Introduction to Fine-Tuning

Yogesh Haribhau Kulkarni

YHK

Outline

YHK

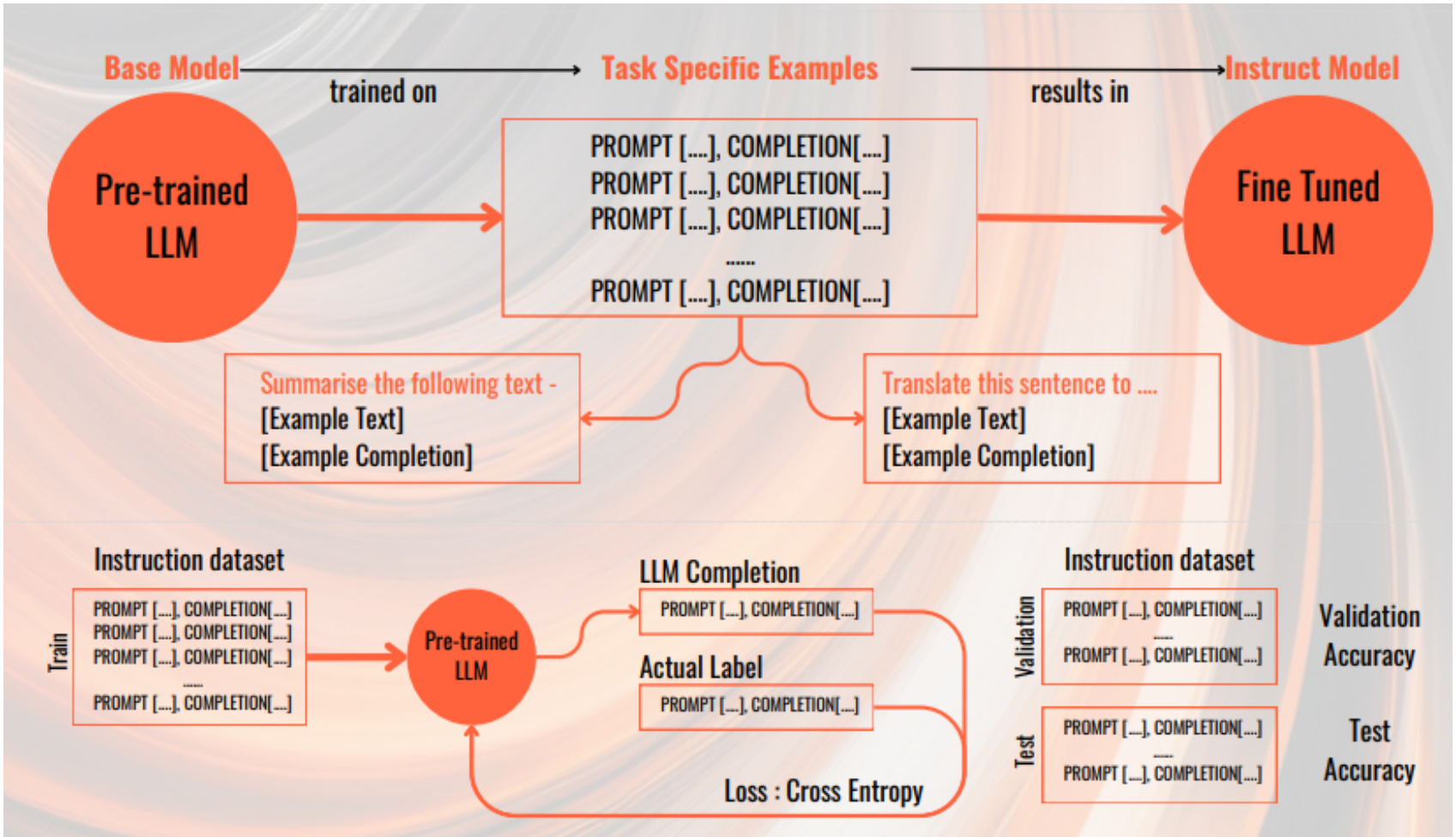
Fine-tuning LLMs (Large Language Models)

Introduction to Fine-Tuning

- **Definition:** Process of training pre-existing models on smaller, domain-specific datasets to enhance task or domain performance.
- **Objective:** Refine capabilities and adapt models for specific applications.
- **Illustration with GPT-3:**
  - GPT-3 designed for diverse NLP tasks but may lack optimization for specific domains.
  - Example: Healthcare organization fine-tunes GPT-3 for generating patient reports, adapting to medical terminologies.
- **Applicability Beyond LLMs:**
  - Not exclusive to language models; applicable to various machine learning models.
  - Example: Convolutional neural network fine-tuning for detecting trucks on highways.
- **Key Principle:** Leverage pre-trained models, recalibrate parameters using novel data for new contexts or applications.
- **Considerations:** Beneficial when training data distribution significantly differs from specific application requirements.

YHK

What is a fine-tuned LLM?



(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

YHK

Why Fine-Tuning?

- **General Models vs. Specialization:**
  - Large language models designed for versatility, not task mastery.
  - Fine-tuning essential for exceptional proficiency in specific tasks or domains.
- **Versatility vs. Mastery:**
  - Generic models proficient in multiple tasks but lack mastery.
  - Fine-tuned models optimized for specific tasks, achieving mastery.
- **Decision to Fine-Tune:**
  - Driven by the need for superior performance in targeted applications.
  - Highly effective specialists in designated areas.

YHK

Why Fine-Tuning?

- Pre-trained models are trained and built with general-purpose tasks, with Fine-tuning we can improve the performance of pre-trained models in wide range of domain-specific tasks.
- Fine-tuning is a technique where a pre-trained model is trained on a new dataset.
- Fine Tuning has many approaches, one uses adjusting last layer with retraining on custom data, another has adopter auxiliary network which gets trained on custom data, some have full retraining with frozen earlier weights.
- Fine-tuning leverages a pre-trained model's knowledge and capabilities, saving significant time and resources compared to training a model from scratch.
- Improve factual response by utilizing Domain-specific data.
- Reduce Hallucinations.

YHK

What is a fine-tuned LLM?

- **Context Learning Limitation:**
  - Through in-context learning or prompting, only a limited performance level is achievable.
- **Challenges of Few-Shot Learning:**
  - Few-shot learning may not be effective for smaller LLMs.
  - Consumes significant space in the context window.
- **Fine Tuning Overview:**
  - Supervised learning process adjusting LLM weights.
  - Uses a labeled dataset of prompt-completion pairs.
- **Instruction Fine Tuning:**
  - Trains LLM on examples of instructions and desired responses.
  - Improves performance on instruction-specific tasks.
- **Full Fine Tuning:**
  - Updates all LLM parameters.
  - Requires sufficient memory for storing and processing gradients and components.

(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

YHK

Reasons for Fine-Tuning - Part 1

- **Domain-Specific Adaptation:**
  - Pre-trained LLMs not optimized for specific tasks or domains.
  - Fine-tuning allows adaptation to nuances, enhancing performance.
  - Example: Fine-tuning for document analysis in the legal domain.
- **Shifts in Data Distribution:**
  - Models may not generalize well to out-of-distribution examples.
  - Fine-tuning aligns the model with new data distribution, improving performance.
  - Example: Fine-tuning a sentiment analysis model for social media comments.
- **Cost and Resource Efficiency:**
  - Training from scratch requires a large dataset, fine-tuning is more efficient.
  - Example: Adapting a pre-trained model for small e-commerce platform recommendations.

YHK

Reasons for Fine-Tuning - Part 2

- **Out-of-Distribution Data Handling:**
  - Fine-tuning mitigates suboptimal performance with modest dataset.
  - Example: Fine-tuning a speech recognition model for a new regional accent.
- **Knowledge Transfer:**
  - Fine-tuning transfers general knowledge from pre-trained models to specific tasks.
  - Example: Transferring medical knowledge to a healthcare chatbot.
- **Task-Specific Optimization:**
  - Fine-tuning optimizes model parameters for specific task objectives.
  - Example: Optimizing a pre-trained model for code generation in software development.

YHK

Reasons for Fine-Tuning - Part 3

- **Adaptation to User Preferences:**
  - Fine-tuning aligns the model with user preferences and task requirements.
  - Example: Fine-tuning a virtual assistant model for user-specific language and tone.
- **Continual Learning:**
  - Fine-tuning supports continual learning, adapting to evolving data and user needs.
  - Example: Continually updating a news summarization model for evolving news topics.

YHK

Challenges and Solutions: In-Context Learning

- Prominent technique for task-specific adaptation.
- Challenges include limited context window and real-time optimization.
- Context window restricts processing of long sequences.
- Real-time optimization is difficult during progression through the context window.

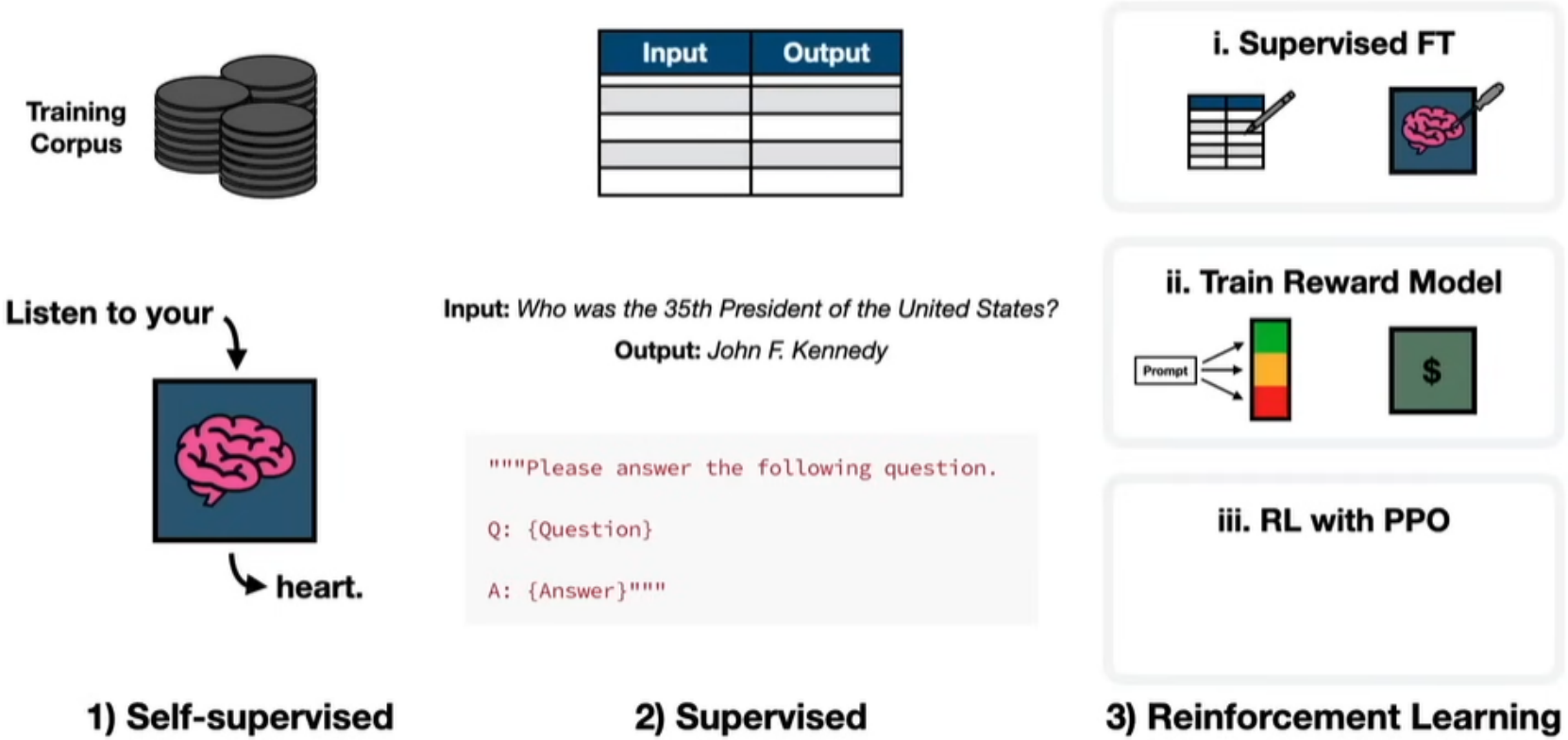
YHK

Challenges with Few-Shot Learning

- Effective for larger LLMs, less so for smaller models.
- Smaller models struggle to learn complex patterns from few examples.
- Few-shot learning demands a large context window.
- Resource-intensive and impractical for memory-constrained environments.

YHK

Ways to fine-tune

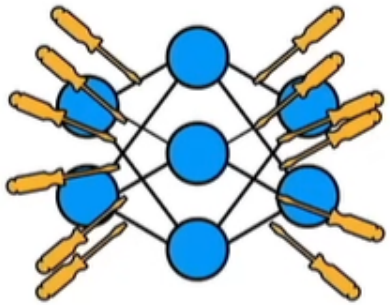


(Ref: Fine-tuning Large Language Models (LLMs) — w/ Example Code - Shaw Talebi)

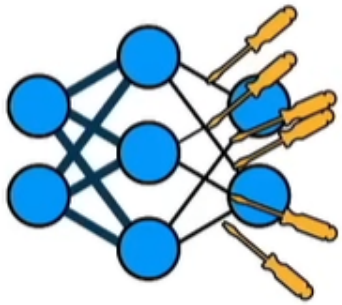
YHK

Ways to change parameters/weights

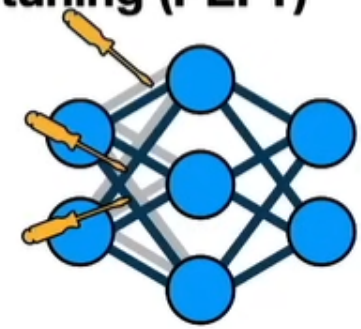
1) Retrain all parameters



2) Transfer Learning



3) Parameter Efficient Fine-tuning (PEFT)



ɔ][6]

(Ref: Fine-tuning Large Language Models (LLMs) — w/ Example Code - Shaw Talebi)

YHK

Understanding Fine-Tuning

- Widely used for customizing pre-trained LLMs to specific tasks.
- Involves training on labeled prompt-completion pairs.
- Allows model to adjust weights for task alignment.

YHK

Unsupervised Fine-Tuning Methods

- **Unsupervised Full Fine-Tuning:**
  - Relevant for updating LLM knowledge base without changing existing behavior.
  - Example: Fine-tuning on legal literature or adapting to a new language using unstructured datasets.
- **Contrastive Learning:**
  - Trains the model to discern between similar and dissimilar examples in the latent space.
  - Beneficial for tasks requiring nuanced understanding of similarities and distinctions.

YHK

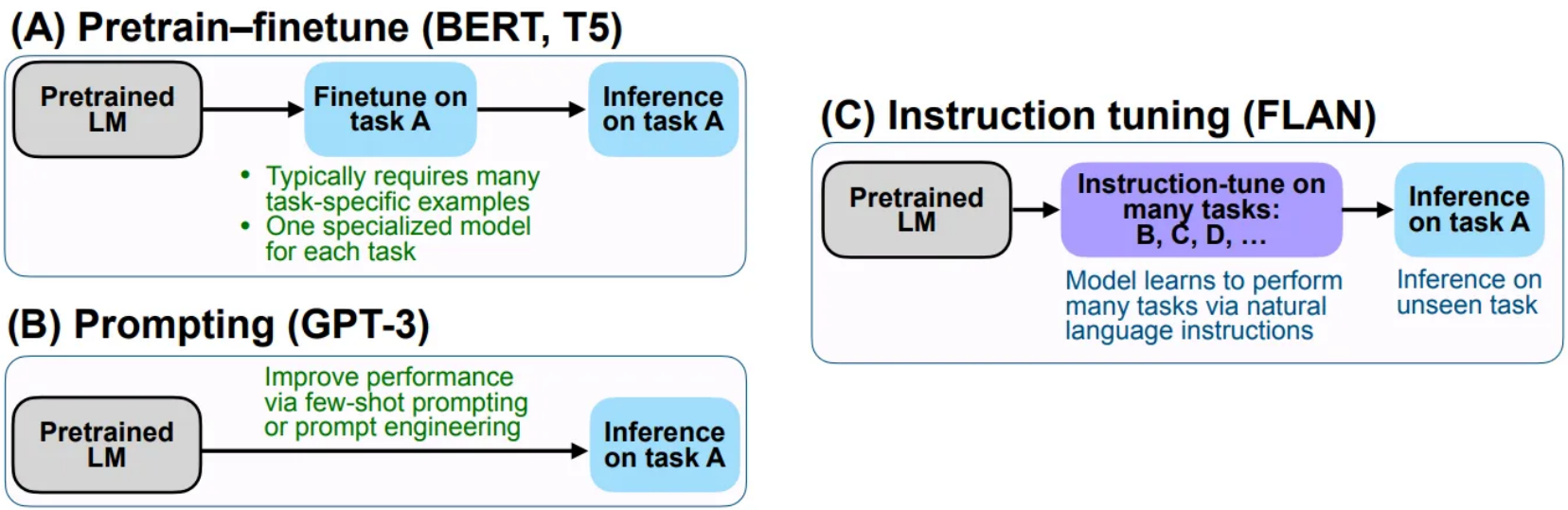
Supervised Fine-Tuning Methods

- **Parameter-Efficient Fine-Tuning (PEFT):**
  - Aims to reduce computational expenses by selectively updating a small set of parameters.
  - Example: Low-rank adaptation (LoRA) technique focuses on updating only relevant parameters.
- **Supervised Full Fine-Tuning:**
  - Involves updating all parameters of the language model during training.
  - Resource-intensive but ensures thorough adaptation of the entire model to the task or domain.
- **Instruction Fine-Tuning:**
  - Involves training the model using examples with explicit instructions for specific queries or tasks.
  - Suitable for applications where precise task execution is essential.
- **Reinforcement Learning from Human Feedback (RLHF):**
  - Incorporates reinforcement learning principles with human evaluators providing ratings.
  - Ratings serve as rewards, guiding the model to optimize parameters based on human preferences.

YHK

Instruction Fine-Tuning

- **Overview:**
  - Instruction fine-tuning enhances LLMs for real-world applications.
  - Differs from standard supervised fine-tuning by augmenting examples with explicit instructions.
  - Instruction-tuned models generalize effectively to new tasks, providing additional context.
- **Value in NLP and ML:**
  - Instruction fine-tuning is crucial in the evolving landscape of NLP and ML.
  - Enables LLMs to adapt to specific tasks with nuanced instructions.



(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

YHK

Natural Instructions Dataset

- **Dataset Overview:**
  - "Natural Instructions" dataset consists of 193,000 instruction-output examples.
  - Sourced from 61 existing English NLP tasks.
  - Structured approach with crowd-sourced instructions aligned to a common schema.
- **Unique Features:**
  - Each instruction associated with a task, providing explicit guidance for the model.
  - Covers fields like definition, things to avoid, positive and negative examples.
  - Structured nature makes it valuable for fine-tuning, offering clear and detailed instructions.

YHK

Natural Instructions Example

question generation (from MC-TACO)

- **Title:** Writing questions that involve commonsense understanding of "event duration".
- **Definition:** In this task, we ask you to write a question that involves “event duration", based on a given sentence. Here, event duration is defined as the understanding of how long events typically last. For example, “brushing teeth”, usually takes few minutes.
- **Emphasis & Caution:** The written questions are not required to have a single correct answer.
- **Things to avoid:** Don't create questions which have explicit mentions of answers in text. Instead, it has to be implied from what is given. In other words, we want you to use "instinct" or "common sense".

Positive Example

- **Input:** Sentence: Jack played basketball after school, after which he was very tired.
- **Output:** How long did Jack play basketball?
- **Reason:** the question asks about the duration of an event; therefore it's a temporal event duration question.

Negative Example

- **Input:** Sentence: He spent two hours on his homework.
- **Output:** How long did he do his homework?
- **Reason:** We DO NOT want this question as the answer is directly mentioned in the text.
- **Suggestion:** -

- **Prompt:** Ask a question on "event duration" based on the provided sentence.

Task Instance

- **Input:** Sentence: Still, Preetam vows to marry Nandini if she meets him again.
- **Expected Output:** How long had they known each other?

answer generation (from Winogrande)

- **Title:** Answering a fill in the blank question on objects
- **Definition:** You need to answer a given question containing a blank (_). Your answer must be one of the two objects mentioned in the question for example "trophy" and "suitcase".
- **Emphasis & Caution:** -
- **Things to avoid:** Your answer must not contain a word that is not present in the question.

Positive Example

- **Input:** Context word: fit. Question: The trophy doesn't fit into the brown suitcase because _ is too large.
- **Output:** trophy
- **Reason:** Answer is one of the objects ("trophy" and "suitcase") in the question. Since the blank is a "large" object that didn't fit the "suitcase", the answer must be "trophy".

Negative Example

- **Input:** Context word: fit. Question: The trophy doesn't fit into the brown suitcase because _ is too large.
- **Output:** bottle
- **Reason:** The issue is that the answer is not one of the objects present in the question which are "trophy" and "suitcase". Note that, a valid answer must be one of the objects present in the question.
- **Suggestion:** -

- **Prompt:** Answer a fill in the blank question that is based on a provided context word.

Task Instance

- **Input:** Context Word: Story. Question: After watching the movie Kelly began to work on her own story. The _ was for her research.
- **Expected Output:** movie

(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

YHK

Leveraging Instruction Fine-Tuning

- Extension of traditional fine-tuning.
- Trains model on examples of instructions and desired responses.
- Offers improved interpretability, controlled outputs, and reduced biases.
- Enables explicit task specification learning for enhanced performance.

YHK

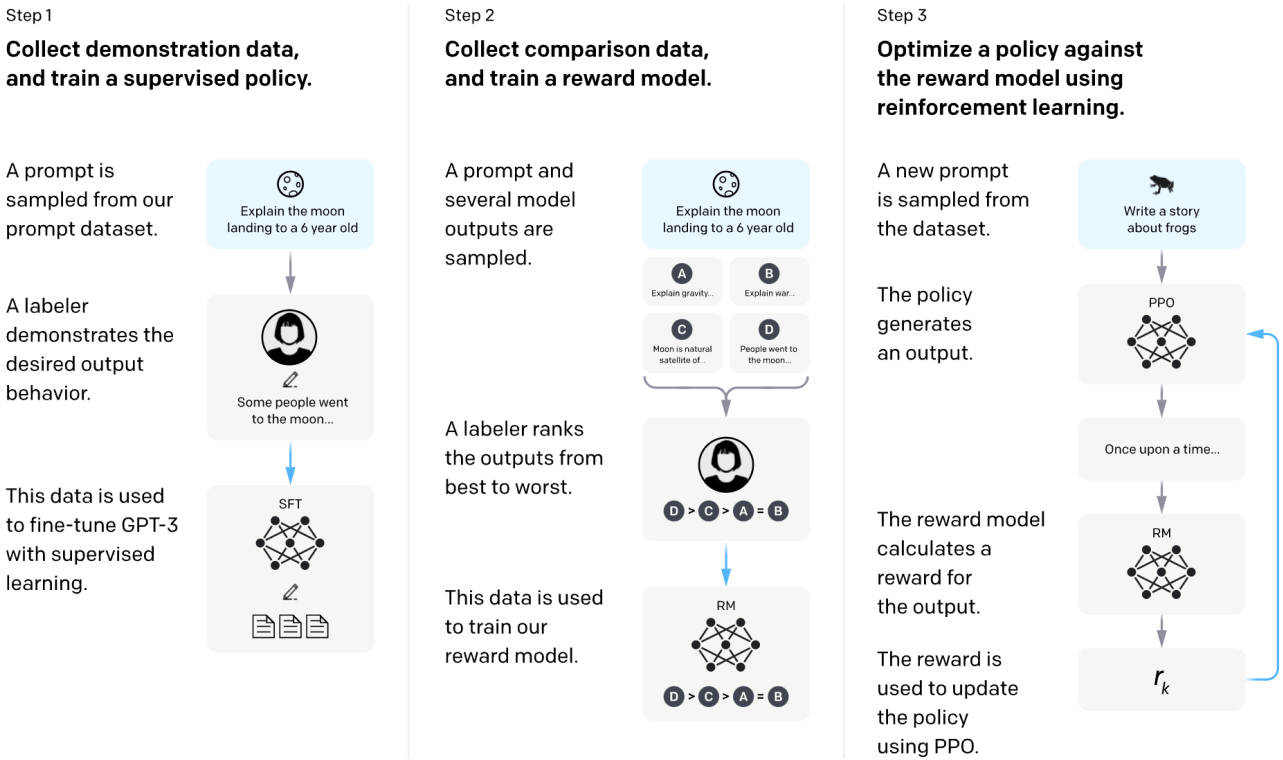
Full Fine-Tuning Potential

- Updates all parameters of the language model during training.
- Enhances adaptability to specific tasks for potentially better performance.
- Demands significant memory for processing gradients and other components.
- Challenging to implement on resource-constrained devices or environments.

YHK

Reinforcement Learning from Human Feedback (RLHF)

- **Overview:**
  - RLHF enhances language models by incorporating human feedback.
  - Aims to align models more closely with intricate human values.



(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

YHK

RLHF Process - Step 1

- **Pretraining Language Models (LMs):**
  - RLHF starts with a pretrained LM, typically achieved through classical pretraining objectives.
  - Initial LM, flexible in size, can undergo optional fine-tuning on additional data.
  - Crucial to have a model with positive response to diverse instructions.

YHK

RLHF Process - Step 2

- **Reward Model Training:**
  - Involves generating a reward model (RM) calibrated with human preferences.
  - RM assigns scalar rewards to text sequences, reflecting human preferences.
  - Dataset for training RM generated by sampling prompts, passing through initial LM, and ranking by human annotators.
  - Reward function combines preference model and a penalty on RL policy vs. initial model difference.

YHK

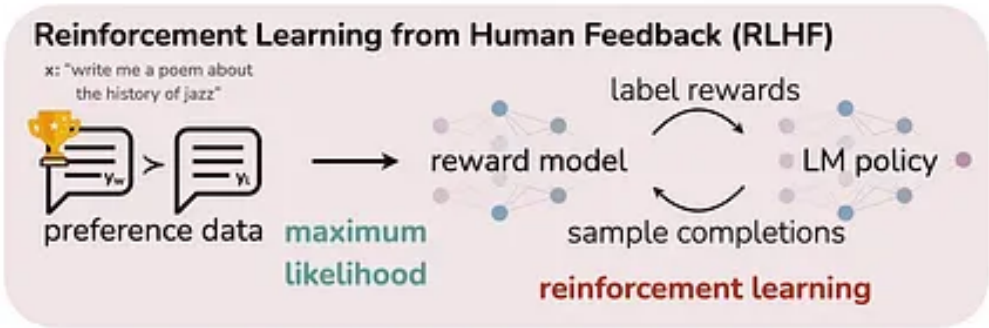
RLHF Process - Step 3

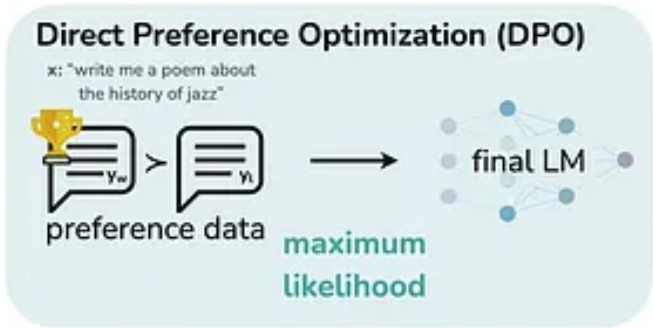
- **Fine-Tuning with RL:**
  - Final step involves fine-tuning the initial LLM using reinforcement learning.
  - Proximal Policy Optimization (PPO) commonly used for RL algorithm.
  - RL policy is LM that takes prompt, produces text, with actions corresponding to tokens in LM's vocabulary.
  - PPO updates LM's parameters to maximize reward metrics, aligning model with human preferences.
  - Some parameters frozen due to computational constraints.

YHK

Direct Preference Optimization (DPO)

- **Overview:**
  - DPO is equivalent to RLHF and gaining significant traction.
  - Offers a straightforward method for fine-tuning large language models based on human preferences.
  - Eliminates the need for a complex reward model, directly incorporating user feedback into the optimization process.
- **User-Friendly Approach:**
  - Users compare two model-generated outputs and express preferences.
  - LLM adjusts its behavior accordingly, simplifying the optimization process.
  - Advantages include ease of implementation, computational efficiency, and greater control over LLM's behavior.





(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

YHK

Comparison: DPO vs RLHF

- **DPO Approach:**
  - Directly optimizes LM based on user preferences without a separate reward model.
  - Users compare two model-generated outputs to guide the optimization process.
- **Maximum Likelihood in LLM Training:**
  - Maximum likelihood is a principle used during LLM training.
  - Involves adjusting model's parameters to maximize the likelihood of generating actual sequences observed in training data.
  - Helps LLM learn to generate text similar to examples it was trained on.
- **RLHF Approach:**
  - Follows a more structured path, leveraging reinforcement learning principles.
  - Involves training a reward model to identify and reward desirable LM outputs.
  - Reward model guides LM's training process, shaping its behavior towards positive outcomes.
- **DPO - A Simpler Approach:**
  - Direct Policy Optimization (DPO) takes a straightforward path.
  - Sidesteps the need for a complex reward model in fine-tuning Large Language Models (LLMs).
  - Optimizes LLM directly based on user preferences by comparing two outputs.

YHK

DPO Advantages

- **Ease of Implementation:**
  - DPO is more user-friendly, eliminating the need for a separate reward model.
  - Accessible to a broader audience due to its simplicity.
- **Computational Efficiency:**
  - Operates directly on the LLM, leading to faster training times.
  - Lower computational costs compared to RLHF.
- **Greater Control:**
  - Users have direct control over LLM's behavior without complexities.
  - Enables guidance toward specific goals and preferences.
- **Faster Convergence:**
  - Due to its simpler structure and direct optimization, DPO often achieves faster results.
  - Suitable for tasks with rapid iteration needs.
- **Improved Performance:**
  - Recent research suggests DPO can outperform RLHF in sentiment control and response quality.
  - Particularly effective in summarization and dialogue tasks.

YHK

RLHF - A More Structured Approach

- **More Structured Path:**
  - Reinforcement Learning from Human Feedback (RLHF) follows a more structured path.
  - Leverages reinforcement learning principles in three training phases.
- **Complexity:**
  - RLHF can be more complex and sometimes unstable.
  - Demands more computational resources and deals with convergence, drift, or uncorrelated distribution problems.
- **Flexibility in Defining Rewards:**
  - RLHF allows more nuanced reward structures, beneficial for precise control over LLM's output.
- **Handling Diverse Feedback Formats:**
  - RLHF can handle various forms of human feedback, including numerical ratings or textual corrections.
  - DPO primarily relies on binary preferences for user feedback.
- **Handling Large Datasets:**
  - RLHF can be more efficient in handling massive datasets, especially with distributed training techniques.

YHK

Summary

- **Choice Depends On:**
  - Specific task requirements.
  - Available computational resources.
  - Desired level of control over LLM's behavior.
- **Strengths and Weaknesses:**
  - Both methods offer strengths and weaknesses in different contexts.
  - Evolving and enhancing fine-tuning processes for LLMs.

YHK

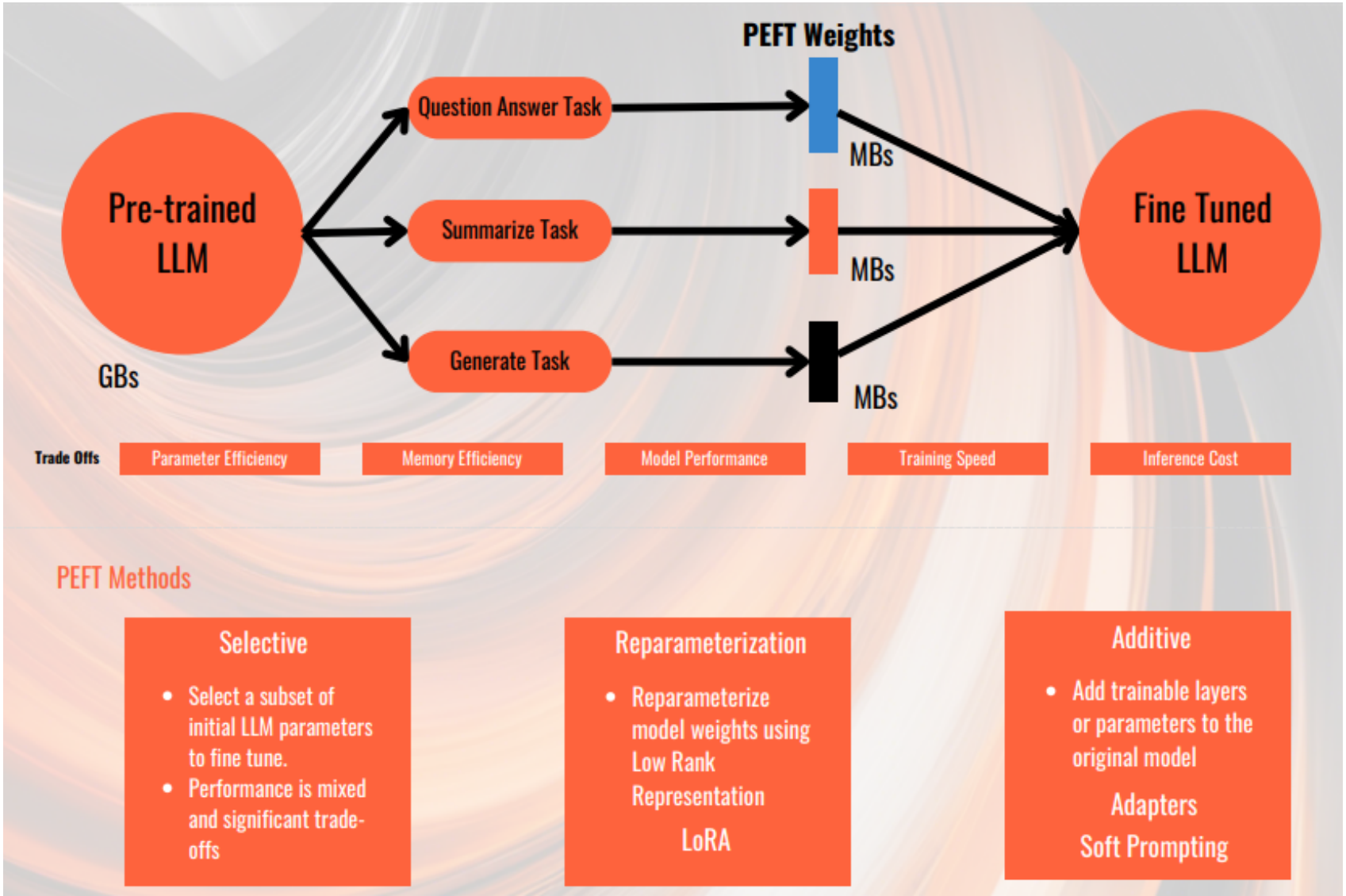
What is Parameter Efficient Fine Tuning?

- **Full Fine Tuning:**
  - Requires memory for the entire model, optimizers, gradients, etc.
  - Similar memory demands as pre-training.
- **Parameter-Efficient Fine Tuning (PEFT):**
  - Fine-tunes only a subset of model parameters.
  - In some cases, leaves the original weights untouched.

(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

YHK

What is Parameter Efficient Fine Tuning?



YHK

Parameter Efficient Fine-Tuning (PEFT)

- **Addressing Resource Intensity:**
  - PEFT addresses the resource-intensive nature of fine-tuning Large Language Models (LLMs).
  - Full fine-tuning modifies all parameters, while PEFT fine-tunes only a small subset, minimizing computational demands.

YHK

Advantages of PEFT

- **Computational Efficiency:**
  - PEFT fine-tunes LLMs with significantly fewer parameters than full fine-tuning.
  - Feasible on less powerful hardware or in resource-constrained environments.
- **Memory Efficiency:**
  - Freezing pretrained model weights minimizes excessive memory usage.
  - Suitable for tasks with memory constraints.
- **Catastrophic Forgetting Mitigation:**
  - PEFT prevents catastrophic forgetting observed in full fine-tuning.
  - Ensures retention of valuable information during adaptation to new tasks.
- **Versatility Across Modalities:**
  - PEFT is effective in various modalities such as computer vision and audio.
  - Applicable to a wide range of downstream tasks beyond natural language processing.

YHK

Advantages of PEFT (Contd.)

- **Modular Adaptation for Multiple Tasks:**
  - PEFT's modular nature allows the same pretrained model to be adapted for multiple tasks.
  - Small task-specific weights are added, avoiding the need for full copies for different applications.
- **INT8 Tuning:**
  - PEFT includes INT8 (8-bit integer) tuning, showcasing adaptability to different quantization techniques.
  - Enables fine-tuning even on platforms with limited computational resources.

YHK

Summary of PEFT

- PEFT offers a **practical and efficient solution** for fine-tuning large language models.
- Addresses **computational and memory challenges** while maintaining performance on downstream tasks.

YHK

Limitations of Fine-Tuning

- Catastrophic Forgetting: Fine-tuned models may forget some aspects of their pre-trained knowledge as they adapt to the new task.
- Computational requirements: Fine Tuning LLMs requires A100 GPU support.
- Full Fine-Tuning learning parameter dimensions is equal to the pre-trained learning parameters

YHK

Can, Cannot

Over time, LLMs will improve in:

- Better instructions following
- Large Contexts
- Simple strategies to fine tune

But, LLMs will not improve in:

- Information Extraction/retrieval: cant have real time data access, not 100% trustworthy

Need knowledge base support

(Ref: Combining LLMs with Knowledge Bases to Prevent Hallucinations - Scott Mackie - LLMs in Prod Con 2)

YHK

Fine-tuning Using Ludwig

(Ref: Efficiently Build Custom LLMs on Your Data - Piero Molino, Arnav Garg)

Introduction

- Large Language Models (LLMs) have revolutionized NLP tasks.
- Fine-tuning LLMs for specific tasks is crucial.
- Ludwig framework from Pedibase offers efficient fine-tuning.

YHK

Large Language Modeling Fine-Tuning using Ludwig

- Ludwig: Open-source AI toolbox by Uber
- Supports fine-tuning of pre-trained language models
- Supports popular models like BERT, RoBERTa, GPT-2
- Provides easy-to-use data preprocessing and model training
- Supports multi-task learning and transfer learning
- Flexible data input formats (CSV, JSON, pandas DataFrame)
- Automatic metric computation and visualizations
- Distributed training support (Horovod, Ray)
- Serialization and deployment of trained models
- Active development and community support

YHK

Setup

- Install Ludwig framework: `pip install ludwig`.
- Prepare data in required format.
- Define model architecture and parameters.

YHK

Fine-tuning Process

- Load pre-trained LLM (e.g., GPT-3) using Ludwig.
- Specify task-specific data for fine-tuning.
- Train the model with Ludwig's `train` command.

YHK

Task-Specific Tuning

- Adapt pre-trained LLM to specific tasks (e.g., text generation, classification).
- Configure task-specific parameters (e.g., learning rate, batch size).
- Fine-tune on task-specific data.

YHK

Evaluation

- Assess fine-tuned model performance using evaluation metrics.
- Validate against task-specific benchmarks.
- Iterate fine-tuning process if necessary.

YHK

Hyperparameter Optimization

- Tune hyperparameters for optimal performance.
- Ludwig supports hyperparameter search.
- Utilize techniques like grid search or random search.

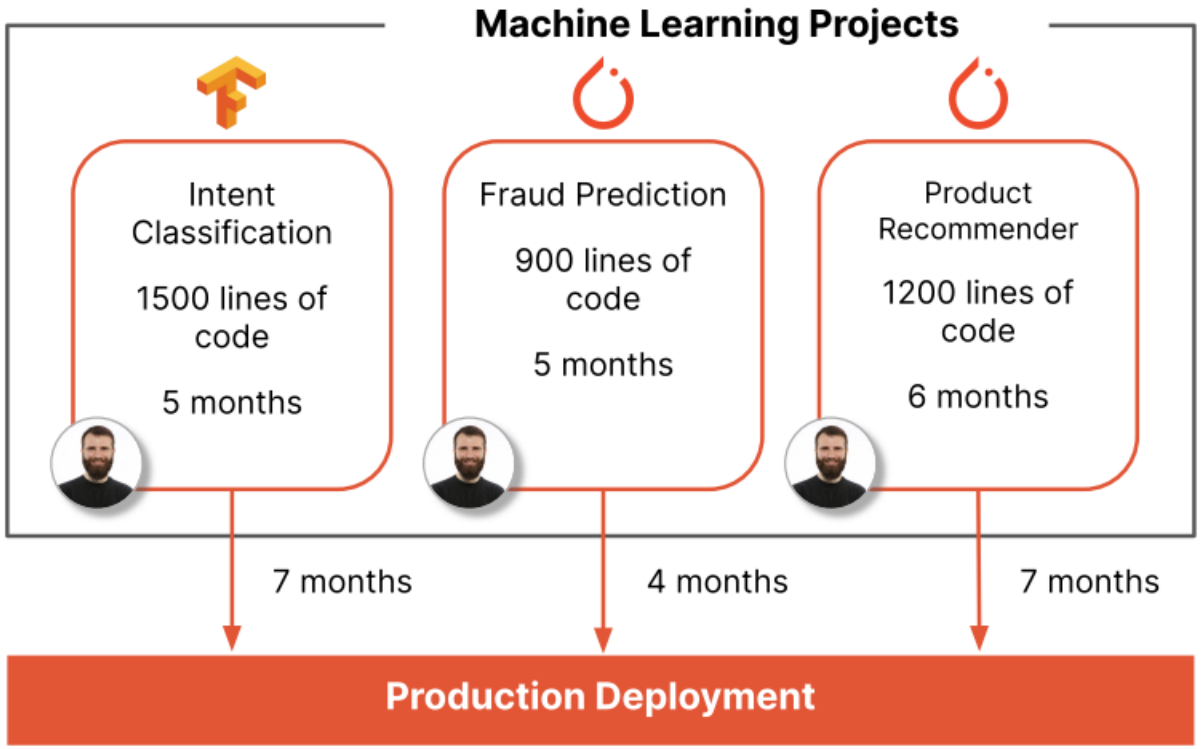
YHK

Deployment

- Deploy fine-tuned model for inference.
- Integrate with existing applications or services.
- Monitor model performance in production.

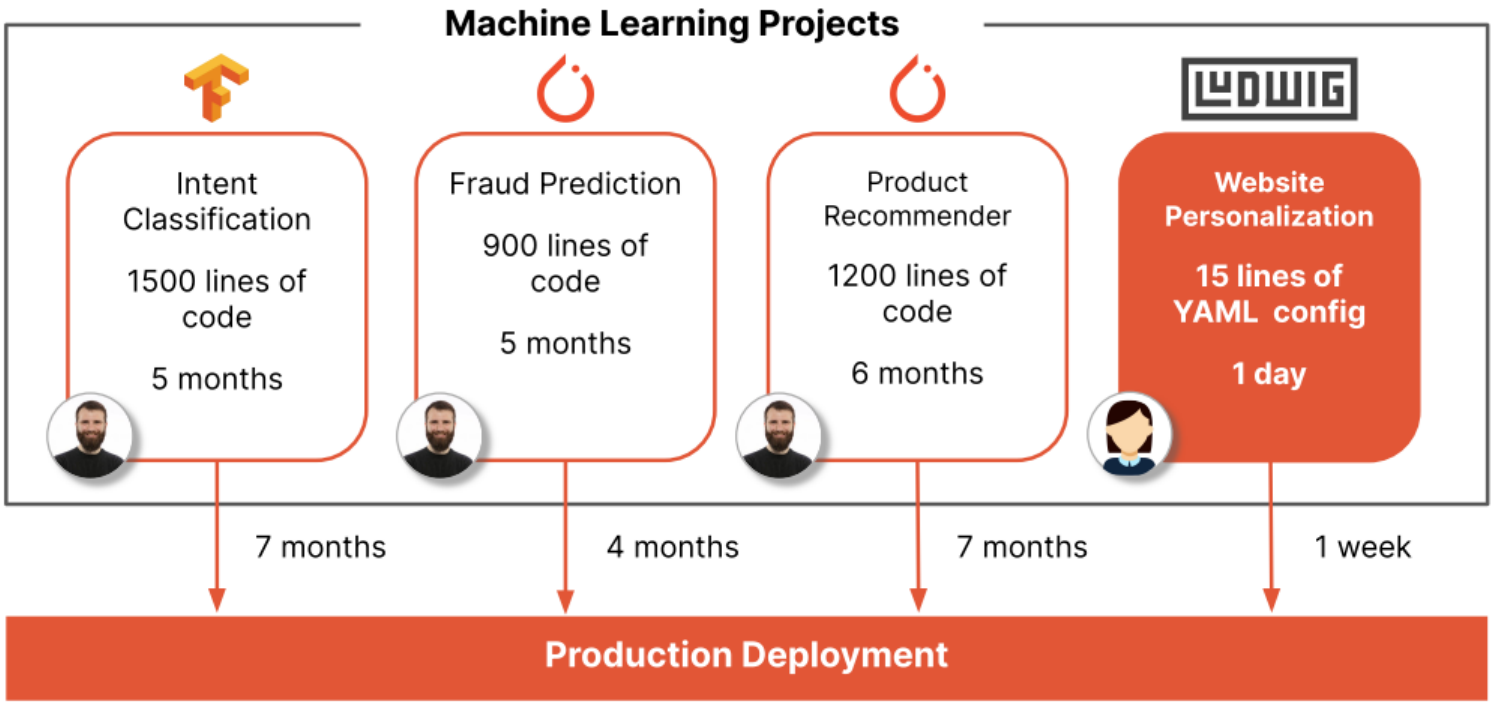
YHK

Current State of ML Projects



There must be a better way

Solution by LUDWIG



Flexibility Levels

An open-source declarative ML framework started at Uber

Easy to start

```
input_features:
  name: sentence
  type: text
output_features:
  name: intent
  type: category
```

From months to days
No ML code required
Readable & Reproducible

Expert level control

```
input_features:
  name: sentence
  type: text
  encoder: bert
output_features:
  name: intent
  type: category
trainer:
  regularize: 0.1
  dropout: 0.05
```

Easy to Iterate
Extensible

Advanced functionalities

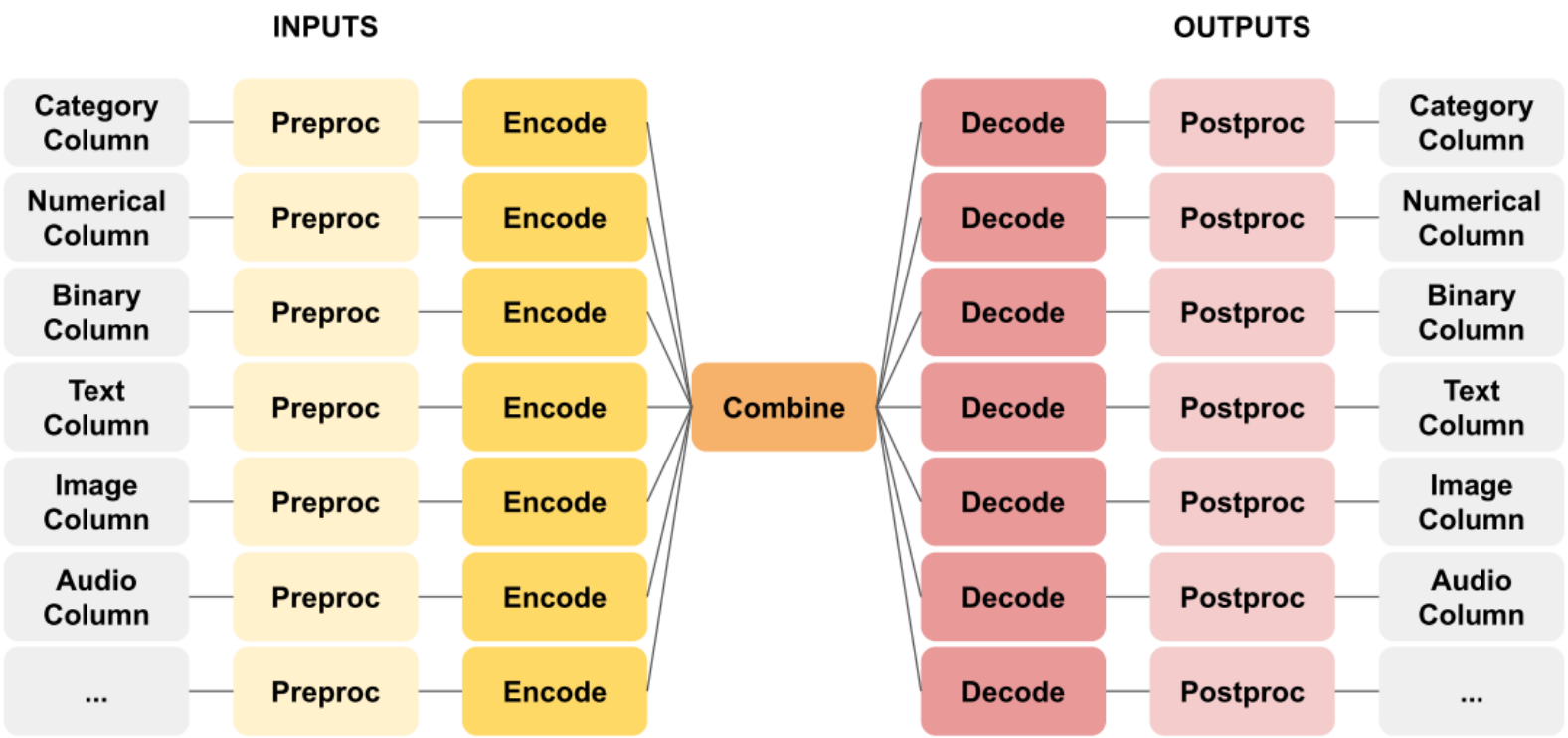
```
input_features:
  name: sentence
  type: text
output_features:
  name: intent
  type: category
hyperopt:
  dropout: [0.1, ...]
  encoder: [llama, ...]
  ...
```

Hyperparameter search
State-of-the-art models
Distributed training

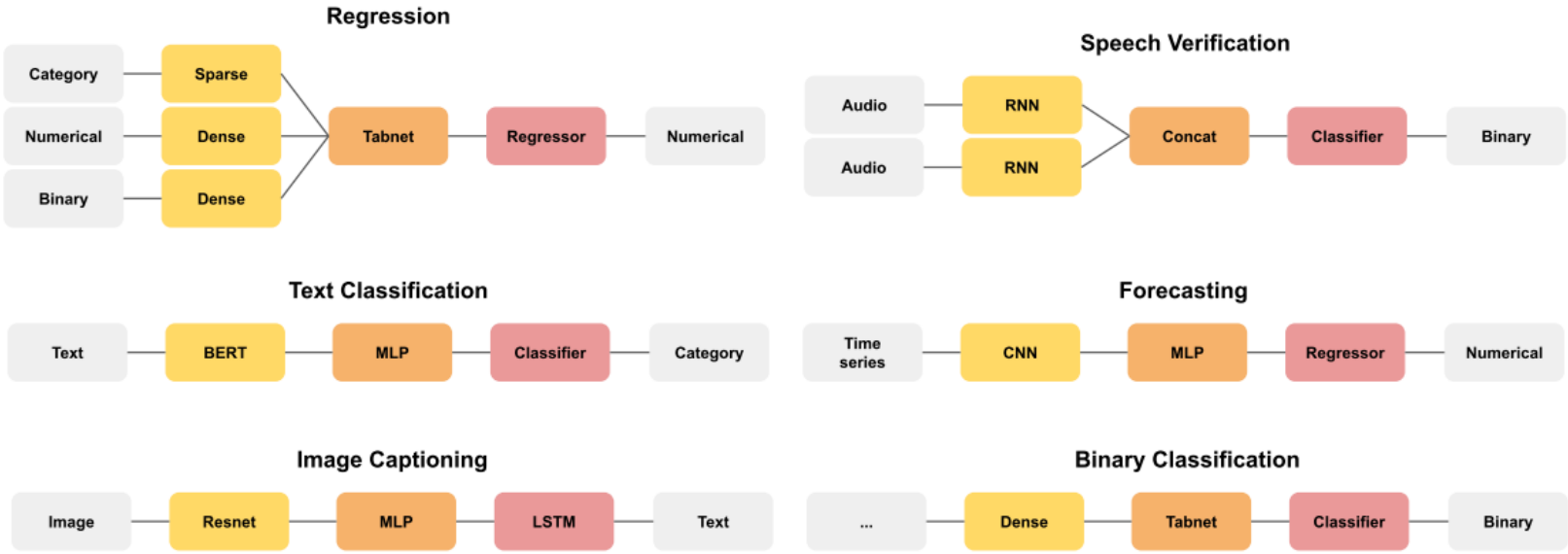
YHK

Ludwig Architecture

Many NLP tasks can be abstracted to Sequence To Sequence model



Ludwig Applications



Prompt Templating

Prompt Template Definition

```
model_type: llm
base_model: Llama-2-7b-hf
prompt:
 task: "Rate this book review with from 1 to 5"
 template: |
   Task: {task}.
   Review: “{title} {review}”.
   What score would you assign?
```

Data

title	review	score
Amazing story!	This book made me dream of …	4

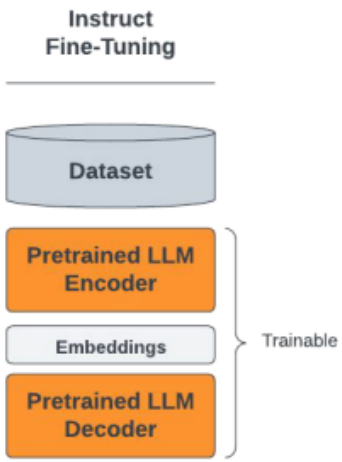
Input to LLM

```
Task: Classify this book review with a score from 1 to 5.
Review: “Amazing story! This book made me dream of …”.
What score would you assign?
```

```
llm = LudwigModel(config)
llm.create_model()
results = llm.predict(df)
```

YHK

Declaratively Fine-Tune LLMs

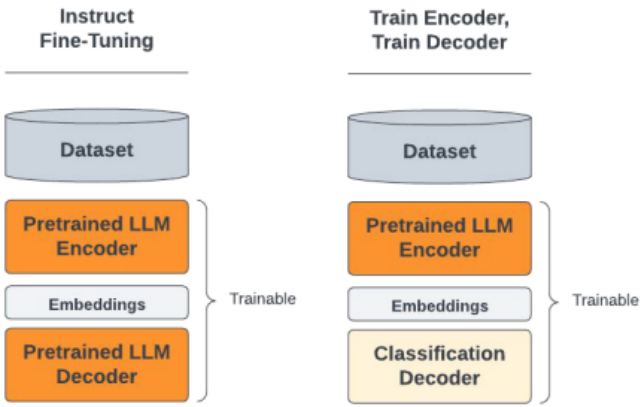



```yaml
model_type: llm
base_model: Llama-2-7b-hf

input_features:
  - name: input
    type: text

output_features:
  - name: output
    type: text

trainer:
  type: finetune
  learning_rate: 0.0003
  batch_size: 1
  gradient_accumulation_steps: 8
  epochs: 3
```

```
llm = LudwigModel(config)
results = llm.train(df)
```

YHK

Declaratively Fine-Tune LLMs

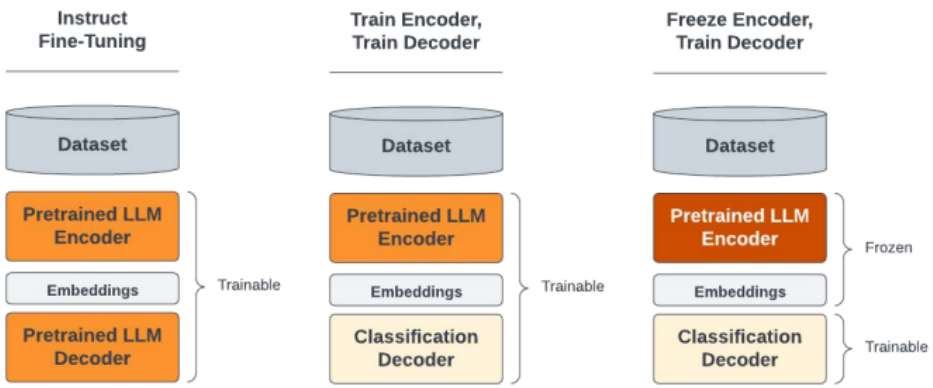



```yaml
input_features:
  - name: review
    type: text
    encoder:
      type: auto_transformer
      pretrained_model_name_or_path: Llama-2-7b-hf
      trainable: true

output_features:
  - name: sentiment
    type: category
```

```
llm = LudwigModel(config)
results = llm.train(df)
```

YHK

Declaratively Fine-Tune LLMs




```yaml
input_features:
  - name: review
    type: text
    encoder:
      type: auto_transformer
      pretrained_model_name_or_path:
Llama-2-7b-hf
      trainable: false
      preprocessing:
        cache_encoder_embeddings: true


output_features:
  - name: sentiment
    type: category
```

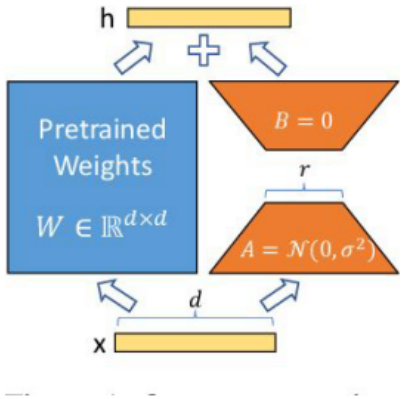
```
llm = LudwigModel(config)
results = llm.train(df)
```

YHK

Parameter efficient fine-tuning

- LoRA
- AdaLoRA
- Adaptation Prompt

  (aka, LLaMA Adapter)
- QLoRA

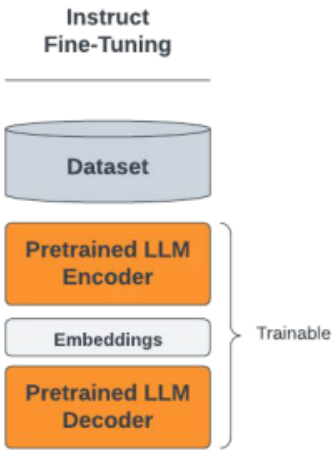



```yaml
adapter:
  type: lora
  r: 16
  alpha: 32
  dropout: 0.1
```

```yaml
adapter:
  type: lora

quantization:
  bits: 4
```

YHK

Putting it all together




```yaml
model_type: llm
base_model: Llama-2-7b-hf

adapter:
  type: lora
quantization:
  bits: 4

prompt:
  template: |
    ### Instruction:
    {instruction}

    ### Input:
    {input}

    ### Response:

input_features:
  - name: input
    type: text

output_features:
  - name: output
    type: text

trainer:
  type: finetune
  learning_rate: 0.0003
  batch_size: 1
  gradient_accumulation_steps: 8
  epochs: 3
```

```
llm = LudwigModel(config)
results = llm.train(df)
```

YHK

Example Application

Hands-on Tutorial available at:
https://medium.com/google-cloud/gemma-for-gst-4595d5f60b6b

YHK

Conclusion

- Ludwig framework simplifies fine-tuning of LLMs.
- Enables efficient adaptation to diverse tasks.
- Empowers practitioners to leverage state-of-the-art language models effectively.

YHK

References

Many publicly available resources have been refereed for making this presentation. Some of the notable ones are:

- When to Fine-Tuning Large Language Models? - Thierry Teisseire
- Fine-tuning Large Language Models - Deeplearning ai https://learn.deeplearning.ai/courses/finetuning-large-language-models/lesson/1/introduction
- Fine-tuning LLMs with PEFT and LoRA https://youtu.be/Us5ZFp16PaU
- Adapters: the game changer for fine-tuning - Geoffrey Angus https://youtu.be/BQYD9HivFLY
- Fine-tuning Large Language Models (LLMs) — w/ Example Code - Shaw Talebi https://www.youtube.com/watch?v=eC6Hd1hFvos
- LLM Crash Course Part 1 - Finetune Any LLM for your Custom Usecase End to End in under[1 hour]!! - Neural Hacks with Vasanth. https://www.youtube.com/watch?v=whbuNo6APVs , https://github.com/Vasanthengineer4949/NLP-Projects-NHV/LLMs Related/LLM Crash Course/Finetune_any_LLM_crash_course.ipynb

YHK

Thanks . . .

- Search **"Yogesh Haribhau Kulkarni"** on Google and follow me on LinkedIn and Medium
- Office Hours: Saturdays, 2 to 5pm (IST); Free-Open to all; email for appointment.
- Email: yogeshkulkarni at yahoo dot com

(Generated by Hugging Face QR-code-AI-art-generator, with prompt as "Follow me")

YHK