Artificial Intelligence
Module 5 Assignment

Q. No: 1. Briefly describe the disadvantages of Flattening (as the first layer or after Input) and how you overcome those issues.

Answer: The main disadvantage of using flattening as the first layer or immediately after the input layer in a deep neural network is the loss of spatial information. By converting a multi-dimensional input (such as an image) into a one-dimensional array, information about the spatial arrangement of features in the original data is lost. This can negatively impact the performance of the model, especially in tasks where spatial relationships are important, such as image classification and segmentation.

To overcome this issue, there are several approaches that can be used,

a. Using Convolutional Neural Networks (CNNs):

- Instead of flattening the input immediately, CNNs use convolutional and pooling layers to maintain the spatial information in the data.
- This allows the model to learn more robust features and improve its performance in tasks where spatial information is important.

b. Using Recurrent Neural Networks (RNNs):

- RNNs are designed to handle sequential data and maintain the spatial information in the data by passing information from one step to the next.
- This can be useful in tasks where spatial information is important, such as video classification or speech recognition.

c. Using Autoencoders:

- Autoencoders are neural networks that are trained to reconstruct the input data from a lower-dimensional representation.
- The lower-dimensional representation, known as the bottleneck, can be used to preserve the spatial information in the data while reducing its dimensionality.

d. Using Multiple Flattening Layers:

Instead of flattening the data immediately after the input layer, multiple flattening layers can be used, each preserving some of the spatial information in the data. This allows the model to learn a hierarchy of features, with the later layers capturing more abstract features while the earlier layers capture more spatial information.

These are just a few examples of approaches that can be used to overcome the disadvantages of flattening. The specific approach used will depend on the requirements of the task and the data being processed.

Q. No: 2. Stanford Vision department, has collected and annotated images of 120 breeds of dogs from ImageNet. Dataset can be found here.

a. Prepare an EDA sheet & explain what you understand from this data. What would be your approach?

b. With a limited number of training images per class, what extra steps will you incorporate for a good model?

c. Iterate over and try different architectures and topologies and preserve the results of each experiment using TensorBoard. Also, save major metrics in Excel (or try W&B).

Answer: With this dataset, I performed the EDA, in which 10222 image files were available having the file names labeled with a total of 120 classes.

I have trained the model with 2 different architectures, and another using a pre-trained model, which has collected all the loss & accuracy metrics in Excel files with two different optimizers i.e., SGD and Adam.

Refer to the "Final_assignment_files" folder to see the ipynb, TensorBoard log, and metrics accuracy with different architectures.

Q. No: 3. Achieve the above problem on Kaggle itself and submit your results.

Answer: Kindly visit by clicking this, **Dogs Breed Classification**

Q. No: 4. What are Convolutional Neural Networks and how are they better than regular only Fully Connected Neural Networks?

Answer: Convolutional Neural Networks (ConvNets or CNNs) are a class of deep neural networks, mainly used for computer vision tasks such as image classification, object detection, and segmentation. They have also been applied to other areas, such as natural language processing and speech recognition.

A CNNs is designed to process data with a grid-like topology, such as an image, which can be thought of as a matrix of pixels. A key feature of CNNs is that they use convolutional layers, which apply filters to the input data and produce feature maps. These filters can be thought of as detecting certain features, such as edges, corners, or textures, in the input data.

The convolutional layers are followed by non-linear activation functions, such as the Rectified

Linear Unit (ReLU), which introduce non-linearity into the model. Pooling layers are also commonly used to reduce the spatial dimensions of the feature maps, making the model more robust to translations in the input data and reducing the computational cost.

Compared to fully connected neural networks (FCNs), CNNs have several advantages:

a. Spatial Hierarchy: ConvNets are able to learn and exploit the spatial hierarchy of features in the input data, where simple features, such as edges and corners, are combined to form more complex features, such as shapes and parts of objects.

b. Sparsity of Connections: In ConvNets, each neuron in a layer only receives input from a small local region of the previous layer, reducing the number of parameters and making the model more computationally efficient.

c. Translation Invariance: Pooling layers in ConvNets help the model become more invariant to translations in the input data, meaning that it can recognize the same object even if it appears in different locations in the image.

d. Shared Weights: In ConvNets, the same filters are applied to each region of the input data, which allows for the same features to be detected regardless of the location in the input data. This makes the model more robust to changes in the input data and reduces the number of parameters in the model.

In conclusion, CNNs are better suited for image-related tasks due to their ability to learn spatial hierarchies of features, their sparsity of connections, translation invariance, and shared weights.

Q. No: 5. The major components of a CNN topology are Filter/ Kernel, Strides, etc. Explain mathematically the output of a single CNN layer and how it is impacted by strides?

Answer:

A Convolutional Neural Network (CNN) layer is designed to detect local patterns in the input data and extract features from them. The output of a single CNN layer is the result of applying a set of filters, also known as kernels, to the input data. The convolution operation between the input and the filters results in a feature map.

Mathematically, the output of a single CNN layer can be represented as follows:

$$O(i, j, k) = \text{sum over all } m,n \text{ of } (I(i+m, j+n, k) * W(m, n, k)) + b(k)$$

Where:

O(i, j, k) is the value of a pixel in the output feature map at position (i, j) and channel k.

I(i, j, k) is the value of a pixel in the input at position (i, j) and channel k.
W(m, n, k) is the value of the filter at position (m, n) and channel k.
b(k) is the bias term for channel k.

The strides refer to the number of pixels that the filter moves in the input data for each computation. The larger the strides, the fewer the number of computations, resulting in a reduction of computational cost. However, this reduction in computational cost comes at the cost of reducing the spatial resolution of the output feature map.

For example, if the stride is set to 2, then the output feature map will have half the spatial resolution in both dimensions compared to the input.

Mathematically, the impact of strides on the output feature map can be represented as follows:

$$O(i, j, k) = \text{sum over all } m,n \text{ of } (I(is+m, js+n, k) * W(m, n, n)) + b(k)$$

Where:

s is the stride length.

In summary, strides in CNNs trade off the spatial resolution of the output feature map with the computational cost of the convolution operation.

Q. No: 6. For a single channel image, size (9x9) and a filter of size (2x2) and stride (2x2), calculate the output. Is padding required? What is the output size of this convoluted feature?

Answer:

The output size of a convolution with a filter of size (2x2) and stride (2x2) on an image of size (9x9) can be calculated using the formula:

$$\text{Output size} = \{(\text{input size} - \text{filter size}) / \text{stride}\} + 1$$

In this case:

$$\text{Output size} = \{(9 - 2) / 2\} + 1 = 4$$

Therefore, the output size of this convolution will be (4x4).

Padding is not required in this case as the output size is not an issue. If the goal is to preserve the spatial dimensions of the input, then padding might be required. The formula for padding calculation is:

$$\text{Padding} = (\text{filter size} - 1) / 2$$

In this case:

padding = (2 - 1) / 2 = 0.5 (not possible)

So, padding is not required and the output size of this convolution will be (4x4).

Q. No: 7. For a particular problem if the error is not dipping below a standard point, what are the techniques used to reduce error ? Explain the methodology in depth and with examples.

Answer:

There are several techniques used to reduce the error in deep neural networks, including the following:

a. Regularization: Regularization is a method used to prevent overfitting by adding a penalty term to the loss function. This penalty term discourages the model from learning the noise in the data and helps to reduce the error. Examples of regularization techniques include L1 regularization, L2 regularization, and dropout.

b. Early Stopping: Early stopping is a technique that involves monitoring the validation loss and stopping the training process when the validation loss stops improving. This can help to prevent overfitting and reduce the error.

c. Data augmentation: Data augmentation involves artificially generating new training data by transforming the existing data. This can help to increase the size of the training dataset and reduce overfitting, resulting in a reduction of error. Examples of data augmentation techniques include random rotation, flipping, and scaling.

d. Hyperparameter tuning: Hyperparameters are parameters that control the behaviour of the model and are set before training begins. Poorly chosen hyperparameters can lead to overfitting and high error. Hyperparameter tuning involves searching for the best combination of hyperparameters to reduce the error. This can be done using techniques such as grid search or random search.

e. Transfer learning: Transfer learning is a technique that involves using a pre-trained model as the starting point for a new model. The pre-trained model is fine-tuned for the new task, allowing it to learn from the existing knowledge and reducing the error.

f. Ensemble methods: Ensemble methods involve combining multiple models to form a single model. This can help to reduce the error by combining the strengths of different models and reducing the variance. Examples of ensemble methods include bagging, boosting, and stacking.

In conclusion, reducing errors in deep neural networks requires a combination of techniques such as regularization, early stopping, data augmentation, hyperparameter tuning, transfer learning,

and ensemble methods. The choice of technique depends on the specific problem and the nature of the data, but using a combination of techniques can lead to the best results.