Settings: Mac running macOS Sonoma 14.1.1, Python 3.10.4 64-bit, Anaconda virtual environment; only recorded results from last epoch

Run	Learning Rate	Batch Size	Epochs	Dropout Rate	Hidden Layer Size	Training Loss	Training Accuracy	val_loss	val_accuracy	(Test) Loss	(Test) Accuracy
Control (Default)	0.001	32	10	0.2	64	0.4283	0.8826	0.2674	0.9265	0.3089	0.9137
1	0.001	32	10	0.5	64	0.4553	0.8733	0.2652	0.9242	0.3187	0.9126
2	0.001	32	10	0.2	16	0.4490	0.8704	0.2568	0.9288	0.3043	0.9177
3	0.001	32	10	0.2	32	0.4308	0.8761	0.2600	0.9275	0.3158	0.9137
4	0.001	32	10	0.2	128	0.4793	0.8816	0.3194	0.9172	0.3683	0.9072
5	0.001	32	10	0.2	256	0.6298	0.8790	0.4667	0.9002	0.5460	0.8907
6	0.001	32	10	0.4	16	0.5086	0.8537	0.2671	0.9282	0.3274	0.9108
7	0.001	32	10	0.1	32	0.4031	0.8865	0.2482	0.9302	0.2968	0.9208
8	0.001	32	10	0.3	128	0.4811	0.8799	0.3027	0.9170	0.3514	0.9090
9	0.001	32	10	0.0	256	0.5811	0.8906	0.5451	0.9017	0.6608	0.8898
10	0.001	32	10	0.5	256	0.7067	0.8688	0.4317	0.9060	0.5216	0.8994

Analysis:

In this assignment, I decided to observe the effects of changing the hyperparameters “dropout rate” and “hidden layer size”. To begin, I changed either the former or latter as the sole hyperparameter to observe the effects of each (runs 1 and 2). From there, I altered just the “hidden layer size” in runs 2-5 and then included “dropout rate” in runs 6-10. From the data provided above and the control condition ran with the default settings, we can draw the following conclusions:

Run 1 - Impact of increasing dropout rate:
In Run 1, increasing the dropout rate to 0.5 resulted in a slight increase in training loss and a decrease in training accuracy, suggesting that there may be underfitting. Additionally, the test accuracy slightly decreased, but the validation accuracy remained relatively stable.

Runs 2 and 3 - Reducing hidden layer size:
Comparing Runs 2 and 3 with the control, we see that reducing the hidden layer size to 16 in Run 2 marginally affected training loss and accuracy, but slightly improved validation accuracy and test accuracy.

When we changed the hidden layer size to 32 (half of the default settings) in Run 3, we maintained a training loss and accuracy close to the control run and had no negative affect on validation and test metrics.

Runs 4 and 5 - Increasing hidden layer size:
In Run 4, the hidden layer size of the default settings was doubled to 128, resulting in increased training loss and decreased validation accuracy, suggesting the onset of overfitting. Test accuracy slightly dropped compared to the control.

When the settings of Run 4 were doubled for Run 5, a hidden layer size of 256 significantly increased training loss and reduced both validation and test accuracy, reinforcing the overfitting trend.

Runs 6-10 - Combining dropout rate with hidden layer size variations:
Run 6 had a dropout of 0.4 and a smaller hidden layer size (16), showing increased training loss but maintained high validation and test accuracy, which may indicate a good balance between capacity and regularization.

Run 7, with a dropout of 0.1 and a hidden layer size of 32, yielded the lowest validation loss and maintained high test accuracy, suggesting that less regularization is needed for smaller network sizes.

Run 8 with a dropout rate of 0.3 and a larger hidden layer size (128) demonstrated an increase in validation loss compared to the control, hinting that this level of regularization may not be sufficient for larger networks.

Removing dropout entirely (Run 9) with the largest hidden layer (256) led to a large increase in validation and test loss, indicating significant overfitting.

Finally, Run 10 with a high dropout rate (0.5) and the largest hidden layer size (256) also showed substantial increases in training and validation loss, and a decrease in test accuracy, which suggests that even a high dropout rate may not prevent overfitting in a very large network.

Conclusion:
Overall, there is a consistent trend where a larger hidden layer size and an increased network capacity without proper regularization leads to overfitting as evidenced by increased training loss and decreased validation and test accuracy. Through altering the dropout rate, we see that it has a nuanced effect on network performance where too much can result in underfitting while too little can lead to overfitting, particularly in larger networks (e.g. larger hidden layer size).

From these general observations, I found it quite interesting how much fine tuning there needs to be in order for a neural network to work well/similar to an actual human brain. For example, one of the main concepts that has been reinforced throughout my pre-medical career is that an increase in the density of neurons lead to faster responses. In general, more neurons = better. If we were to directly cross-apply this concept to AI neural networks, that would mean that a larger hidden layer size would result in the highest test accuracy and lowest test loss. However, based on the data and observations above, we see that this is not the case.

As such, this analysis highlights the importance of hyperparameter tuning to achieve good generalization. Especially given that Run 7 performed the best in terms of generalization to validation and test sets, we can conclude that a balanced approach to network capacity and regularization is critical to avoid overfitting and maintain the network's ability to learn from training data.