1. Create a neural network with at least two hidden layers for a classification task. The dataset could be chosen arbitrarily, e.g. MNIST.

Output:

```
Epoch [1/5], Loss: 1.0660
Epoch [2/5], Loss: 0.3901
Epoch [3/5], Loss: 0.3255
Epoch [4/5], Loss: 0.2914
Epoch [5/5], Loss: 0.2659
```

```
Baseline Model Accuracy: 92.78%, Execution Time: 30.89 seconds
```

2. Experiment with three activation functions (one linear and one nonlinear) and report (i) accuracy and (ii) execution time.

Output:

```
Epoch [1/5], Loss: 0.8168
Epoch [2/5], Loss: 0.3669
Epoch [3/5], Loss: 0.3285
Epoch [4/5], Loss: 0.3131
Epoch [5/5], Loss: 0.3042
Activation Function: Identity, Accuracy: 91.72%, Execution Time: 29.47 seconds
Epoch [1/5], Loss: 1.0329
Epoch [2/5], Loss: 0.3847
Epoch [3/5], Loss: 0.3233
Epoch [4/5], Loss: 0.2915
Epoch [5/5], Loss: 0.2675
Activation Function: ReLU, Accuracy: 92.37%, Execution Time: 29.51 seconds
Epoch [1/5], Loss: 1.0227
Epoch [2/5], Loss: 0.4168
Epoch [3/5], Loss: 0.3323
Epoch [4/5], Loss: 0.2932
Epoch [5/5], Loss: 0.2665
Activation Function: Tanh, Accuracy: 92.60%, Execution Time: 29.37 seconds

Summary of Results:
Activation: Identity, Accuracy: 91.72%, Execution Time: 29.47 seconds
Activation: ReLU, Accuracy: 92.37%, Execution Time: 29.51 seconds
Activation: Tanh, Accuracy: 92.60%, Execution Time: 29.37 seconds
```

Activation Function	Accuracy	Exec Time
Identity	91.72%	29.47
ReLU	91.72%	29.51
Tanh	92.60%	29.37

3. Experiment with three optimizers and report (i) accuracy and (ii) execution time.

Output:

```
Epoch [1/5], Loss: 0.9902
Epoch [2/5], Loss: 0.3823
Epoch [3/5], Loss: 0.3236
Epoch [4/5], Loss: 0.2884
Epoch [5/5], Loss: 0.2623
Optimizer: SGD, Accuracy: 92.33%, Execution Time: 30.71 seconds
Epoch [1/5], Loss: 0.4109
Epoch [2/5], Loss: 0.2005
Epoch [3/5], Loss: 0.1468
Epoch [4/5], Loss: 0.1151
Epoch [5/5], Loss: 0.0995
Optimizer: Adam, Accuracy: 96.83%, Execution Time: 43.21 seconds
Epoch [1/5], Loss: 0.3818
Epoch [2/5], Loss: 0.1864
Epoch [3/5], Loss: 0.1368
Epoch [4/5], Loss: 0.1102
Epoch [5/5], Loss: 0.0923
Optimizer: RMSprop, Accuracy: 97.16%, Execution Time: 33.95 seconds

Summary of Results:
Optimizer: SGD, Accuracy: 92.33%, Execution Time: 30.71 seconds
Optimizer: Adam, Accuracy: 96.83%, Execution Time: 43.21 seconds
Optimizer: RMSprop, Accuracy: 97.16%, Execution Time: 33.95 seconds
```

Optimizers	Accuracy	Exec Time
Baseline	92.78%	30.89
SGD	92.33%	30.71
Adam	96.83%	43.21
RMS prop	97.16%	33.95

4. Experiment with “Dropout layer”, “BatchNorm” and “Weight initialization” and report changes in accuracy and execution time.

```
Epoch [1/5], Loss: 1.5487
Epoch [2/5], Loss: 0.7958
Epoch [3/5], Loss: 0.6195
Epoch [4/5], Loss: 0.5351
Epoch [5/5], Loss: 0.4880
Model: Dropout, Accuracy: 92.36%, Execution Time: 30.98 seconds
Epoch [1/5], Loss: 0.5722
Epoch [2/5], Loss: 0.2195
Epoch [3/5], Loss: 0.1557
Epoch [4/5], Loss: 0.1210
Epoch [5/5], Loss: 0.0996
Model: BatchNorm, Accuracy: 97.36%, Execution Time: 29.36 seconds
Epoch [1/5], Loss: 0.5805
Epoch [2/5], Loss: 0.3061
Epoch [3/5], Loss: 0.2560
Epoch [4/5], Loss: 0.2244
Epoch [5/5], Loss: 0.2018
Model: WeightInit, Accuracy: 94.48%, Execution Time: 30.62 seconds

Summary of Results:
Model: Dropout, Accuracy: 92.36%, Execution Time: 30.98 seconds
Model: BatchNorm, Accuracy: 97.36%, Execution Time: 29.36 seconds
Model: WeightInit, Accuracy: 94.48%, Execution Time: 30.62 seconds
```

Regularization	Accuracy	Exec Time
Baseline	92.78%	30.89
Dropout	92.36%	30.98
BatchNorm	97.36%	29.36
WeightInit	94.48%	30.62

Key Observations

Based on the experiments conducted with different activation functions, optimizers, and regularization techniques, here are the observations and findings:

Observations on Accuracy

1. **Activation Functions**: Among the activation functions tested, **Tanh** and **ReLU** provided higher accuracies compared to the **Identity** function. Non-linear activation functions, such as ReLU and Tanh, introduce non-linearity into the model, enabling it to learn more complex patterns, which often results in better performance on classification tasks.
2. **Optimizers**: **Adam** and **RMSprop** optimizers achieved significantly higher accuracies compared to **SGD**. This is due to their adaptive learning rate properties, which help in faster convergence and better handling of complex error surfaces, especially in deep networks.
3. **Regularization Techniques**: Applying **Batch Normalization** and **Weight Initialization** noticeably improved accuracy. Batch Normalization helps stabilize and accelerate training by normalizing layer inputs, making it easier for the network to learn. Proper Weight Initialization ensures that weights start at a scale conducive to stable gradient flow, preventing issues like vanishing or exploding gradients.

Observations on Execution Time

1. **Activation Functions**: There was minimal difference in execution time across activation functions, with Tanh being slightly faster than Identity and ReLU. The choice of activation function generally has a small impact on overall execution time.
2. **Optimizers**: Execution time varied significantly across optimizers, with **SGD** being the fastest and **Adam** taking the longest. This is because Adam and RMSprop involve more computations per update (e.g., maintaining per-parameter learning rates and computing moving averages of gradients), which makes them computationally heavier than SGD.
3. **Regularization Techniques**: Batch Normalization slightly reduced execution time, likely due to faster convergence. Dropout, however, added to execution time because it deactivates random neurons during training, requiring additional computation to handle dropped connections. Weight Initialization had minimal impact on execution time.

Factors Impacting Accuracy

- **Choice of Optimizer**: Optimizers like Adam and RMSprop adaptively adjust learning rates, which improves accuracy by effectively navigating the loss landscape.
- **Activation Function**: Non-linear functions (ReLU, Tanh) enable the network to learn complex features, improving accuracy.
- **Regularization**: Techniques like BatchNorm and Weight Initialization help stabilize training, leading to better convergence and accuracy.

Factors Impacting Execution Time

- **Optimizer Complexity**: More complex optimizers (Adam, RMSprop) increase execution time due to additional computations.
- **Regularization Techniques**: Techniques like Dropout add overhead, as they require additional computations to manage neuron dropout during training.