**Deep Learning Assignment 1 - Report**

Using MNIST dataset to train a 4 layer [20,7,5,10] neural network. 80/20 training/validation split. Learning rate .

Without dropout, without batch normalization:

Train cost at every **hundredth** training step:

|  |  |
| --- | --- |
| 0 | 2.302835 |
| 1 | 1.606642 |
| 2 | 0.954348 |
| 3 | 0.756562 |
| 4 | 0.712141 |
| 5 | 0.644852 |
| 6 | 0.755451 |
| 7 | 0.391164 |
| 8 | 0.501066 |
| 9 | 0.454734 |
| 10 | 0.39164 |
| 11 | 0.356354 |
| 12 | 0.346846 |
| 13 | 0.322747 |
| 14 | 0.323141 |
| 15 | 0.306486 |
| 16 | 0.343067 |
| 17 | 0.291438 |
| 18 | 0.30731 |
| 19 | 0.327239 |
| 20 | 0.278005 |
| 21 | 0.297851 |
| 22 | 0.296844 |

Validation cost at every hundredth training step:

|  |  |
| --- | --- |
| 0 | 2.302632 |
| 1 | 1.864374 |
| 2 | 0.980276 |
| 3 | 0.787528 |
| 4 | 0.790899 |
| 5 | 0.605452 |
| 6 | 0.695781 |
| 7 | 0.464273 |
| 8 | 0.568781 |
| 9 | 0.420555 |
| 10 | 0.426309 |
| 11 | 0.380158 |
| 12 | 0.462368 |
| 13 | 0.353711 |
| 14 | 0.343152 |
| 15 | 0.347292 |
| 16 | 0.327627 |
| 17 | 0.325491 |
| 18 | 0.315529 |
| 19 | 0.350373 |
| 20 | 0.321341 |
| 21 | 0.310428 |
| 22 | 0.29364 |

Without dropout, with batch normalization:

Train cost at every hundredth training step:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 2.303816 | 40 | 0.429964 |
| 1 | 1.91337 | 41 | 0.428645 |
| 2 | 1.398711 | 42 | 0.477325 |
| 3 | 1.187164 | 43 | 0.479846 |
| 4 | 1.121736 | 44 | 0.459907 |
| 5 | 1.058642 | 45 | 0.460469 |
| 6 | 0.963447 | 46 | 0.420112 |
| 7 | 0.993939 | 47 | 0.438401 |
| 8 | 0.882722 | 48 | 0.423202 |
| 9 | 0.869025 | 49 | 0.412506 |
| 10 | 0.87573 | 50 | 0.439239 |
| 11 | 0.84589 | 51 | 0.454198 |
| 12 | 0.764897 | 52 | 0.484234 |
| 13 | 0.791282 | 53 | 0.391856 |
| 14 | 0.73481 | 54 | 0.431525 |
| 15 | 0.666769 | 55 | 0.386696 |
| 16 | 0.698542 | 56 | 0.404034 |
| 17 | 0.653303 | 57 | 0.417405 |
| 18 | 0.680577 | 58 | 0.437557 |
| 19 | 0.667047 | 59 | 0.425493 |
| 20 | 0.658973 | 60 | 0.394435 |
| 21 | 0.665638 | 61 | 0.383711 |
| 22 | 0.683381 | 62 | 0.387912 |
| 23 | 0.617397 | 63 | 0.331957 |
| 24 | 0.630611 | 64 | 0.384929 |
| 25 | 0.604111 | 65 | 0.335417 |
| 26 | 0.608583 | 66 | 0.356191 |
| 27 | 0.571169 | 67 | 0.315448 |
| 28 | 0.584191 | 68 | 0.358154 |
| 29 | 0.548952 | 69 | 0.437778 |
| 30 | 0.555011 | 70 | 0.340787 |
| 31 | 0.549862 | 71 | 0.431074 |
| 32 | 0.588857 | 72 | 0.357219 |
| 33 | 0.482044 | 73 | 0.30211 |
| 34 | 0.508165 | 74 | 0.315508 |
| 35 | 0.504503 | 75 | 0.324533 |
| 36 | 0.51781 |  |  |
| 37 | 0.509374 |  |  |
| 38 | 0.527292 |  |  |
| 39 | 0.49239 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 2.238248 | 41 | 0.480519 |
| 1 | 1.735848 | 42 | 0.484735 |
| 2 | 1.39945 | 43 | 0.486609 |
| 3 | 1.155333 | 44 | 0.502392 |
| 4 | 1.104301 | 45 | 0.459333 |
| 5 | 1.057838 | 46 | 0.474964 |
| 6 | 0.978949 | 47 | 0.457673 |
| 7 | 0.996023 | 48 | 0.445621 |
| 8 | 0.889536 | 49 | 0.441675 |
| 9 | 0.866308 | 50 | 0.443978 |
| 10 | 0.834734 | 51 | 0.436239 |
| 11 | 0.85883 | 52 | 0.464891 |
| 12 | 0.772824 | 53 | 0.442018 |
| 13 | 0.762185 | 54 | 0.436358 |
| 14 | 0.752199 | 55 | 0.42531 |
| 15 | 0.729713 | 56 | 0.420592 |
| 16 | 0.71444 | 57 | 0.437718 |
| 17 | 0.695973 | 58 | 0.435028 |
| 18 | 0.692386 | 59 | 0.425185 |
| 19 | 0.688517 | 60 | 0.42217 |
| 20 | 0.699202 | 61 | 0.416233 |
| 21 | 0.673482 | 62 | 0.426973 |
| 22 | 0.661522 | 63 | 0.421847 |
| 23 | 0.640364 | 64 | 0.411728 |
| 24 | 0.636237 | 65 | 0.405351 |
| 25 | 0.616563 | 66 | 0.411447 |
| 26 | 0.598062 | 67 | 0.406404 |
| 27 | 0.594594 | 68 | 0.407207 |
| 28 | 0.590271 | 69 | 0.411935 |
| 29 | 0.581457 | 70 | 0.396942 |
| 30 | 0.572925 | 71 | 0.432959 |
| 31 | 0.54992 | 72 | 0.400284 |
| 32 | 0.55367 | 73 | 0.392505 |
| 33 | 0.540868 | 74 | 0.40113 |
| 34 | 0.53008 | 75 | 0.39085 |
| 35 | 0.541504 |  |  |
| 36 | 0.526441 |  |  |
| 37 | 0.514959 |  |  |
| 38 | 0.503397 |  |  |
| 39 | 0.494171 |  |  |
| 40 | 0.499734 |  |  |

Validation cost at every hundredth training step:

With dropout, without batch normalization:

Train cost at every hundredth training step:

|  |  |
| --- | --- |
| 0 | 2.3023 |
| 1 | 1.8427 |
| 2 | 1.4104 |
| 3 | 0.8567 |
| 4 | 0.7669 |
| 5 | 0.6849 |
| 6 | 0.5972 |
| 7 | 0.7102 |
| 8 | 0.5014 |
| 9 | 0.4612 |
| 10 | 0.4506 |
| 11 | 0.4371 |
| 12 | 0.4472 |
| 13 | 0.4265 |
| 14 | 0.4063 |
| 15 | 0.4000 |
| 16 | 0.3755 |
| 17 | 0.3684 |
| 18 | 0.3521 |
| 19 | 0.3507 |
| 20 | 0.3544 |
| 21 | 0.3418 |
| 22 | 0.3328 |
| 23 | 0.3237 |
|  |  |

Validation cost at every hundredth training step:

|  |  |
| --- | --- |
| 0 | 2.301746 |
| 1 | 1.908302 |
| 2 | 1.582575 |
| 3 | 1.134884 |
| 4 | 0.923005 |
| 5 | 0.789325 |
| 6 | 0.707162 |
| 7 | 0.656775 |
| 8 | 0.557648 |
| 9 | 0.583395 |
| 10 | 0.496129 |
| 11 | 0.45377 |
| 12 | 0.45299 |
| 13 | 0.540898 |
| 14 | 0.396508 |
| 15 | 0.384217 |
| 16 | 0.375366 |
| 17 | 0.365486 |

With dropout, with batch normalization:

Train cost at every hundredth training step:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 2.313603 | 41 | 0.658543 |
| 1 | 1.62804 | 42 | 0.586342 |
| 2 | 1.348924 | 43 | 0.619272 |
| 3 | 1.319491 | 44 | 0.572216 |
| 4 | 1.294935 | 45 | 0.580883 |
| 5 | 1.208565 | 46 | 0.578458 |
| 6 | 1.173591 | 47 | 0.594751 |
| 7 | 1.207909 | 48 | 0.5818 |
| 8 | 1.085703 | 49 | 0.559022 |
| 9 | 1.083743 | 50 | 0.50977 |
| 10 | 1.101592 | 51 | 0.558622 |
| 11 | 0.939393 | 52 | 0.537563 |
| 12 | 0.966971 | 53 | 0.622783 |
| 13 | 1.047765 |  |  |
| 14 | 0.911651 |  |  |
| 15 | 0.879397 |  |  |
| 16 | 0.936079 |  |  |
| 17 | 0.836693 |  |  |
| 18 | 0.789876 |  |  |
| 19 | 0.841867 |  |  |
| 20 | 0.732771 |  |  |
| 21 | 0.733228 |  |  |
| 22 | 0.709112 |  |  |
| 23 | 0.777478 |  |  |
| 24 | 0.737935 |  |  |
| 25 | 0.777339 |  |  |
| 26 | 0.724041 |  |  |
| 27 | 0.727299 |  |  |
| 28 | 0.730436 |  |  |
| 29 | 0.697491 |  |  |
| 30 | 0.673578 |  |  |
| 31 | 0.681525 |  |  |
| 32 | 0.71068 |  |  |
| 33 | 0.662112 |  |  |
| 34 | 0.697638 |  |  |
| 35 | 0.628272 |  |  |
| 36 | 0.750395 |  |  |
| 37 | 0.66016 |  |  |
| 38 | 0.675069 |  |  |
| 39 | 0.678057 |  |  |
| 40 | 0.661687 |  |  |

Validation cost at every hundredth training step:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 2.279029 | 41 | 0.646176 |
| 1 | 1.567996 | 42 | 0.621111 |
| 2 | 1.379224 | 43 | 0.61165 |
| 3 | 1.303258 | 44 | 0.608445 |
| 4 | 1.238453 | 45 | 0.61026 |
| 5 | 1.221512 | 46 | 0.598354 |
| 6 | 1.200537 | 47 | 0.597091 |
| 7 | 1.139931 | 48 | 0.595117 |
| 8 | 1.085335 | 49 | 0.611531 |
| 9 | 1.125545 | 50 | 0.611473 |
| 10 | 1.076921 | 51 | 0.57706 |
| 11 | 1.011548 | 52 | 0.575673 |
| 12 | 0.975963 | 53 | 0.560878 |
| 13 | 0.974066 |  |  |
| 14 | 0.933226 |  |  |
| 15 | 0.932282 |  |  |
| 16 | 0.891232 |  |  |
| 17 | 0.863555 |  |  |
| 18 | 0.838626 |  |  |
| 19 | 0.853913 |  |  |
| 20 | 0.812848 |  |  |
| 21 | 0.793571 |  |  |
| 22 | 0.771078 |  |  |
| 23 | 0.760639 |  |  |
| 24 | 0.758922 |  |  |
| 25 | 0.759173 |  |  |
| 26 | 0.738044 |  |  |
| 27 | 0.733735 |  |  |
| 28 | 0.723708 |  |  |
| 29 | 0.726207 |  |  |
| 30 | 0.717037 |  |  |
| 31 | 0.721666 |  |  |
| 32 | 0.748228 |  |  |
| 33 | 0.681034 |  |  |
| 34 | 0.688931 |  |  |
| 35 | 0.674521 |  |  |
| 36 | 0.684695 |  |  |
| 37 | 0.682406 |  |  |
| 38 | 0.653937 |  |  |
| 39 | 0.645623 |  |  |
| 40 | 0.635651 |  |  |

Comparing the four variants:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variant | Without dropout, without batch normalization | Without dropout, with batch normalization | With dropout, without batch normalization | With dropout, with batch normalization |
| # iterations | **2299** | 7580 | 3300 | 5346 |
| # epochs | **49** | 162 | 71 | 114 |
| Batch size | 1024 | 1024 | 1024 | 1024 |
| Time (seconds) | **288** | 632 | 338 | 403 |
| Train accuracy | **0.9162** | 0.9050 | 0.9256 | 0.8621 |
| Validation accuracy | **0.9095** | 0.8936 | 0.9093 | 0.8542 |
| Test accuracy | **0.9105** | 0.8933 | 0.9080 | 0.8518 |

Modifications for dropout:

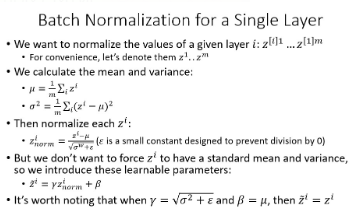
We implemented dropout in the linear\_forward function, as part of the forward propagation phase. With a probability of 0.25, each neuron from the previous layer may be dropped. In practice, this is done by multiplying the activation vector of the previous layer with a vector comprised of ones and zeros with about a 75/25 split.

We don’t apply dropout to the inputs and to the last layer before the output layer.

In order to apply Batchnorm

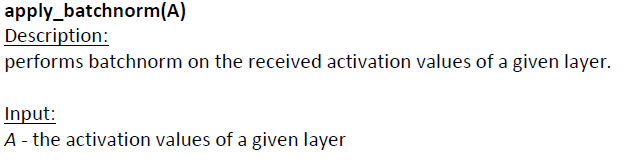
We passed the flag “use\_batchnorm” to **L\_model\_forward** and activated the batchnorm after the activation function (**linear\_activation\_forward**). Moreover, in order to pass use\_batchnorm flag to L\_Model\_Forward,we have added use\_batchnorm param to the signature of “**L\_layer\_model**” & “**Predict”**, in order to provide it to L\_model\_forward.

It is important to note that in class we learnt that batch normalization normally being used over Z, before the activation function,



While the assignment described





In order to apply batchnorm as described above, there is a need of applying batchnorm at “Linear\_forward” function, just after calculating Z.

Auxiliary Functions:

**def create\_dataset(X, Y , num\_of\_classes, test\_size = 0.2)**

* **Split dataset to validation and train sets,**
* **Reshapes images into 28\*28 pixels**

**def get\_strongest\_index(AL, index):**

* **Returns the strongest index within the classes,**

**e.g. [0,0,0,0,0,0,0,0,0,1] will return 10 as the strongest index.**

**Will play major role in the prediction of the class.**

**def initalize\_weights(dim\_size, input\_length):**

* **Init weights with random values**

**def initalize\_bias(dim\_size):**

* **Init bias as a zero np array.**