1. After each stride-2 conv, why do we double the number of filters?

**Ans : A stride 2 conv with the default padding (1) and ks (3) will reduce the activation map dimension by half. Formula: (n + 2\*pad - ks)//stride + 1. As the activation map dimension reduces by half we double the number of filters. This results in no overall change in computation as the network gets deeper and deeper.**

2. Why do we use a larger kernel with MNIST (with simple cnn) in the first conv?

**Ans : We can already see that convolutional layers drastically reduce the number of weights needed. The number of weights is dependent on the kernel size instead of the input size which is really important for images. Convolutional layers reduce memory usage and compute faster.**

3. What data is saved by ActivationStats for each layer?

**Ans : ActivationStats hook to grab the activations from my chosen layers and generate the histogram data used for the visualisations.**

4. How do we get a learner's callback after they've completed training?

**Ans : A callback is an object that can perform actions at various stages of training (e.g. at the start or end of an epoch, before or after a single batch, etc). You can use callbacks to: Write TensorBoard logs after every batch of training to monitor your metrics.**

5. What are the drawbacks of activations above zero?

**Ans : The output is always between 0 and 1, that means that the output after applying sigmoid is always positive hence, during gradient-descent, the gradient on the weights during backpropagation will always be either positive or negative depending on the output of the neuron. As a result, the gradient updates go too far in different directions which makes optimization harder.**

**Sigmoid neurons get saturated on the boundaries and hence the local gradients at these regions is almost zero. To give you a more intuitive example to understand this, consider the inputs to the sigmoid function to be +15 and -15. The derivative of sigmoid function is `sig(z) \* (1 — sig(z))`. As mentioned above, the large positive values are squashed near 1 and large negative values are squashed near 0. Hence, effectively making the local gradient to near 0. As a result, during backpropagation, this gradient gets multiplied to the gradient of this neurons’ output for the final objective function, hence it will effectively “kill” the gradient and no signal will flow through the neuron to its weights. Also, we have to pay attention to initializing the weights of sigmoid neurons to avoid saturation, because, if the initial weights are too large, then most neurons will get saturated and hence the network will hardly learn.**

6. Draw up the benefits and drawbacks of practicing in larger batches?

**Ans : Larger batch sizes make larger gradient steps than smaller batch sizes for the same number of samples seen. For the same average Euclidean norm distance from the initial weights of the model, larger batch sizes have larger variance in the distance. Higher batch sizes leads to lower asymptotic test accuracy.**

7. Why should we avoid starting training with a high learning rate?

**Ans : If you’re learning rate is set too high, it can cause undesirable divergent behaviour in your loss function.**

8. What are the pros of studying with a high rate of learning?

**Ans : Generally, a large learning rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train.**

9. Why do we want to end the training with a low learning rate?

**Ans : Generally, a large learning rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train.**