Knowledge Injection via XAI: A Spark-based Framework

Distributed framework for Vision Transformer adapter selection using explainability metrics

Author: Domenico Lacavalla

---

1 Introduction and Scientific Context

1.1 Scenario: The Dominance of Vision Transformers

The Transformer architecture, originating in NLP, has revolutionized Computer Vision as well. Models such as **ViT** [1] and self-supervised approaches like **DINOv2** [2] offer feature extraction capabilities superior to traditional CNNs. However, the use of these *Foundation Models* in real-world or sensitive contexts presents three fundamental critical issues:

1. **Adaptation cost:** Full fine-tuning of models with hundreds of millions of parameters is computationally expensive and risks *catastrophic forgetting*.
2. **Poor interpretability:** The self-attention mechanism, while powerful, does not guarantee reliable explanations for why a decision was made.
3. **Validation difficulty:** Evaluating semantic robustness requires analyzing the model's behavior on large datasets, an operation often prohibitive on a single machine.

1.2 The Solution: Big Data for Model Selection

The project proposes a Big Data approach based on **Apache Spark** to horizontally scale the calculation of Explainable AI (XAI) metrics. The central idea is to use explainability not just to "look" at the model, but as a quantitative metric to select the best one. The pipeline includes:

- The use of **PEFT** (Parameter-Efficient Fine-Tuning) techniques, specifically **LoRA** [3], to rapidly create diverse lightweight model variants.
- Distribution of inference and XAI metric calculation (computationally intensive) on a Spark cluster.
- Training a meta-model that, based on explainability profiles, predicts which configurations are most robust on *Out-of-Distribution* (OOD) data.

1.3 Objectives

The case study aims to address two challenges:

1. **Scalability:** Demonstrate how the integration between PyTorch and Spark [4] can drastically reduce calculation times for complex XAI metrics (e.g., Deletion Score).
2. **Meta-Learning:** Verify the hypothesis that metrics such as attention entropy or heatmap faithfulness are reliable predictors of model robustness.

2 System Architecture and Distributed Pipeline

The architecture follows the **Medallion Architecture** pattern (Bronze–Silver–Gold), adapted to manage the model evaluation lifecycle.

2.1 Data Flow

1. Bronze Layer: Ingestion and Pre-computation.
This level manages raw image ingestion. Since the DINOv2 backbone (ViT-B/14) is used in *frozen* mode, the base features do not change between experiments.

- **Process:** Image loading in Spark, resize/normalization, and extraction of the *CLS token* and *Patch Embeddings*.
- **Output:** Optimized Parquet files containing serialized tensors. This avoids recalculating the backbone pass for every LoRA configuration.

2. Silver Layer: Distributed XAI Extraction.
This is the core of the system. Here, LoRA adapters are dynamically applied, and XAI metrics are calculated.

- **Input:** Join between embeddings (Bronze) and the **Adapter Zoo** (set of LoRA configurations).
- **Distributed Computation:** Use of **Pandas UDF (User Defined Functions)** to vectorize operations. Each Spark worker receives a batch of images and adapter weights, performs inference, and calculates explainability metrics.
- **Output:** Analytical table with metrics per single image (e.g., *Attention Entropy*, *Deletion Score*).

3. Gold Layer: Meta-Dataset.
Point metrics are aggregated by model (calculating mean, variance, percentiles) to create the final dataset used for training the Meta-Learner.

2.2 Enabling Technologies and Optimizations

- **Apache Arrow:** Used to minimize serialization/deserialization overhead between the Spark JVM and the Python/PyTorch environment.
- **GPU Resource Management:** Use of *Scalar Iterator UDF* to load the model into VRAM only once per partition (instead of for every record), drastically reducing I/O times.
- **Broadcast Variables:** LoRA adapter weights (a few MBs compared to the GBs of the base model) are sent to worker nodes via broadcast to reduce network traffic.

3 Methodology: Explanatory Feature Extraction

The system transforms the "visual intuitions" of heatmaps into numerical vectors $\mathbf{v}_{xai}$ usable by machine learning algorithms.

3.1 Attention Map Analysis

From the Vision Transformer, we extract the attention matrix A of the last layer, focusing on how the classification token (`[CLS]`) interacts with image patches. By averaging over the different transformer heads, we obtain a heatmap H indicating the importance of each image area.

3.2 Metric 1: Attention Entropy (Focus)

We calculate Shannon entropy on the attention distribution H:

$$S(H) = -\sum_{i=1}^{n} H_i \log_2(H_i)$$

High entropy values indicate the model "looks everywhere" (diffused/confused attention), while low values indicate a precise focus on specific areas.

3.3 Metric 2: Deletion Score (Faithfulness)

This metric, inspired by RISE [5], measures how much the areas deemed important by the model are *actually* necessary for prediction. The process, distributed on Spark, proceeds as follows:

1. Image patches are sorted based on importance defined by the heatmap H.
2. Perturbed versions of the image are generated by progressively removing the most important patches.
3. Inference is re-run on perturbed images, and the drop in probability of the correct class (AUC) is measured.

A robust model should show a rapid drop in confidence when key features are removed (low Deletion Score). If confidence remains high even after removing the main object, the model is likely exploiting background bias.

4 Meta-Learning and Experimental Validation

The final goal is to estimate a model's robustness without testing it on infinite external datasets, but by analyzing its internal properties.

4.1 Problem Definition

Let $\mathcal{M} = \{M_1, \ldots, M_K\}$ be the set of trained LoRA adapters. Each model M_k is characterized by an aggregated feature vector $\mathbf{x}_k$ derived from the Silver Layer (e.g., mean entropy, mean deletion score). We want to train a regressor f such that:

$$\hat{y}_k^{OOD} = f(\mathbf{x}_k)$$

where $\hat{y}_k^{OOD}$ is the estimated performance on an *Out-of-Distribution* dataset.

4.2 Experimental Setup

- **Source Dataset (Domain A):** Oxford-IIIT Pet (standard images).
- **Target Dataset (Domain B - OOD):** Generated by applying synthetic corruptions (Gaussian noise, blur, contrast changes) to the original dataset.
- **Meta-Model:** Since the number of adapters to evaluate is in the order of tens/hundreds, we use a **Random Forest Regressor**, capable of handling tabular data with heterogeneous features and providing feature importance.

4.3 Expected Results

1. **Computational Efficiency:** The Spark implementation is expected to scale linearly with the number of images, making the calculation of Deletion Score (which requires N inferences per image) feasible on large datasets.
2. **XAI-Robustness Correlation:** We expect an inverse correlation between Deletion Score and OOD Accuracy. A model that correctly focuses on the object (low Deletion Score, low Entropy) should degrade less when the background or image style changes (see Table 1).

Adapter	**Test Accuracy**	**Deletion Score**	**OOD Accuracy (Target)**
LoRA (r=4)	92.5%	**0.15 (Low)**	**88.1% (High)**
LoRA (r=32)	**93.1%**	0.42 (High)	76.4% (Low)

Table 1: Hypothetical scenario: the r=32 model overfits (high accuracy, but high deletion score), while r=4 generalizes better on OOD data.

5 Conclusions

The project demonstrates how the integration of Big Data technologies (Spark) and advanced Deep Learning (Transformers, LoRA) allows for the automation of complex evaluation processes. The proposed framework does not just select the most accurate model, but the most "aware" one, introducing quantitative explainability metrics into large-scale MLOps pipelines.

References

[1] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[2] M. Oquab et al., "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[3] E. J. Hu et al., "Lora: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2021.

[4] B. Chambers and M. Zaharia, *Spark: The definitive guide: Big data processing made simple*. O'Reilly Media, 2018.

[5] V. Petsiuk, A. Das, and K. Saenko, "Rise: Randomized input sampling for explanation of black-box models," *arXiv preprint arXiv:1806.07421*, 2018.