1. Neural Networks exhibit a lot of weight space symmetry. These are less weights to train and this can simplify training. If we have a deep NN that is K layers deep with L hidden units, then there are in fact $L!^K$ ways to rearrange all hidden units in model without changing models output at all. So we have an unfathomably large number of equivalent local minima that all have the same objective value. This is not the only phenomena that leads to weight-space symmetry or infinite local minima in our objective function. To say, Local minima aren't the only thing mirrored by these symmetries, they're just the thing we care most about.

2. Learning Rate Schedulers can help by decreasing the step size as we are descending our gradient. This is helpful because as gradient gets harder to navigate (think vanishing/exploding gradients) we want to take the appropriate step size to continue our optimization.

3. A performance decay learning rate schedule means reducing the learning rate when you run out of patience. Running out of patience means that your accuracy is not improving on your dev set after so many updates and the loss graph has flattened out. Here we turn down the learning rate because the gradient is now harder to navigate.

4. You might initialize the bias vectors differently when the layer is followed by a ReLU activation than when it is followed by any other different activation function. This is because in ReLU you want to avoid the 0 slope, the common value to initialize is 0.1. For example, if it were tanh activation, we would want to use the xavier_uniform initialization. A good initialization helps to avoid slow convergence and ensure that you do not keep oscillating off/around the minima.

5. Dropout and ensembles are similar because they are both regularization techniques. A way in which dropout is different from ensembles is that they are dropping out, or omitting data/weights, while ensembles are culminations of multiple results. They both regularize, but they do so in different ways.

6. Dropout functions different at test time because we DO NOT dropout during test time. However, during training phase we will perform dropout which gives us the mechanism to emulate the effect of ensembling many models without the computational cost. The likelihood we dropout in training is dictated by an extra hyperparameter called keep_prob, which is the probability that we do not drop out on each unit. In the testing phase we want scale each layer with the keep_prob. We do this to make sure the network is outputting a similarly weighted model at test time compared to training time.

7. Multitask learning is where you are learning multiple tasks at the same time. You are doing this to either learn the relationships between each task or help one task regularize by also learning another task. For example, image classification that also does scene classification on the background. One needs to be careful to have sufficient capacity in the common layers. Even then, you have risk of underfitting on possibly both tasks.

8. Shortcut/Skip connections help train deep models by helping with vanishing and exploding gradients. You are propagating (taking from one layer and feeding to a much deeper layer) "good" activations further through the network faster so they will not suffer from vanishing or exploding gradients.

9. The problem that batch norm addresses is it makes neural networks more stable (robust). By normalizing the input features we can speed up training drastically. This normalization

is not only applied to input layer, but can also be applied periodically throughout the hidden layers.

10. Learned weights might differ while utilizing L1 regularization during training compared to L2 regularization in the sense that L1 regularization weights are going to be non-differentiable. They will either be a defined derivative of 1 or -1. This keeps pushing the gradients by the same amount since L1 takes the absolute value which results in a V shaped function (graphical representation of abs value) whereas L2 regularization is differentiable since we square the absolute value resulting in a quadratic function resulting in a more parabolic shaped graphical representation. L1 can be nice however to create sparsely connected layers which will create efficient hardware between layers and results in some models being trained first with L1 regularization.