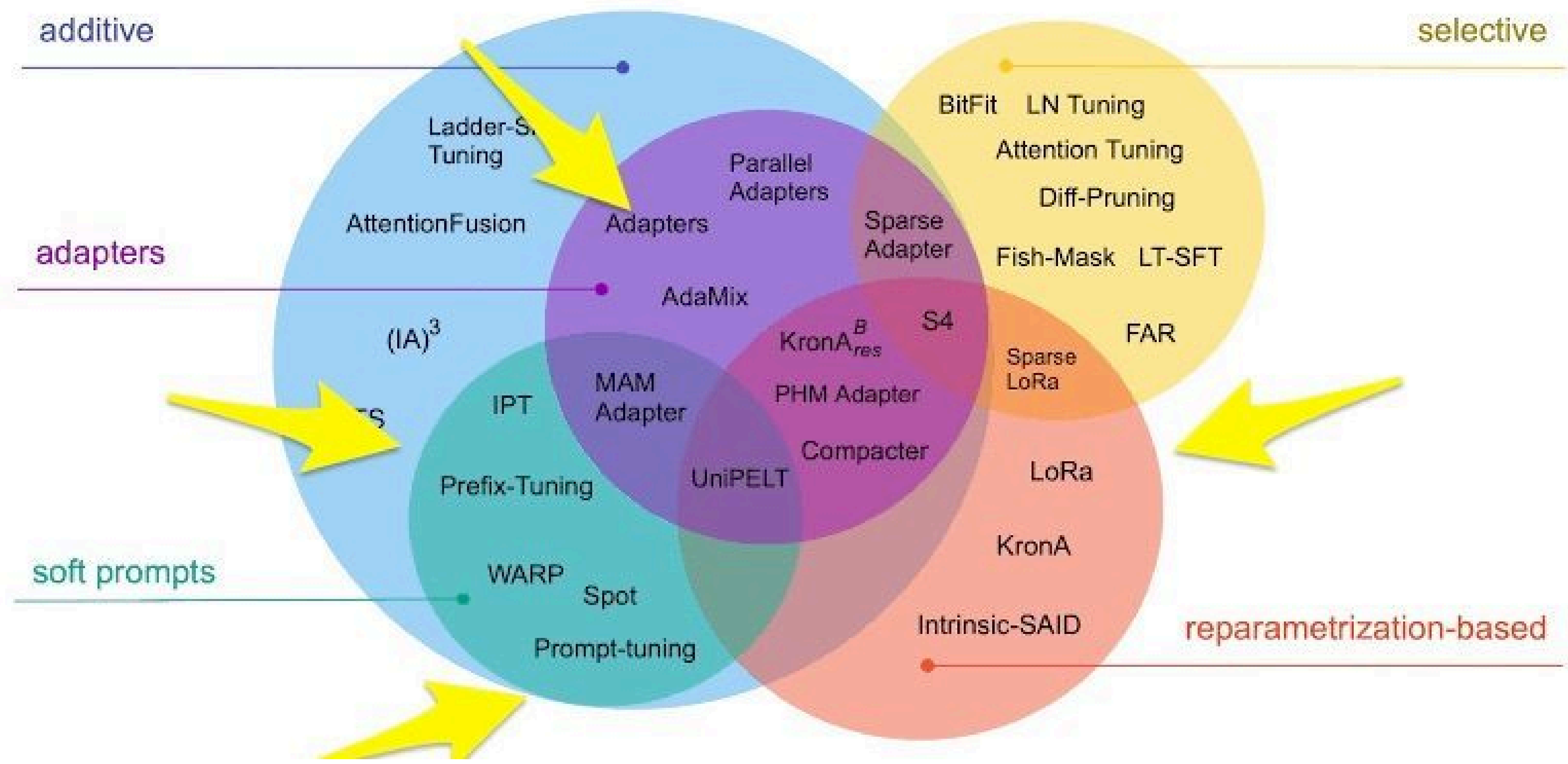
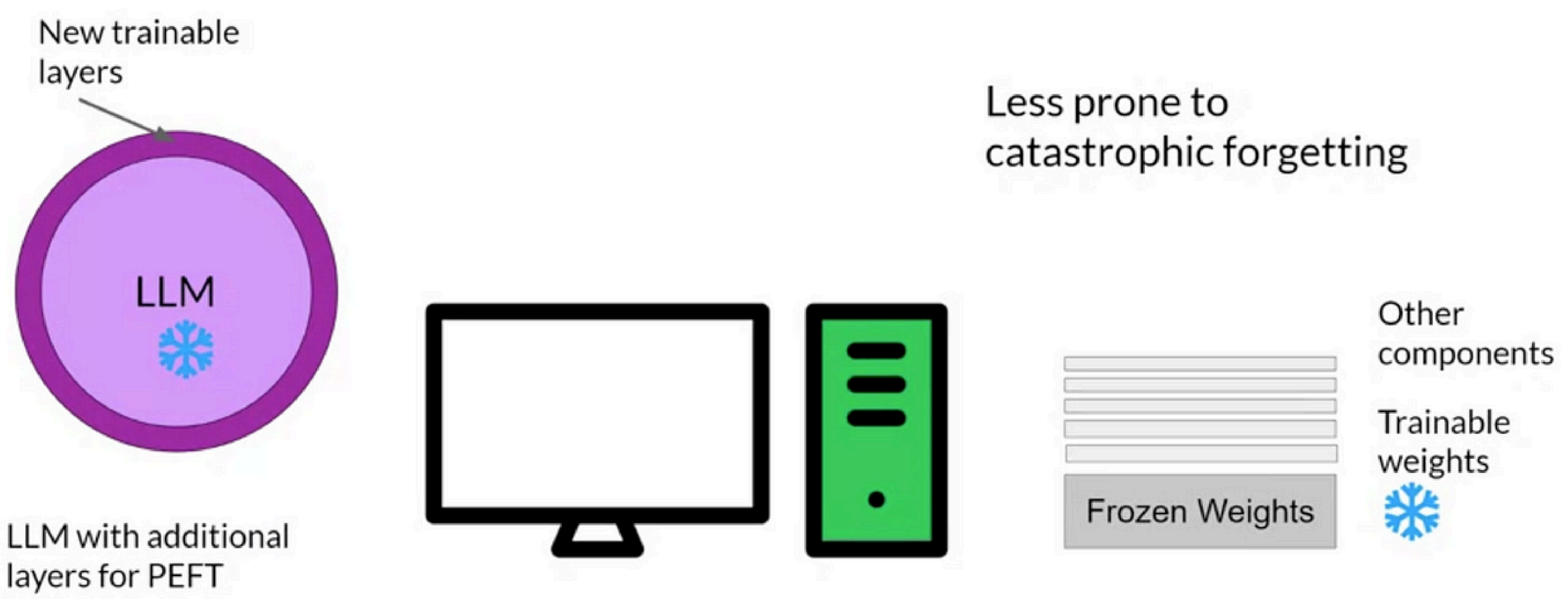
Mastering LLMs

Analytics Vidhya

Day 24: Parameter Efficient Fine-Tuning (PEFT)





What is PEFT?

Parameter Efficient Fine-Tuning (PEFT) is a technique designed to fine-tune large pre-trained models by modifying only a small subset of parameters. Instead of updating the entire model, PEFT methods adapt specific components to achieve similar or even better task performance with significantly lower computational costs.

Why PEFT?

Fine-tuning large models like BERT, GPT, and Vision Transformers can be resource-intensive due to the vast number of trainable parameters. PEFT addresses these challenges by:

- Reducing memory and processing requirements.
- Enabling faster adaptation to new tasks.
- Improving generalization by fine-tuning fewer parameters.
- Supporting efficient multi-task learning with minimal additional storage.

How it Works?

PEFT works by introducing small trainable components into the frozen pre-trained model. Several techniques are commonly used:

- **Adapters**: Small neural layers inserted between transformer layers that learn task-specific features while keeping the main model fixed.
- **LoRA** (Low-Rank Adaptation): Introduces low-rank decomposition of weight updates to reduce trainable parameters.
- **Prefix Tuning**: Adds learnable prefix tokens to model inputs, allowing efficient tuning without modifying core model weights.
- **Prompt Tuning**: Optimizes input prompt embeddings rather than modifying model weights.
- **BitFit**: Fine-tunes only the bias terms of the model to achieve efficiency.

Advantages of PEFT

- **Computational Efficiency**: Reduces training time and hardware requirements.
- **Lower Storage Costs**: Since only a few parameters are updated, storing multiple task-specific versions becomes feasible.
- **Faster Deployment**: PEFT methods enable rapid adaptation to new tasks without re-training entire models.
- **Better Generalization**: Limits overfitting by constraining parameter updates.

Disadvantages of PEFT

- **Limited Capacity**: Since only a subset of parameters are fine-tuned, performance may be slightly lower than full fine-tuning in some cases.
- **Task Dependency**: Some PEFT methods might not generalize well across all task types.
- **Implementation Complexity**: Requires additional architectural modifications to incorporate adapter modules or prompt embeddings.

```python
from transformers import AutoModelForCausalLM, AutoTokenizer
from peft import LoraConfig, get_peft_model

# Load base model and tokenizer
model_name = "meta-llama/Llama-2-7b"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Define LoRA configuration
lora_config = LoraConfig(
    r=8,  # Rank of LoRA matrices
    lora_alpha=32,  # Scaling factor
    lora_dropout=0.1,  # Dropout probability
    target_modules=["q_proj", "v_proj"]  # Target transformer layers
)

# Apply PEFT
peft_model = get_peft_model(model, lora_config)

# Print model summary
peft_model.print_trainable_parameters()
```

Analytics Vidhya

Stay Tuned for **Day 25** of

Mastering LLMs