**Assignment-2**

**Q.1) Define Bias and Variance?**

* **Bias:** bias refers to the error introduced by a neural network's inability to approximate the true underlying function or relationship between the input data and the output. It represents the model's simplifications or assumptions that may not capture the complex patterns present in the data. A model with high bias fails to learn the intricacies of the data and tends to underfit.
* **Characteristics of high bias in deep learning:**
* The model is too simple and lacks complexity to capture the underlying patterns. Inadequate representation of the data, resulting in poor performance. High training error and similar performance on training and test data. Underfitting occurs, leading to suboptimal predictions.
* **Variance:** Variance in deep learning refers to the extent of fluctuations or instability in a neural network's predictions due to its sensitivity to variations in the training data. It represents the model's tendency to overfit the training data and capture noise or random variations. A model with high variance is too flexible and complex, leading to poor generalization on unseen data.
* **Characteristics of high variance in deep learning:**
* The model is too complex and captures noise or random variations in the training data. Low training error but high test error. Difficulty in generalizing to unseen data. Overfitting occurs, resulting in poor predictive performance.

**Q.2) Define Overfitting and Underfittin with suitable example?**

* **Overfitting :** Overfitting in deep learning occurs when a neural network model becomes too complex and starts to memorize the noise and specific details of the training data rather than learning the underlying patterns. The model becomes highly specialized to the training data, resulting in poor generalization to unseen data.
* **Example of overfitting in deep learning:** Consider a convolutional neural network (CNN) trained on a dataset of images of cats and dogs for classification. If the model is excessively large or trained for too long, it may start to overfit the training data. It might learn specific features or textures unique to the training images, including noise or irrelevant details that are not indicative of the broader cat/dog discrimination. As a result, the model may achieve high accuracy on the training set but fail to generalize well to new images of cats and dogs.
* **Underfitting :** Underfitting in deep learning occurs when a neural network model is not complex enough to capture the underlying patterns in the training data. The model lacks the capacity to represent the relationships between the features and the target variable, leading to poor performance on both the training and test data.
* **Example of underfitting in deep learning:**
* Continuing with the same image classification task, suppose you train a simple shallow neural network with few layers and parameters. If the model is too simple, it may not have enough capacity to capture the intricate features that differentiate cats from dogs. Consequently, it might fail to achieve high accuracy even on the training data and perform poorly on unseen images as well.
* To address overfitting in deep learning, techniques such as regularization (e.g., L1, L2 regularization), dropout, early stopping, or reducing model complexity (e.g., using fewer layers or parameters) can be employed. Underfitting can be mitigated by increasing model complexity, adding more layers or parameters, or using more advanced architectures like deeper neural networks or convolutional neural networks.

**Q.3) Explain various ways to handle Under fitting and Over fitting?**

* Underfitting and overfitting are common challenges in machine learning models. Handling these issues effectively is crucial for improving model performance and achieving good generalization. Here are various ways to address underfitting and overfitting:
* **Underfitting:**
* **Increase model complexity:** If the model is too simple and lacks the capacity to capture the underlying patterns, increasing model complexity can help. This can involve adding more layers, increasing the number of parameters, or using more advanced architectures.
* **Add more features:** Insufficient features may result in underfitting. Adding more relevant features can provide the model with more information to learn from and improve its ability to capture complex relationships in the data.
* **Reduce regularization:** If regularization techniques such as L1 or L2 regularization are excessively penalizing the model's parameters, it can lead to underfitting. Adjusting the regularization strength or removing it entirely can allow the model to learn more from the data.
* **Decrease bias:** Bias refers to the simplifications or assumptions made by the model. Reducing bias involves allowing the model to have more flexibility and freedom to fit the data. This can be achieved by increasing model capacity or reducing regularization.
* **Overfitting:**
* **Increase regularization:** Regularization techniques such as L1 or L2 regularization can help reduce overfitting by adding penalties to the model's parameters. This discourages the model from fitting noise or irrelevant details in the training data.
* **Use dropout:** Dropout is a regularization technique where random neurons or connections are temporarily dropped during training. This helps prevent the model from relying too heavily on specific neurons and encourages more robust and generalized representations.
* **Early stopping:** Training a model for too long can lead to overfitting. Monitoring the performance of the model on a validation set during training and stopping the training process when the validation error starts increasing can prevent overfitting.
* **Increase training data:** Insufficient training data can contribute to overfitting. Acquiring more diverse and representative training data can help the model generalize better and reduce the tendency to overfit specific examples.
* **Data augmentation:** By applying various transformations or perturbations to the existing training data, data augmentation can increase the effective size of the training set. This helps the model learn more generalized representations and reduces overfitting.
* **Simplify model architecture:** If the model is overly complex, reducing the number of layers, parameters, or using simpler architectures can prevent overfitting. This promotes a more balanced representation of the data and reduces the risk of memorizing noise.

**Q.4) Difference between fully/densely connected NN and CNN?**

|  |  |  |
| --- | --- | --- |
|  | **FCNNs** | **CNN** |
| **Connectivity** | FCNNs are fully connected, meaning that every neuron in one layer is connected to every neuron in the previous layer. | In contrast, CNNs use convolutional layers that apply filters to local regions of the input data, resulting in sparse connectivity. |
| **Input Data** | FCNNs are typically used for tasks with fixed-size input data, such as image classification or sentiment analysis | CNNs are specifically designed for image processing and computer vision tasks, where the input data is often in the form of images or other spatial data. |
| **Feature Extraction** | FCNNs learn features from the raw input data, while CNNs use convolutional layers to extract spatial features from the input data. | This allows CNNs to capture local patterns and spatial correlations in the data, which is particularly useful in image processing. |
| **Parameter Sharing** | CNNs use parameter sharing in the convolutional layers to reduce the number of parameters and improve generalization. | In contrast, FCNNs have a separate weight for every connection, which can lead to overfitting. |
| **Training** | FCNNs are typically trained using backpropagation and gradient descent | While CNNs often use variants of these algorithms such as stochastic gradient descent with momentum or Adam optimization. |

**Q.5) Explain Applications of CNN?**

* Convolutional Neural Networks (CNNs) are a specialized type of neural network that are specifically designed for image processing and computer vision tasks. Here are some applications of CNNs in more detail:
* **Image Classification:** One of the most common applications of CNNs is image classification, where the task is to assign a label or category to an input image. CNNs can learn to recognize patterns and features in the input image and use them to make accurate predictions.
* **Object Detection:** Object detection is the task of identifying and localizing objects in an image. CNNs can be used to detect the presence and location of objects in an image, which is useful in applications such as self-driving cars, surveillance systems, and robotics.
* **Facial Recognition:** CNNs can be used for facial recognition, which involves identifying and verifying the identity of a person from a digital image or video. This application is used in security systems, law enforcement, and social media.
* **Medical Image Analysis:** CNNs can be used to analyze medical images such as X-rays, CT scans, and MRI scans. This can aid in the diagnosis of diseases and conditions, as well as the development of personalized treatment plans.
* **Natural Language Processing:** While CNNs are primarily used in image processing, they can also be used in natural language processing tasks such as text classification and sentiment analysis. CNNs can be used to learn patterns and features in text data, which can be used to classify or analyze text.
* **Video Analysis:** CNNs can also be used for video analysis, such as detecting and tracking objects in video streams or recognizing actions in videos. CNNs are widely used in applications related to image processing, computer vision, and natural language processing.

**Q.6) Explain building Blocks of (Pooling, Convolution, Relu, Padding,etc) in CNN.**

* **In Convolutional Neural Networks (CNNs), there are several building blocks that are used to extract features from images. Here are some of the main building blocks of CNNs:**
* **Convolution:** A convolution layer is used to extract features from the input image by applying a set of learnable filters to local regions of the image. These filters are moved across the entire image to generate a feature map that captures the presence of specific patterns or features in the input image. Convolution layers can be stacked to capture more complex features.
* **Example:** Let's say we have an input image of size 32x32x3 (height x width x channels) and a filter of size 5x5x3. We apply the filter to the image by sliding it over the image with a stride of 1, producing an output feature map of size 28x28x1.
* **Pooling:** Pooling layers are used to reduce the spatial size of the feature map while preserving the most important information. The most common type of pooling is max pooling, which takes the maximum value of each sub-region of the feature map. This reduces the spatial dimensions of the feature map while retaining the most salient features.
* **Example:** Let's say we have an input feature map of size 28x28x64 (height x width x channels) and we apply max pooling with a pool size of 2x2 and a stride of 2. This reduces the spatial dimensionality by half, producing an output feature map of size 14x14x64.
* **ReLU Activation:** The Rectified Linear Unit (ReLU) activation function is used to introduce non-linearity into the CNN. It applies an element-wise function to the output of the convolution layer, which sets all negative values to zero and leaves all positive values unchanged. This helps the network to learn more complex and discriminative features.
* **Example:** Let's say we have an input feature map of size 28x28x64 (height x width x channels) and we apply ReLU activation to it. All negative values in the feature map will be set to zero, while positive values will remain unchanged.
* **Padding:** Padding is a technique used to preserve the spatial dimensions of the feature map after convolution and pooling. It involves adding extra rows and columns of zeros around the edges of the input image, which allows the filters to see the pixels on the edge of the input image and generate a feature map with the same spatial dimensions as the input image.
* **Example:** Let's say we have an input image of size 32x32x3 and we apply a filter of size 5x5x3 with a stride of 1 and no padding. This produces an output feature map of size 28x28x1. If we want to preserve the spatial dimensionality and produce an output feature map of size 32x32x1, we need to add a border of 2 pixels around the input image (1 pixel on each side). This is called zero-padding, since the extra border pixels are typically set to zero.

**Q.7) Why Relu Activation is preferred in hidden layer of CNN?**

* **The Rectified Linear Unit (ReLU) activation function is commonly used in the hidden layers of Convolutional Neural Networks (CNNs) because of several advantages it offers:**
* **Non-linearity:** ReLU introduces non-linearity into the CNN by allowing the network to model more complex functions. This is because the ReLU function is non-linear and has a piecewise linear structure, which can approximate any non-linear function to a high degree of accuracy.
* **Computationally Efficient:** The ReLU function is computationally efficient to compute compared to other activation functions such as sigmoid or hyperbolic tangent. This is because the ReLU function only involves simple element-wise operations, which can be parallelized and optimized to run efficiently on modern hardware.
* **Sparse Activation:** ReLU produces sparse activations, where only a fraction of the neurons in the hidden layer are activated at any given time. This can help to reduce the problem of overfitting by preventing co-adaptation between neurons. Sparse activations also help to reduce the computational cost of training and inference.
* **Robustness to Vanishing Gradient:** ReLU is less susceptible to the vanishing gradient problem that can occur in deep neural networks. This is because the ReLU function has a derivative that is either 0 or 1, which ensures that gradients are not exponentially diminished as they propagate through the network.
* **Biological Plausibility:** ReLU has been shown to be biologically plausible and to model the behavior of real neurons in the human brain. This suggests that ReLU may be a more natural and effective activation function for modeling complex visual processing tasks such as image recognition.

The ReLU activation function is preferred in the hidden layers of CNNs because of its non-linearity, computational efficiency, sparse activation, robustness to vanishing gradients, and biological plausibility. These properties make ReLU a powerful tool for learning complex and discriminative features from images and other types of data.

**Q.8) Explain Lenet Architecture, With Diagram and Trainable params?**

* **LeNet** is a pioneering Convolutional Neural Network (CNN) architecture developed by Yann LeCun et al. in 1998 for handwritten digit recognition. The architecture consists of seven layers, including convolutional layers, max pooling layers, and fully connected layers.
* The LeNet architecture has a total of approximately 60,000 trainable parameters. Let's break down the number of trainable parameters in each layer:

|  |
| --- |
|  |

* **Convolution layer 1:** This layer has 156 trainable parameters, which correspond to the weights of the 6 filters of size 5x5. Each filter has 5x5=25 weights, and there are 6 filters in total. Additionally, there are 6 biases, one for each filter.
* **Average pooling layer 1:** This layer has no trainable parameters.
* **Convolution layer 2:** This layer has 2,416 trainable parameters, which correspond to the weights of the 16 filters of size 5x5. Each filter has 5x5x6=150 weights, since there are 6 input feature maps from the previous layer. There are 16 filters in total, so the total number of weights is 150x16=2,400. Additionally, there are 16 biases, one for each filter.

|  |
| --- |
|  |

* **Average pooling layer 2**: This layer has no trainable parameters.
* **Fully connected layer 1:** This layer has 48,120 trainable parameters, which correspond to the weights of the 120 neurons in the layer. The input to this layer is a flattened vector of size 400, which is the concatenation of the 16 feature maps from the previous layer. Thus, each neuron has 400 weights, and there are 120 neurons in total. Additionally, there are 120 biases, one for each neuron.
* **Fully connected layer 2:** This layer has 10,164 trainable parameters, which correspond to the weights of the 84 neurons in the layer. The input to this layer is the output of the previous layer, which is a vector of size 120. Thus, each neuron has 120 weights, and there are 84 neurons in total. Additionally, there are 84 biases, one for each neuron.
* **Output layer:** This layer has 850 trainable parameters, which correspond to the weights of the 10 neurons in the layer. The input to this layer is the output of the previous layer, which is a vector of size 84. Thus, each neuron has 84 weights, and there are 10 neurons in total. Additionally, there are 10 biases, one for each neuron.

**Q.9) Explain ALexnet Architecture, With diagram and Trainable params?**

* The AlexNet architecture is a deep convolutional neural network that achieved state-of-the-art performance on the ImageNet dataset in 2012.

|  |
| --- |
|  |

* **Convolutional layer 1:** This layer has 96 filters of size 11x11x3 (where 3 is the number of input channels for RGB images), with a stride of 4 pixels and padding of size 0. This layer has approximately 35 million trainable parameters.
* **Max pooling layer 1:** This layer performs max pooling over non-overlapping regions of size 3x3, with a stride of 2 pixels. This layer has no trainable parameters.
* **Convolutional layer 2**: This layer has 256 filters of size 5x5x48 (where 48 is the number of output channels from the previous layer), with a stride of 1 pixel and padding of size 2. This layer has approximately 614,000 trainable parameters.
* **Max pooling layer 2:** This layer performs max pooling over non-overlapping regions of size 3x3, with a stride of 2 pixels. This layer has no trainable parameters.
* **Convolutional layer 3:** This layer has 384 filters of size 3x3x256, with a stride of 1 pixel and padding of size 1. This layer has approximately 885,000 trainable parameters.
* **Convolutional layer 4:** This layer has 384 filters of size 3x3x192, with a stride of 1 pixel and padding of size 1. This layer has approximately 1.3 million trainable parameters.
* **Convolutional layer 5:** This layer has 256 filters of size 3x3x192, with a stride of 1 pixel and padding of size 1. This layer has approximately 884,000 trainable parameters.
* **Max pooling layer 3:** This layer performs max pooling over non-overlapping regions of size 3x3, with a stride of 2 pixels. This layer has no trainable parameters.
* **Fully connected layer 1:** This layer has 4,096 neurons, each connected to all 6x6x256=92,160 neurons in the previous layer. This layer has approximately 167 million trainable parameters.
* **Dropout layer 1:** This layer randomly drops out 50% of the activations from the previous layer during training, in order to prevent overfitting. This layer has no trainable parameters.
* **Fully connected layer 2:** This layer has 4,096 neurons, each connected to all 4,096 neurons in the previous layer. This layer has approximately 67 million trainable parameters.
* **Dropout layer 2:** This layer randomly drops out 50% of the activations from the previous layer during training. This layer has no trainable parameters.
* **Output layer:** This layer has 1,000 neurons, each corresponding to one of the classes in the ImageNet dataset. This layer has approximately 4 million trainable parameters.
* The AlexNet architecture has a total of approximately 60 million trainable parameters, making it significantly larger than the LeNet architecture. The majority of the parameters are in the fully connected layers, which have a large number of neurons.

**Q.10) How ResNet(Residual Network) is Different from other CNN? Explain in details?**

* The main difference between ResNet and other CNNs is the use of residual connections, which allow information to bypass some layers in the network and be directly propagated to subsequent layers. This helps to alleviate the problem of vanishing gradients, which can occur when training deep neural networks.
* To understand the importance of residual connections, it is helpful to first consider how traditional CNNs process information. In a typical CNN, the input data is passed through a series of convolutional layers, each followed by a non-linear activation function (such as ReLU) and a pooling layer. This process is repeated several times, with the output of each layer being passed as input to the next layer.
* The problem with this approach is that as the network gets deeper, it becomes harder to train, due to the vanishing gradient problem. This occurs when the gradients flowing through the network become very small, making it difficult for the network to learn.
* ResNet addresses this problem by introducing "shortcut connections" (also called "skip connections" or "identity mappings") that allow information to bypass some layers and be directly propagated to subsequent layers. This is achieved by adding an identity mapping between two layers, so that the output of one layer is added to the input of the next layer. This allows gradients to flow more easily through the network and reduces the risk of them vanishing.
* Another important aspect of ResNet is its use of "bottleneck" layers, which reduce the number of parameters in the network and improve its computational efficiency. A bottleneck layer consists of three convolutional layers: a 1x1 convolution that reduces the dimensionality of the input, a 3x3 convolution that performs the main processing, and another 1x1 convolution that increases the dimensionality back to the original size.
* ResNet has been shown to be highly effective at image classification and other computer vision tasks. It has won several ImageNet challenges and has been widely adopted in both industry and academia. Its ability to train very deep neural networks has led to improvements in many areas of computer vision, and its ideas have been extended to other types of neural networks as well.

**Q.11) How InceptionNet is different from other CNN? Explain in details?**

* **InceptionNet** is a type of convolutional neural network (CNN) that is designed to be computationally efficient while still achieving high accuracy on image classification tasks. It is different from other CNNs in the way it processes information by using a combination of convolutional layers of different sizes and pooling operations in parallel.
* In a traditional CNN, the input data is passed through a series of convolutional layers, followed by pooling layers and non-linear activation functions. These layers typically have fixed filter sizes, which limits the flexibility of the network to capture features at different scales. In contrast, InceptionNet uses a set of convolutional layers of different sizes (1x1, 3x3, and 5x5) in parallel, allowing it to capture features at different scales.
* Another key aspect of InceptionNet is the use of "inception modules," which are blocks of convolutional layers that are designed to capture different types of features. Each inception module contains a series of parallel convolutional layers, each with a different filter size, as well as a pooling layer. The outputs of these layers are then concatenated and passed on to the next layer. The use of parallel convolutional layers and pooling operations allows InceptionNet to capture features at different scales and to efficiently process large volumes of data. It also allows it to achieve high accuracy on image classification tasks while using fewer parameters and computational resources than other CNNs.
* One downside of InceptionNet is that it can be more difficult to train than other CNNs due to the complexity of its architecture. To address this issue, InceptionNet has been extended with various modifications, such as the use of batch normalization, residual connections, and other techniques to improve training stability and performance.
* InceptionNet represents a significant advancement in the field of computer vision, and its ideas have been widely adopted in other CNN architectures. Its ability to efficiently process large volumes of data and achieve high accuracy on image classification tasks has made it a popular choice for both research and practical applications.