Experiment No: 06

Aim: Implementation of object detection using transfer learning of CNN architectures

Problem Statement:

Implementation of object detection using transfer learning of CNN architecture.

a. Load in a pre-trained CNN model trained on a large dataset

b. Freeze parameters (weights) in model's lower convolutional layers

c. Add custom classifier with several layers of trainable parameters to model

d. Train classifier layers on training data available for task

e. Fine-tune hyperparameters and unfreeze more layers as needed

Objectives:

To detect the objects in an image by implementing a transfer learning deep learning approach using existing CNN architectures.

Theory:

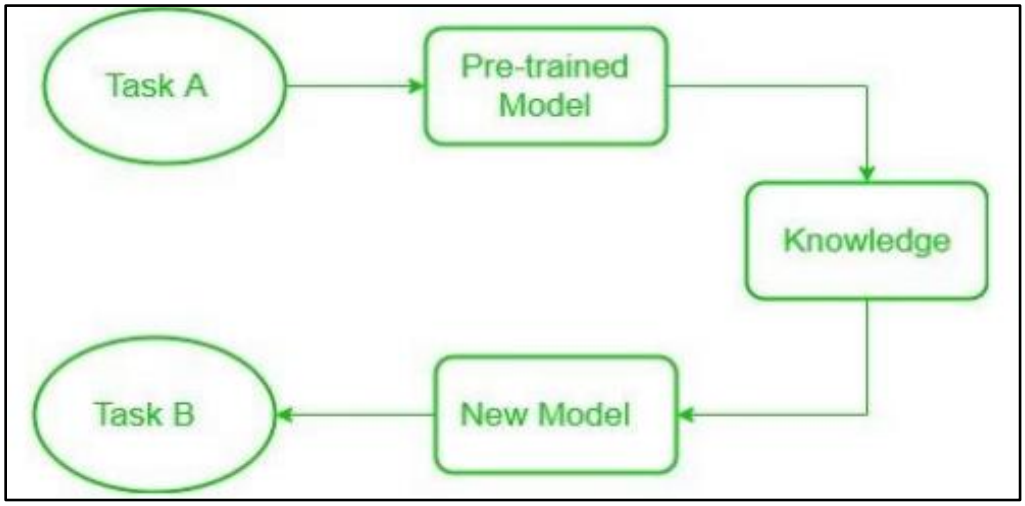
Whenever we encounter a new problem or a task, we recognize it and apply our relevant knowledge from our previous learning experiences. This makes our work easy and fast to finish. For instance, if you know how to ride a bicycle and if you are asked to ride a motorbike which you have never done before. In such a case, our experience with a bicycle will come into play and handle tasks like balancing the bike, steering, etc. This will make things easier compared to a complete beginner. Following the same approach, a term was introduced Transfer Learning. This approach involves the use of knowledge that was learned in some task and applying it to solve the problem in the related target task.

Need of transfer learning:

Many deep neural networks trained on images have a curious phenomenon in common: in the early layers of the network, a deep learning model tries to learn a low level of features, like detecting edges, colors, variations of intensities, etc. Such kind of features appears not to be specific to a

particular dataset or a task because no matter what type of image we are processing either for detecting a lion or car. In both cases, we have to detect these low-level features. All these features occur regardless of the exact cost function or image dataset. Thus learning these features in one task of detecting lion can be used in other tasks like detecting humans.

Necessity for transfer learning: Low-level features learned for task A should be beneficial for learning of model for task B. This is what transfer learning is. Nowadays, it is very hard to see people training a whole convolutional neural network from scratch, and it is common to use a pre-trained model trained on a variety of images in a similar task, e.g models trained on ImageNet.
The Block diagram is shown below as follows:

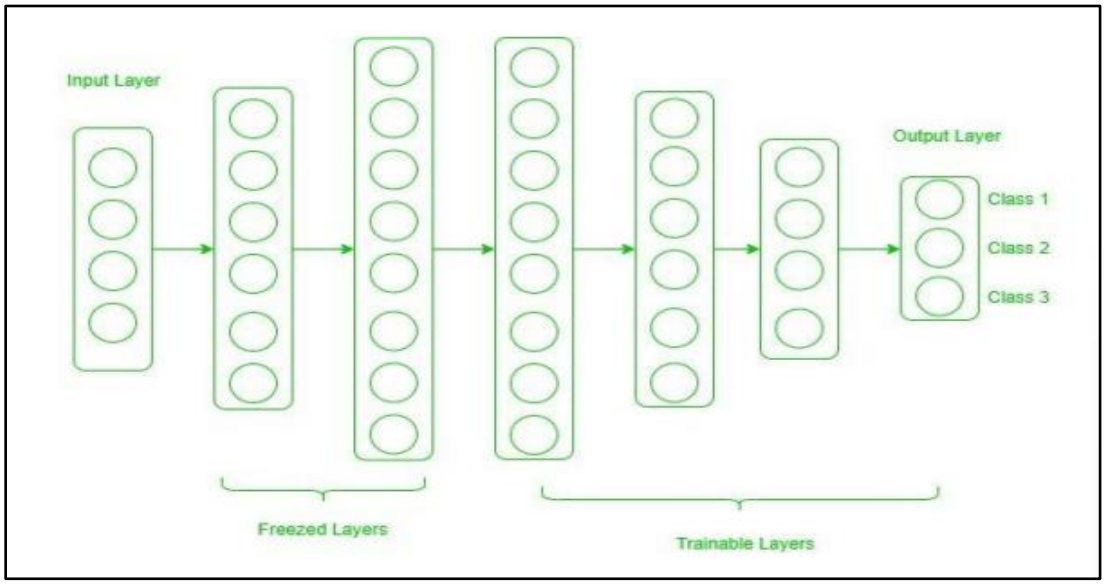


When dealing with transfer learning, we come across a phenomenon called freezing of layers. A layer, it can be a CNN layer, hidden layer, a block of layers, or any subset of a set of all layers, is said to be fixed when it is no longer available to train. Hence, the weights of freezed layers will not be updated during training. While layers that are not freezed follows regular training procedure. When we use transfer learning in solving a problem, we select a pre-trained model as our base model. Now, there are two possible approaches to use knowledge from the pre-trained model. First way is to freeze a few layers of pre-trained model and train other layers on our new dataset for the new task. Second way is to make a new model, but also take out some features from the layers in the pre-trained model and use them in a newly created model. In both cases, we take out some of the learned features and try to train the rest of the model. This makes sure that the only feature that

may be same in both of the tasks is taken out from the pre-trained model, and the rest of the model is changed to fit new dataset by training.

Now, one may ask how to determine which layers we need to freeze and which layers need to train. The answer is simple, the more you want to inherit features from a pre-trained model, the more you have to freeze layers. For instance, if the pre-trained model detects some flower species and we need to detect some new species. In such a case, a new dataset with new species contains a lot of features similar to the pre-trained model. Thus, we freeze less number of layers so that we can use most of its knowledge in a new model. Now, consider another case, if there is a pre-trained model which detects humans in images, and we want to use that knowledge to detect cars, in such a case where dataset is entirely different, it is not good to freeze lots of layers because freezing a large number of layers will not only give low level features but also give high-level features like nose, eyes, etc which are useless for new dataset (car detection). Thus, we only copy low-level features from the base network and train the entire network on a new dataset. Let's consider all situations where the size and dataset of the target task vary from the base network.

Freezed and Trainable Layers:



Target dataset is small and similar to the base network dataset: Since the target dataset is small, that means we can fine-tune the pre-trained network with target dataset. But this may lead

to a problem of overfitting. Also, there may be some changes in the number of classes in the target task. So, in such a case we remove the fully connected layers from the end, maybe one or two, and add a new fully-connected layer satisfying the number of new classes. Now, we freeze the rest of the model and only train newly added layers.

Target dataset is large and similar to base training dataset: In such case when the dataset is large and it can hold a pre-trained model there will be no chance of overfitting. Here, also the last full-connected layer is removed, and a new fully-connected layer is added with the proper number of classes. Now, the entire model

is trained on a new dataset. This makes sure to tune the model on a new large dataset keeping the model architecture the same.

Target dataset is small and different from the base network dataset: Since the target dataset is different, using high-level features of the pre-trained model will not be useful. In such a case, remove most of the layers from the end in a pre-trained model, and add new layers the satisfying number of classes in a new dataset. This way we can use low-level features from the pre-trained model and train the rest of the layers to

fit a new dataset. Sometimes, it is beneficial to train the entire network after adding a new layer at the end.

Target dataset is large and different from the base network dataset: Since the target network is large and different, best way is to remove last layers from the pre-trained network and add layers with a satisfying number of classes, then train the entire network without freezing any layer. Transfer learning is a very effective and fast way, to begin with, a problem. It gives the direction to move, and most of the time best results are also obtained by transfer learning.

Conclusion:

Thus we have studied to implement transfer learning deep learning approach whereby a neural network model is first trained on a problem similar to the problem that is being solved. One or more layers from the trained model are then used in a new model trained on the problem of interest. We also observed that transfer learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error.