CS776A Assignment 1
Name:- Debdeep Paul Chaudhuri
Roll:- 21111413

Neural Network from Nuts and Bolts

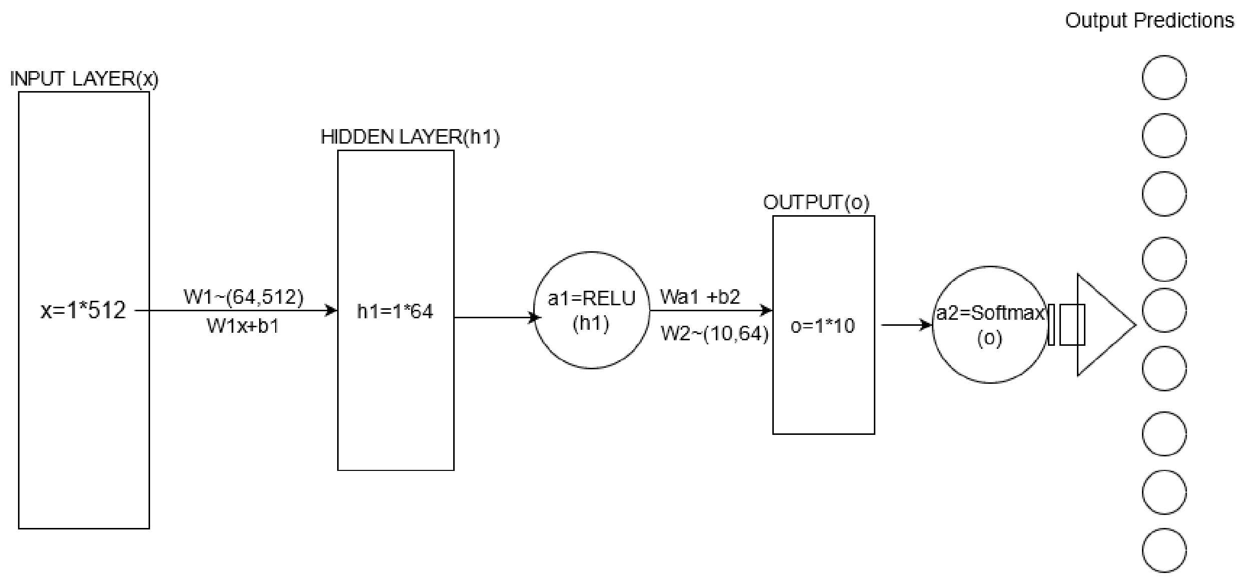
A BRIEF INTRO ABOUT THE PROBLEM:

We're working on a benchmark image processing dataset known as CIFAR-10(10 indicates number of different labels present in the image namely Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck) which contains 50,000 images curated from a variety of sources, and it also has 10,000 test images.

MODEL ARCHITECTURE

A MLP is a neural network that has an input layer followed by multiple hidden layers and an output layer. To introduce non-linearity we include activation functions like relu, tanh, sigmoid, softmax etc. These activation functions have a wide range of applications, and in our model we're using ReLU in the hidden layer and Softmax in the output layer to generate a probability distribution across all the labels.

1. Input Layer(1*512)
2. Weight matrix 1(W1~[64,512]), bias 1(b1~[64,1])
3. Hidden Layer(1*512)
4. Activation 1(a1=RELU)
5. Weight matrix 2(W2~[10,64]), bias 2(b2~[10,1])
6. Output Layer(1*10)
7. Activation 2(a2=Softmax)



LEARNING RATE:

Learning rate is set to 0.01, which facilitates fast convergence in mini_batch gradient descent, demanding less epochs for completion.

EPOCHS USED FOR TRAINING:

I've used 120 epochs for training on both un-augmented and augmented dataset, and epochs are purely empirical in nature, 120 seems to be a sweet spot and a reasonable tradeoff b/w time taken and performance obtained. Increasing the number of epochs without any bound will take a greater amount of time and comes at the risk of increasing chances of overfitting.

EVALUATION METRICS

Categorical-Cross Entropy Loss:- This loss function is widely used in multi-class classification problems. In our case, the final layer produces a probability distribution when passed through Softmax activation, the output labels are converted to one hot encoded format(making it suitable for comparison). The loss function attempts to model the difference between 2 different probability distributions namely the activations(a2) and the encoded labels(y).

Softmax is given by:- $f(s)_i = \frac{e^{s_i}}{\Sigma_j^C e^{s_j}}$, where C=no. of classes

Cross Entropy Loss:- $CE = -\Sigma_i^C y_i log(f(s_i))$

DERIVING GRADIENTS AND UPDATE EXPRESSIONS

We need to find derivative of the Loss function w.r.t every tunable parameters such as

W_1, W_2, b_1, b_2, and to achieve that we'll be needing a lot of derivative values along the way, and we'll do so using chain rule of derivatives and moving from the output layer all the way to the weights and biases.

Our Calculation flow will be as follows:-

$$\frac{dL}{dh_2} = \frac{dL}{da_2} \times \frac{da_2}{dh_2} ....(1)$$

$$\frac{dL}{dW_2} = \frac{dL}{dh_2} \times \frac{dh_2}{dW_2} ...(2)$$

$$\frac{dL}{db_2} = \frac{dL}{dh_2} \times \frac{dh_2}{db_2} ...(3)$$

$$\frac{dL}{da_1} = \frac{dL}{dh_2} \times \frac{dh_2}{da_1} ....(4)$$

$$\frac{dL}{dz_1} = \frac{dL}{da_1} \times \frac{da_1}{dz_1} .....(5)$$

$$\frac{dL}{dW_1} = \frac{dL}{dz_1} \times \frac{dz_1}{dW_1} .....(6)$$

$$\frac{dL}{db_1} = \frac{dL}{dz_1} \times \frac{dz_1}{db_1} .....(7)$$

Differentiating L w.r.t h_2

$$\frac{dL}{dh_i} = \frac{d}{dh_i}\left[\sum_{k=1}^{c} y_k log(a_k)\right]$$

$$= \sum_{k=1}^{c} y_k \frac{d\big(log(a_k)\big)}{dh_i}$$

$$= \sum_{k=1}^{c} y_k \frac{d\big(log(a_k)\big)}{da_k} . \frac{da_k}{dh_i}$$

$$= \sum_{k=1}^{c} \frac{y_k}{a_k} . \frac{da_k}{dh_i}$$

$$= \left[\frac{y_i}{a_i} . \frac{da_i}{dh_i} + \sum_{k=1, k\neq i}^{c} \frac{y_k}{a_k} \frac{da_k}{dh_i}\right]$$

$$= \frac{y_i}{a_i} . a_i(1 - a_i) \quad \sum_{k=1, k\neq i}^{c} \frac{y_k}{a_k} . (a_k a_i)$$

$$= y_i + y_i a_i + \sum_{k=1, k\neq i}^{c} y_k a_i$$

$$= a_i \big(y_i + \sum_{k=1, k \neq i}^{c} y_k\big) y_i$$

$$= a_i + \sum_{k=1}^{c} y_k - y_i$$

$$= a_i .1 - y_i \text{ , since } \sum_{k=1}^{c} y_k = 1$$

$$= a_i - y_i$$

Differentiating h2 w.r.t W2

$$\frac{dh_2}{dW_2} = a1$$

Differentiating h2 w.r.t b2

$$\frac{dh_2}{db_2} = IdentityMatrix$$

Differentiating h2 w.r.t a1

$$\frac{dh_2}{da_1} = W_2$$

Differentiating a1 w.r.t z1

$$\frac{da_1}{dz_1} = D \; where \; D_{ii} = 1 \; if \; z_i > 0 \; else \; 0$$

Differentiating z1 w.r.t W1

$$\frac{dz_1}{dW_1} = x$$

Differentiating z1 w.r.t b1

$$\frac{dz_1}{db_1} = Identity$$

Putting all of the values we got in eqn. (1), (2), (3), (4), (5), (6), (7), gradients w.r.t each parameters are:-

$$dW_2 = (a_2 - y) * a1.Shape(10, 64)$$

$$db_2 = (a_2 - y).....Shape(10, 1)$$

$$dW_1 = (a_2 - y) * W_2 * D * x..Shape(64, 512)$$

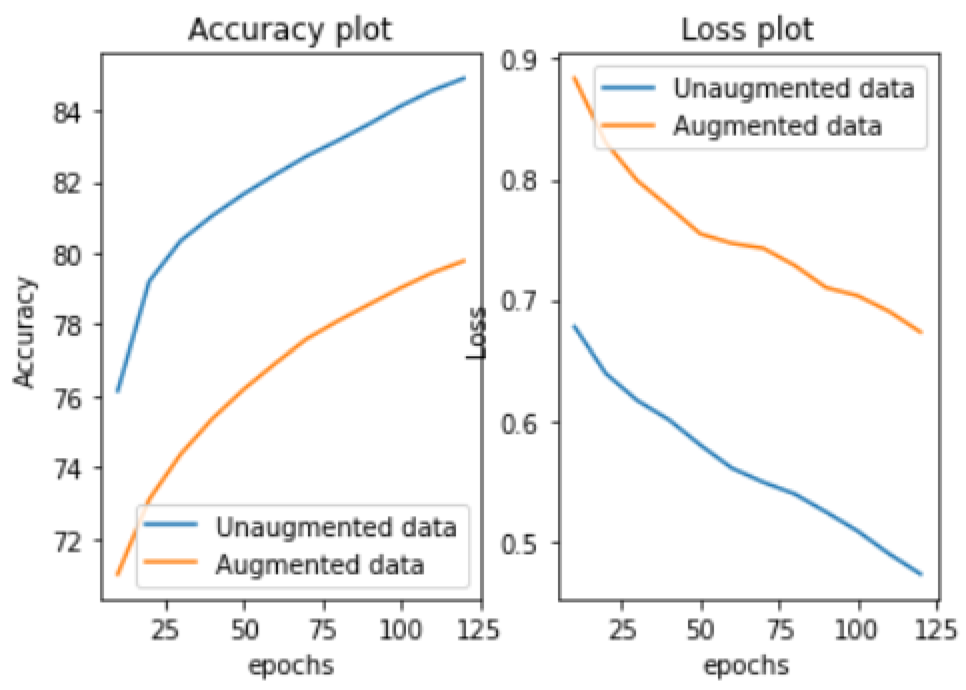
$$db_1 = (a_2 - y) * W_2 * D..Shape(64, 1)$$

RESULTS AND CONCLUSIONS:

	Training Loss	Training Accuracy	Testing Loss	Testing Accuracy
Original Data	0.47	84.9%	0.60	79.8%
Augmented Data	0.56	79.78%	0.59	80.2%

Although there isn’t much difference in Testing accuracy of Original and Augmented data, still the difference can be explained. When we train without augmentation, we are limited to the images in the training set. These images will be shown to the model over and over again. As a result, *the model can start to pick up on specific details in the training data*. When we train for long enough, these details will be less and less related to the actual object shown in the image and more related to the background or how the object is placed in the scene. This is clearly not what we want the model to learn. So therefore, when we vary the image slightly by using augmentation, *we make it harder for the model to pick up on these specific features*. E.g. When we train with a horizontal flip, we don't destroy any information relevant to the model, but we vary the information that we don't want the model to pick up. This information is for example low-level features at specific locations or the pose of the object we wish to classify. However this only occurred once or twice, and the trained parameters might be slightly off than what's expected in an ideal case.

Here are some plots for visual demonstration:



REFERENCES

1. The CIFAR-10 dataset.
2. For derivation of backpropagation
3. For understanding the flow of code

4. Why Augmentation is done