**COMPUTER VISION ASSIGNMENT\_6**

**1.What is the difference between TRAINABLE and NON-TRAINABLE PARAMETERS?**

Trainable and non-trainable parameters refer to different types of parameters in a deep learning model.

Trainable parameters are the weights and biases in a model that are updated during training. These parameters are updated in response to the optimization algorithm's assessment of the model's performance on the training data, and they capture the patterns in the data that the model uses to make predictions.

Non-trainable parameters are parameters that are not updated during training. For example, the mean and standard deviation of the input data used to normalize the input can be considered non-trainable parameters. Some layers in a deep learning model may also have non-trainable parameters, such as the number of filters in a Convolutional Neural Network (CNN) layer.

In summary, trainable parameters are parameters that can be updated during training to improve the model's performance, while non-trainable parameters are parameters that are fixed and cannot be updated during training.

**2. In the CNN architecture, where does the DROPOUT LAYER go?**

The dropout layer in a Convolutional Neural Network (CNN) can be placed anywhere in the network, but it is commonly inserted between the dense layers, or fully connected layers, of the network. The dropout layer helps to prevent overfitting by randomly dropping out some neurons during training, forcing the model to rely on other neurons to make predictions.

For example, a common architecture for a CNN might include the following layers:

1. Convolutional layer
2. Pooling layer
3. Convolutional layer
4. Pooling layer
5. Flatten layer
6. Dense layer with dropout
7. Dense layer with dropout
8. Output layer

In this example, the dropout layer is placed between the two dense layers. The dropout rate, or the fraction of neurons to drop out, can be set as a hyperparameter and fine-tuned during training.

**3. What is the optimal number of hidden layers to stack?**

The optimal number of hidden layers to stack in a deep learning model is a question without a definite answer as it largely depends on the complexity of the problem and the size of the dataset.

More hidden layers can lead to a more complex model that can capture more intricate patterns in the data. However, adding more layers also increases the risk of overfitting, which can result in poor generalization performance on new data.

A common approach is to start with a small number of hidden layers and gradually increase the number until the model performance plateaus, or begins to deteriorate. Some researchers have also found that deeper models with more hidden layers tend to perform better than shallow models with fewer layers.

Ultimately, the best number of hidden layers will depend on the specific problem and dataset, and finding the optimal number often requires experimentation. To prevent overfitting, it's important to use regularization techniques such as dropout or early stopping.

**4. In each layer, how many secret units or filters should there be?**

The number of units or filters in each layer of a deep learning model is a hyperparameter that can be adjusted to optimize model performance. The optimal number of units or filters depends on the specific problem and dataset, and it is often determined through experimentation.

In general, a larger number of units or filters can allow the model to capture more complex patterns in the data, but it also increases the risk of overfitting and the computational cost of training the model.

As a starting point, a common practice is to set the number of units or filters in the first layer to a relatively large value, such as 32 or 64, and to gradually decrease the number of units or filters in subsequent layers, such as 16, 8, 4, and so on. The final number of units or filters will depend on the size of the dataset and the complexity of the problem.

Ultimately, the best number of units or filters will depend on the specific problem and dataset, and finding the optimal number often requires experimentation.

**5. What should your initial learning rate be?**

The initial learning rate is a hyperparameter in deep learning models that determines the step size at which the optimization algorithm updates the model parameters. The optimal learning rate depends on the specific problem and dataset and can be determined through experimentation.

A high learning rate can lead to rapid convergence, but the optimization algorithm may not find the optimal solution and can overshoot the minimum. A low learning rate, on the other hand, can converge to the optimal solution, but it may take a long time to converge and can get stuck in local minima.

As a starting point, a common initial learning rate value is 0.001. If the model does not converge or is underfitting, the learning rate can be reduced, while if the model is overfitting or converging too slowly, the learning rate can be increased.

In practice, it's often helpful to use learning rate schedules or learning rate annealing techniques to dynamically adjust the learning rate during training, such as reducing the learning rate over time, or reducing the learning rate if the optimization stagnates. This can help to improve convergence and find a better solution.

**6. What do you do with the activation function?**

The activation function in a deep learning model is a mathematical function that transforms the inputs of a neuron into its outputs. It is used to introduce non-linearity into the model, allowing it to capture complex patterns in the data.

The choice of activation function depends on the specific problem and the type of deep learning model being used. Some common activation functions include:

Sigmoid: maps any input to the range of 0 and 1, and is often used in binary classification problems.

ReLU (rectified linear unit): sets all negative inputs to 0 and is commonly used in feedforward neural networks.

Tanh (hyperbolic tangent): maps inputs to the range of -1 and 1, and is similar to the sigmoid activation function.

Leaky ReLU: similar to ReLU but with a small positive slope for negative inputs, helping to prevent the vanishing gradient problem.

It's worth noting that the activation function should be chosen based on the specific problem, and different activation functions may perform better or worse depending on the dataset and model architecture.

In practice, it's often helpful to experiment with different activation functions to find the one that works best for a specific problem and dataset.

**7. What is NORMALIZATION OF DATA?**

Data normalization is a preprocessing step in deep learning and other machine learning algorithms that scales the input data to a specific range. This can help to ensure that all input features are on a similar scale, which can improve the performance of the model and reduce the risk of overfitting.

There are several techniques for normalizing data, including:

Min-Max Normalization: scales the data between 0 and 1.

Z-Score Normalization: scales the data to have a mean of 0 and a standard deviation of 1.

Log Normalization: applies a logarithmic transformation to the data.

The choice of normalization technique depends on the specific problem and the type of data being used. Min-Max normalization is often used for image data, while Z-Score normalization is often used for text or numerical data.

It's worth noting that normalization should be applied to the training data and then applied to the validation and test data using the same normalization parameters. This helps to ensure that the model is evaluated on data that has been transformed in the same way as the training data.

**8. What is IMAGE AUGMENTATION and how does it work?**

Image augmentation is a technique used in computer vision and deep learning to artificially increase the size of a dataset by creating new, transformed versions of the existing images. This helps to prevent overfitting, which can occur when a model is trained on a small dataset, and can also improve the generalization performance of the model.

Image augmentation works by applying various transformations to the images in the dataset, such as flipping, rotating, scaling, and adding noise. The goal is to create new, diverse images that capture variations in the appearance of the objects in the original images.

1. There are several common image augmentation techniques, including:
2. Flipping: flipping the image horizontally or vertically.
3. Rotation: rotating the image by a specified degree.
4. Scaling: resizing the image by a specified factor.
5. Cropping: cropping the image to a smaller size.
6. Adding noise: adding random noise to the image.

In practice, image augmentation is often performed as part of the data preprocessing step, and the transformed images are used as additional inputs to the deep learning model during training. The choice of augmentation techniques depends on the specific problem and the type of data being used, and it is often helpful to experiment with different augmentation techniques to find the ones that work best for a specific problem and dataset.

**9. What is DECLINE IN LEARNING RATE?**

Decline in learning rate is a technique used in deep learning to adjust the learning rate over time during training. The goal is to reduce the learning rate as training progresses, helping the model to converge more quickly and effectively.

There are several techniques for declining the learning rate, including:

Step Decay: the learning rate is reduced by a fixed factor after a fixed number of epochs.

Exponential Decay: the learning rate is reduced by a factor proportional to the exponential decay of the number of epochs.

1/t Decay: the learning rate is reduced by a factor proportional to the inverse of the number of epochs.

Cyclical Learning Rates: the learning rate is increased and decreased over time in a cyclical manner, allowing the model to make larger updates in the beginning of training and smaller updates as training progresses.

In practice, declining the learning rate can help to improve the performance of the model and reduce the risk of overfitting. However, the specific technique used and the schedule for declining the learning rate will depend on the specific problem and dataset, and it is often helpful to experiment with different declining learning rate schedules to find the one that works best for a specific problem and dataset.

**10.What does EARLY STOPPING CRITERIA mean?**

Early stopping criteria is a technique used in deep learning to stop the training process before the specified number of epochs is reached. The goal is to prevent overfitting, which can occur when a model is trained for too long and begins to memorize the training data rather than generalize to new, unseen data.

The idea behind early stopping is to monitor the performance of the model on a validation set during training. If the performance on the validation set has not improved after a certain number of epochs, the training process is stopped. This helps to prevent overfitting and ensures that the model is not trained for too long.

There are several techniques for determining the early stopping criteria, including:

* Monitoring the validation loss: if the validation loss has not improved after a certain number of epochs, the training process is stopped.
* Monitoring the validation accuracy: if the validation accuracy has not improved after a certain number of epochs, the training process is stopped.
* Monitoring the difference between the training and validation loss: if the difference between the training and validation loss begins to increase after a certain number of epochs, the training process is stopped.

In practice, early stopping is a powerful technique for preventing overfitting and improving the generalization performance of deep learning models. The specific criteria for early stopping will depend on the specific problem and dataset, and it is often helpful to experiment with different early stopping criteria to find the one that works best for a specific problem and dataset.