1. Is it OK to initialize all the weights to the same value as long as that value is selected

randomly using He initialization?

No. All weights should be initialized to different random values and should not have the same initial value. If weights are symmetrical, meaning they have the same value, it makes it almost impossible for backpropagation to converge to a good solution.

2. Is it OK to initialize the bias terms to 0?

It is possible and common to initialize the biases to be zero, since the asymmetry breaking is provided by the small random numbers in the weights.

3. Name three advantages of the SELU activation function over ReLU.

Advantages of Relu are-

1. it does not have an abrupt change like relu hence the smoothness
2. It solves the dying relu problem by accepting negative input values
3. Always has a non zero derivative.

4. In which cases would you want to use each of the following activation functions: SELU, leaky

ReLU (and its variants), ReLU, tanh, logistic, and softmax?

For classification problem-

1.Relu,tanh-hidden layer

2.sigmoid-output layer of binary classification.

3. softmax- binary layer of multiclass classification.

For Regression problem-

No activation function at output .

5. What may happen if you set the momentum hyperparameter too close to 1 (e.g., 0.99999)

when using an SGD optimizer?

Deep learning neural networks are trained using the stochastic gradient descent algorithm.

Stochastic gradient descent is an optimization algorithm that estimates the error gradient for the current state of the model using examples from the training dataset, then updates the weights of the model using the back-propagation of errors algorithm, referred to as simply backpropagation.

The amount that the weights are updated during training is referred to as the step size or the “learning rate.”

Specifically, the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0.

The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs.

A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.

The challenge of training deep learning neural networks involves carefully selecting the learning rate. It may be the most important hyperparameter for the model.

6. Name three ways you can produce a **sparse model**.

A sparse model is that where most weights are equal to 0. There's a couple ways of achieving that effect You can train the model normally then zero out tiny weights.

For more sparsity, you can apply l1 regularization during training, which pushes the optimizer towards sparsity.

Finally, you can combine l1 regulatization with dual averaging using TensorFlow's

7. Does dropout slow down training? Does it slow down inference (i.e., making predictions on

new instances)? What about MC Dropout?

8. Practice training a deep neural network on the CIFAR10 image dataset:

a. Build a DNN with 20 hidden layers of 100 neurons each (that’s too many, but it’s the

point of this exercise). Use He initialization and the ELU activation function.

b. Using Nadam optimization and early stopping, train the network on the CIFAR10

dataset. You can load it with keras.datasets.cifar10.load\_​data(). The dataset is

composed of 60,000 32 × 32–pixel color images (50,000 for training, 10,000 for

testing) with 10 classes, so you’ll need a softmax output layer with 10 neurons.

Remember to search for the right learning rate each time you change the model’s

architecture or hyperparameters.

c. Now try adding Batch Normalization and compare the learning curves: Is it

converging faster than before? Does it produce a better model? How does it affect

training speed?

d. Try replacing Batch Normalization with SELU, and make the necessary adjustements

to ensure the network self-normalizes (i.e., standardize the input features, use

LeCun normal initialization, make sure the DNN contains only a sequence of dense

layers, etc.).

e. Try regularizing the model with alpha dropout. Then, without retraining your model,

see if you can achieve better accuracy using MC Dropout.