Uralp Ergin 2448355 Part3 Questions

1) **What type of measure or measures have you considered to prevent overfitting?**

I tuned the hypermaraters which are hidden layer number, hidden layer size, learning rate, epoch number , activation function.

The hyperparameter set I consider was

num_hidden_layers = [1, 2, 3]

hidden_size = [64, 128, 256]

learning_rates = [0.001, 0.01, 0.1]

num_epochs = [10, 20, 30]

activations = ['relu', 'tanh', 'sigmoid']

and monitor the loss values while running.

2) **How could one understand that a model being trained starts to overfit?**
   - Increasing validation loss: Monitor the validation loss during training. Overfitting is indicated if it begins to increase while the training loss keeps decreasing.
   - Decreasing validation accuracy : Overfitting is indicated if, like with validation loss, validation accuracy begins to decline while training accuracy keeps rising. A common sign is high training accuracy paired with low validation accuracy.

3) **Could we get rid of the search over the number of iterations (epochs) hyperparameter by setting it to a relatively high value and doing some additional work? What may this additional work be?**

   Yes, we can.

   Early Stopping: Throughout training, it's critical to keep an eye on the model's performance on a validation set. putting early stopping into practice, which halts training when the validation loss or another selected metric begins to decline. In the process, training time is reduced and the model is kept from overfitting the training set.

   Learning Rate Scheduling: Putting learning rate schedules into place to adjust the learning rate as training goes on, such as learning rate decay. By doing so, one may keep the model's convergence steady and stop it from going above its ideal weight range.

4) **Is there a ”best” learning rate value that outperforms the other tested learning values in all hyperparameter configurations?**

No, there isn't. The efficiency of the learning rate depends on other configurations and parameters. Different learning rates can outperforms others in different configurations.

5) **Is there a "best" activation function that outperforms the other tested activation functions in all hyperparameter configurations?**
No, there isn't. The efficiency of the activation function depends on other configurations and parameters. Different activation functions can outperforms others in different configurations

6) **What are the advantages and disadvantages of using a small learning rate?**

Advantages:
Stability: A training procedure that is more steady and predictable can result from a modest learning rate. It lessens the possibility of significant weight updates leading to oscillations or divergences in the loss.

Improved Generalization: Because the model is trained more slowly at smaller learning rates, it is able to catch important patterns in the data instead of fitting noise, which frequently leads to greater generalization.

Robustness: Small learning rates can withstand noisy or poorly conditioned data better and are less susceptible to the choice of initial weights.

Disadvantages:

Slower Convergence: Training with a small learning rate requires more iterations to reach convergence, which can significantly increase training time, especially for large datasets or complex models.

Risk of Getting Stuck: If the learning rate is too small, the optimization process can get stuck in a local minimum, or it may take an excessively long time to escape a saddle point.

7) **What are the advantages and disadvantages of using a big learning rate?**

Advantages:

Faster Convergence: Faster learning and faster convergence to a solution are the outcomes of a high learning rate for the model. When it comes to training time, this can be helpful.

Escaping Local Minima: A large learning rate can facilitate the optimizer's escape from saddle points and local minima, allowing the model to locate better minima in the loss landscape.

Disadvantages:

Poor Generalization: A big learning rate can cause the model to overfit the training data, as it quickly adapts to the noise and specifics of the training set, resulting in poor generalization to unseen data.

Sensitivity to Initialization: Models with large learning rates are more sensitive to the choice of initial weights and may require more careful initialization.

8) **Is it a good idea to use stochastic gradient descent learning with a very large dataset? What kind of problem or problems do you think could emerge?**

Large datasets can benefit greatly from the practical and effective use of SGD, particularly when parallelization is feasible and CPU resources are constrained.
Problems that may arise include:

Slow Convergence: Because each update is based on a small subset of the data, training with a very big dataset using SGD may take longer to converge than training with batch gradient descent. To reach convergence, lots of epochs might be needed.

Increased Variability: The training process may become more variable due to SGD's stochastic character. Greater swings in the loss may make it more difficult to track and manage the training.

9) **In the given source code, the instance features are divided by 255 (Please recall that in a gray scale-image pixel values range between 0 and 255). Why may such an operation be necessary? What would happen if we did not perform this operation? (Hint: These values are indirectly fed into the activation functions (e.g sigmoid, tanh) of the neuron units. What happens to the gradient values when these functions are fed with large values?)**

This operation is necessary for several reasons:

Scale Invariance: More scale-invariance can be achieved in the neural network by scaling the pixel values to the range [0, 1]. This implies that minor changes in pixel values won't have a big impact on the functionality of the network. Instead of being sensitive to changes in pixel intensity, it enables the network to concentrate on the patterns and characteristics in the image.

Compatibility with Activation Functions: Certain activation functions, such the tanh and sigmoid, are more affected by the size of their inputs. Application of these functions is facilitated by normalizing inputs to a specified range (e.g., [0, 1] or [-1, 1] for tanh).

If we don’t perform this operation, several issues may arise:

Unstable Training: The network may undergo slower and less stable convergence in the absence of normalization. It can cause explosion gradients, which are extremely large gradients that make it difficult to train the network.

Inefficient Optimization: To reach the required degree of accuracy, additional training iterations may be needed if the optimization procedure is less effective.

If we supplied high values to the activation functions: For high inputs, some activation functions have diminishing gradients.