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

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

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

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

5. What are the drawbacks of activations above zero?

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

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

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

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

Answer:

1. Doubling the number of filters after each stride-2 conv is a common practice in convolutional neural networks because it helps to maintain the spatial resolution of the output while increasing the complexity of the learned features. With each stride-2 conv, the spatial dimensions of the input are reduced by a factor of two, so doubling the number of filters helps to compensate for the loss of information.
2. A larger kernel size in the first convolutional layer of a simple CNN for MNIST can help to capture more global features of the input image, such as edges and contours, that may be important for classification. However, a larger kernel size also means more parameters to learn and can increase the computational cost of the network.
3. ActivationStats is a callback in fastai library that saves the mean, standard deviation, and histogram of the activations for each layer during training. This data can be useful for diagnosing issues with the network, such as vanishing or exploding gradients, and for gaining insights into how the network is processing the input data.
4. In fastai library, we can use the **on\_train\_end** method of a learner's callback to execute code after the training is completed. For example, if we want to print a message after the training is done, we can define a custom callback with an **on\_train\_end** method that prints the message, and then pass this callback to the **Learner** object using the **add\_cb** method.
5. Activations above zero can lead to saturation of the neurons in a neural network, where the output of the neuron is pushed to either the minimum or maximum value of the activation function. This can cause the gradient to vanish or explode during backpropagation, making it difficult or impossible to train the network effectively.
6. Practicing in larger batches can have several benefits, such as faster training times, better parallelism, and smoother gradients. However, using larger batches also requires more memory and computational resources, and can lead to lower generalization performance if the batch size is too large.
7. Starting training with a high learning rate can lead to unstable training dynamics, where the network oscillates between large weight updates and small weight updates, or even diverges completely. This can happen because large weight updates can cause the network to overshoot the optimal weights, leading to a worse performance.
8. Studying with a high learning rate can help the network to converge faster and reach a good performance in fewer iterations. This can be especially useful for large datasets or complex models, where training can take a long time with a low learning rate.
9. Ending the training with a low learning rate can help the network to fine-tune its weights and improve its performance on the validation set. This is because a low learning rate allows the network to make small, incremental adjustments to its weights, which can lead to better generalization performance. Additionally, a low learning rate can help to prevent the network from overfitting to the training data.