# Few-Shot Learning: Training with Limited Data


Presented by

Rishabh Jain

M.Tech. 3rd Semester

Reg. No: 2023DS18

Computer Science & Engineering Department,

Motilal Nehru National Institute of Technology

Allahabad, Prayagraj


December 21, 2024

# Table of Contents

# Why Do We Need Zero-Shot Learning?

- **Handling Unseen Classes:** Zero-shot learning allows models to classify classes that were never seen during training.
- **Data Scarcity:** Useful in scenarios where labeled data for new classes is unavailable or costly.
- **Semantic Understanding:** Leverages semantic relationships between known and unknown classes to classify unseen data.
- **Generalization:** Helps models generalize better, enabling them to work in real-world settings with evolving data.

# Zero-Shot Learning Overview

- **Definition:** Zero-shot learning (ZSL) allows models to classify classes that have not been seen during training.
- **Core Idea:** Leverages pre-trained models and semantic embeddings to understand and classify unseen classes.
- **Application:** Used in situations where it is impractical to train a model on every possible class.

# N-Way, K-Shot Learning in Zero-Shot Learning

- **N-Way Classification:** Refers to the number of candidate classes in a classification task. For example, in a 3-way classification, there are 3 classes: "panda," "bear," and "cat."
- **K-Shot Learning:** Refers to the number of examples available for each class during training. In a few-shot setting, the model is given only a few examples for each class.
- **Example Scenario:**
  - **3-way, 0-shot:** The model is given 3 classes (panda, bear, cat) but has never seen "panda" during training.
  - The model must classify an image of a panda into one of the 3 classes based on semantic embeddings, even without specific examples of "panda."

# Problem Setup

- **Given Classes:** We have three classes: **panda**, **cat**, and **bear**.
- **Training Data:** The model has been trained on classes such as **cat**, **dog**, and **car**, but not on **panda**.
- **Task:** The goal is to classify an image of a **panda** into one of these three classes.

# Pre-trained Embeddings

- **Embeddings:** The model uses vector representations for known classes like **cat** and **bear**.
- **Vector Representation:** These embeddings capture the semantic meaning of each class.
- **Example Embeddings:**

$$f(\text{cat}) = [0.3, 0.5, 0.1], \quad f(\text{bear}) = [0.6, 0.2, 0.7]$$

# Semantic Embedding for Panda

- **Semantic Embedding:** Since **panda** is not part of the training data, we generate its embedding using a description.
- **Text Embedding Model:** Natural language models like BERT or GloVe are used to generate the embedding from a description.
- **Example Description:** "Panda is a bear-like mammal with black-and-white fur."
- **Embedding for Panda:**

$$f(\text{panda}) = [0.7, 0.4, 0.5]$$

# Image Embedding through CNN

- **Input Image:** The image of a **panda** is processed using a pre-trained Convolutional Neural Network (CNN).
- **Feature Extraction:** The CNN extracts relevant features (color, shape, texture) to create an image embedding.
- **Example Image Embedding:**

$$f(x) = [0.68, 0.38, 0.52]$$

# CNN Model Architecture Flow

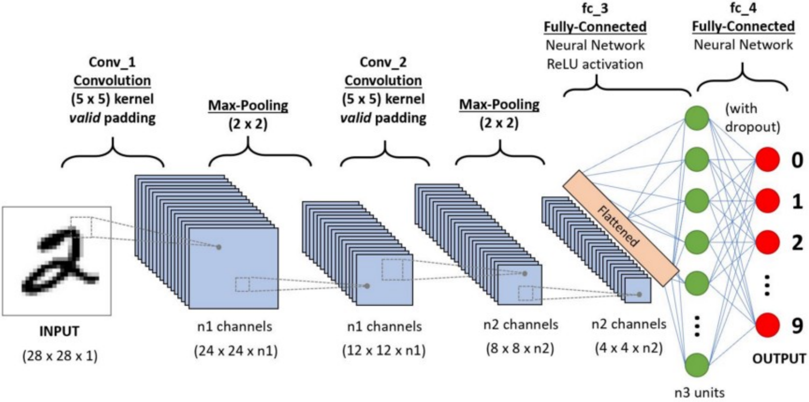



Figure: Flow diagram of the CNN Model Architecture

# Cosine Similarity Calculation

- **Cosine Similarity:** A similarity measure to compare embeddings. It calculates how similar two vectors are by measuring their cosine angle.
- **Formula:**

$$\text{Cosine Similarity}(a, b) = \frac{a \cdot b}{\|a\|\|b\|}$$

  where $a$ and $b$ are two vectors, $\cdot$ is the dot product, and $\| \cdot \|$ is the norm of the vector.

# Calculating Similarities

- **Cosine Similarity Scores:** The cosine similarity is computed between the image embedding $f(x)$ and the class embeddings.
- **Example Similarities:**

$$\text{Similarity}(f(x), f(\text{panda})) = 0.95$$

$$\text{Similarity}(f(x), f(\text{bear})) = 0.85$$

$$\text{Similarity}(f(x), f(\text{cat})) = 0.50$$

# Prediction

- **Highest Similarity:** The class with the highest similarity score is selected.
- **Prediction:** In this case, the highest similarity is with **panda**.
- **Output:** The model predicts that the image belongs to the class **panda**.

# Final Output

- **Final Prediction:** The model classifies the image as a **panda**, despite having never seen this class during training.
- **Key Achievement:** The model leverages zero-shot learning to identify novel classes using semantic relationships.

# Zero-Shot Learning Summary

- **Key Steps:**
  - Generate embeddings for both seen and unseen classes.
  - Generate an embedding for the input image.
  - Compute the cosine similarity between the image embedding and class embeddings.
  - Classify the image based on the highest similarity score.
- **Conclusion:** Zero-shot learning enables models to classify unseen classes based on semantic embeddings.

## Technologies Used

- **Pre-trained CNN models:** Feature extraction models such as ResNet, VGG.
- **Text-based Embeddings:** BERT, GloVe for generating semantic class embeddings.
- **Cosine Similarity:** Used to compare and rank embeddings.
- **Zero-shot Learning:** A framework that generalizes across unseen classes.

# Why Do We Need Few-Shot Learning?

- **Data Scarcity:** Labeled data is expensive or rare in many real-world scenarios (e.g., medical imaging, niche languages). Few-shot learning helps overcome this scarcity by using just a few examples to train the model.
- **Rapid Adaptation:** Few-shot learning allows models to adapt quickly to new classes or tasks with minimal data, unlike traditional models that require large datasets.
- **Resource Efficiency:** Saves time, cost, and effort in data collection and labeling by learning from a small number of labeled examples.
- **Human-Like Learning:** Mimics human learning by enabling models to generalize and learn new concepts from just a few examples, similar to how humans learn from limited exposure.
- **Contrast with Zero-Shot Learning:** Unlike zero-shot learning, where no examples are available, few-shot learning allows for learning from a small set of labeled examples, making it useful when some data is available but not enough for full training.

# Why is Few-Shot Learning Important?

- **Practicality:** Essential in fields with limited data availability.
  - **Medical Field:** Identifying rare diseases.
  - **Industry:** Adapting models for specific products.
- **Scalability:** Enables rapid deployment of models across domains without retraining.
- **Cost-Effective:** Reduces dependency on large datasets and labeling efforts.

# Background and Evolution

- **Traditional Learning Paradigm:**
    - Relies heavily on large datasets.
    - Struggles with generalization from limited data.
- **Transfer Learning:**
    - Uses pre-trained models fine-tuned on specific tasks.
    - Addresses data scarcity but requires some labeled data.
- **Meta-Learning:**
    - "Learning to Learn" across tasks.
    - Enables quick adaptation to unseen tasks.

# What is Few-Shot Learning?

- A subset of machine learning aimed at training models with minimal labeled data for specific tasks.
- **Example:** Classifying objects with only a few images or examples per class.

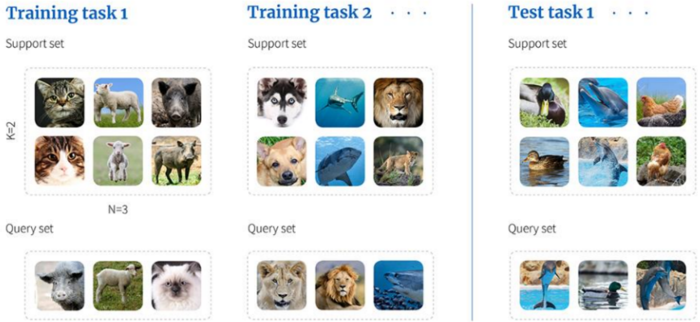


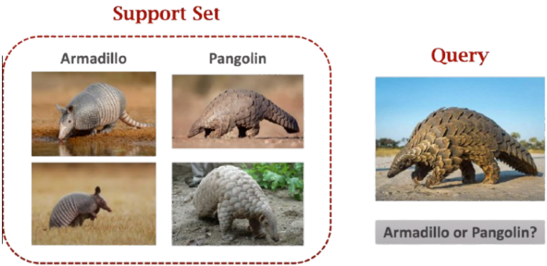

Figure: Few-Shot Learning

# Core Concepts and Terminology

- **Few-Shot Learning:** Learning from 1–5 labeled examples per class.
- **N-Shot, K-Way:** Tasks involve $N$ examples per $K$ classes.
- **Meta-Learning:** Training models to generalize across tasks.
- **Support Set & Query Set:** Key components in episodic training.

# Support Set and Query Set in Few-Shot Learning

- **Support Set:**
  - A small set of labeled examples provided for each class during training.
  - Helps the model learn about the different classes.
  - Contains $N$ examples per class (e.g., 5 labeled images per class).
- **Query Set:**
  - An unlabeled set of examples the model needs to classify.
  - Evaluates how well the model generalizes from the support set.
  - Contains examples that are unseen during training.



# How Does Few-Shot Learning Work?

Few-Shot Learning uses special techniques to generalize from a few examples:

- **Meta-Learning ("Learning to Learn"):**
  - Pre-trains a model on diverse tasks to develop general skills.
  - Adapts quickly to new tasks using a few examples.
  - **Example:** Learning to classify animals and adapting to a new bird species with minimal examples.
- **Feature Reuse:** Leverages pre-trained networks (e.g., ResNet, BERT) to extract useful features.
- **Metric-Based Learning:**
  - Measures similarity between examples (e.g., Siamese Networks, Prototypical Networks).
  - **Example:** Comparing features to classify whether two images show the same object.
- **Data Augmentation:** Expands the dataset by transformations (e.g., rotating, flipping images).

# Introduction to Meta-Learning

**Meta-Learning: Learning to Learn**

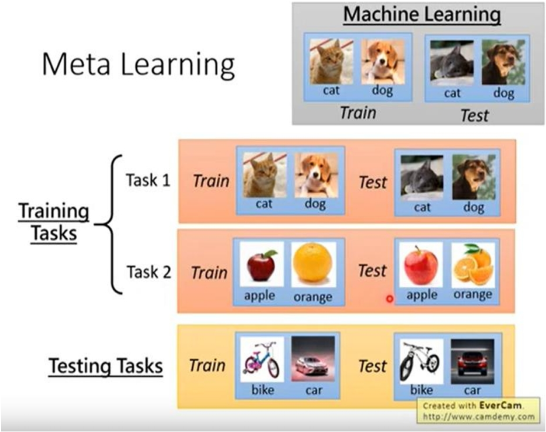
- Meta-learning is a machine learning paradigm where a model is trained to adapt quickly to new tasks using limited data.
- Focuses on learning general strategies or initialization states that can be reused across various tasks.

**Key Levels:**

- **Task-Specific Learning:** Adapts to individual tasks with minimal data.
- **Meta-Level Learning:** Learns how to adapt effectively across multiple tasks.

**Applications:**

- Few-shot learning, Transfer learning, Robotics and reinforcement learning.



## Approaches in Few-Shot Learning

- **Metric Learning:**
  - Learns a distance metric to compare examples.
  - **Examples:** Siamese Networks, Triplet Loss, Prototypical Networks.
- **Gradient-Based Meta-Learning:**
  - Optimizes the model's ability to quickly adapt to new tasks.
  - **Example:** MAML (Model-Agnostic Meta-Learning).
- **Self-Supervised Learning:** Learns representations from unlabeled data for FSL tasks.

# Meta-Learning Approaches: Model-Agnostic Meta-Learning (MAML)

**Goal:** Train a model on a variety of tasks such that it can adapt quickly to new tasks with minimal data.

**Key Features:**

- Task-agnostic initialization for neural networks.
- Uses gradient descent for fast adaptation.

**Training Process:**

- **Inner Loop:** Learn task-specific parameters.
- **Outer Loop:** Optimize initialization using meta-objective.

**Applications:**

- Few-shot learning.
- Reinforcement learning.

# Meta-Learning Approaches: Reptile Algorithm

**Overview:** A simpler, first-order approximation of MAML.

**Key Features:**

- Does not require second-order gradients.
- Works by moving weights closer to those effective for sampled tasks.

**Advantages:**

- Computationally efficient compared to MAML.
- Easy to implement.

**Applications:**

- Similar to MAML, including robotics and image classification.

# Embedding-Based Methods: Prototypical Networks

**Core Idea:** Learn a metric space where classification is performed by computing distances to prototype representations of each class.

**Key Features:**

- **Prototype:** Mean embedding of all examples in a class.
- Works well with Euclidean distance or cosine similarity.

**Advantages:**

- Simple yet effective for few-shot tasks.
- Requires fewer parameters than traditional models.

**Applications:**

- Few-shot image recognition.
- Natural language processing.

# Prototypical Networks

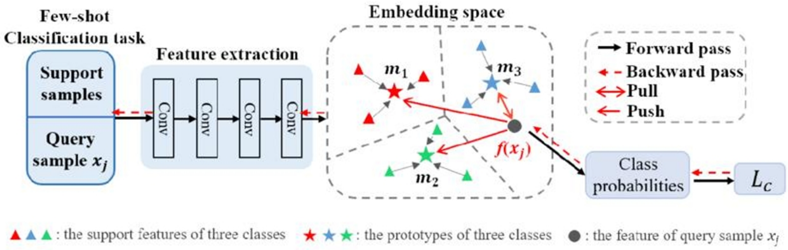



Figure: Prototypical Networks

# Embedding-Based Methods: Siamese Networks

**Core Idea:** Compare pairs of inputs to decide if they belong to the same class.

**Key Features:**

- Twin networks share parameters for generating embeddings.
- Uses contrastive or triplet loss for training.

**Advantages:**

- Effective for verification tasks.
- Handles imbalance by focusing on pairs rather than classes.

**Applications:**

- Signature verification.
- Face recognition.
- Text similarity.

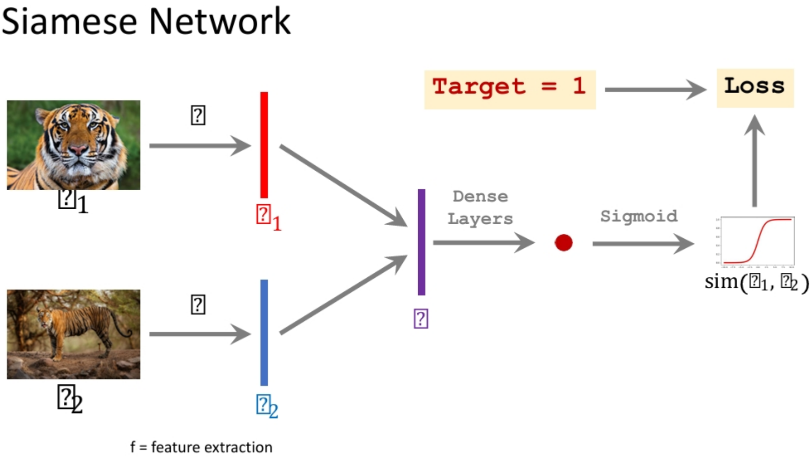



Figure: Siamese Networks

# Benefits of Few-Shot Learning

**Key Benefits:**

- **Reduced Dependency on Large Datasets:**
  - Few-shot learning allows models to generalize well even with limited labeled data, reducing the need for large annotated datasets.
- **Cost-Effective and Adaptable:**
  - Reduces the cost and effort involved in labeling large datasets, making it more adaptable to new tasks with minimal additional data.
- **Generalization to Unseen Scenarios:**
  - Enables models to generalize to new, unseen tasks with minimal additional data, improving real-world adaptability.
- **Effective in Resource-Constrained Settings:**
  - Well-suited for applications where data is scarce or labeling is expensive (e.g., healthcare, robotics).

# Key Benefits

# Limitations of Few-Shot Learning

**Key Limitations:**

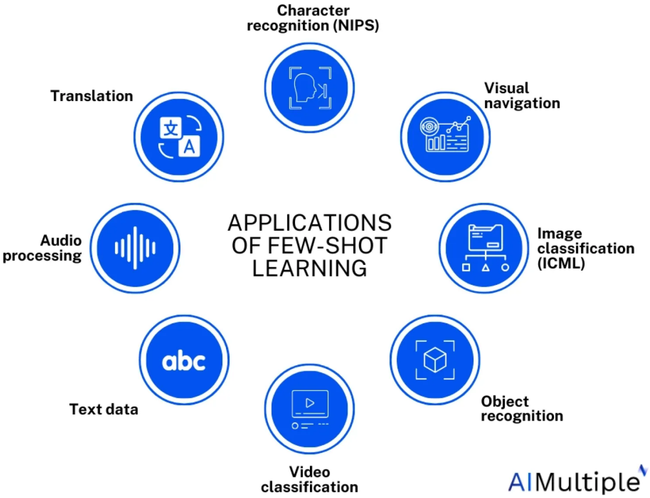
- **Overfitting Risks with Small Datasets:**
  - With limited labeled data, models may overfit to the available examples, leading to poor generalization on unseen data.
- **Computational Demands for Advanced Models:**
  - Many few-shot learning approaches (e.g., meta-learning) require significant computational resources, particularly during training.
- **Difficulty in Cross-Domain Generalization:**
  - Few-shot models may struggle to generalize across different domains or tasks if there is a large domain shift.
- **Limited Robustness to Fine-Grained Class Distinctions:**
  - Few-shot learning may face challenges when distinguishing between very similar or subtle class distinctions in complex datasets.

# Applications of Few-Shot Learning

**Key Application Areas:**

- **Computer Vision:**
  - Image classification with limited labeled data.
  - Object detection in rare or unseen categories.
- **Natural Language Processing (NLP):**
  - Sentiment analysis for low-resource languages.
  - Named entity recognition (NER) in specialized domains.
- **Healthcare:**
  - Diagnosis using scarce medical imaging data.
  - Personalized treatment recommendations.
- **Robotics:**
  - Learning novel tasks with minimal demonstrations.
  - Adaptation to new environments or tools.
- **Finance:**
  - Fraud detection in rare transaction patterns.
  - Forecasting for low-volume financial instruments.

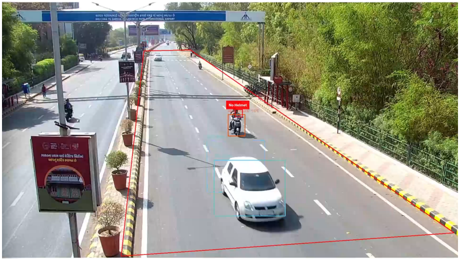
# Applications



# Case Study – Traffic Rule Violation Detection Using Few-Shot Learning

**Overview:**

- Traffic rule violation detection is a critical application in smart city and surveillance systems.
- Few-shot learning (FSL) is particularly beneficial in this domain due to the scarcity of annotated examples for rare or complex violations (e.g., illegal parking in specific zones, vehicles running red lights under unusual conditions).



# Challenges in Traffic Rule Violation Detection

**Challenges:**

- **Lack of Annotated Data:** Limited labeled data for specific traffic violations like driving on the wrong side or crossing restricted lanes.
- **Real-Time Detection:** The system must quickly adapt to new scenarios (e.g., different traffic environments or rules).
- **Fine-Grained Differences:** Distinguishing between minor infractions (e.g., a partial stop vs. a complete stop).

# Methodology: Dataset Preparation

**Support Set:**

- Labeled examples of known violations (e.g., 5 examples of wrong-side driving).

**Query Set:**

- Unlabeled traffic videos/images to classify.

**Synthetic Data Augmentation:**

- Using GANs to simulate rare violations under different conditions (e.g., lighting, weather).

# Methodology: FSL Model

**Prototypical Networks:**

- Represent each type of violation with a prototype in the feature space.
- Classify new violations based on their proximity to these prototypes.

**Meta-Learning (MAML):**

- Train the system on a variety of traffic scenarios (e.g., urban, rural, highway) for better generalization.

# Implementation: Overview

**Input:**

- Real-time video feeds or image captures from surveillance cameras.

**Preprocessing:**

- Object detection models (e.g., YOLO) to isolate vehicles and pedestrians.

**FSL-Based Classification:**

- Identify violations such as signal jumping, lane breaches, and illegal turns.

# Results: Performance and Adaptability

**Performance:**

- Achieved 90% accuracy in detecting violations with only 10 labeled examples per class.

**Adaptability:**

- The system adapted to new traffic rules with minimal additional labeled data.

**Efficiency:**

- Reduced annotation effort by 80% compared to traditional supervised methods.

# Future Trends in Few-Shot Learning

**Emerging Trends in Few-Shot Learning:**

- **Hybrid Models (Few-Shot and Zero-Shot Learning)**:
  - Combine few-shot and zero-shot learning to address unseen classes with minimal labeled data.
  - Enhances adaptability and generalization across different domains.
- **Advanced Data Augmentation**:
  - Use of generative models such as GANs and VAEs to synthesize data under various real-world conditions (e.g., lighting, weather).
  - Improves model robustness and generalization to rare scenarios.
- **Multimodal Learning**:
  - Integration of text, image, and audio data to create richer feature representations.
  - Enhances performance across a broader range of tasks, including cross-modal applications.
- **Lightweight Models for Edge Devices**:
  - Focus on creating efficient models that can run on resource-constrained devices such as smartphones or IoT devices.
  - Aims to reduce computational load while maintaining real-time performance.

# Conclusion

**Key Takeaways:**

- Few-shot learning enables models to generalize effectively with minimal labeled data, making it a valuable approach for real-world applications with scarce data.
- While it has significant advantages, such as cost-effectiveness and adaptability, few-shot learning also faces challenges, including overfitting and computational demands.
- Future advancements in hybrid models, data augmentation, and multimodal learning will further enhance the capabilities of few-shot learning, making it more robust and scalable.

# References


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998–6008. DOI: `10.48550/arXiv.1706.03762`.

Howard, J., Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*. DOI: `10.48550/arXiv.1801.06146`.

Cai, J. (2024). Introduction to few-shot learning. D-VELOP Workshop Series- Spring 2024, Purdue University. Available at: `https://guides.lib.purdue.edu/d-velop/workshops`.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D. (2016). Matching networks for one-shot learning. In *Advances in Neural Information Processing Systems* (NeurIPS), 29, 3630–3638.

Ravi, S., Larochelle, H. (2017). Optimization as a model for few-shot learning. In *Proceedings of the 5th International Conference on Learning Representations* (ICLR).

# Thank You!