EECS 182 Deep Neural Networks
Fall 2025 Anant Sahai and Gireeja Ranade — Discussion 12

1. Entropy, Cross-Entropy, Kullback - Leibler (KL)-divergence

Entropy is a measure of expected surprise. For a given discrete Random variable Y, we know that from Information Theory that a measure the surprise of observing that Y takes the value k by computing:

$$\log \frac{1}{p(Y=k)} = -\log[p(Y=k)]$$

As given:

- if $p(Y=k) \to 0$, the surprise of observing k approaches ∞
- if $p(Y=k) \to 1$, the surprise of observing k approaches 0

The Entropy of the distribution of Y is then the expected surprise given by:

$$H(Y) = E_Y\Big[-\log\big(p(Y=k)\big)\Big] = -\Sigma_k\Big[p(Y=k)\log[p(Y=k)]\Big]$$

On the other hand, Cross-entropy is a measure building upon entropy, generally calculating the difference between two probability distributions p and q. it is given by:

$$\begin{aligned} H(p,q) &= E_{p(x)}\Big[\frac{1}{\log\big(q(x)\big)}\Big] \\ &= \Sigma_x\Big[p(x)\log[\frac{1}{q(x)}]\Big] \end{aligned}$$

Relative Entropy also known as KL Divergence measures how much one distribution diverges from another. For two discrete probability distributions, p and q, it is defined as:

$$D_{KL}(p||q) = \Sigma_x\Big[p(x)\log[\frac{p(x)}{q(x)}]\Big]$$

(a) Let's define the following probability distributions given by:

$$p(x) = \begin{cases} 0.5 & \text{when x = 1} \\ 0.5 & \text{when x = -1} \end{cases}, \text{ and } q(x) = \begin{cases} 0.1 & \text{when x = 1} \\ 0.9 & \text{when x = -1} \end{cases}.$$

Note that $H(p) = -\log 0.5 = 1$ bit and $H(q) = -0.1\log 0.1 - 0.9\log 0.9 \approx 0.47$ bits. Show that KL-divergence is not symmetric and hence does not satisfy some intuitive attributes of distances.

Solution:

To show this, we need to show that:

$$D_{KL}(p||q) \neq D_{KL}(q||p)$$

$$\begin{aligned} D_{KL}(p||q) &= 0.5 \times \log\left[\frac{0.5}{0.1}\right] + 0.5 \times \log\left[\frac{0.5}{0.9}\right] \approx 0.74 \\ D_{KL}(q||p) &= 0.1 \times \log\left[\frac{0.1}{0.5}\right] + 0.9 \times \log\left[\frac{0.9}{0.5}\right] \approx 0.53, \end{aligned}$$

hence $D_{KL}(p||q) \neq D_{KL}(q||p)$

(b) Re-write $D_{KL}(p||q)$ in term of the Entropy $H(p)$ and the cross entropy $H(p, q)$.

Solution:

$$\begin{aligned} D_{KL}(p||q) &= \Sigma_x \Big[p(x) \log[\frac{p(x)}{q(x)}]\Big] \\ &= \Sigma_x \Big[p(x)[\log\big(p(x)\big) - \log\big(q(x)\big)]\Big] \\ &= E_{p(x)}\Big[\log\big(p(x)\big)\Big] - E_{p(x)}\Big[\log\big(q(x)\big)\Big] \\ &= -E_{p(x)}\Big[\log\big(q(x)\big)\Big] + E_{p(x)}\Big[\log\big(p(x)\big)\Big] \\ &= E_{p(x)}\Big[\frac{1}{\log\big(q(x)\big)}\Big] - E_{p(x)}\Big[\frac{1}{\log\big(p(x)\big)}\Big] \\ &= H(p, q) - H(p) \end{aligned}$$

2. Catastrophic Forgetting

The neural networks are vulnerable to the distributional shift. Many questions in AI/ML are related to the distributional shift: out-of-distribution (OoD), domain adaptation/generalization, meta-learning, and so on. In this discussion, we study one of those problems, catastrophic forgetting, alternatively called catastrophic interference. The catastrophic forgetting is the tendency of neural networks to lose information about previously learned tasks when learning the new one. This is also referred to as stability-plasticity dilemma[1]. One potential drawback of a model that is too stable is that it will not be able to consume new information from the future training data. Conversely, a model with too much plasticity may suffer from large weight changes and forgetting of previously learned representations.

[1] https://www.sciencedirect.com/science/article/pii/S1364661399012942

(a) What happens in parameter space

Figure 1a is the cartoon parameter space of the given model and tasks. The dark orange oval is the simplified low-loss region of task 0, and the light blue oval is that of task 1. The model is trained on the task 0, and θ_0^* is the trained parameter of the model, which minimizes the loss for that task. We now train the model with task 1 data.

i. Mark the θ_1^* on the Figure 1a, which is the trained parameter for task 1 if the model is too plastic.

Solution: If the model is too plastic, the model lose every information that it learned from task 0. Therefore, θ_1^* is the middle point of the light blue oval and goes outside of the dark orange oval.

ii. Mark the $\tilde{\theta_1}$ on the Figure 1a, which is the trained parameter for task 1 if the model is too stable.

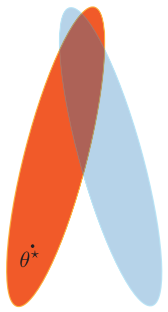
Solution: If the model is too stable, the model cannot learn any new information from the task 1. Therefore, $\tilde{\theta_1} = \theta_0^*$. This is called catastrophic remembering. The model suffering from catastrophic remembering cannot transfer any knowledge for learning the new task (e.g. learning rate is too small).

iii. What are the problems if the model is too plastic or stable?

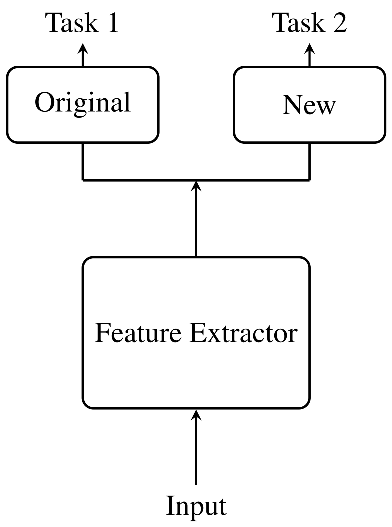
Solution: Plastic: catastrophic forgetting. Stable: catastrophic remembering

iv. Where do we want to go?

Solution: We want to find the θ_1^* that is in the middle of the dark orange and light blue ovals. The model should be able to learn new information from the new task while keeping the old task information.



(a) The cartoon parameter space for feature extractor.



(b) Feature extractor architecture with task-specific heads.

Figure 1: Catastrophic forgetting: (a) Parameter space visualization showing loss landscapes for different tasks, and (b) Feature extractor architecture with task-specific heads.

(b) Dealing with catastrophic forgetting

We consider three traditional approaches for learning the new tasks: feature extraction, fine-tuning, and joint training.

i. frozen feature extraction - The model trained with the previous tasks is frozen. The output of this model with the new task input is used for to train the new classifier for the new task.

ii. full fine-tune on new task - The model trained with the previous tasks is trained with the new task. To prevent the large shift, the learning rate is typically low.

iii. joint training full fine-tune - The model is trained with both previous task data and new task data

Let's study pros and cons of those methods. Fill in the table below.

Category	frozen feature extraction	full fine-tune on new task	joint training
New task performance	**Solution:** Good	**Solution:** Best in most cases	Good
Old task performance	Good	**Solution:** Bad	Good
Storing old task data	No	No	**Solution:** yes

Solution:
The solution to the first row "**New task performance**" is more subtle than a universal rule of one being better than the other. Fine-tuning should perform the best in the new task since the model only has to fit the new task data instead of considering *both* the old and new task, which is often a more complex distribution. This narrative is valid when the fine-tuning dataset is large enough (the concept of enough varies among applications). However, when the fine-tuning data is too small, feature extraction can perform better than finetuning if the model capacity is so large that overfitting to the new task is likely. Some examples are when one uses BERT features and CLIP (Contrastive Language Image Pretraining) scores, which are both trained on immense dataset, it is very unlikely to finetune the entire BERT or CLIP model on a reasonable budget.

In conclusion, whether feature extraction or finetuning is better for your application depends on the size of the old and new task datasets, and the model capacity.

3. How to read research papers

One critical skill for improving yourself in deep learning is to quickly read papers. Being able to read papers efficiently and to identify the key contributions regardless of what the authors claimed are some critical skills as you work in a field where some fundamental theory is still an open problem. In this question, you will be reading Learning without Forgetting (Li et al.), a paper that proposes a simple but effective strategy to alleviate the problem of Catastrophic Forgetting.

But first, let's talk about how to read a deep learning paper. Every researcher has a different strategy, but one recurring pattern is to *read with multiple passes*.

- First pass: Read the **title**, **abstract** and **figures**
- Second pass: Read the **introduction**, **conclusion** and **figures** again
- Third pass: Read the **method** (skip or skim the math) and skim over the **results** (skip ablation study)
- Fourth pass: Read everything else but skip parts that do not make sense (unless you are doing research in that particular field)

Solution:
Here are some more concrete guidelines (many of which adapted from Andrew Ng's CS230 lecture at Stanford in 2018):

- **Why start with figures?** A common pattern in ML/DL papers is that often there is one or two figures that summarizes the entire paper well, either the architecture or method.

- **Skipping *Related Work*.** Some reason to do this are (1) Related Work section is meant to be a place to look for extension of ideas if you're interested in this topic, so chances are unless you are familiar with this topic beforehand, you won't understand the brief description of related papers well (2) due to the nature of peer review, sometimes people cite lots of papers that one believes could potentially be their reviewers, which might not all be helpful material for learning.
- **Why skip parts that do not make sense?** This might be a seemingly counterintuitive statement. The main reason is that papers are often state-of-the-arts research and sometimes even the authors do not know exactly why a new phenomenon emerges. There are tons of examples of unimportant modules in influential papers: Transducers in LeNet-5 (set many foundations for ConvNets), Memory bank in Contrastive Learning... etc. If one is doing research on the same topic as the paper, then understanding thoroughly can be a good idea. Yet most students will be reading ***many more*** papers outside of their research / project, in those cases, focusing on the main idea is more effective.
- **Why skip the math first?** Beginners often get caught up by one or two math equations before understanding the entire picture of the paper. However, whether you understand every single equation does not necessarily indicate your knowledge on the main idea of this paper, and not all papers are well-written. Knowing the high-level goals and methods first can help one understand the math much easier later.
- **Why skim over results?** Results are important. Yet most of the good papers (either accepted to conferences or being viral on Twitter) are state-of-the-arts in some way already. Focusing on the Method section teaches more for your own future projects than the results do.

Now spend some time to reading the Learning without Forgetting paper with this multiple-pass strategy, and discuss with your classmates for the following questions.

(a) What was the challenge authors are trying to overcome?

Solution: The authors are trying to overcome The authors proposed Learning without Forgetting (LwF), which allows a network to learn new tasks without forgetting old tasks. The method leverages knowledge distillation, where the original network is used as a "teacher" to guide the learning of a new network. The LwF approach involves training the network on new tasks while simultaneously trying to maintain its performance on the old tasks by minimizing the differences between the outputs of the original and updated networks.

(b) How is Learning without Forgetting (LwF) different from standard fine-tuning and joint training (multitask learning)?

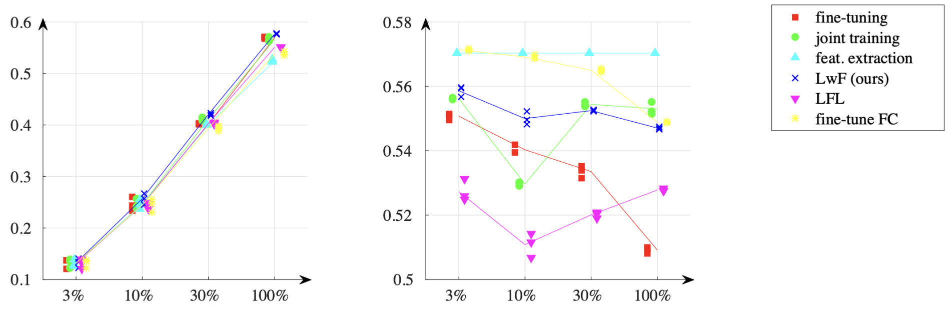
Solution: Standard finetuning is the most commonly used adaptation method, but also the most likely to suffer from catastrophic forgetting among the methods mentioned in Q1. LwF can preserve the model's performance better by using knowledge distillation on the "old task branch". LwF differs from joint training because it allows the model to learn new tasks sequentially, without access to the previous tasks' data.

(c) Is LwF better than feature extraction in both new and old tasks? How does LwF compare with joint training in terms of accuracy?

Solution: LwF performs better in new tasks, but worse in the old tasks. This is reasonable since feature extraction freezes the features which leaves much smaller flexibility for the model to fit the new tasks. On the other hand, LwF slightly outperforms joint training on new tasks, but slightly underperforms on old tasks.

(d) Does the new task dataset size affect LwF's effectiveness? What about the old task dataset size?

Solution: In this figure, we see that all methods suffer from the decrease of new task's data, while



(a) CUB accuracy (new) (b) ImageNet accuracy (old)

Fig. 5. Influence of subsampling new task training set on compared methods. The x-axis indicates diminishing training set size. Three runs of our experiments with different random θ_n initialization and dataset subsampling are shown. Scatter points are jittered horizontally for visualization, but line plots are not jittered to facilitate comparison. Differences between LwF and compared methods on both the old task and the new task decrease with less data, but the observations remain the same. LwF outperforms fine-tuning despite the change in training set size.

Figure 2: the influence of the dataset size

feature extraction suffers the most. LwF compares favorably with joint training in all sizes of new task's dataset size. As the size of old task's dataset changes, we observe that fine-tuning is affected most drastically, where joint training and feature extraction remain fairly robust.

Contributors:

- Jerome Quenum.
- Anant Sahai.
- Suhong Moon.
- Kumar Krishna Agrawal.
- Sultan Daniels.
- Kevin Li.