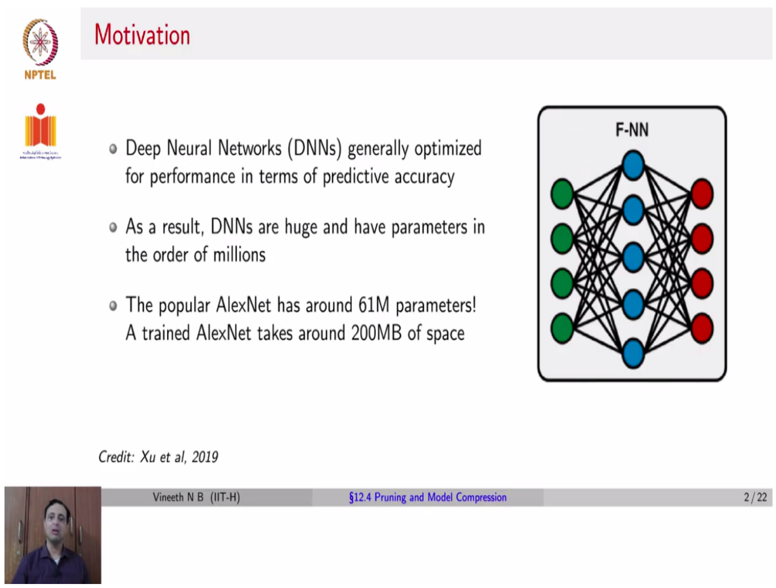
Deep Learning for Computer Vision
Professor. Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture No. 77
Pruning and Model Compression

Moving on from adversarial robustness, we will now talk about Pruning and Model Compression. Another important component in taking Deep Learning models to in the wild real world applications.

(Refer Slide Time: 00:32)



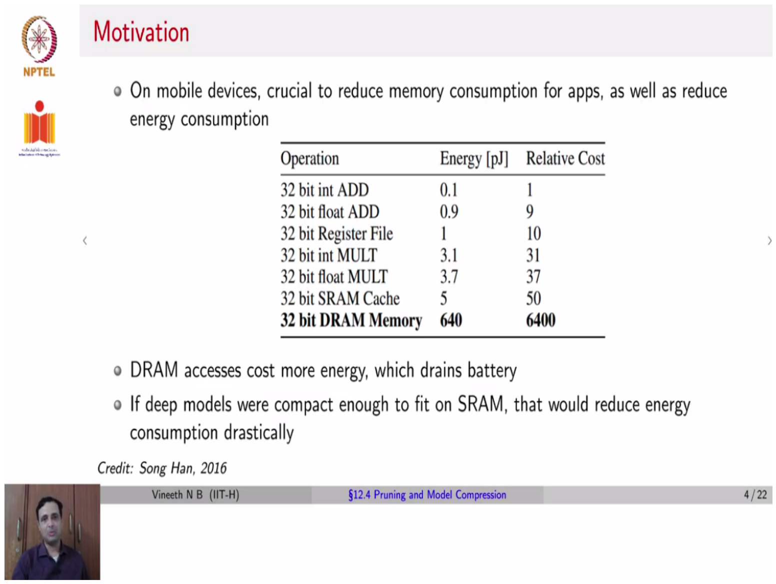
Neural Networks in general are optimized to improve predictive accuracy. Be it accuracy for classification models, mean average precision for detection models or pixel wise classification accuracy for segmentation. Trying to chase accuracy alone makes neural networks very large. As a result, the models that are state of the art today have very large number of parameters, often of the order of millions.

Recall that we said that AlexNet has over 61 million parameters, occupying about 200 MB of space in the memory. VGG occupies up to 500 MB of space, just to store the weights in the model in your memory.

(Refer Slide Time: 01:26)

Is this really a problem? When you train your models, it is alright to have very high storage footprint, memory footprint, and one can use powerful GPUs to train these models. However, expecting the availability of heavy compute at test time or inference may be limiting. If one considers the deployment of Deep Learning models in low compute applications, such as mobile phones, drones, or unmanned aerial vehicles, or IoT devices, which could be deployed in any edge at the corner of the world, even in harsh conditions, having bulky Neural Network models becomes a limiting factor in taking their success to these kinds of compute platforms.

(Refer Slide Time: 02:22)

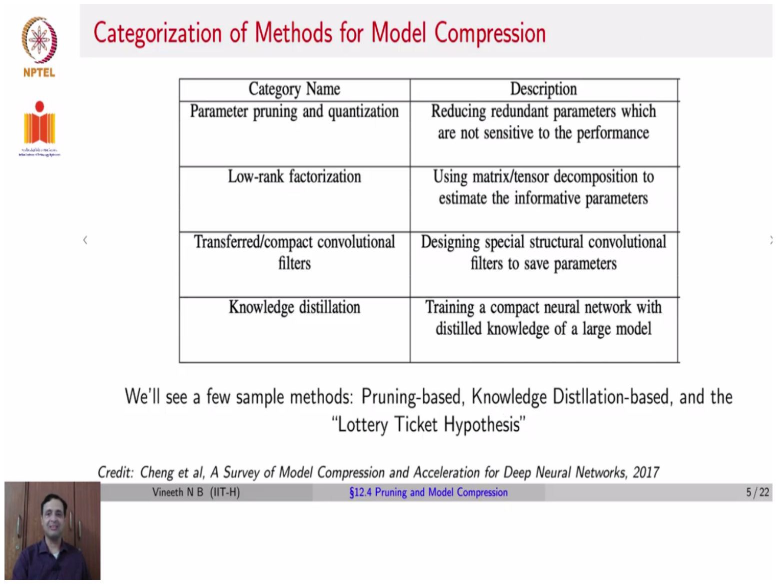


Another way of viewing this, is from the viewpoint of the energy expended for carrying out such operations in memory. An interesting analysis was done by Song Han, who came up with one of the most popular papers for deep model compression. And the analysis here shows on this table that a 32 bit integer addition consumes about 0.1 Pico joules. Pico joules 10^{-12} of energy. A 32 bit float addition operation consumes 0.9 Pico joules. And if you keep going further and further, a 32 bit SRAM cache access operation consumes 5 Pico joules. And when you go to the DRAM, you significantly go up in orders of magnitude. And now things go up to 640 Pico joules.

Accessing DRAM or dynamic RAM is significantly more costly than accessing your SRAM. Why are we talking about this, which means when we talk about low compute devices, we ideally would like these Deep Learning models to be housed in the SRAM and not have to go to the DRAM, because accessing them could cause a lot of energy requirements, especially in environments like drones, or edge devices, IoT devices, where battery also becomes a concern to deploy these models.

So, one key requirement that emerges now is the need to be able to prune these bulky neural network models into smaller memory footprints that can be deployed in low compute environments. This category of methods are broadly called model compression methods, where a trained model is compressed into a smaller memory footprint for deployment in low compute devices.

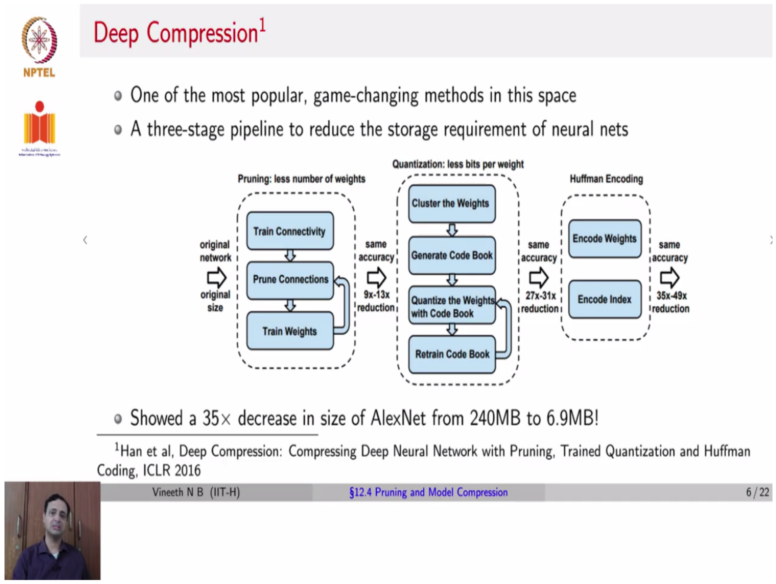
(Refer Slide Time: 04:34)



Over the last few years, several efforts have been taken have been taken by different researchers. And a broad categorization of these methods can be given as parameter pruning and quantization. That is one family of methods, which focuses on reducing redundant parameters that do not affect performance. A second family of methods is based on Low-rank factorization, where matrix and tensor decomposition methods are used to estimate only the informative parameters and discard the rest.

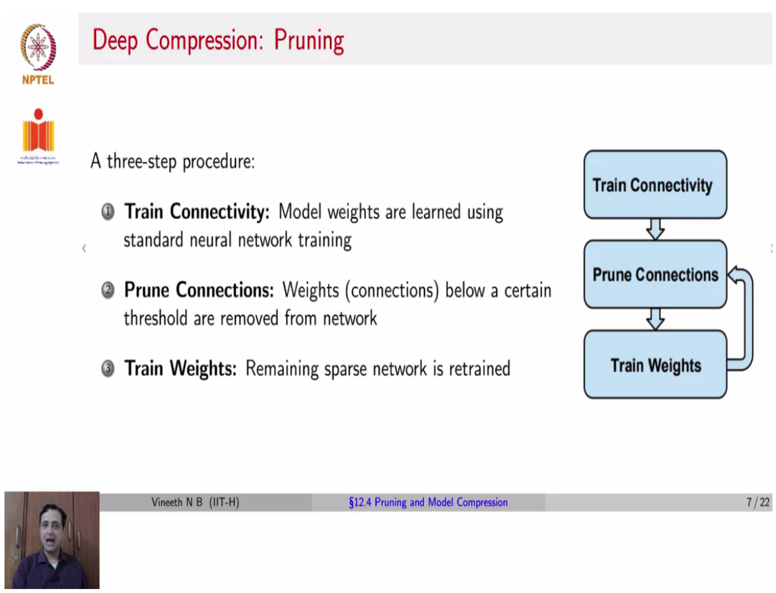
Transferred or compact convolution filters are a family of methods where special structural convolution filters are designed to save parameters. And finally, an interesting family of methods called Knowledge distillation methods that use an idea of distilling knowledge from a large neural network model into a small student neural network model. We would not see all of them in this lecture, but see a few ones briefly and point to other resources for more reading. Specifically, we will see a very popular pruning based approach, a knowledge distillation approach and a more recent approach called lottery ticket hypothesis.

(Refer Slide Time: 06:05)



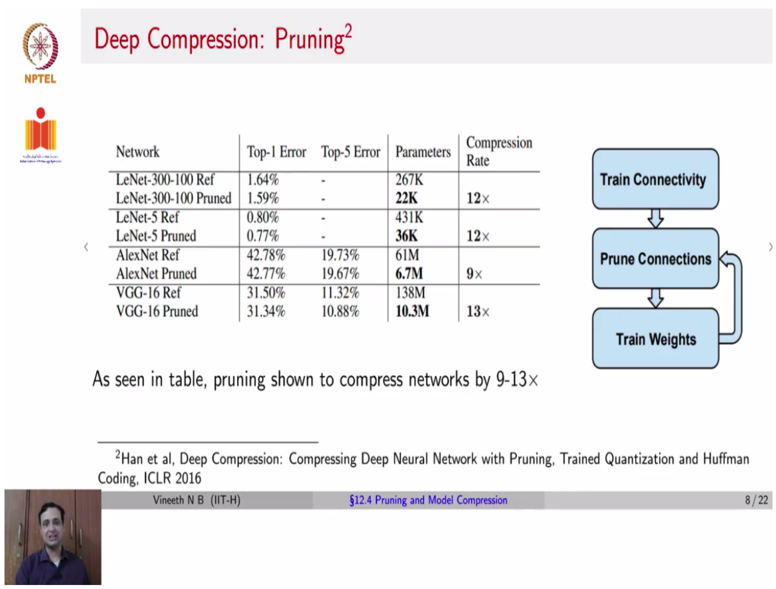
One of the most popular and reasonably early methods for model compression is called Deep Compression, developed by Song Han in ICLR of 2016. It was a game-changing method, which also took the method forward to hardware design. And this uses a 3-stage pipeline to reduce storage requirement of neural nets. The first step being pruning of a trained model, then, quantization of the weights and finally, an Huffman encoding step, which provides a model that reduces the size by 35 to 50 X with very minimal loss in accuracy.

(Refer Slide Time: 06:54)



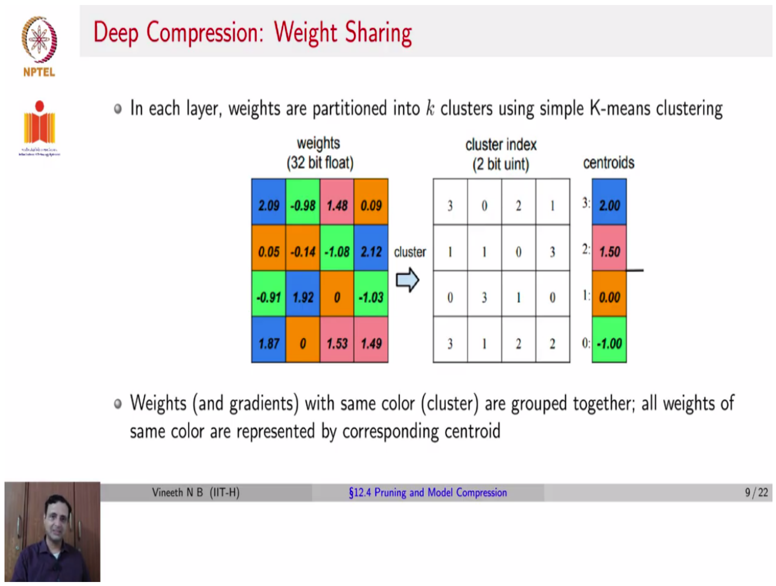
Let us see each of these steps. The first step was to prune the model. What does pruning mean here? Once you train the full model, which is the first step, weights with values below a certain threshold are removed from the network. So, if any weight is lower than say 10 power minus 5, that weight is removed from the network and the remaining sparse network with only the other connections is retrained to get a better network. Once again, this is an iterative process. Once again, in that new retrained sparse network, if any weights are below a certain threshold, they are removed. And once again the remaining sparse network is retrained and this step is done iteratively.

(Refer Slide Time: 07:45)



And just with this simple step alone, the authors showed that many of the popular networks could be reduced in size significantly. For example, you see here, AlexNet, while the original size is 61 million, using the simple pruning step, the size comes down to 6.7 million, which is a 9x compression. And when one looks at the top-1 error or top-5 error on ImageNet, you notice that there is no significant drop in performance, because of this reduction in parameters. In case of VGG, pruning alone, reduced number of parameters by 13x. This was also observed for smaller networks, such as LeNet.

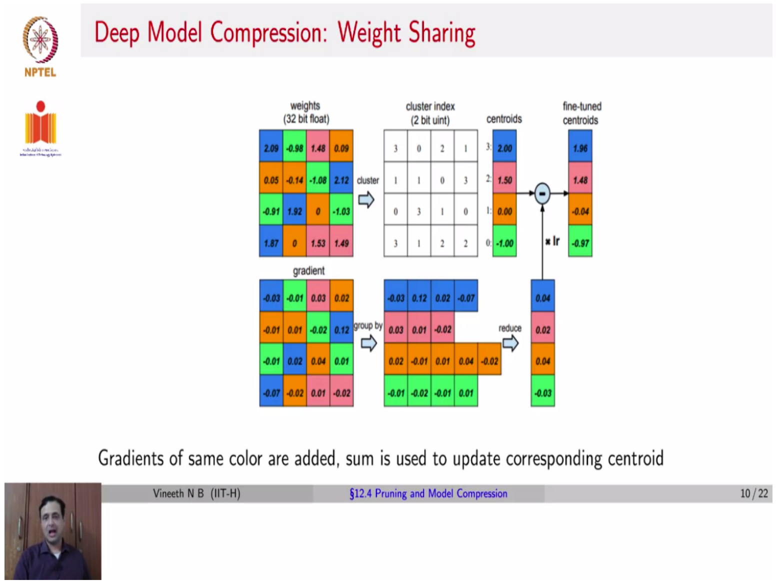
(Refer Slide Time: 08:39)



The second step after pruning is, a step known as weight sharing, where in each layer, the weights in that layer are partitioned into k clusters using simple K-means clustering and each weight is replaced by the centroid of the cluster that it belongs to. So, here you see an example of a 4 by 4 matrix of weights and the cluster assignment in the subsequent matrix here and the cluster centroid value, which is shown for each of these clusters.

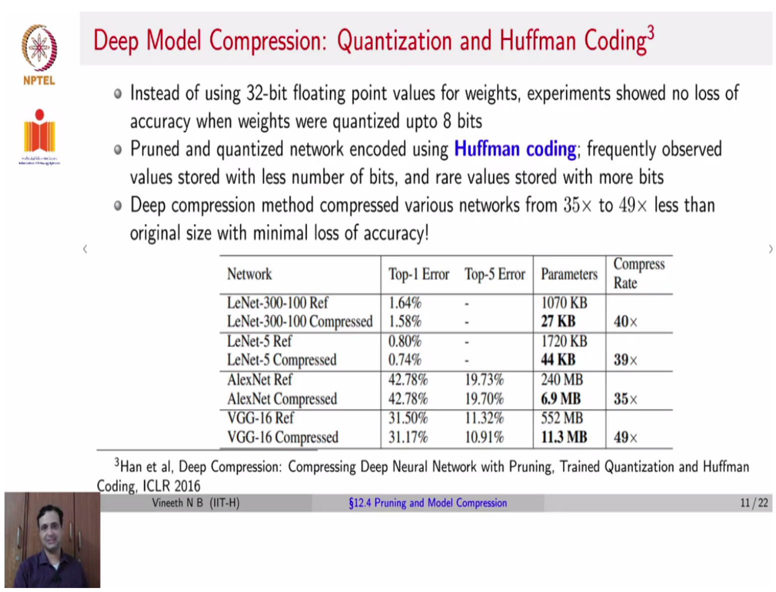
At the end, each of these blue weights are replaced by the cluster centroid of the blue color here, and so on and so forth for each of the colors. How does it help? We need to store fewer values to represent this layer's weights. A subsequent question is if the weights changed and are clustered like this, what happens to the gradients?

(Refer Slide Time: 09:45)



The gradients also follow a similar process. So, if you have a certain values of gradients for each of these locations in that particular layer. The gradients are also clustered and the cluster centroid value for the gradient is then used to subtract from the original weight to get the new weight. So, in the even the gradients participate in this weight sharing exercise in the same way.

(Refer Slide Time: 10:18)

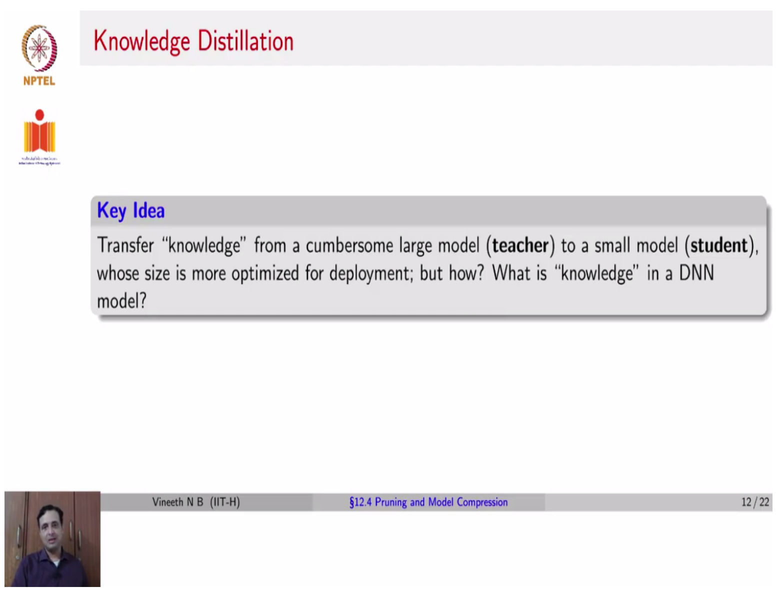


Having done pruning and weight sharing, the next step that the authors used was Quantization. This was based on an empirical observation that instead of using 32-bit float values, if we used just 8 bits, the performance really did not reduce much. So, this is the quantization step that was then used to still further reduce the storage footprint. And the final step was to use Huffman coding. Huffman coding is a popular coding compression method in computer science, where a frequently occurring pattern is stored with lesser number of bits and a rarely occurring pattern is stored with more number of bits to capture it is additional information.

Huffman coding is a long standing compression method, which is used here to once again reduce the storage footprint. With these methods sequentially, one after the other. The overall approach showed a 35 to 49 x reduction in parameters with minimal loss of accuracy. As you can see here, AlexNet went from being 240 MB to 6.9 MB in this particular case, and VGG went from 552 MB to 11.3 MB, which was a 49x reduction in storage parameters.

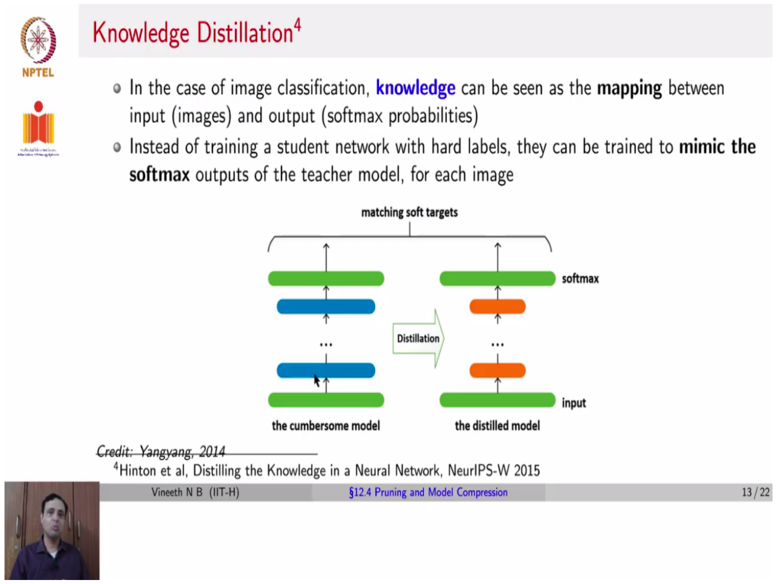
With if you see the error rates here, there is no significant loss in error because of this compression, which is the main objective. So, once you get to 6.9 MB, or 10 MB, these models now become amenable to deploy on low compute devices.

(Refer Slide Time: 12:17)



A second method that we will talk about is that of Knowledge Distillation. The key intuition of this family of methods is to transfer knowledge from a cumbersome, large model to a small model, which we call a student model, whose size is more optimized for deployment. The question obviously here is, what do we mean by knowledge in a Deep Neural Network.

(Refer Slide Time: 12:51)

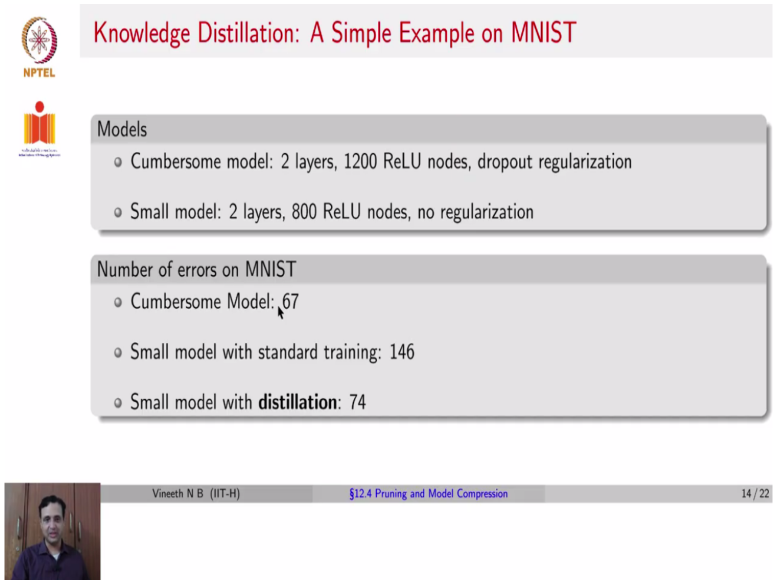


And the first idea that was used here was that knowledge can be viewed as the mapping between inputs and the softmax probabilities. Instead of while you are looking, if a cat, if there was an image of a cat, and you would like the class, for the cat to be 1 and everything else to be 0, a neural network may not necessarily give you that output, it may say, the probability for a cat is 0.8, and the probability for other class labels could be 0.01, 0.05, so on and so forth.

Now, these outputs represent the knowledge that the Neural Network has gained over the process of training. So, in knowledge distillation, the idea now is to take a small shallow student network and instead of training this network with hard labels, or one hot labels, we ask the student to target and predict the softmax probabilities or even the logits of the teacher network.

You can see here you have the cumbersome model, and through distillation, the distilled model's objective is to match the soft outputs or targets of the teacher model. In this sense, the knowledge gained by the teacher model is distilled into the student model, which performs as well with a smaller storage footprint.

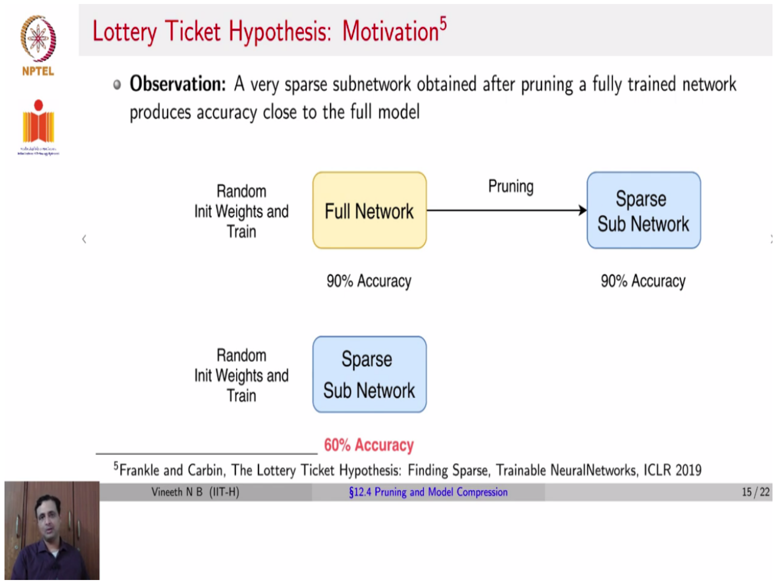
(Refer Slide Time: 14:25)



Here is a simple experimental example. So given a cumbersome model, this is on MNIST of 2 layers, with 1200 ReLU, nodes and dropout. And a small model of 2 layers and 800 ReLU nodes smaller model at least with no regularization. It is simple to observe that the number of errors on MNIST with the Cumbersome model is 67. If you train the small model using standard training, it makes 146 errors on the test set of MNIST and the small model with distillation makes only 74 errors, which is close to the bulky model.

Over the years, knowledge distillation has resulted in several variants, where instead of matching only the logits or only this outputs, probabilistic outputs of the teacher model, you can also match intermediate representations of hidden layers, you can add some noise, and try to ensemble, multiple teachers, so on and so forth, which are provided in the references for further reading.

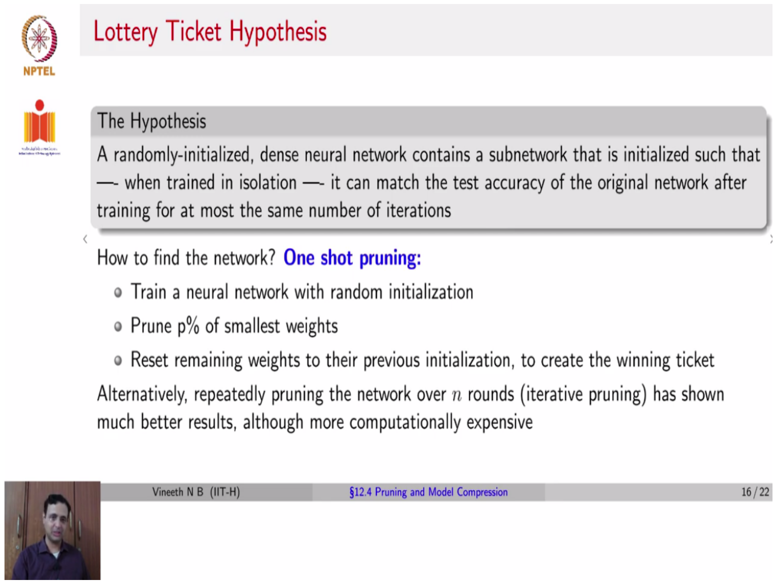
(Refer Slide Time: 15:35)



A third approach that we will talk about here is a recent one published in ICLR 2019, called Lottery Ticket Hypothesis. As the name says, this was based on an observation that when you train a full bulky Deep Neural Network, often a very sparse sub network, obtained after pruning produces accuracy, close to the full model. So, you randomly initialize weights and train a full network, you get 90 percent accuracy and you prune, you get a sparse sub network with 90 percent accuracy.

But you took the same kind of a Sub network, randomly initialized it and trained, you get only 60 percent accuracy. So, there seems to be something about training the full network, and then pruning.

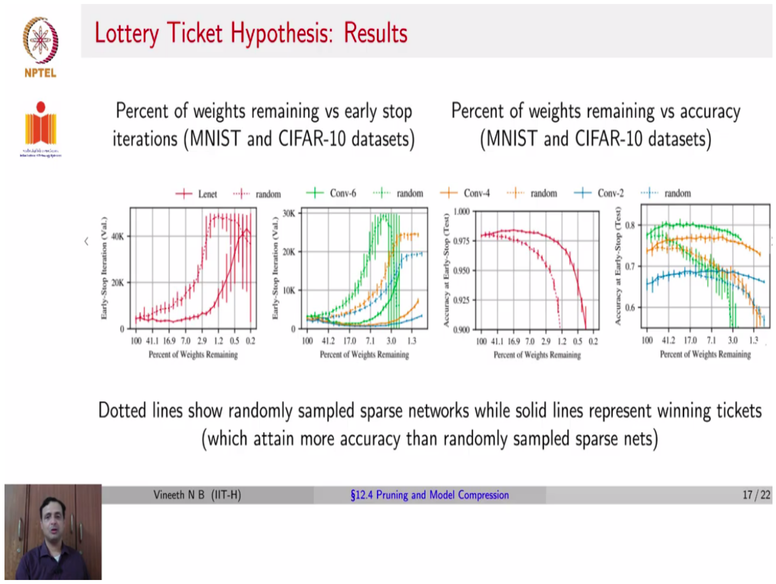
(Refer Slide Time: 16:32)



So, this work made a hypothesis that a randomly initialized Dense Neural Network contains a sub network that is initialized such that when it is trained in isolation, it can match the test accuracy of the original network, after training for at most the same number of iterations. The obvious question now for us is, how do you find the sub network? To do this, this approach, proposed a simple idea, which is called One Shot pruning where you first train a full network with random initialization. You prune a certain p percentage of the smallest weights of the full network. You reset the remaining weights to their previous initialization to create the winning ticket.

They showed that following this procedure helps us find the lottery ticket, which is that one random sparse sub network which seems to match the accuracy of the complete network. One could also repeatedly prune the network over multiple rounds, similar to the iterative pruning that we spoke about for deep compression. This does get better results, but of course, requires more computation.

(Refer Slide Time: 18:06)

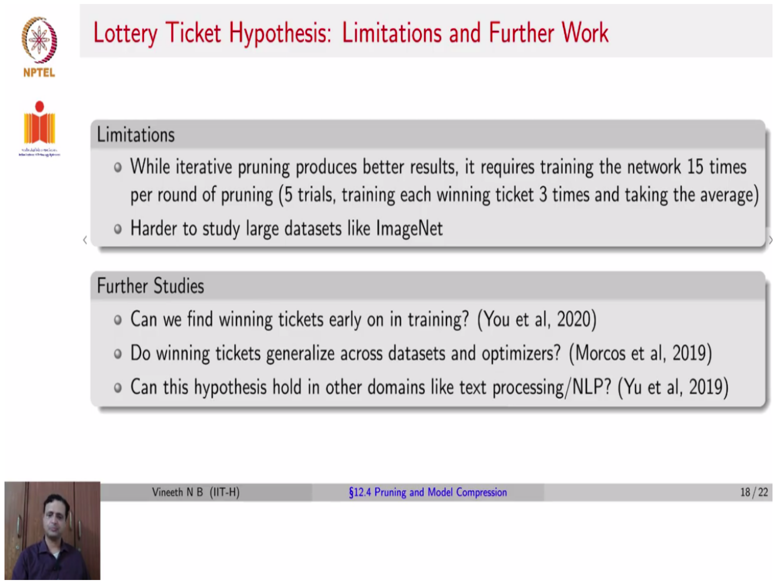


Here are some results that were shown in this work. On the left, what you see are percentage of weights remaining versus early stop iterations for MNIST and CIFAR-10. You see here, that for if you look at these two curves, one of them the dotted lines, is a randomly sampled sparse network and the bold line is the one obtained by Lottery Ticket Hypothesis. You see that even when the number of early stop iterations is very low, the percentage of weights remaining for the Lottery Ticket Hypothesis remains high using this kind of an approach.

A similar pattern is also observed for CIFAR-10. On the other hand, more importantly, a plot of the percentage of weights remaining versus accuracy is again favorable, favorable for the Lottery Ticket Hypothesis, you can see that the dotted line here and the bold line, here dotted line is a randomly sampled sparse sub network and the Bold Line is the one obtained using Lottery Ticket Hypothesis.

You see that as the percentage of weights remaining decreases, you can see here it goes from 100 to 0.2. The dotted network sub network random one has a quicker fall in accuracy, while the Lottery Ticket Hypothesis maintains a higher accuracy for a longer period of time. A similar result is also seen on CIFAR-10 with different kinds of layers, shown for the percentage of weights remaining.

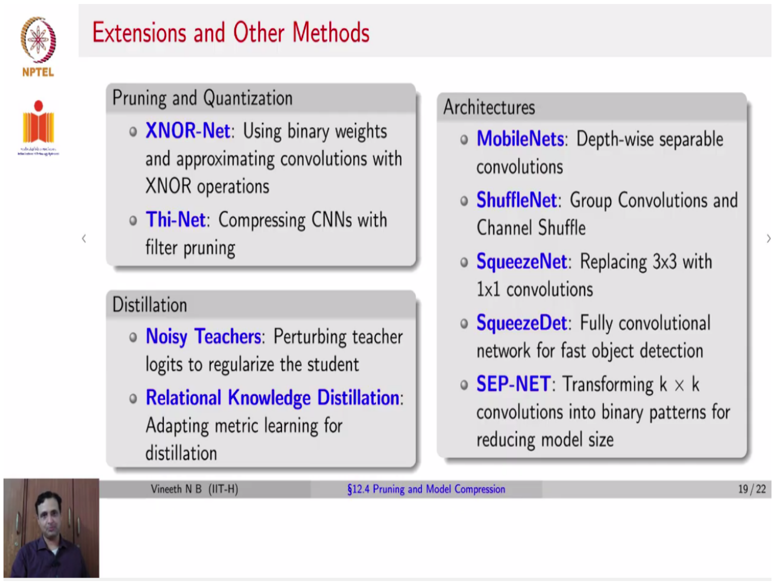
(Refer Slide Time: 19:57)



So, is that always a good strategy? Not really. While iterative pruning produces better results, it requires training the network perhaps about 15 times per round of pruning. On the other hand, finding Lottery Ticket Hypothesis, while can be done sometimes may not give you as good a performance as the original network. If you did iterative pruning it is going to be harder to study large data sets such as ImageNet.

So, over the last few years, there have been improvements over the Vanilla Lottery Ticket Hypothesis work, where researchers have studied if these winning lottery tickets can be found early on in training, rather than wait for too many iterations. Can such winning tickets generalize to newer datasets and optimizers and does this hypothesis hold in other domains such as text processing, or NLP.

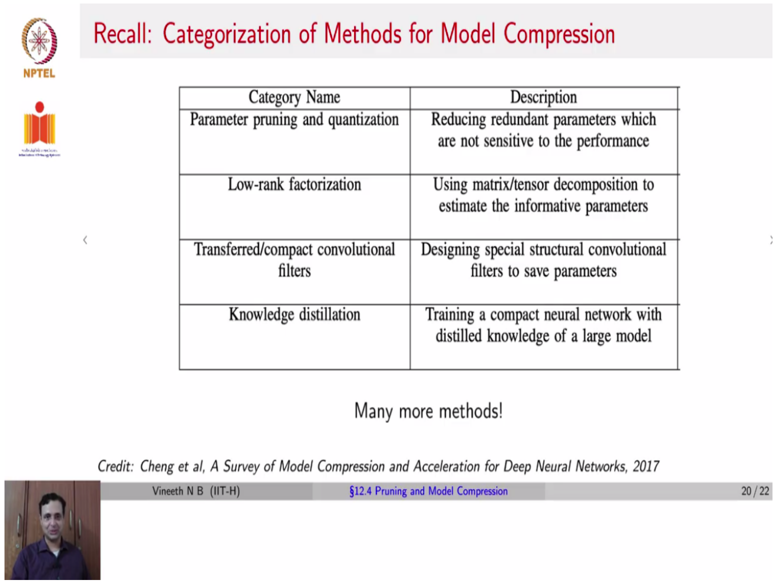
(Refer Slide Time: 21:01)



These methods have also been extended in several different ways. XNOR-Net is a popular compression method where binary weights are used. The understanding here is you do not need the precision of representing each weight using too many bits just using binary weights and replacing convolution with XNOR operations can make Neural Networks attain a reasonable amount of accuracy with a very small memory footprint. Thi-Net compressor compresses CNNs with filter pruning.

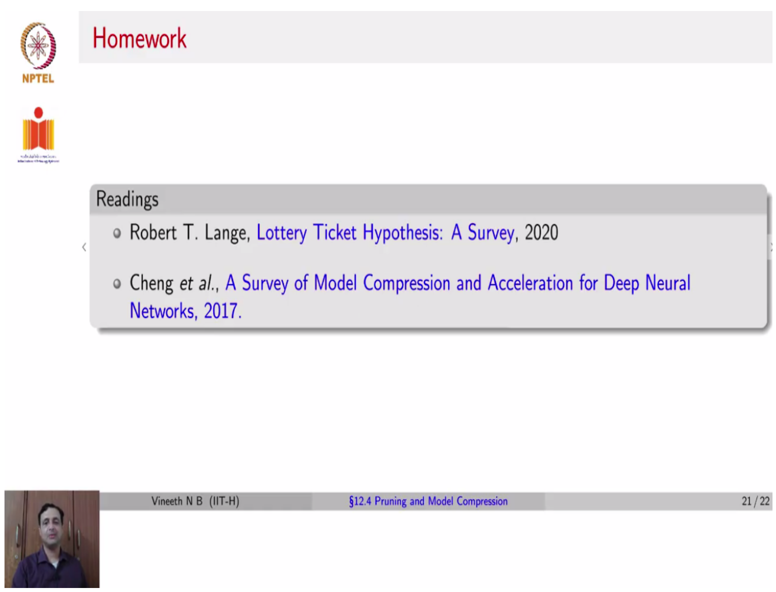
Knowledge Distillation methods have used noisy teachers where the teacher logits are perturbed to get the effect of multiple teachers to train a student. Relational Knowledge Distillation adapts metric learning in distillation, in distillation, and there have also been specific architectures. We discussed a few of them when we discussed CNNs, Mobile Nets, Shuffle Net, SqueezeNet, SqueezeDet for detection and SEP-NET so on and so forth, which have been used for model compression.

(Refer Slide Time: 22:18)



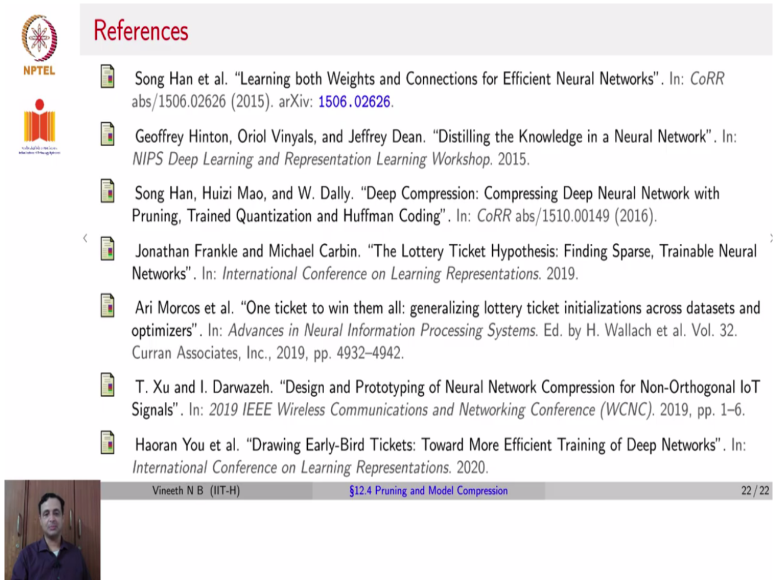
As we said earlier, the space is fairly large. There are also Low rank factorization methods. There are also methods that design convolutional filters in a particular way to save parameters, so on and so forth, which we leave it for reading in this lecture.

(Refer Slide Time: 22:34)



So, the homework for you is to read a very nice survey of the Lottery Ticket Hypothesis and this Comprehensive survey of different Model Compression and Acceleration Methods for Deep Neural Networks.

(Refer Slide Time: 22:50)



Here are some references.