```
    # compile the model
    model.compile(optimizer=tf.keras.optimizers.SGD(0.01),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

Gradient Clipping

Gradient clipping is another technique for hemming the problem of vanishing and exploding gradients mostly seen in recurrent networks due to training via backpropagation across a large number of deep recurrent layers. Gradient clipping involves trimming the computed gradients so that they remain within a specific range; in doing so, the gradients are prevented from saturating as the network trains across multiple deep layers.

Gradient clipping is implemented in TensorFlow 2.0 by adjusting the **'clipnorm'** or **'clipvalue'** parameters of the selected optimizer from the **'tf.keras.optimizers'** package. **'clipnorm'** clips the gradients by norm, while **'clipvalue'** clips the gradients by value.

This chapter introduces some important techniques that are employed to improve the performance of a neural network by further mitigating the issue of vanishing and exploding gradients. In the next chapter, we will see more optimization techniques for training deep neural network model.

CHAPTER 33

More on Optimization Techniques

In this chapter, we'll go over some other optimization techniques for improving the ability of a neural network to learn complex patterns in a dataset.

Momentum

Momentum is a technique for improving the convergence speed of stochastic gradient descent (SGD) optimization. Remember that stochastic gradient works by learning the direction of steepest descent by evaluating a training example at each time step to optimize the weights of the network. Momentum improves on this by calculating the average of previous gradients in a process called exponentially smoothed averages. It then uses this computed average to continue to move in the direction of steepest descent. By doing so, it quickens the learning process. In computing this exponentially decayed average, a momentum hyper-parameter is introduced to control how the weight parameters are updated. Figure 33-1 shows an example of stochastic gradient descent with and without momentum as it converges in a function space. In TensorFlow 2.0, momentum is added to a SGD optimizer by adjusting the **'momentum'** parameter of the **SGD method, 'tf.keras.optimizers.SGD(momentum=[float >=0])'**. The momentum value must be a float value that is greater or equal to 0 that accelerates SGD in the relevant direction and dampens oscillations.

E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*,
https://doi.org/10.1007/978-1-4842-4470-8_33