Lec 8 Fine tuning, Instruction tuning

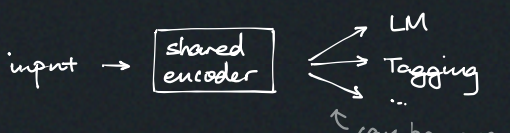
Some tasks have limited naturally occurring data

Generally naturally occurring text works pretty well
└ e.g. scrapped data already has translation, etc. e.g. output Romanised Japanese
GPT could translate... but also learns to translate unprofessionally

If not naturally occurring... hand label + supervise training

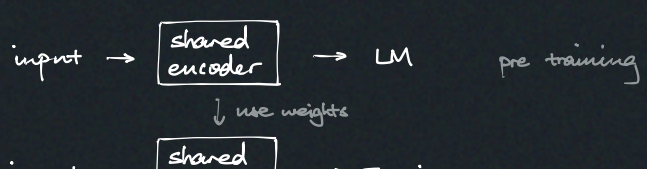
Approaches

▷ Multitask Learning



↑ can be one task per batch or just mixed together

▷ Fine tuning



Good:
- Can fine tune on cleaner data
- Cheaper than training from scratch

Problems:
- Multitask may be more informative than only learning one thing. Model can learn better representation
  └ e.g. find optima in multitask

▷ Prompting → last lec

▷ Instruction Tuning ← Combines prompting & fine-tuning

→ Various instructions for different tasks, and fine tune

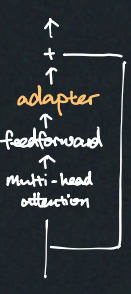
Fine-tuning

Just continue training on desired data

Issue — if large model, high memory requirement
└ train 65B LLaMa1 ... 1000-1400 VRAM on 16-bit precision
no single GPU can do that probably...
→ Multi-GPU training
└ DeepSpeed ZeRo partion strategy
optimizer state, gradient, params
usually partitioning this doesn't hurt perf that much

▷ Param-efficient fine tuning

→ Prefix tuning
→ Adapter
└ freeze transformer, insert simple adapter networks in the middle that can be fine tuned



etc.

adapter:

- Adapter fusion — have multiple adapters trained separately then have an attention over the different adapters
  e.g. per task adapter / per lang adapter

- LoRa — similar idea. No linear layer, just downscale + upscale matrix.
  After finishing LoRa, learnt weight matrix can be added to original model's weight

- QLoRa — 4 bit quantisation of transformer params
  add paging to save VRAM
  then train LoRa

→ BitFit — freeze everything except the biases, just fine tune biases

Some NLP Tasks to fine tune towards

- Context-free Q&A
  └ answer questions without lookup
  └ Datasets: MMLU
- Contextual Q&A
  └ answer question given document(s)
  └ retrieval based: given all documents
- Code generation
  └ Dataset: HumanEval
- Summarisation
  └ document / documents
  we do well pretty open problem
  └ e.g. survey / report generation
- Info extraction
  └ entity recognition / linking / reference
  └ dataset: OntoNotes
- Translation
  └ eval based on similarity to reference
  └ FLORES dataset (101 langs)
- General Purpose
  └ language task across many tasks
  └ e.g. BIGBench

Eval task complexity
Active research on how to calculate & control this

Note we try to not have the test data show up in train data.
Heuristic: change order of options, change numbers, see if purtabation in perf
Test datasets can leak :(

Instruction Tuning

This usually generalise to new tasks
← supervised on many task, follow similar format
→ Basic: tune to follow instructions on many tasks
inference on unseen tasks ← Yes they do better at new tasks
→ Learn to learn from in-context examples
└ just train on prompts with in-context examples

▷ Instruction-tuned models:

-	FLAN-T5 good at input-output	11B params
-	LLaMa2 Chat	70B
-	Mistral instruct	45B

Dataset Generation

→ Self instruct: start from some tasks, generate some more
→ ORCA: generated explanations for chain of thought