DEPARTMENT OF SOFTWARE ENGINEERING

COLLEGE OF COMPUTING AND INFORMATICS

HARAMAYA UNIVERSITY

Big Data Group assignments

Name:-

1. Aliyu Yunus:- 0349/13
2. Abdinago Ashiga:- 0045/13
3. Entonyos Solomon:-0975/13

Submitted to: -Mr. Matias (MSc)

Apr, 2025

Haramaya

1. How can machine learning techniques be applied to improve image analytics tasks such as detection, segmentation, and classification? Discuss specific algorithms that are commonly used?

Image analytics involves extracting meaningful information from images. Machine Learning, especially deep learning, has revolutionized this domain by automating and enhancing accuracy in tasks like **Object Detection, Image Segmentation, Image Classification.**

These tasks are crucial in areas such as **medical diagnostics**, **autonomous vehicles**, **surveillance**, **agriculture**, and **manufacturing**. Now let us see each one of them in detail: -

1. Object Detection

Object detection involves identifying and localizing multiple objects within an image by drawing bounding boxes around them and assigning class labels to each detected object.

Common Algorithms:

- **YOLO (You Only Look Once**)

Type: Single-shot detector.

Strength: Extremely fast and efficient. Ideal for real-time detection.

How it works: Splits image into grids and predicts bounding boxes and class probabilities.

Use case: Autonomous vehicles, drone surveillance.

- **Faster R-CNN (Region-based Convolutional Neural Network)**

Type: Two-stage detector.

Strength: High accuracy; especially in complex images.

How it works: First proposes object regions (RPN), then classifies and refines bounding boxes.

Use case: Medical imaging, document analysis.

- **SSD (Single Shot Multibox Detector)**

Type: One-shot detection.

Strength: Balances speed and accuracy.

How it works: Uses feature maps of different scales to detect objects of varying sizes.

Use case: Surveillance cameras, mobile apps.

2. Image Segmentation

Image segmentation is the process of dividing an image into multiple segments or regions, usually by assigning a class label to every pixel.

Common Algorithms:

- **U-Net**

Type: Semantic segmentation.

Strength: Very effective for small datasets; excellent in biomedical fields.

How it works: Encoder-decoder architecture with skip connections that help retain spatial features.

Use case: Cell segmentation, MRI scan analysis.

- **Mask R-CNN**

Type: Instance segmentation.

Strength: Detects object and generates a pixel-level mask.

How it works: Builds upon Faster R-CNN by adding a branch for predicting segmentation masks.

Use case: Retail (shelf scanning), robotics (object grasping).

- **DeepLab (e.g., DeepLabv3+)**

Type: Semantic segmentation.

Strength: Excellent edge detection using atrous convolutions and spatial pyramid pooling.

Use case: Satellite imagery, urban scene understanding.

3. Image Classification

Image classification is the task of assigning a single label to an entire image, identifying the most prominent object or concept present.

Common Algorithms:

- **CNN (Convolutional Neural Networks)**

Type: Backbone architecture for most image classification models.

Strength: Automatically learns spatial hierarchies of features.

Popular Architectures:

AlexNet: Early breakthrough in deep learning image classification.

VGGNet: Uses small (3x3) filters, very deep.

ResNet: Introduces residual connections to solve vanishing gradients.

EfficientNet: Scalable and efficient for mobile and edge devices.

Use case: Plant disease detection, handwritten digit recognition, medical diagnosis (e.g., X-rays).

- **Transfer Learning (using pretrained models)**

What it is: Using a model trained on a large dataset (like ImageNet) and fine-tuning it on a smaller, specific dataset.

Benefits:

Saves time and computation.

Improves performance with limited data.

Common Pretrained Models: ResNet50, InceptionV3, MobileNet, EfficientNet.

Use case: Industrial defect detection, art classification.

2. How does Natural Language Processing (NLP) facilitate the transformation of unstructured text into actionable information?

Natural Language Processing (NLP) enables the transformation of unstructured text into actionable information by leveraging computational techniques to analyze, interpret, and extract meaningful insights from human language. Here’s how NLP facilitates this process:

2.1. Text Preprocessing & Normalization

NLP cleans and standardizes raw text through:

- **Tokenization**: Breaking text into words, phrases, or sentences.
- **Stopword Removal**: Filtering out common but insignificant words (e.g., "the," "and").
- **Lemmatization/Stemming**: Reducing words to their base forms (e.g., "running" → "run").
- **Noise Removal**: Handling typos, special characters, and irrelevant data.

2.2 Syntactic & Semantic Analysis

- **Part-of-Speech (POS) Tagging**: Identifies grammatical roles (nouns, verbs, etc.) to understand structure.
- **Dependency Parsing**: Analyzes sentence relationships (e.g., subject-action-object).
- **Named Entity Recognition (NER)**: Extracts entities (people, organizations, dates) for context.
- **Word Embeddings (e.g., Word2Vec, BERT)**: Represents words as vectors to capture meaning.

2.3 Information Extraction & Structuring

- **Keyphrase Extraction**: Identifies important terms (e.g., "customer complaint").
- **Relation Extraction**: Detects connections between entities (e.g., "Apple [company] launches iPhone").
- **Sentiment Analysis**: Determines emotional tone (positive/negative/neutral) for feedback analysis.
- **Topic Modeling (e.g., LDA)**: Groups documents by themes (e.g., "support tickets about shipping delays").

2.4 Advanced NLP for Actionable Insights

- **Machine Translation**: Converts text between languages for global use.
- **Question Answering (QA)**: Extracts precise answers from documents (e.g., chatbots).
- **Text Summarization**: Generates concise summaries of long reports.
- **Intent Detection**: Classifies user requests (e.g., "refund status" → customer support ticket).

2.5 Integration with Downstream Applications

Processed text is fed into:

- **Business Intelligence (BI) Tools**: For trend analysis.
- **CRM Systems**: To automate customer support.
- **Recommendation Engines**: Personalizing content based on user behavior.
- **Fraud Detection**: Flagging suspicious activity in financial reports.

Example Use Cases

- **Healthcare**: Extracting diagnoses from clinical notes.
- **Finance**: Analyzing earnings reports for investment signals.
- **Retail**: Parsing reviews to improve products.

By converting unstructured text into structured, analyzable data, NLP empowers organizations to make data-driven decisions, automate processes, and enhance user experiences.

3. Explain the Strength and weaknesses of Stream Processing with example.

Stream Processing

Stream Processing is a computing paradigm where data is processed **in real-time** (or near real-time) as it flows into the system.

Unlike **batch processing** (which collects data first and then processes it), stream processing handles data **event by event**, often within milliseconds or seconds of its arrival.

1. Strengths of Stream Processing

- **Real-Time Data Processing**

Process and react to data immediately.

Enables instant decision-making, which is crucial for time-sensitive applications.

Example*:* Fraud detection systems can block suspicious credit card transactions **before** they are approved.

- **Low Latency**

Minimal delay between data arrival and result generation.

Enables interactive and responsive systems.

Example*:* Live dashboard showing user activity on an e-commerce site (clicks, purchases, searches).

- **Continuous Data Ingestion**

Handles data that never stops coming (e.g., IoT sensors, social media).

Systems can process an infinite flow of data without stopping or waiting.

Example*:* Monitoring smart city traffic sensors to optimize signal timing.

- **Scalability**

Stream processing frameworks (like **Apache Kafka**, **Apache Flink**, and **Apache Spark Streaming**) are designed to scale across distributed systems.

They can handle millions of events per second by distributing workload across nodes.

Example*:* Video streaming platforms processing usage data from millions of users concurrently.

- **Supports Event-Driven Architectures**

Enables reactive applications where actions are triggered by data events.

Ideal for microservices and responsive systems.

Example*:* When a user posts a tweet, systems instantly trigger notifications, trend updates, or moderation checks.

2. Weaknesses of Stream Processing

- **Complexity in Design and Debugging**

Writing stream processing logic is **more complex** than batch processing.

Requires handling data order, concurrency, and state management.

Example: An e-commerce site analyzing real-time cart updates must manage user state carefully.

- **Difficulty in Handling Late or Out-of-Order Data**

Real-world data can arrive late (network lag, retries) or out of sequence.

Ensuring **correctness** (e.g., computing averages, counts) becomes harder.

Example*:* In stock trading, receiving a price update **after** a transaction can corrupt analytics unless managed carefully.

- **Resource-Intensive**

Requires constant computation and memory usage.

Costs can grow rapidly with scale due to 24/7 uptime needs.

Example*:* Real-time video analytics for surveillance can become expensive due to high-resolution video feeds.

- **Challenging Error Recovery**

Handling failures (e.g., node crash or network partition) is difficult.

Systems need **checkpointing**, **rollback**, or **exactly-once semantics**, which adds complexity.

Example*:* If a message broker like Kafka fails mid-stream, the application must know what was processed and what wasn't**.**

- **Not Ideal for Heavy Computations**

  Complex analytics (e.g., deep learning) are better suited for batch processing.

  Example: Training a recommendation model is done in batches, while real-time predictions use streams.