Rethinking fine-tuning to mitigate feature distortion

Aditi Raghunathan
Large-scale pretraining

Step one: pretraining

Diverse (typically unlabeled) data

Pretrained model

Step two: adaptation

Specialize to narrow distribution

Bommasani et al. 2021
Robustness to distribution shifts

A core challenge for reliable machine learning in the wild

Train

Pedestrians using a crosswalk

Deploy

Skateboarders

Important pedestrians

Pedestrians jaywalking
Distribution shifts are everywhere

Train Deploy
Satellite remote sensing (different regions)

Train Deploy
Wildlife conservation (different forests)

Train Deploy
Tumor detection (new hospitals)

Sim-to-real

The generalization challenge

From scratch

Pretraining

Fine-tuning
The promise of large-scale pretraining

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ImageNet ResNet101</th>
<th>CLIP ViT-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet</td>
<td>76.2%</td>
<td>76.2%</td>
</tr>
<tr>
<td>ImageNet V2</td>
<td>64.3%</td>
<td>70.1%</td>
</tr>
<tr>
<td>ImageNet Rendition</td>
<td>37.7%</td>
<td>88.9%</td>
</tr>
<tr>
<td>ObjectNet</td>
<td>32.6%</td>
<td>72.3%</td>
</tr>
<tr>
<td>ImageNet Sketch</td>
<td>25.2%</td>
<td>60.2%</td>
</tr>
<tr>
<td>ImageNet Adversarial</td>
<td>2.7%</td>
<td>77.1%</td>
</tr>
</tbody>
</table>

More data generally helps
The generalization problem revisited

Step one: pretraining

Diverse (typically unlabeled) data

Step two: adaptation

Specialize to narrow distribution

Pretrained model

Supervision during adaptation is still coming from limited data
The generalization challenge revisited

From scratch

Pretraining

Fine-tuning

How to retain information beyond the limited data used for adaptation?
The “art” of neural network training

- What parameters to update (model family)
- Loss function
- Optimization hyperparameters
The “art” of neural network training

- What parameters to update (model family)
- Loss function
- Optimization hyperparameters
Linear probing vs (full) fine-tuning

Pop quiz!

Kumar, Raghunathan, Jones, Ma, and Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. ICLR 2022.
Dataset: BREEDS Living-17

**Task:** classify into animal categories

**Train distribution:** one subset of ImageNet hierarchy tree with animal category as root

**Test distribution:** other subset of ImageNet hierarchy tree with animal category as root

**Pretrained model:** MoCo-V2, which has seen *unlabeled* ImageNet images (including various types of animals)

Santurkar et al. 2020
Pop quiz: living-17

<table>
<thead>
<tr>
<th>Living-17</th>
<th>ID</th>
<th>OOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scratch</td>
<td>92.4%</td>
<td>58.2%</td>
</tr>
<tr>
<td>Linear probing</td>
<td>96.5%</td>
<td>?</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>97.1%</td>
<td></td>
</tr>
</tbody>
</table>

Does linear probing do better than scratch OOD?
### Pop quiz: living-17

<table>
<thead>
<tr>
<th>Living-17</th>
<th>ID</th>
<th>OOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scratch</td>
<td>92.4%</td>
<td>58.2%</td>
</tr>
<tr>
<td>Linear probing</td>
<td>96.5%</td>
<td>82.2%</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>97.1%</td>
<td></td>
</tr>
</tbody>
</table>

Does linear probing do better than scratch OOD? **Yes!**
Pop quiz: living-17

<table>
<thead>
<tr>
<th>Living-17</th>
<th>ID</th>
<th>OOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scratch</td>
<td>92.4%</td>
<td>58.2%</td>
</tr>
<tr>
<td>Linear probing</td>
<td>96.5%</td>
<td>82.2%</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>97.1%</td>
<td>?</td>
</tr>
</tbody>
</table>

Does fine-tuning do better than linear probing OOD?

Kumar, Raghunathan, Jones, Ma, and Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. ICLR 2022.
### Pop quiz: living-17

<table>
<thead>
<tr>
<th>Living-17</th>
<th>ID</th>
<th>OOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scratch</td>
<td>92.4%</td>
<td>58.2%</td>
</tr>
<tr>
<td>Linear probing</td>
<td>96.5%</td>
<td>82.2%</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>97.1%</td>
<td>77.7%</td>
</tr>
</tbody>
</table>

Does fine-tuning do better than linear probing OOD? **No!**

Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.
Dataset: CIFAR 10.1

**Task:** classify into CIFAR-10 categories

**Train distribution:** original CIFAR-10 dataset

**Test distribution:** recent near-replication of the pipeline

**Pretrained model:** MoCo-V2, which has seen *unlabeled* ImageNet images

Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.
Pop quiz: CIFAR10.1

<table>
<thead>
<tr>
<th>Living-17</th>
<th>ID</th>
<th>OOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear probing</td>
<td>91.8%</td>
<td>82.7</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>97.3%</td>
<td>?</td>
</tr>
</tbody>
</table>

Does linear probing do better than fine-tuning OOD?
Pop quiz: CIFAR10.1

<table>
<thead>
<tr>
<th>Living-17</th>
<th>ID</th>
<th>OOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear probing</td>
<td>91.8%</td>
<td>82.7</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>97.3%</td>
<td>92.3%</td>
</tr>
</tbody>
</table>

Does linear probing do better than fine-tuning OOD? **No!**
Linear probing vs fine-tuning summary

Kumar, Raghunathan, Jones, Ma, and Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. ICLR 2022.

Which method does better?
Linear probing vs fine-tuning summary

<table>
<thead>
<tr>
<th></th>
<th>ID</th>
<th>OOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear probing</td>
<td>82.9%</td>
<td></td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>85.1%</td>
<td></td>
</tr>
</tbody>
</table>

Averaged over 10 datasets

Common wisdom is fine-tuning works better than linear probing

Kumar, Raghunathan, Jones, Ma, and Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. ICLR 2022.
# Linear probing vs fine-tuning summary

<table>
<thead>
<tr>
<th></th>
<th>ID</th>
<th>OOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear probing</td>
<td>82.9%</td>
<td>66.2%</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>85.1%</td>
<td>59.3%</td>
</tr>
</tbody>
</table>

*Averaged over 10 datasets*

LP performs better than FT OOD on 8 out of 10 datasets.
Intuition for theoretical result

Kumar, Raghunathan, Jones, Ma, and Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. ICLR 2022.
Intuition for theoretical result

Pretrained Features

Fine-tuning: features for ID examples change in sync with the linear head

Features for OOD examples change less

Kumar, Raghunathan, Jones, Ma, and Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. ICLR 2022.
Intuition for theoretical result

Pretrained Features

Fine-tuning: features for ID examples change in sync with the linear head

Features for OOD examples change less
Intuition for theoretical result

Pretrained Features

Fine-tuning: features for ID examples change in sync with the linear head

Features for OOD examples change less

Kumar, Raghunathan, Jones, Ma, and Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. ICLR 2022.
Intuition for theoretical result

Pretrained Features

Fine-tuning: features for ID examples change in sync with the linear head

Features for OOD examples change less

Kumar, Raghunathan, Jones, Ma, and Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. ICLR 2022.
Intuition for theoretical result

Pretrained Features

Fine-tuning: features for ID examples change in sync with the linear head

Linear probing: freezes pretrained features

Head performs poorly on OOD examples

Head is decent on OOD examples
Key takeaway

A larger change in parameters can **distort** pretrained features

How to retain information beyond the limited data used for adaptation?
Best of both worlds

Why does FT do better ID?

Training data may not be linearly separable in the space of pre-trained features i.e. imperfect pre-trained features

Why does FT do worse OOD?

Features can change a lot to accommodate a randomly initialized head

Can we refine features without distorting them too much?
Method to achieve best of both worlds

Idea: modify pre-trained features only as necessary

Step 1: Linear probe

Step 2: Fine-tune

Kumar, Raghunathan, Jones, Ma, and Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. ICLR 2022.
Method to achieve best of both worlds

Idea: modify pre-trained features only as necessary

Step 1: Linear probe
Step 2: Fine-tune

LP-FT method

Can prove that LP-FT dominates both LP and FT under the simple setting of perfect features
Improving fine-tuning

<table>
<thead>
<tr>
<th>Method</th>
<th>ID</th>
<th>OOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear probing</td>
<td>82.9%</td>
<td>66.2%</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>85.1%</td>
<td>59.3%</td>
</tr>
<tr>
<td>LP-FT</td>
<td>85.7%</td>
<td>68.9%</td>
</tr>
</tbody>
</table>

LP-FT obtains better than the best of both worlds

Kumar, Raghunathan, Jones, Ma, and Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. ICLR 2022.
The “art” of neural network training

- What parameters to update (model family)
- Loss function
- Optimization hyperparameters
The loss function

Contrastive pretraining

Maximize scores

Can we reduce distortion?
Revisiting the fine-tuning loss function

**Contrastive pretraining**

- Image Encoder
  - $I_1$, $I_2$, ..., $I_B$
- Text Encoder
  - $T_1$, $T_2$, ..., $T_B$

- $I_1 \cdot T_1$, $I_1 \cdot T_2$, ..., $I_B \cdot T_B$
- $I_2 \cdot T_1$, $I_2 \cdot T_2$, ..., $I_B \cdot T_B$
- $\vdots$
- $I_B \cdot T_1$, $I_B \cdot T_2$, ..., $I_B \cdot T_B$

- Maximize scores

**Finetune like you pretrain (FLYP)**

- $I_1$, $I_2$, ..., $I_B$
- $I_B$ (Image Encoder)
- $T_1$, $T_2$, ..., $T_B$

- $I_1 \cdot T_1$, $I_1 \cdot T_2$, ..., $I_B \cdot T_B$
- $I_2 \cdot T_1$, $I_2 \cdot T_2$, ..., $I_B \cdot T_B$
- $\vdots$
- $I_B \cdot T_1$, $I_B \cdot T_2$, ..., $I_B \cdot T_B$

- Text Encoder
  - $y_1$, $y_2$, ..., $y_B$

- “A photo of a [label]”
Fine-tune like you pretrain

Same pretraining loss can reduce distortion and improve robustness.

- Full finetuning
- L2-sp (baseline)
- LP-FT
- FLYP (ours)
Fine-tune like you pretrain

Also see gains in few-shot learning

<table>
<thead>
<tr>
<th></th>
<th>PatchCamelyon</th>
<th>SST2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero shot</td>
<td>56.5%</td>
<td>60.5%</td>
</tr>
<tr>
<td>FT</td>
<td>63.1%</td>
<td>61.1%</td>
</tr>
<tr>
<td>LP-FT</td>
<td>62.7%</td>
<td>60.9%</td>
</tr>
<tr>
<td>FLYP</td>
<td>66.9%</td>
<td>61.3%</td>
</tr>
</tbody>
</table>
Summary

• Pretrained models give large improvements in accuracy, but how we fine-tune them is key

• General principle: minimize distortion while fine-tuning

• Two simple ways to do that
  • LP-FT (only change features once the head is trained)
  • FLYP (keep the fine-tuning loss identical to the pretraining loss)
Thanks!

Ananya Kumar  Robbie Jones  Tengyu Ma  Percy Liang

Sachin Goyal  Sankalp Garg  Zico Kolter

Apple  Google  Schmidt Futures  Open Philanthropy