**DL theory : Assingments-15**

1. Deep Learning: a. To build a DNN with five hidden layers of 100 neurons each, He initialization, and the ELU activation function, you can use a deep learning framework such as TensorFlow or PyTorch. The code for building the DNN would involve defining the architecture of the network, including the number of layers and neurons per layer, and the activation function. He initialization can be achieved by initializing the weights with a normal distribution with a mean of 0 and a standard deviation of sqrt(2/n), where n is the number of input neurons. b. To train the DNN on MNIST digits 0 to 4, you can use the Adam optimization algorithm and early stopping to prevent overfitting. You can also save checkpoints at regular intervals and save the final model for reuse later. The output layer should have a softmax activation function with five neurons, one for each of the digits 0 to 4. c. To tune the hyperparameters, you can use techniques such as cross-validation to try different combinations and see which one achieves the best precision. d. To add Batch Normalization, you can insert batch normalization layers between the hidden layers and the activation function. Compare the learning curves to see if it converges faster and produces a better model. e. To check for overfitting, you can evaluate the model's performance on a validation set. If the model is overfitting, you can try adding dropout to every layer to reduce overfitting
2. **Transfer learning.**
3. a. To create a new DNN that reuses all the pretrained hidden layers of the previous model, we can use the TensorFlow function tf.keras.Model.layers to create a new model instance and then set the trainable attribute of the hidden layers to False to freeze them. The new model should have a new softmax output layer with five neurons, corresponding to the five digits (0-4) in the original model.
4. b. To train this new DNN on digits 5 to 9, we can use a small subset of the MNIST dataset that only contains images of these digits. We can use the Adam optimization algorithm and early stopping to train the model, and measure the precision using a validation set.
5. c. To improve the training speed, we can cache the frozen layers of the model and use them as a starting point for training the new output layer. This can significantly reduce the computation time required to train the model.
6. d. By reusing fewer hidden layers and training the remaining layers with a small dataset, we may be able to achieve a higher precision. This is because the model is able to learn more specific features of the new task with fewer layers.
7. e. By unfreezing the top two hidden layers and continuing to train, we may be able to improve the model's performance even further. This is because the model is able to learn more specific features of the new task with a smaller dataset by fine-tuning the already trained layers.

Name three popular activation functions. Can you draw them?

Three popular activation functions are ReLU, sigmoid, and tanh.

ReLU (Rectified Linear Unit) is defined as f(x) = max(0,x) and it is usually used in the hidden layers of a network. It is a linear function for positive inputs and zero for negative inputs.

Sigmoid function is defined as f(x) = 1/(1+e^-x) and it is commonly used in the output layer of a binary classification problems. It produces an output between 0 and 1, and can be interpreted as a probability.

tanh (hyperbolic tangent) is defined as f(x) = (e^x - e^-x)/(e^x + e^-x) and it is similar to sigmoid function but it produces an output between -1 and 1. It is commonly used in the output layer of a binary or multi-class classification problems.

I am unable to draw the graph of these activation functions**.**