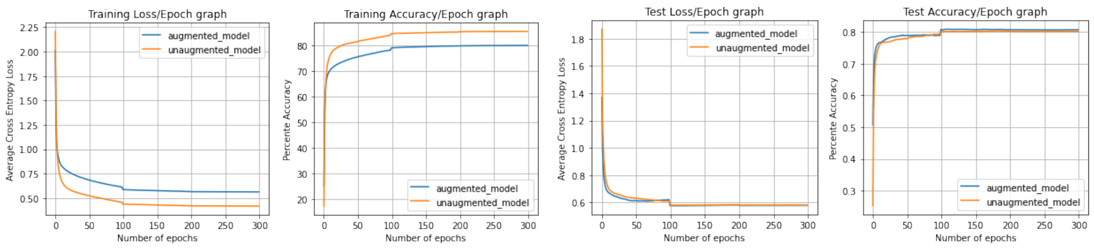

Figure 4: (a) Training Loss and Accuracy for augmented model and original model. (b) Test Loss and Accuracy for augmented and original model.

Model	**Training Accuracy**	**Training Loss**	**Test Accuracy**	**Test Loss**
Original Dataset	85.546%	0.417	80.140%	0.580
Augmented Dataset	80.128%	0.562	80.680%	0.579

Table 1: Comparative Study of both Classification models

Key points

1. The model trained with the original dataset converges faster than that trained with the augmented dataset.
2. After 300 epochs, the training accuracy of the augmented model is 80%, whereas the accuracy of the unaugmented model is more than 85.
3. The model that used the augmented dataset gives better test set accuracy than the model trained with the original dataset.
4. **Adding more (augmented) images in the dataset improves the model's performance. Only using the original dataset overfits the model, whereas the augmented dataset gives the effect of regularizer and reduces the problem of overfitting.**
5. **Using the original dataset, the model sees the same image repeatedly and picks some specific feature of the image and overfit the model for those features. Adding additional images with some augmentation performed on them makes it harder for the model to pick those specific features.**[9]
6. After 100 epochs, the learning rate is reduced by 0.1, which improves the model's learning capability, which justifies the sudden decrease in the loss and increase in the accuracy at $100^{th} epoch$.

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%20is%20Model%20Training%3F,function%20over%20the%20prediction%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/