

Machine Learning for Engineering and Science Applications
Professor Dr. Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of technology Madras
Transfer Learning

In this video we will look at Transfer Learning. One of the strategies used for training deep neural networks when the number of data points available for training for the particular task that you are interested in is very low.

(Refer Slide Time: 0:31)

Modes of using a Pre-Trained Model

- 1) **CNN as a feature Extractor:**
 - a) Remove the classification layer in the network. ✓
 - b) Forward pass the data through a pre-trained network (say VGG-16) and store the embedding (4096-D representation of the data)
 - c) Use the embeddings to train traditional machine learning algorithms such as SVM or Logistic regression to classify data appropriately.
- 1) **Fine-tuning of CNN:**
 - a) Initialize a network with pre-trained weights ✓
 - b) Modify the classification layers from 1000 neurons to number of classes in the new dataset. (Random Initialization)
 - c) Train the network. This is called transfer learning ✓
↓
fine tuning



So, the idea here is, to use what we can refer to as a Pre-Trained model. Okay, so for instance, let us say you have some data wherein you are trying to figure out for a particular task, okay, it's not an ImageNet challenge. For instance, you are trying to determine a species of birds or easiest thing you have data, you have a limited number of data to determine the make and model of a car depending from the pictures. This is some hypothetical task but then you don't have enough data, maybe you have thousands of data points but that's not sufficient for training a deep neural network.

So, what you do is, you would take a network like Alex net or VGG or Inception. We just been trained on image ImageNet database. Do a forward pass through a pre-trained network, (())(1:38) forward pass the data, right? And store the embedding. So if in the case of Alex net or VGG you have about 4096 neurons in the output layer before the classification there. To

take this as a representation of your data. So now, your input data is now represented by 4096 length feature vector. That is the feature that represent the data.

And you use this as input to another machine learning framework let's say an SVM support vector machine or binary trees or maybe just another neural network that you can train with this data and classified appropriately. Okay, so this is a strategy that works very well, that is taking the embedding in this case this 4096-D vector would be the embedding that you get from your CNN. So this strategy works very well if the task at hand is a data for the task at hand or the task at hand is similar to something that ImageNet accomplishes. In this case since we are talking about ImageNet, we will take that as the standard.

So, ImageNet networks, so the networks strained on ImageNet data. Pretty much it's a classification net or image recognition network strained on thousands of types of images, okay. And you can safely assume that it is learned all kinds of possible features, okay. Corresponding to the images from the wild present in the ImageNet database.


Now, if your input data is very similar, in our case example I quoted that no types of type and make and model of the car or let's say even type of species of birds or for instance species of cats or dogs, okay. That data is kind of similar to Imagenet data. So if you have a network trainer ImageNet data maybe then you can use this 4096-D representation as a reduce representation of your input data. And then use another machine learning paradigms to train it in order to accomplish your task.

However in some cases, the data might not exactly be the same or maybe there is probably have a slightly larger dataset. In that case what you can do is, you can take the network with pre-trained weights, okay. Modify the classification layers from 1000 neurons to number of classes in your new dataset that your task demands, and then train the network. So this portion of training the network is basically training the network with whatever data you have for your task. And this process is this training with your excess data is referred to as fine tuning, okay.

And in general this is what people refer to as transfer learning, okay. So we start with a network that has been trained on a very large database and most often the not that image database is very similar or the task to be accomplished is very similar to the task that you have now proposing to do with your current deep CNN. And then you use your limited


dataset to modify the weights in your network appropriately by using backprop. And this typically works very well in many cases.

(Refer Slide Time: 5:25)



Transfer Learning

- Convolutional Neural Networks are data hungry & generating an extremely large database of labeled data is expensive.
- In deep CNNs:
 - Deeper Layers learn task specific features such as features specific to certain breed of dog or cat. These layers are easier to train as error from classification layer can be easily backpropagated
 - Initial layers learn more generic features such as edges, patterns. Tougher to train when compared to deeper layers due to vanishing gradients.
- Transfer learning : Knowledge of an already trained Machine Learning model is applied to a different but related problem.



So the reason it works as I stated earlier is that, if you have a fairly large database which is labelled as like the ImageNet database, okay. Considerably effort has gone into making that so, and if you have a network trained on it, couple of things. One is that, given the size of the database and the depth of the network you have trained on. Safe to assume that the network has learned all kinds of low-level features which are transferable to other tasks.

So here the deeper layers let's say for some CNN trained on ImageNet, the deeper layers learnt task specific features. So for instance features specific to certain breed of dog or cat. These are easier to train as these are closer to the classification layer so there is back propagate faster. The initial layers learn more generic low-level features like edges, blocks, some kind of patterns in the picture. And generally difficult to train because of the errors being difficult to backprop the errors from the output layer to the innermost layers.

So, if you have a network which is trained on a fairly large database on a task which is kind of very similar closer to what your, at least related to what you are going to do, then it is safe to assume that at least that the lower level features learned by that network are transferable to your task, okay. So we just try to exploit that features learned by this already trained pre-trained network for your task.

And since the deeper layers are easier to train compared to the innermost layers, the earlier layers in the network. Fine tuning your pre-trained network with the data that you have will hopefully modify your outermost layers and adapt it to the task at hand.

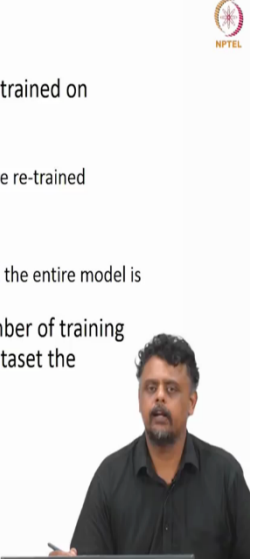
(Refer Slide Time: 7:43)

Modes of Transfer Learning

- Pre-trained models (VGG-16, ResNets, DenseNets) are trained on extremely large corpus of data for days or weeks.
- During transfer learning either:
 - Initial layers are frozen and last few layers in the network are re-trained

Or

- The entire model is initialized with pre-trained weights and the entire model is set to trainable mode. ✓ Chest x-ray scans
- The mode of transfer learning is dependent on the number of training examples in the new dataset and similarity of it with dataset the network was pre-trained on.

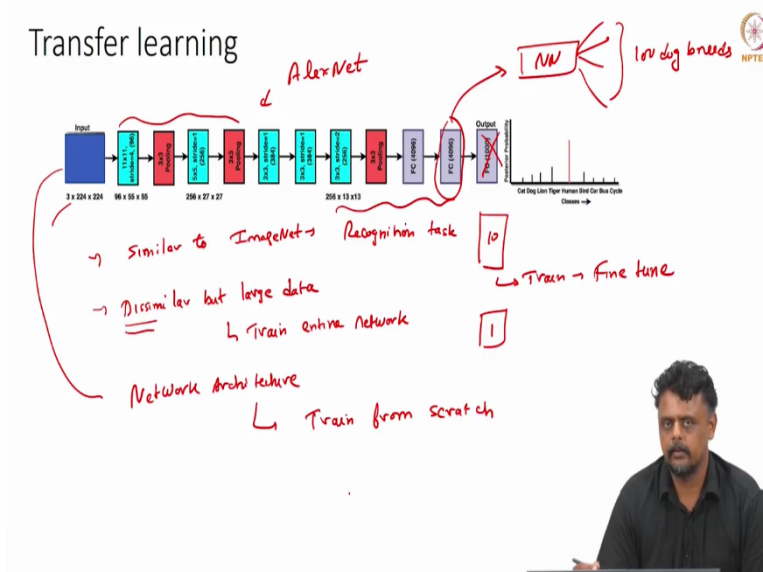


So in general when you try to do this transfer learning, one strategy is to freeze the initial layers because as I mentioned earlier would expect that the low-level features are also transferable to your data. In some cases, it's possible that even the, you are using the pre-trained network because you feel that, you can say that the weights have been initialized to a pretty good value after training with that large database.

But then your task or your input data is totally not related to the network you are using. So for example for the second case, let's say you are trying to accomplish a radiology task, we will look at a case study later on. So you are trying to classify chest x-ray scans, right? Right, so we're looking at chest x-ray scans are being abnormal or normal, okay.

Now this data, chest x-ray scan data is not similar to ImageNet data at all, it's quite (())(8:57) different. But then you it's still like to exploit you know the deep network that has been initialized with pre-trained weights. But then in this case you would have to necessarily train from scratch. Okay, so but in that case we assumption is that you have enough data to do so and this is another mode of transfer learning even though you were used the pre-trained network as a good initialization of weights and you would still go ahead and train all the weights in all the layers you see the data available to you.

(Refer Slide Time: 9:31)



So just to illustrate some of the points that we have talked about earlier. So ideally if you have data which is let's say similar to, in this case since I am looking at a this is Alex net, okay, we looked at this before. So your data is similar to ImageNet and then you have a classification recognition task, right? Then what you would do is, you would just drop this out, okay.

And then replace it by say in this case let's say you want looking at 10 breeds of dogs instead of thousands will have 10 output, and then trained or fine tune with whatever data that you have for the task, okay. But then let's say your input is radiology data, chest x-ray, so dissimilar but large data. So you take the, of course you would still go ahead and modify the output layer.

Let's say you need only normal versus abnormal, so instead of 10 you will have just one output, right? You can say 0 or 1 and you would train the entire network. This is assuming that you have (11:20) different data but then you have enough of it. So because there are if you know Alex net has several million parameters, 60 million and if they want to train them you need to have as many datasets of course you also do data augmentation in order to increase the size of a dataset, artificially.

So, given that the data is dissimilar on the task of chest x-ray is a good example or medical data in general, medical imaging data in general or some form of spectral data which is not similar to ImageNet, then you would have to train the network from scratch but you can use

the pre-trained network on the pre-trained weights as a good initialization. Okay, so that's typically this recommended.

So in any case if you have enough data let's say you have hundreds of tens of millions of data points let's say images for instance for some particular task. One thing you can still do is to keep the network architecture retained network architecture. This case I have shown network but you feel that use inception or some other model that's fine.

But you train from scratch. So that way you have you do have a structure a (12:53) architecture that has been proven to work with large datasets image datasets. So you can use that same architecture even try to see if the some of the training hyper parameters can be applicable or not and use your data to train it from scratch.

Only if in the case where you have dissimilar data but you don't have enough of it, okay. so the sense that it's not millions but maybe of several thousands and then you can do data augmentation may be of tens of thousands then you still, the idea would be to freeze some of this layers, okay and also (13:42) that's to train the final layers let's say we trained this few layers using your data and hope for the best, okay. Assumption behind Tri-freezing these layers is that at least you have believed that the local features the low-level features are transferable to your task that you are interested in.

So all these strategies put together typically is referred to as transfer learning wherein you put in simple terms you take a pre-trained network and see if you can just modify it slightly to accomplish your task of interest. Okay, so the other scenario that I think I referred (14:28) to early on is when your data is actually very similar to let say in this case you are looking at Alex net trained on ImageNet data.

Maybe you have a very similar data since let's say you're looking at cat breeds of dog breeds you want to identify different type of dog breeds based on just the picture, you can just then as we discussed earlier just take this embedding out and put it through let's say another neural network and you have either 10 outputs hundred dog breeds let's say or 10 dog breeds, okay. Such a thing is possible, okay. So if it's very similar then you don't have to maybe even train from scratch or even do fine tuning can take the embedding from the network, okay.

This is done in many cases so for instance if you are trying to do task like video segmentation, video frame segmentation or trying to do object detection then many of the imports look like the objects in your inputs look like objects in let's say ImageNet, then you

can use image net directly in such a, in the sense the networks trained on ImageNet directly in such applications, okay. When you move across applications and you be careful about how you're going to accomplish transfer learning.