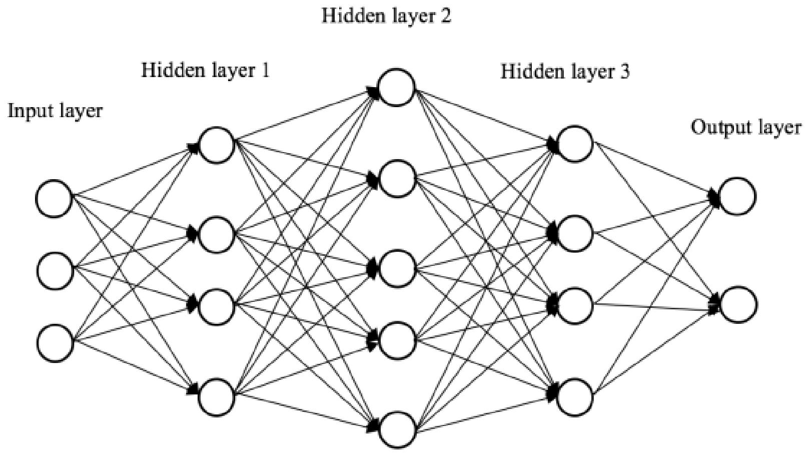
CS 541: Artificial Intelligence

Instructor: Jie Shen — Lecture 9

Scribe: David Staronka and Omkar Sinha — November 9, 2021

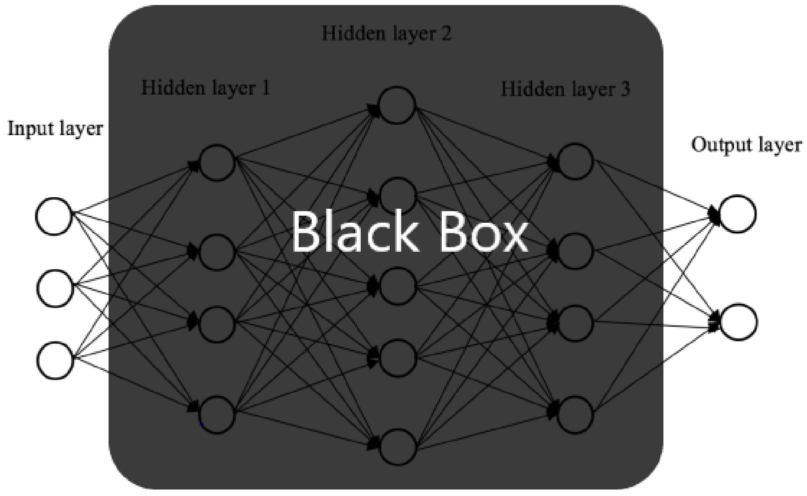
1 Deep Learning Model

A typical deep learning model is as follows:



Here the Nodes are one of more weighted input connections and produce an output connection. They're organised into layers to comprise a network. Many such layers, together form a Neural Network, i.e. the foundation of Deep Learning. The input layer receives the input and the output layer produces results.

However the intermediate layers workings are not recognizable to us, and are called the hidden layers and together constitute the black box as follows:



2 Optimization of Deep Learning Model

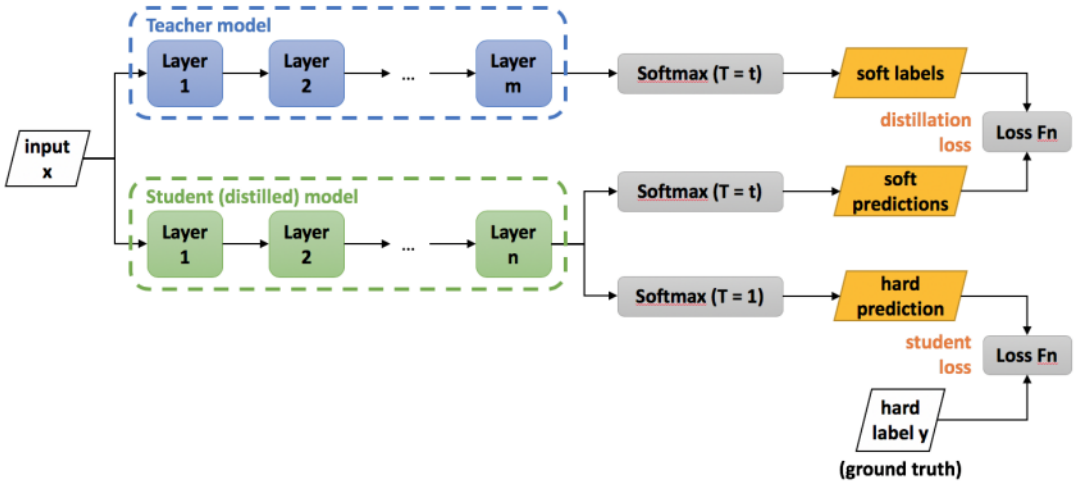
Since the model is like a black box it is challenging for us to effectively optimize. One way to optimize the model could be to simply increase the number of parameters, however such an approach will lead to increase in model complexity which will make it costlier to train. When we take the opposite approach, we face overfitting.

So the question is how to find the optimum size of the model?
We take one approach, where we recognize the overfitting to be caused by the model learning some extra features. We can use model compression to filter those nodes learning extra information. We can do this manually using techniques like node pruning or by using a function low rank factorization which uses matrix/tensor decomposition to project the initial d-dimension layer to lower rank compressed k-dimension layer.

While this is helpful it has the possibility of us filtering out the important parameters. So now the question arises as how to develop a deep learning model which can itself determine which information is important or not.

3 Knowledge Distillation

A technique to develop a model that meets these requirement was proposed in 2015 by Geoffrey Hinton. This is called knowledge distillation which can also be called teacher-student model for simplicity. This has one complex 'teacher' model which well-trained in advance and the other smaller distilled student model.



In the above knowledge distillation representation, the student model is trained by the comparing its output with that of the teacher model. Here we call the loss observed between the student and teacher model as distillation loss. We also compare the 1st layer of the student model with ground truth and record the loss as student loss. The loss function $\pounds$ is given as:

$$\pounds = \lambda \pounds(f_t, f_s) + (1 - \lambda)\pounds(f_s, y)$$

where f_s = outcome of student model, f_t = outcome of teacher model and y = ground truth

The loss function is divided into two parts - the first student model learned from teacher and the second part shows student model learns from ground truth.

3.1 Response-based Knowledge:

According to the model proposed by Hinton, the knowledge is defined as the output of teacher model known as logits, which are also called response-based knowledge. Acc. to this the softmax is defined as follows:

$$p(x_i, T) = \frac{exp(x_i/T)}{\sum_j exp(x_j/T)} \rightarrow \pounds_{CE}(p(f_t(x), T), p(f_t(x), T))$$

where, T is temperature or margin controller that depicts the relative values of the parameters:
$0 < t < 1$ one factor is enhanced
$t > 1$ factors are similar
when CE(cross-entrophy) = 0, student learns well from teacher model.

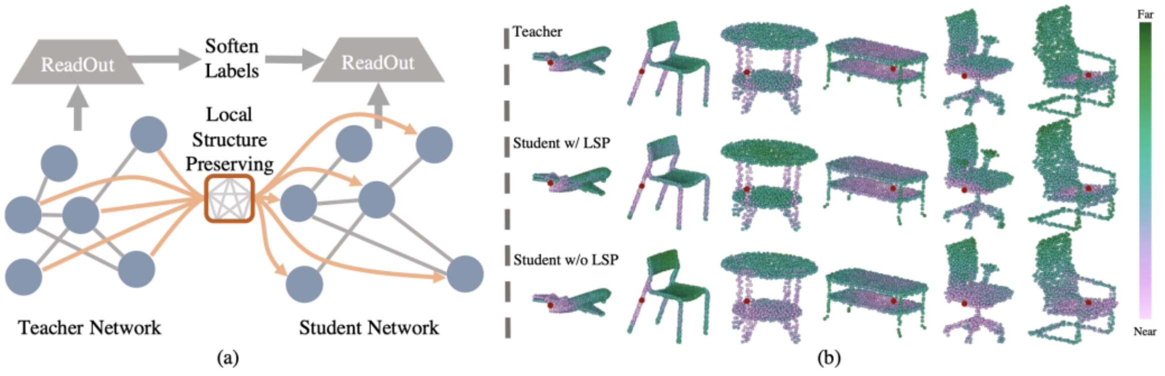
An example of response based learning below shows the difference of image recognition between using hard label and soft label.

	Hard Label	Soft Label
Cat	0	0.4
Dog	0	0
Car	1	0.6

While the cat and car are easily recognized, the third image of a cat inside a car, has quite different results between hard label and soft label.

3.2 Feature-based Knowledge:

We can also use the data from the intermediate hidden layers in the teacher model to train the student model. An example of this techniques when used in computer vision with local structure preserving(LSP) is stated below:



Acc. to Yang et al, the local structure is denoted by formula:

$$LS_{ij} = \frac{e^{||z_i - z_j||_2^2}}{\sum_{j:(j,i)\in\epsilon}(||z_i - z_j||_2^2)}$$

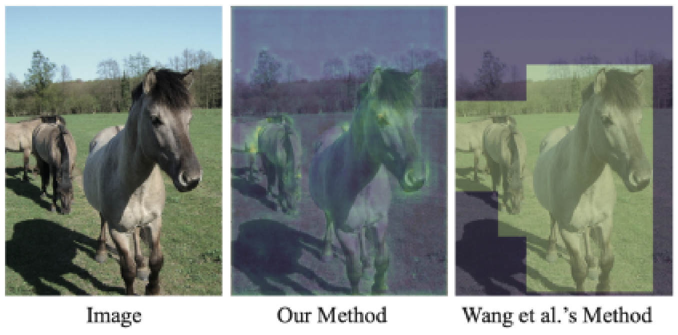
Acc. to Zhang et al(2021), we have Attention and Global Relation (C: Channel; H: Height; W: Width) as follows:

$$Attention : G^c(A) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}|A_{\cdot,i,j}|$$

$$G^S(A) = \frac{1}{C}\sum_{i=1}^{C}|A_{i,\cdot,\cdot}|$$

$$Relation : r_{i,j} = \frac{1}{WH}\sum_{i'=1}^{H}\sum_{j'=1}^{W} f(A_{\cdot,i,j}, A_{\cdot,i',j'})g(A_{\cdot,i',j'})$$

This is shown to improve object detection with Feature-based Knowledge Distillation giving accurate and efficient detectors.



3.3 Relation-based Knowledge:

In this approach, we do not evaluate the layers, but instead compare pairs of layers to find their instance relation and distill that information from teacher to student model. Here we have:

$$\pounds_{R'}(F_t, F_s) = \pounds(\phi_t(t_i, t_j), \phi_s(s_i, s_j))$$

where $\phi(f_i, f_j)$: Relation between f_i and f_j
$\pounds : l1norm, l2norm, l_{CE}$

4 Distillation Schemes

There are 3 types of knowledge distillation:

1) Offline distillation: here we assume the teacher model is well trained and use it train student. Whatever we discussed till now was offline distillation.

2) Online distillation: here we simultaneously train the teacher model and enhance the student network.

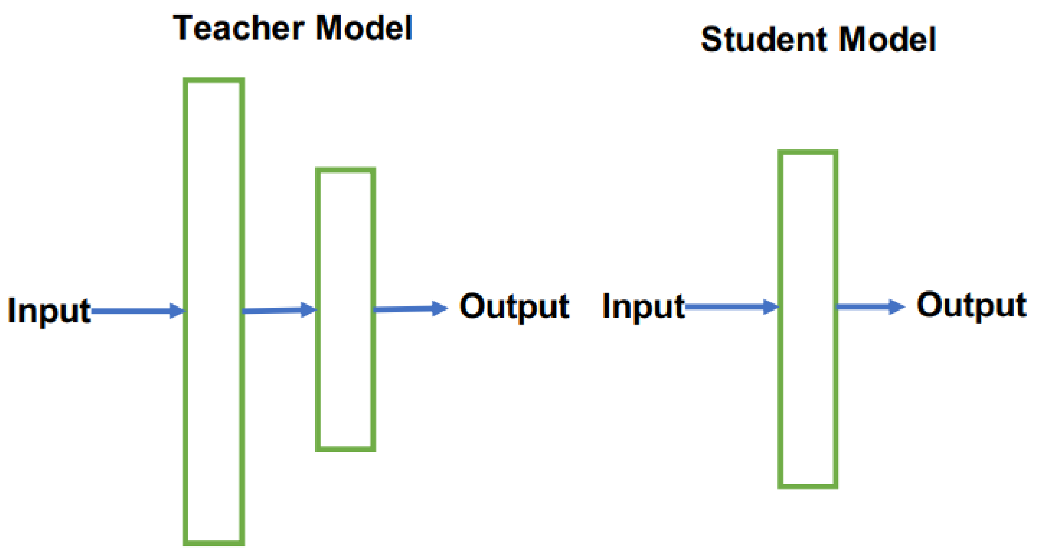
3) Self distillation: here the student model teaches itself through repeated iterations of learning from itself.

5 Teacher-Student Architecture

There are 3 ways to choose student architecture:

1) Student model is chosen from the simplified version of the teacher model. Here our aim is to preserve the features of the specific model while simplifying it.

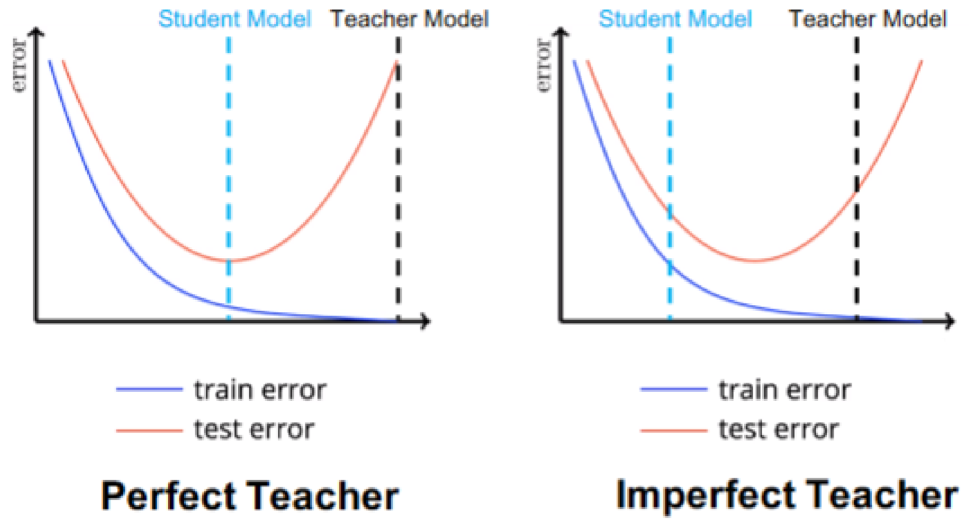
2) Smaller size of the teacher model is used as the architecture for the student model. This is represented below:



3) Using the teacher model as the student model unchanged. In cases where we do not require a simplified version, this is chosen.

6 Why Knowledge Distillation Works

In the Under Parameters Regime:
One hot risk: $\hat{R}(f)$
Bayes Distilled Risk: $\hat{R}_*(f)$
Overfitting is caused by low-bias and high-variance.
$\mathbb{V}[\hat{R}_*(f)] \leq \mathbb{V}[\hat{R}(f)]$



7 Generalization Bound Perspective

Observation: The Generalization bound is not good for deep models.
This is because of the large Vapnik–Chervonenkis (VC) dimension.

Distillation error: Given a multi-class predictor f: $\mathbb{R}^d \to \mathbb{R}^k$, the goal is to find another predictor g: $\mathbb{R}^d \to \mathbb{R}^k$ which is simpler, but still similar in distillation error $\phi_{\gamma,n}$, defined along with ϕ_γ, the softmax with temperature $\gamma > 0$, as:

$$\phi_\gamma(f(x_i), y_i) = \frac{\exp(f(x_i)_{y_i}/\gamma)}{\sum_{j=1}^{k} \exp(f(x_i)_j/\gamma)}.$$

$$\Phi_{\gamma,n} = \frac{1}{n}\sum_{i=1}^{n} |\phi_\gamma(f(x_i), y_i) - \phi_\gamma(g(x_i), y_i)|\,.$$

The distillation process can be thought as matching the G_j one by one until the distillation error is small. Then we announce that f inherits the generalization properties of G_j.

Lemma: With any set of multiclass predictors G_j where $g \in G$, with probability at least $1 - \delta$,

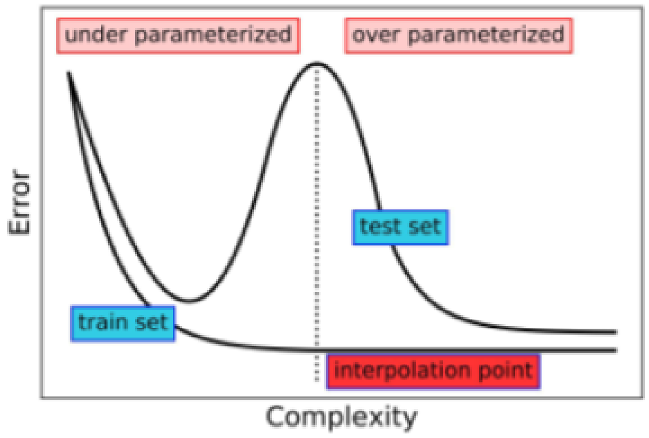
$$\Pr[f(x) \neq y] \leq \frac{2}{n}\sum_{i}(1 - \phi_\gamma(g(x_i), y_i)) + \frac{16}{n} + 6\sqrt{\frac{\ln(2/\delta) + \ln \lg(n)}{n}} +$$
$$\inf_{j\geq 1, g\in\mathcal{G}_j}\left[8\Phi_{\gamma,n}(f,g) + \frac{1}{\gamma}\text{Rad}\left(\{(x,y) \to g(x)_y - \max_{i\neq y} g(x)_i : g \in \mathcal{G}_j\}\right)\right] +$$
$$6\sqrt{\frac{\ln(1/p_j)}{n}} \quad (1)$$

Therefore,

$$\text{T}_{\text{test error}} \leq \text{S}_{\text{training error}} + \text{distillation error} + \text{distillation complexity}$$

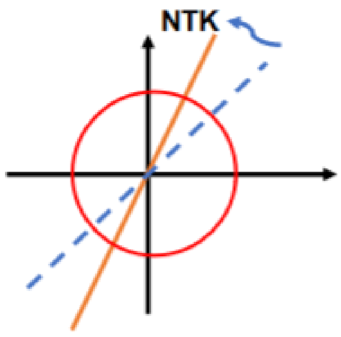
8 Neural Tangent Kernel in Wide Neural Network

This method uses the Kernel to explain the Artificial Neural Network where the neural network can be thought of as a kernel function which maps $\mathbb{R}^d \to \mathbb{R}^k$ when $k \gg d$



Assume that x is the input, θ is all parameters in the network, and $f(\theta, x)$ is the output of the network:
$\frac{df(\theta_t,x_t)}{dt} = -H(t) * (f(\theta_t, x_t) - y)$, where the (i, j)-th entry of $H(t)$ is $\langle \frac{\partial f(\theta_t,x_i)}{\partial \theta}, \frac{\partial f(\theta_t,x_j)}{\partial \theta} \rangle$

Thus $H(t)$ is the kernel matrix: $ker_t(x, x') = \langle \frac{\partial f(\theta_t,x)}{\partial \theta}, \frac{\partial f(\theta_t,x')}{\partial \theta} \rangle, \forall x, x' \in \mathbb{R}^d$
When the width of the network is infinite, the time-varying kernel would approximate to a fixed kernel of the form: $ker(x, x') = \mathbb{E}_{\theta\ w}[\langle \frac{\partial f(\theta_t,x)}{\partial \theta}, \frac{\partial f(\theta_t,x')}{\partial \theta} \rangle]$ Which is the Neural Tangent Kernel (NTK).



Theorem: The transfer risk is bounded by: $\mathbb{R}_n \leq p(\frac{\pi}{2} - \bar{\alpha}_n = \mathcal{O}(n^{-b/2})$

9 Empirical Perspective

Observation: Knowledge Distillation and Ensemble do not aid generalization when the input is Gaussian or Gaussian-like. Consider a binary classification problem, where the data contains 4 features, $v = \{v_1, v_2, v_3, v_4, \}$

For the first label, we have:

$$\begin{cases} v_1, v_2 = 1, v_3, v_4 = 0.1 & \text{w.p. } 80\% \\ v_1 = 1, v_3, v_4 = 0.1 & \text{w.p. } 10\% \\ v_2 = 1, v_3, v_4 = 0.1 & \text{w.p. } 10\% \end{cases}$$

And for the second label, we have:

$$\begin{cases} v_3, v_4 = 1, v_1, v_2 = 0.1 & \text{w.p. } 80\% \\ v_3 = 1, v_1, v_2 = 0.1 & \text{w.p. } 10\% \\ v_4 = 1, v_1, v_2 = 0.1 & \text{w.p. } 10\% \end{cases}$$

The training process for a single model is in two phases:

1. The network will quickly pick up one of the features $v \in \{v_1, v_2\}$ for the first label, and one of the features $v' \in \{v_3, v_4\}$ for the second label
2. The network will memorize the remaining 10% of the training examples without learning any new features

Single Model: 100% training accuracy, 90% testing accuracy.
v_1, v_2: first label. v_3, v_4: second label
Conclusion: Knowledge Distillation forces the model to learn every feature of the label.