This Content is licensed under a Creative Commons
Attribution-NonCommercial-ShareAlike 4.0 International License.

For accessing this content for free (no charge),
visit : nptel.ac.in

This Content is licensed under a Creative Commons
Attribution-NonCommercial-ShareAlike 4.0 International License.

For accessing this content for free (no charge),
visit : nptel.ac.in

Deep Learning for Computer Vision

Finetuning in CNNs

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Limitations of Working with CNNs

Practical concerns of working with CNNs:

- Optimization of parameters in deep models (characteristic of CNNs) very hard, requires careful parameter initializations and hyperparameter tuning

So all this can become even more difficult when you have to design an architecture for a new domain, in which case, you have to design an architecture, run it for about a week or several days, see how it works, go back and change the hyper parameter, again, run it for one more week, see how it works.And this kind of an approach to designing new architectures of CNN architectures will not scale.

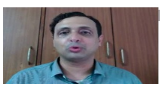
Limitations of Working with CNNs

Practical concerns of working with CNNs:

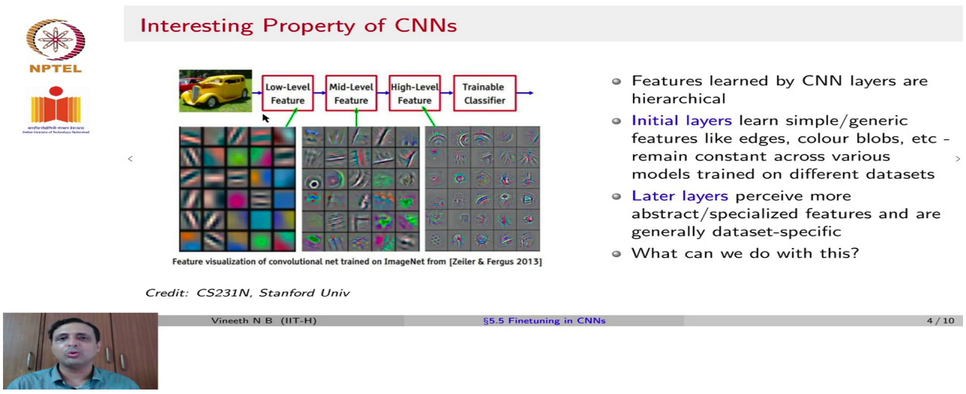
- Optimization of parameters in deep models (characteristic of CNNs) very hard, requires careful parameter initializations and hyperparameter tuning
- Can suffer from overfitting, as data samples used for training are lesser compared to parameters being trained
- Require a long time and computational power to train

Model	Parameters
AlexNet	60m
VGG	138m
Inception v1	5m
Inception v3	23m
Resnet 50	25m

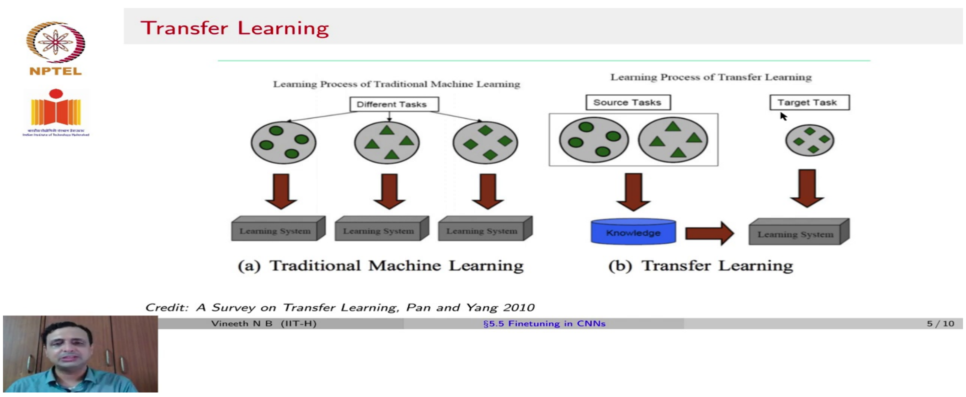
Model	Training time	Hardware
AlexNet	5-6 days	two GTX 580 3GB
VGG	2-3 weeks	four NVIDIA Titan Black
Inception v1	1 week	not mentioned



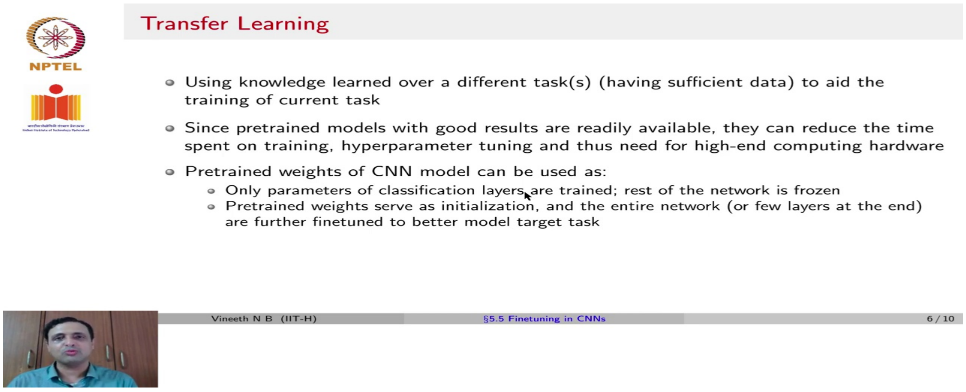
So how do we manage doing this for newer tasks and newer domains? Some things you can try is perhaps try a good weight initialisations.As we said earlier, this approach of unsupervised greedy layer wise pre-training is not used these days due to the increased computational power and also data set sizes, it takes a long time to do the greedy layer wise pre-training as only an initialization method.



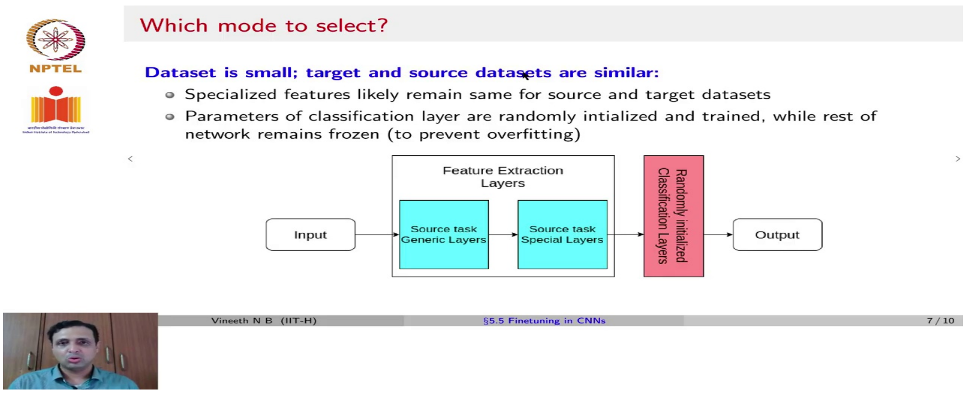
The question we asked now is, what can we do with this? Can we use this in some way? Can we use this understanding of how CNN filters are learning to be able to design architectures better for newer tasks and newer domains? The answer there is a setting in machine learning known as transfer learning.In transfer learning, before we go to transfer learning in traditional machine learning, if one had different tasks, you would have a data set for each of these tasks.And using a data set for each task, you would have a different model through a learning system that you learn.In transfer learning, you have a set of source tasks that are given to you.And a target task on which you want to perform well, that is the new domain, a new task that you want to solve the problem on.So when we say source and target, for example, you could imagine now that you have trained Alex net, on the image net data set and tomorrow, you have data coming from a different domain.



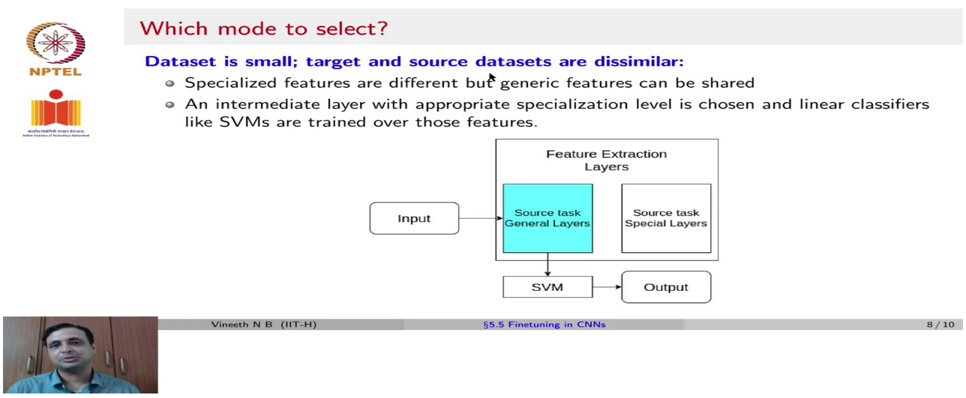
So how do you now transfer the model that you learnt on image net to a model that you want to develop on in a new domain? In this case, you try to use the data in the target domain, as well as the model or the knowledge that you got from the source domain to be able to learn a model for the target domain.So this we call as transfer learning to be able to transfer knowledge from an earlier source task to a target task that we are currently interested in.How does this relate to the hierarchical learning of abstractions of features in CNNs, we will see that in a moment.So we know now that using knowledge learned over a different task to aid the training of current task is transfer learning.So what we can do now is, you could take a pre trained neural network such as an Alex net to a newer domain, and keep all of those weights of Alex net exactly the same.Do not update them for this new task, instead take only the last layer, which we call as the classification layer, and train only those parameters for the newer task.



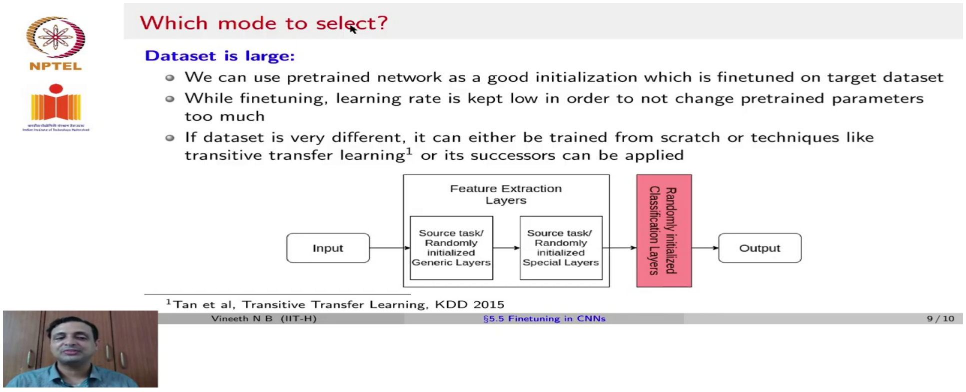
We call this to be fine tuning, where you are initializing the weights of a network for a new task based on some pre trained weights of another model, and you only fine tune certain layers in the new network for the new task.You could also use the pre trained weights as an initialization, and then fine tune the entire network or certain layers at the end of the CNN, not just the classification layer, you could take the last 3 layers or last 4 layers, anything of your choice to fine tune to better model the target task.Which one do you choose depends on your data set size, and how similar is your target task to the source task.If your data set was fundamentally small, and your target and source data sets are fairly similar to each other.



What you can do now is because this target and source datasets are similar to each other, you could say that the specialized features are likely to remain the same in both these datasets, for models trained on both these datasets.So what you can do now is you take the source tasks, generic layers, you take the source tasks special layer, so this entire block that you see in the middle could be an entire pre trained network, such as an Alex net, or Resnet, that you trained on another data set, which we call as a source task.We leave that as it is and we only train you randomly initialize the classification layer alone and train only that one for the outputs in your new task.This is because your data set is small, you feel you may not have enough data to retrain or fine tune all these feature extraction layers, that as well focus only on training the classification layer properly.So in this case, we are initializing the new network with a pre trained weights and we are fine tuning only the classification layer.Another setting is where the data set is small and also the target and the source data sets are dissimilar this time, which means the specialized features the general features could still be common.



What can we do here, we now consider the entire pre trained CNN because we know that the feature maps of the generic layers which are by General layers, we mean the early layers of the CNN, you can consider that from your pre train network, take those feature maps as output, and then train another classifier such as support vector machine, or could also be a neural network to give you the output for the target task.



You are saying that I still want my overall network weights to be similar to the weights that I learned from my source task.But I have the data, I am going to update it but let me keep the learning rate low so that I do not make too many updates to my weights from what they were from a source task.On the other hand, if data set is very different, and the data set is large, there are explicit methods for this, such as what is known as transitive transfer learning or things like that, where you can either train from scratch, you do not need to fine tune at all, you can randomly initialize using Clorox initialization and train from scratch on the new target task because you have a large data set, you are saying that the target task and the source tasks are not similar to each other so you may as well train from scratch, or use methods such as transitive transfer learning, which is a known method in traditional machine learning and that can help you improve performance.There are many hyper parameters, number of layers with resolution, number of filters also learning hyper parameters to try to come up with the appropriate combination for a domain and a task by itself can be very difficult task.And using the idea of transfer learning and fine tuning could help you take an existing architecture that has solved a different problem and adapt it to a new problem.