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

**Doubling the number of filters after each stride-2 convolution is a common practice in convolutional neural networks (CNNs) for a few reasons:**

**- It helps increase the network's capacity: As the spatial dimensions of the feature maps decrease (due to stride-2 convolutions or pooling layers), the number of filters is increased to capture more diverse and abstract features in the smaller spatial context.**

**- It allows the network to learn hierarchical features: By increasing the number of filters, the network can learn low-level features in the early layers and high-level features in the deeper layers, enabling it to recognize complex patterns.**

**- It helps with information flow: Doubling the number of filters maintains a balanced flow of information as the spatial dimensions decrease, preventing information loss.**

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

**Using a larger kernel (e.g., 5x5) in the first convolutional layer of a CNN for MNIST can be beneficial because MNIST images are relatively simple and have a low spatial resolution (28x28 pixels). A larger kernel allows the network to capture larger and more abstract patterns or features in the input images. Starting with a larger receptive field helps the network learn basic shapes, edges, and higher-level features that might span multiple pixels. As you go deeper into the network, you can gradually reduce the kernel size to capture finer details.**

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

**The `ActivationStats` callback in deep learning frameworks like fastai is used to record statistics about activations (outputs of neurons) for each layer during training. For each layer, it typically saves the following data:**

**- Mean Activation: The mean value of activations for that layer over the training batch.**

**- Standard Deviation Activation: The standard deviation of activations for that layer over the training batch.**

**- Histogram of Activations: A histogram or distribution of activation values for that layer over the training batch. This can provide insights into the spread of activation values.**

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

**To get a callback after a learner (model) has completed training, you can define a custom callback class that inherits from the base callback class in the deep learning framework you're using (e.g., `Callback` in fastai or `tf.keras.callbacks.Callback` in TensorFlow/Keras). In this custom callback, you can implement the `on\_train\_end` method, which is called after the training is completed. Inside this method, you can define the specific actions or computations you want to perform after training.**

5. What are the drawbacks of activations above zero?

**Activations above zero (positive activations) are generally desirable in neural networks because they indicate the presence of learned features and signal that the network is activating specific neurons. However, some potential drawbacks or challenges associated with positive activations include:**

**- Vanishing gradients: If activations become too large, gradients during backpropagation can become extremely small (vanishing gradients), making it difficult for the network to update weights effectively.**

**- Saturation: If activations saturate (approach extremely high values), it can lead to numerical instability during training.**

**- Limited range: Positive activations may not capture negative or opposing patterns in the data effectively.**

**- Nonlinearity: Positive activations introduce nonlinearity into the model, which may complicate certain optimization and learning processes.**

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

**Benefits of larger batches:**

**- Faster convergence: Larger batches often lead to faster training convergence, as weight updates are based on more training examples per iteration.**

**- Better hardware utilization: Utilizing larger batch sizes can make better use of high-performance hardware (GPUs/TPUs) and improve training efficiency.**

**- Regularization effect: Larger batches may have a slight regularization effect due to the increased noise in the gradient estimates, potentially reducing overfitting.**

**Drawbacks of larger batches:**

**- Memory requirements: Larger batch sizes require more memory, which may limit training on GPUs with limited memory.**

**- Slower iterations: Larger batches can result in slower iterations, making it harder to fine-tune models or experiment with hyperparameters.**

**- Reduced generalization: Very large batch sizes can sometimes lead to worse generalization performance due to the loss of fine-grained updates and regularization.**

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

**Starting training with a high learning rate can be problematic because it may lead to the following issues:**

**- Divergence: A high learning rate can cause the optimization process to diverge, leading to exploding gradients and loss values.**

**- Unstable training: It can make the training process unstable, making it difficult to find a good model.**

**- Skipping over minima: The optimizer may overshoot the minimum of the loss function and keep oscillating, preventing the model from converging.**

**It's common practice to start with a lower learning rate and gradually increase it during training (learning rate annealing or scheduling) to find a balance between faster convergence and stability.**

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

**While starting with a high learning rate is generally discouraged, there are some potential advantages to using a high learning rate during certain stages of training or for specific tasks:**

**- Faster convergence: A high learning rate can lead to faster initial convergence when the model weights are far from the optimal values.**

**- Escaping local minima: It might help the model escape local minima and explore a larger part of the loss landscape.**

**- Exploration: High learning rates can be useful for exploration in reinforcement learning settings.**

**However, it's crucial to anneal or reduce the learning rate after this initial phase to stabilize training.**

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

**Ending training with a low learning rate is beneficial because:**

**- Fine-tuning: A lower learning rate allows the model to fine-tune its weights and converge to a more precise solution as it gets closer to the optimal values.**

**- Stability: Lower learning rates stabilize the training process, preventing the model from overshooting minima or oscillating.**

**- Generalization: Slower training with a lower learning rate often leads to better generalization and improved model performance on unseen data.**

**- Local optimization: It helps the optimizer perform local optimization to refine the model's parameters.**

**Gradually reducing the learning rate (learning rate scheduling) is a common technique to achieve these benefits and improve the final model's performance.**