207547183
206477952

Assignment 1 - Deep Learning

This report evaluates the design and performance of a neural network trained to classify handwritten digits from the MNIST dataset, comparing models with and without L2 regularization and batch normalization.
Implementation:
The network uses a Gaussian distribution to initialize weights while the biases are initialized to zero to ensure smoother convergence.
The architecture consists of three hidden layers with 20, 7, and 5 neurons. These layers utilize the ReLU activation function, introducing nonlinearity that allows the network to learn complex patterns.
The output layer has 10 neurons, each representing a digit class, and a SoftMax activation function that generates probabilities, with the highest probability selected as the network's prediction.
The MNIST images were aligned to the range [0,1] and flattened into 1D vectors before fed into the network. The l_layer_model function implements a deep neural network training pipeline. Training stops when there is little to no improvement in the validation cost for 100 consecutive training steps, indicating convergence.
The L2 regularization is then added to the cost function, to penalize large weights, simplify the model, and improve its ability to handle unseen data. When batch normalization is enabled, it normalizes the input to each layer across mini-batches. This reduces internal variable shifts, stabilizes training, and can lead to faster convergence.

Results and analysis of findings:

- Experiment number 1: without L2 regularization and without batch normalization:

```
Final Training Cost: 0.4121646285469284
Final Validation Cost: 0.4676450678077275
Accuracy on validation: 0.876
Epoch: 69, Iterations: 13100
Accuarcy on train:  0.88355
Accuarcy on test:  0.8822
The time it takes: 48.68 seconds
```

This configuration does not utilize batch normalization or L2 regularization, relying solely on the model's architecture and optimization process. Training took 48.6 seconds, completing after 69 epochs and 13,100 batches. The model achieved high accuracy, with approximately 88% accuracy on the training, validation, and test datasets. While the results indicate good performance, the absence of regularization and normalization limits the model's ability to generalize further, resulting in slightly lower accuracy compared to more advanced configurations.

- Experiment number 2: with batch normalization, without L2 regularization:

```
Final Training Cost: 0.5025023785429055
Final Validation Cost: 0.4821833730656054
Accuracy on validation: 0.88925
Epoch: 43, Iterations: 8200

Accuarcy on train:  0.8977166666666667
Accuarcy on test:  0.896
The time it takes: 36.95 seconds
```

This configuration incorporates batch normalization into each batch to improve the convergence speed of the training process but excludes L2 regularization. Training was significantly faster, completing in 36.95 seconds over 43 epochs and 8,200 batches, and the model achieved 89% accuracy on the training, validation, and testing datasets. While batch normalization improved efficiency and stabilized convergence. Compared to Experiment 1, accuracy showed a slight improvement, but the most notable gain was the faster convergence time, making this configuration a more efficient alternative.

- Experiment number 3: with L2 regularization, without batch normalization.

```
Final Training Cost: 0.5946603746012806
Final Validation Cost: 0.6170265370224594
Accuracy on validation:  0.8493333333333334
Epoch: 99, Iterations: 18801

Accuracy on train:  0.85385
Accuracy on test:  0.8522
The time it takes: 72.0 seconds
```

This configuration implements L2 regularization to constrain weights and reduce overfitting but does not use batch normalization. Training took the longest, 72 seconds, and finished after 99 epochs and 18,801 samples. The model achieved approximately 85% accuracy on the training, validation, and testing datasets. In this experiment, the optimization is less stable, resulting in slower convergence and lower overall performance compared to the other configurations.

- Experiment number 4: With L2 Regularization combined with batch normalization:

```
Final Training Cost: 0.559853803834573
Final Validation Cost: 0.32641749218004656
Accuracy on validation: 0.9234166666666667
Epoch: 74, Iterations: 14100
Accuarcy on train:  0.9304666666666667
Accuarcy on test:  0.9246
The time it takes: 62.01 seconds
```

This configuration aims to leverage the combined benefits of batch normalization and L2 regularization. Training took 62.01 seconds, completing after 74 epochs and 14,100 batches. The model achieved the highest performance, with approximately 93.05% accuracy on the training set, 92.34% accuracy on the validation set, and 92.46% accuracy on the test set. The results demonstrate the complementary effects of L2 regularization in reducing overfitting and batch normalization in stabilizing training and accelerating convergence. This setup achieved the best balance of accuracy and efficiency.

- Summary table:

	Experiment	Training Cost	Validation Cost	Train Accuracy (%)	Validation Accuracy (%	Test Accuracy (%)	Training Time (s)	Epochs	Iterations
1	Experiment 1: Without L2, Without Batch Norm	0.4122	0.4676	88.36	87.6	88.22	48.68	69	13100
2	Experiment 2: With Batch Norm, No L2	0.5025	0.4822	89.77	88.93	89.6	36.95	43	8200
3	Experiment 3: With L2, Without Batch Norm	0.5947	0.617	85.39	84.93	85.22	72.0	99	18801
4	Experiment 4: With L2 and Batch Norm	0.5599	0.3264	93.05	92.34	92.46	62.01	74	14100

- Comparing the weight values of the architecture with and without L2 norm functionality:

In this comparison, we expect the weights in the model with L2 regularization to be smaller because L2 penalizes models with larger weights.

	Layer	Experiment 1 (Without L2,	Experiment 3 (With L2,
1	Layer 1	0.01926	0.01991
2	Layer 2	0.22563	0.24582
3	Layer 3	0.43374	0.41696
4	Layer 4	0.49053	0.42779

- The code file contains the code that prints the average weights in each layer in the l_layer_model function.

Experiment 1 (Without L2): In the shallower layers, the weights are already small, so there is minimal risk of overfitting. However, in the deeper layers, the weights grow significantly larger (e.g., Layer 4: 0.49053), which can lead to overfitting due to the absence of constraints like L2 regularization.

Experiment 3 (With L2): While L2 regularization effectively penalizes large weights, the weights in the shallower layers (e.g., Layer 1: 0.01991) remain small and are not significantly affected, as there is no "punishment" for already small weights. In the deeper layers, however, L2 regularization successfully reduces weight magnitudes (e.g., Layer 4: 0.42779 compared to 0.49053 in Experiment 1), demonstrating its effectiveness in controlling overfitting where it is most needed.

207547183
206477952

- Short explanation about the changes done in the code:

Cross-Entropy Cost: Now includes an L2 regularization term to penalize large weights.

Gradients: Gradients of weights (dW) are adjusted with the L2 penalty during backpropagation, guiding the optimizer to reduce weight magnitudes.

1. compute_cost Function

- **Original**: The function calculated only the **categorical cross-entropy cost**.
- **Changes**:
  - An additional term for the **L2 penalty** was added to the total cost.
  - A loop iterates through the params dictionary to sum the squared weights across all layers.
  - The penalty is scaled by $\frac{\lambda}{2m}$ where m is the number of examples and λ is the regularization strength.

2. l_model_backward Function

- **Original**: The function calculated the gradients of the weights, biases, and activations without considering L2 regularization.
- **Modified**:
  - An L2 penalty term was added to the gradient of the weights - dW.
  - The penalty is computed as $\frac{\lambda}{m}W$, where m is the number of examples, and added to dW for each layer.