1. A neuralnetworkdesignedusingTensorflowKerasisgivenbelow.Studentsareinstructed to typetheanswersinthetextboxoftheportal.

```
net=tf.keras.layers.Sequential()
net.add(
t f.keras.layers.InputLayer(input_shape=((256 * 2 56 * 3 ),)),
t f.keras.layers.Dense(1048,activation='relu'),
t f.keras.layers.Dense(512,activation='relu'),
t f.keras.layers.Dropout(0.4),
t f.keras.layers.Dense(256,activation='relu'),
t f.keras.layers.Dense(128,activation='relu'),
t f.keras.layers.Dense(64,activation='relu'),
t f.keras.layers.Dropout(0.6),
t f.keras.layers.Dense(32,activation='relu'),
t f.keras.layers.Dense(16,activation='relu'),
t f.keras.layers.Dense(8,activation='softmax'))
```

(a) What istheobjectiveoftheneuralnetwork?Whatistheinputgiventothenetwork? What istheexpectedoutput?Howdeepandwideisthenetwork.[2]

(b) Justifythechoiceactivationfunctioninoutputlayer.InsteadofReluactivationfunction, justifythechoiceofusingTanhactivationfunction.[1]

(c) Twodropoutstatementsareaddedinthecode.Howmanyadditionalparametersare learned becauseofthis?Ifdropoutisaddedafterthelaststatementinthegivencode, howwillitaffectthenetwork?[1]

(d) Writethecodesnippetforaddingtheoptimizerofyourchoice.Justifythechoiceof the optimizer.Assumeanyotherrelevantinformation.[1]

(e) What willthefollowingcodesnippetdotothenetwork.[1]

```
cb=tf.keras.callbacks.EarlyStopping(monitor='loss',patience=7)
h istory=net.fit(epochs=200,batch_size=32,callbacks=[cb])
```

ChatGPT

(a) Objective, Input, Output, Depth, and Width of the Neural Network:

- **Objective**: The objective of the neural network is not explicitly stated in the provided code snippet. However, based on the structure of the network, it appears to be a classification model as it ends with a softmax activation function in the output layer.
- **Input**: The input given to the network has a shape of $256 \times 256 \times 3$, which suggests that it likely represents images with dimensions 256×256 and 3 color channels (RGB).
- **Expected Output**: The output of the network is produced by the last layer with a softmax activation function, which suggests that it's performing multi-class classification. The network has 8 output units with softmax activation.
- **Depth and Width**: The network has a depth of 10 layers, including the input layer and output layer. It consists of several dense layers with varying numbers of neurons, resulting in a wide architecture.

(b) Choice of Activation Functions:

- **Output Layer**: The softmax activation function is commonly used in the output layer for multi-class classification tasks because it normalizes the output into a probability distribution, making it suitable for predicting class probabilities.
- **Justification for Tanh Activation**: If tanh activation function were used instead of ReLU in the hidden layers, it would introduce non-linearity and can be useful in cases where the output range needs to be between -1 and 1, which can be beneficial for better convergence during training.

(c) Dropout Statements and Additional Parameters:

- **Number of Additional Parameters due to Dropout**: Each dropout layer introduces additional parameters equal to the number of weights in the corresponding layer multiplied by the dropout rate. Since the dropout rate is 0.4 and 0.6 for the two dropout layers, respectively, the total number of additional parameters introduced by dropout can be calculated based on the weights of the layers they are applied to.

- **Effect of Dropout after the Last Statement**: Adding dropout after the last statement in the given code would affect the network by introducing regularization only during training. It would randomly drop units from the last layer, which could help prevent overfitting.

(d) Code Snippet for Adding Optimizer and Justification:

python Copy code

```python
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001) net.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
```

- **Choice of Optimizer**: Adam optimizer is chosen for its adaptive learning rate and momentum-based approach, which often leads to faster convergence and better performance in a variety of tasks.

(e) Effect of the Provided Code Snippet:

- The provided code snippet sets up an early stopping callback (`tf.keras.callbacks.EarlyStopping`) that monitors the training loss and stops training if the loss does not improve for 7 consecutive epochs.
- It then fits the neural network (`net`) to the data for a maximum of 200 epochs with a batch size of 32, utilizing the early stopping callback. This means that training will stop early if the loss does not improve for the specified number of epochs (7 in this case).