Activation Function	**Accuracy**	**Exec Time**
Identity (Linear)	0.9101	199.73
ReLU (Non-linear)	0.9737	214.04
Sigmoid	0.9729	205.44

Optimizers	**Accuracy**	**Exec Time**
Baseline (Adam)	0.9735	207.98
SGD	0.9708	188.46
RMSprop	0.9757	193.84
Adagrad	0.9753	219.26

Regularization	**Accuracy**	**Exec Time**
Baseline	0.9758	222.40
Batch Norm	0.9783	199.39
Dropout	0.9472	208.92
Weighted Init	0.9671	208.29

Analysis:

For activation functions, both non-linear ones have significantly better accuracy than the linear activation function, yet higher executions time. Due to the fact that linear activation function will effectively negate the purpose of having multiple layers, and yield a simple y = W*X+B model, the accuracy suffers as well. However, non-linear activation functions are more expensive computationally, thus the higher execution time.

For Optimizers, all optimizers converged with similar accuracy. SGD, unsurprisingly, has the best execution time, due to the fact it only picks a random sample instead of the entirety of the dataset. Therefore, it's missed the best local minimum by a bit, resulting in slightly worse accuracy. RMSprop, like Rprop, uses the second momentum to speed up Adagrad, therefore resulting in a faster execution time than adagrad. However, compared to Adam, it doesn't have the ability to adept to gradient in different directions, yet we don't see the effect of it in this dataset/model. Adagrad uses the square of gradient summed, which is a costly calculation, resulting in the slowest execution time.

Batch Norm reduces the covariate shift and speeds up the execution time. (Sergey Ioffe, Christian Szegedy, [cs.LG] 2 Mar 2015) It ensures that the mean and var of the parameters, no matter how they update, will stay the same.

Dropout layer reduces the size of the network by turning of neurons by random. Thus, the execution time is naturally lower. However, that also causes a lower accuracy as trade off.

Weighted initialization is supposed to help with preventing exploding or vanishing gradient, thus making the training time faster (converge faster). Therefore, it also has a lower execution time than baseline.