CENG 499 Report

Ismail Karabas (2375186)

November 9, 2023

In my algorithm, I designed a series of nested for loops to systematically train the model using various combinations of hyperparameters. For each combination, I evaluated both accuracy and confidence intervals, displaying the results. Additionally, I identified the optimal set of hyperparameters and conducted a final test on the dataset.

1 Results of part3.py

$Hyperparameters$: "$learning_rate$" : 0.01, "$hidden_layer_count$" : 1, "$neuron_count$" : 16, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 82.0328131252501, $ConfidenceInterval$: (81.77999780130453, 82.28562844919567)

$Hyperparameters$: "$learning_rate$" : 0.01, "$hidden_layer_count$" : 1, "$neuron_count$" : 16, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 85.31812725090035, $ConfidenceInterval$: (85.09267589495774, 85.54357860684296)

$Hyperparameters$: "$learning_rate$" : 0.01, "$hidden_layer_count$" : 1, "$neuron_count$" : 32, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 70.49819927971188, $ConfidenceInterval$: (66.82710458996176, 74.169293969462)

$Hyperparameters$: "$learning_rate$" : 0.01, "$hidden_layer_count$" : 1, "$neuron_count$" : 32, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 79.98499399759905, $ConfidenceInterval$: (78.3959330486501, 81.574054946548)

$Hyperparameters$: "$learning_rate$" : 0.01, "$hidden_layer_count$" : 2, "$neuron_count$" : 16, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 15.386154461784713, $ConfidenceInterval$: (12.140099866672628, 18.632209056896798)

$Hyperparameters$: "$learning_rate$" : 0.01, "$hidden_layer_count$" : 2, "$neuron_count$" : 16, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 77.89315726290516, $ConfidenceInterval$: (76.2563810814735, 79.52993344433682)

$Hyperparameters$: "$learning_rate$" : 0.01, "$hidden_layer_count$" : 2, "$neuron_count$" : 32, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 11.15546218487395, $ConfidenceInterval$: (11.000560224089638, 11.310364145658264)

$Hyperparameters$: "$learning_rate$" : 0.01, "$hidden_layer_count$" : 2, "$neuron_count$" : 32, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 76.91276510604243, $ConfidenceInterval$: (75.10301842508, 78.72251178700486)

$Hyperparameters$: "$learning_rate$" : 0.05, "$hidden_layer_count$" : 1, "$neuron_count$" : 16, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$:

75.04301720688275, $ConfidenceInterval$: (74.17946020131404, 75.90657421245146)
$Hyperparameters$: "$learning_rate$" : 0.05, "$hidden_layer_count$" : 1, "$neuron_count$" : 16, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 70.48419367747098, $ConfidenceInterval$: (66.5530156904458, 74.41537166449616)
$Hyperparameters$: "$learning_rate$" : 0.05, "$hidden_layer_count$" : 1, "$neuron_count$" : 32, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 70.80032012805124, $ConfidenceInterval$: (69.76069016992817, 71.83995008617431)
$Hyperparameters$: "$learning_rate$" : 0.05, "$hidden_layer_count$" : 1, "$neuron_count$" : 32, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 40.36114445778311, $ConfidenceInterval$: (37.92132980407502, 42.8009591114912)
$Hyperparameters$: "$learning_rate$" : 0.05, "$hidden_layer_count$" : 2, "$neuron_count$" : 16, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 13.649459783913567, $ConfidenceInterval$: (11.454528944636557, 15.844390623190577)
$Hyperparameters$: "$learning_rate$" : 0.05, "$hidden_layer_count$" : 2, "$neuron_count$" : 16, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 77.79111644657863, $ConfidenceInterval$: (75.57000753668456, 80.0122253564727)
$Hyperparameters$: "$learning_rate$" : 0.05, "$hidden_layer_count$" : 2, "$neuron_count$" : 32, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 10.568227290916365, $ConfidenceInterval$: (10.07289731224161, 11.06355726959112)
$Hyperparameters$: "$learning_rate$" : 0.05, "$hidden_layer_count$" : 2, "$neuron_count$" : 32, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 49.64085634253702, $ConfidenceInterval$: (37.67844299227785, 61.60326969279619)
$Hyperparameters$: "$learning_rate$" : 0.1, "$hidden_layer_count$" : 1, "$neuron_count$" : 16, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 64.55682272909164, $ConfidenceInterval$: (60.29511071056075, 68.81853474762252)
$Hyperparameters$: "$learning_rate$" : 0.1, "$hidden_layer_count$" : 1, "$neuron_count$" : 16, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 60.0860344137655, $ConfidenceInterval$: (48.91224549113021, 71.2598233364008)
$Hyperparameters$: "$learning_rate$" : 0.1, "$hidden_layer_count$" : 1, "$neuron_count$" : 32, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 58.74549819927971, $ConfidenceInterval$: (57.709560589795146, 59.78143580876427)
$Hyperparameters$: "$learning_rate$" : 0.1, "$hidden_layer_count$" : 1, "$neuron_count$" : 32, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 52.163865546218496, $ConfidenceInterval$: (49.32164039014093, 55.00609070229606)
$Hyperparameters$: "$learning_rate$" : 0.1, "$hidden_layer_count$" : 2, "$neuron_count$" : 16, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 11.084433773509403, $ConfidenceInterval$: (10.053037201641093, 12.115830345377713)
$Hyperparameters$: "$learning_rate$" : 0.1, "$hidden_layer_count$" : 2, "$neuron_count$" : 16, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 60.27010804321729, $ConfidenceInterval$: (55.13065567014954, 65.40956041628505)
$Hyperparameters$: "$learning_rate$" : 0.1, "$hidden_layer_count$" : 2, "$neuron_count$" : 32, "$activation_function$" : $Sigmoid()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 11.000400160064027, $ConfidenceInterval$: (10.691391500956703, 11.30940881917135)
$Hyperparameters$: "$learning_rate$" : 0.1, "$hidden_layer_count$" : 2, "$neuron_count$" : 32, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20 $MeanValidationAccuracy$: 31.44357743097239, $ConfidenceInterval$: (22.43285743278536, 40.45429742915942)

Best One and its results on test $BestHyperparameters$: "$learning_rate$" : 0.01, "$hidden_layer_count$" : 1, "$neuron_count$" : 16, "$activation_function$" : $Tanh()$, "$epoch_count$" : 20$MeanTestAccuracy$: 64.659, $ConfidenceInterval$: (57.31623584558392, 72.0017641544161)

2 Measures to prevent overfitting

1. During the training phase, we exclusively employed the training dataset, while for hyperparameter tuning, we relied on the validation dataset. Subsequently, the testing phase involved the dedicated testing dataset. This meticulous separation of data at each stage was implemented to prevent memorization in the model.

2. I implemented L2 regularization, a technique that imposes a penalty on large weights in the model. By discouraging the presence of excessively large weights, L2 regularization fosters a more generalized model. This emphasis on weight moderation contributes significantly to preventing overfitting.

3 Detection of overfitting

The indication of overfitting emerges when the training accuracy demonstrates continuous improvement, yet the validation accuracy begins to decline. This phenomenon signifies that the model has surpassed its optimal learning point and is now memorizing the training data. As a consequence, the model loses its ability to generalize well to unseen data. The diminishing accuracy observed on the validation dataset serves as significant evidence of this overfitting,

4 Eliminating Search Over Epochs

Adding early stopping to our training plan would be a good idea. Instead of sticking to a strict rule, like stopping after the last 10 tries if there is no improvement. I suggest being more flexible. We'll keep an eye on how well the model is doing on a separate set of data. If we don't see much improvement over time, even with more tries, we'll stop training. This careful early stopping helps us avoid ending training too soon or too late. It lets the models show improvement, even if it takes a bit longer. This plan helps us find a balance between saving time and making sure our models are in the best shape possible.

5 Best Learning Rate

There isn't a universally "best" learning rate, as it depends on the specific problem and model architecture. I have tested multiple learning rates (0.01) to find a balance. Very small rates can lead to slow convergence, while very large rates may cause oscillations or divergence.

As you can see, in the following graph we don't have a best learning rate that is best for all the combinations. If that was true there we no need to do grid search

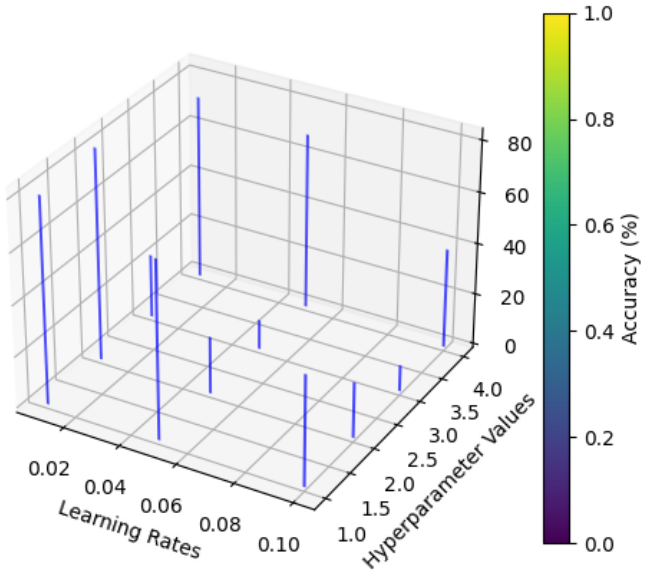

Figure 1: UDP

6 Best Activation Function

Selecting the most suitable activation function depends on the problem and model architecture. Just like learning rate case , there is not a fit-all perfect activation function. I explored the effectiveness of activation functions, including Sigmoid and Tanh, to find the optimal choice for the given context. Different functions may exhibit better performance based on the nature of the data or the architectural nuances.

7 Advantages and Disadvantages of Small Learning Rate

Smaller learning rates can lead to more stable convergence and better generalization, especially in complex models. Training might be slower, and the model could get stuck in local minima.

8 Advantages and Disadvantages of Large Learning Rate

Faster convergence, quicker training times. Risk of overshooting the optimal weights, divergence, and instability. It might also skip over the minimum if the learning rate is too large.

9 Use of SGD with a Large Dataset

Utilizing standard SGD with an extensive dataset can pose computational inefficiencies. Adopting Mini-batch SGD, where data subsets are randomly sampled per iteration, enhances training speed while offering a reliable gradient approximation.

10 Normalization of Pixel Values

The normalization technique of dividing pixel values by 255 is vital for numerical stability, particularly when employing activation functions like sigmoid and tanh. This method safeguards against challenges such as vanishing gradients during the training process.