Assignment no-6

Aim: Object detection using Transfer Learning of CNN architectures:

a. Load in a pre-trained CNN model trained on a large dataset
b. Freeze parameters(weights) in the model's lower convolutional layers
c. Add a custom classifier with several layers of trainable parameters to model
d. Train classifier layers on training data available for the task
e. Fine-tune hyperparameters and unfreeze more layers as needed

Objective: Student will learn:

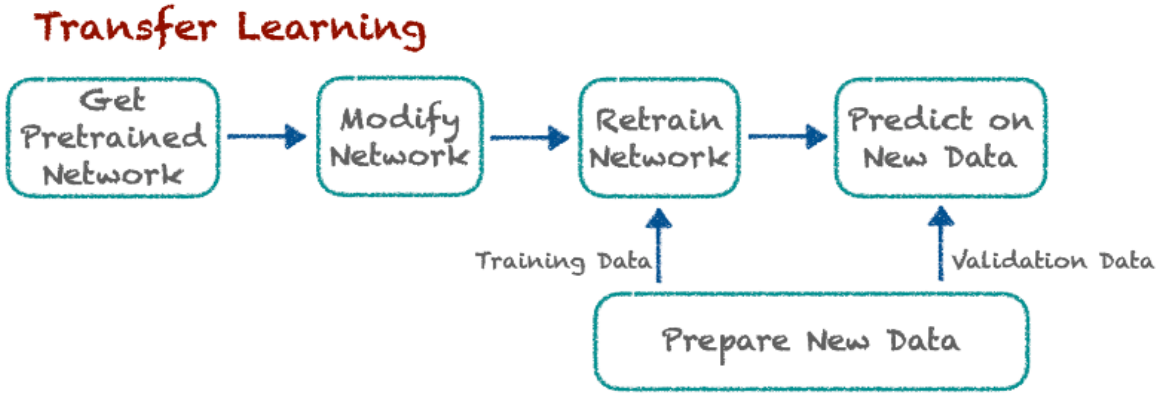
- Understand how to use Tensorflow Eager and Keras Layers to build a neural network architecture.
- Understand how a model is trained and evaluated.
- Transfer learning is flexible, allowing the use of pre-trained models directly as feature extraction preprocessing and integrated into entirely new models.
- Keras provides convenient access to many top performing models on the ImageNet image recognition tasks such as VGG, Inception, and ResNet.
- Our main goal is to train a neural network (using Keras) to obtain > 90% accuracy on Caltech dataset

Methodology to be used

- Deep Learning Convolutional Neural Networks Keras

Theory :

What is transfer learning?
Transfer learning uses pre-trained models from one machine learning task or dataset to improve performance and generalizability on a related task or dataset.
Transfer learning is a machine learning technique in which knowledge gained through one task or dataset is used to improve model performance on another related task and/or different dataset.transfer learning uses what has been learned in one setting to improve generalization in another setting.
Transfer learning is a technique that allows machines to exploit the knowledge gained from a previous task to improve generalization about another. It is a fundamental concept behind the development of models like ChatGPT and Google Gemini, and it aids in many important and useful tasks, like summarizing long documents, drafting complex essays, organizing trips, or even writing poems and songs.



What is Data augmentation?

Data augmentation is a technique that artificially increases the size and variety of a dataset by making small changes to existing data. It's a key part of training machine learning (ML) models, which need large and diverse datasets to learn accurately.

Data augmentation can help: Reduce overfitting, Improve model robustness, Correct imbalanced datasets, Increase dataset size and variability, and Extract additional information from the original dataset.
Data augmentation improves machine learning model optimization and generalization. In other words, data augmentation can reduce overfitting and improve model robustness.
Data augmentation has been widely implemented in research for a range of computer vision tasks, from image classification to object detection. As such, there is a wealth of research on how augmented images improve the performance of state-of-the-art convolutional neural networks (CNNs) in image processing.

Computer vision

Data augmentation is a central technique in computer vision tasks. It helps create diverse data representations and tackle class imbalances in a training dataset.

The first usage of augmentation in computer vision is through position augmentation. This strategy crops, flips, or rotates an input image to create augmented images. Cropping either resizes the image or crops a small part of the original image to create a new one. Rotation, flip, and resizing transformation all alter the original randomly with a given probability of providing new images.

Another usage of augmentation in computer vision is in color augmentation. This strategy adjusts the elementary factors of a training image, such as its brightness, contrast degree, or saturation.

Audio data augmentation

Audio files, such as speech recordings, are also a common field where you can use data augmentation. Audio transformations typically include injecting random or Gaussian noise into some audio, fast-forwarding parts, changing the speed of parts by a fixed rate, or altering the pitch.

Text data augmentation

Text augmentation is a vital data augmentation technique for NLP and other text-related sectors of ML. Transformations of text data include shuffling sentences, changing the positions of words, replacing words with close synonyms, inserting random words, and deleting random words.

Neural style transfer

Neural style transfer is an advanced form of data augmentation that deconstructs images into smaller parts. It uses a series of convolutional layers that separate the style and context of an image, producing many images from a single one.

Adversarial training

Changes on the pixel level create a challenge for an ML model. Some samples include a layer of imperceptible noise over an image to test the model’s ability to perceive the image underneath. This strategy is a preventative form of data augmentation focusing on potential unauthorized access in the real world.

In machine learning, preprocessing involves transforming a raw dataset so the model can use it. This is necessary for reducing the dimension, identifying relevant data, and increasing the performance of some machine learning models. It involves transforming or encoding data so that a computer can quickly parse it.

What are pre trained Neural Network models ?
Pre-trained neural network models are deep learning models that have been trained on large datasets to perform a specific

task. They can be used as a starting point for developing machine learning (ML) models, and can significantly improve the performance of the model.

Some examples of pre-trained neural network models include:

- Convolutional neural networks (CNNs): Used to categorize images into predetermined categories
- Region-based convolutional neural networks (R-CNNs): Used to recognize and categorize items in photos or videos.
- Recurrent neural networks (RNNs): Used to predict the next word in a sequence.
- VGG-16: A CNN model developed by the Visual Geometry Group (VGG) at the University of Oxford

Explain Pytorch library ?

PyTorch is an open-source machine learning library that helps developers build deep learning models.PyTorch is an optimized Deep Learning tensor library based on Python and Torch and is mainly used for applications using GPUs and CPUs. PyTorch is favored over other Deep Learning frameworks like TensorFlow and Keras since it uses dynamic computation graphs and is completely Pythonic.
PyTorch offers two main features:

- Tensor computation: Similar to NumPy, but with GPU acceleration.
- Automatic differentiation: Creates and trains deep neural networks.

What is the PyTorch Transforms module?

The PyTorch Transforms module is a collection of image transformations that can be used to manipulate data and prepare it for training machine learning algorithms:

- ToTensor: Converts a PIL image or NumPy ndarray into a FloatTensor, while also scaling the image's pixel intensity values.
- Compose: Allows you to chain together multiple transformations
- Functional transforms: Provide more granular control over transformations, which can be useful for building more complex transformation pipelines
- Conversion Transforms: Allows you to convert to and from PIL images

Some examples of transformations include:

- Cropping
- Padding
- Rotating
- Flipping
- Random affine transformation
- Random perspective transformation
- Elastic transformations
- Converting to grayscale or RGB
- Adding Gaussian noise
- Inverting colors

What is PyTorch Training?

PyTorch training is the process of using a PyTorch optimizer to update the weights of a neural network model. This is done in order to minimize the value of a loss function. Training is important to ensure that a neural network model is able to generalize well and make predictions for data it has not seen before.

Efficient data handling in PyTorch is achieved via two main classes:

- The dataset
- The data loader

The dataset is responsible for accessing and processing single instances of your data from your dataset. There are a number of datasets available in the PyTorch domain APIs. You can make your own datasets using provided subclasses or by subclassing the dataset parent class yourself.

The data loader pulls instances of data from the dataset either automatically or with a sampler that you define, collects them in batches and returns them for consumption by your training loop. The data loader works with all kinds of data sets regardless of the type of data they contain in the PyTorch domain APIs: torch vision, torch text and torch audio. Give access to a collection of open labeled data sets that you may find useful for your own training purposes.

TorchVision, TorchText and TorchAudio:

Torchvision contains a broad array of data sets labeled for classification object detection and object segmentation. It also contains the convenience classes image folder and dataset folder which allow you to easily create a dataset from images or other data accessible on your file system.

Torchtext offers data sets labeled for a variety of classification translation and analysis tasks.

Torch audio gives access to data sets labeled for transcription and music genre detection. Most of the time, you will know the size of your data set and be able to access arbitrary single instances of it.

Explain VGG-16 model from Pytorch.?

The VGG-16 model is a convolutional neural network (CNN) model in PyTorch that's known for its depth and ability to classify images:

Layers-

The VGG-16 model has 16 layers, including 13 convolutional layers and 3 fully connected layers.

Architecture-

The VGG-16 model is defined as a subclass of nn.Module, which is the base class for all neural network modules in PyTorch.

Convolutional layers-

The first 13 layers are convolutional layers with batch normalization and ReLU activation functions.

Max pooling-

Max pooling is applied after layers 2, 4, 7, and 10 to reduce spatial dimensions.

Fully connected layers-

The last three layers are fully connected layers with dropout and ReLU activation functions.

Training-

To train the model for more than 2 epochs, set resume_training to True after the first run.

Disabling training-

To disable training for the convolutional layers, set requre_grad = False.

The VGG-16 model was created by K. Simonyan and A. Zisserman, researchers at the University of Oxford, in 2014. It won the ImageNet competition in 2014.

Conclusion: In this experiment, we were able to see the basics of using PyTorch as well as the concept of transfer learning, an effective method for object recognition. Instead of training a model from scratch, we can use existing architectures that have been trained on a large dataset and then tunethem for our task. This reduces the time to train and often results in better overall performance. The outcome of this experiment is knowledge of transfer learning and PyTorch that we can build on to build more complex applications.