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) because it helps the network capture increasingly abstract and higher-level features as you go deeper into the network. The intuition behind this is that earlier layers learn low-level features like edges and textures, which are relatively simple. As you move deeper into the network, the receptive field of the filters increases, and more complex features, like object parts and combinations of low-level features, can be learned. By increasing the number of filters, the network can capture a wider variety of features and patterns.

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

Using a larger kernel in the first convolutional layer of the MNIST dataset with a simple CNN can help the network learn more meaningful local features. MNIST images are relatively small (28x28 pixels), and a larger kernel can capture more spatial information and features at a higher level of abstraction. Smaller kernels might be too restrictive in capturing important patterns, while larger kernels can provide a broader view of the data.

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

The ActivationStats callback in fastai or similar frameworks is used to save statistics about activations in each layer during training. For each layer, it saves the following data:

* Mean activation: The average value of activations in the layer.
* Standard deviation of activation: A measure of the spread or dispersion of activations.
* Sparsity: The proportion of zero activations in the layer.

Analyzing these statistics can provide insights into how activations are distributed in the network and whether there are issues like vanishing or exploding gradients, which can affect training stability and convergence.

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

To get a learner's callback after they've completed training, you can use callback hooks provided by the fastai library. Specifically, you can add a callback hook for the "after\_train" stage. This will allow you to execute custom code after the training process is completed. For example, you can use learn.add\_cb to add a callback for the "after\_train" stage and specify the callback function you want to run.

5. What are the drawbacks of activations above zero?

Drawbacks of activations above zero:

* Saturation: Activations above zero can saturate, meaning they reach extremely high values, making it challenging to train the network and causing vanishing/exploding gradients.
* Gradient instability: High activations can lead to gradient instability, making convergence difficult and slowing down training.
* Overfitting: Extremely high activations can lead to overfitting because the network might focus on memorizing the training data rather than generalizing.

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

Benefits and drawbacks of practicing with larger batches in deep learning:

Benefits:

* Faster convergence: Larger batches can lead to faster training convergence since more data is processed in each update step.
* Efficient GPU utilization: Larger batches can fully utilize GPU resources, leading to faster training times on modern hardware.
* Smoother loss curves: Training with larger batches can result in smoother loss curves, which can make it easier to monitor training progress.

Drawbacks:

* Memory requirements: Larger batches require more memory, limiting the batch size on GPUs with limited memory.
* Generalization: Larger batch sizes can sometimes lead to worse generalization, as the model may focus on memorizing the training data instead of learning meaningful features.
* Stiffness: Large batches may lead to slower adaptation to changing patterns in the data, as the model makes infrequent updates.

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

Starting training with a high learning rate can lead to several issues, including:

* Poor convergence: High initial learning rates may cause the training process to diverge, making it difficult to find an optimal solution.
* Overshooting: The optimizer may overshoot the minimum of the loss function, causing oscillations and slow convergence.
* Instability: High learning rates can result in unstable and erratic training behavior, making it challenging to train the model effectively.

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

Pros of studying with a high learning rate:

* Faster convergence: High learning rates can lead to faster initial convergence when the model is far from the optimal solution.
* Exploration: A high learning rate can help the model explore a larger portion of the parameter space in the early stages of training.

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

Ending training with a low learning rate is a common practice known as "learning rate annealing" or "learning rate scheduling." The reasons for this include:

* Fine-tuning: A lower learning rate in later stages of training allows the model to fine-tune its parameters, gradually approaching a more accurate solution.
* Stable convergence: Lower learning rates help ensure stable and precise convergence, avoiding overshooting or diverging from the optimal solution.
* Improved generalization: Slower learning in the later stages can lead to better generalization, as the model can focus on fine-grained adjustments to fit the data.