ICT304 Tutorial 2

Table of Contents

[*1. Types of Data Representation in Machine Learning* 1](#_Toc178118499)

[*1.1 Tabular Data* 1](#_Toc178118500)

[*1.2 Text Data* 1](#_Toc178118501)

[*1.3 Image Data* 1](#_Toc178118502)

[*1.4 Time Series Data* 2](#_Toc178118503)

[*1.5 Graph Data* 2](#_Toc178118504)

[*2. Common Methods for Scaling Data in Machine Learning* 3](#_Toc178118505)

[*2.1 Min-Max Scaling* 3](#_Toc178118506)

[*2.2 Standardisation* 4](#_Toc178118507)

[*2.3 Robust Scaling* 5](#_Toc178118508)

[*3. Pre-Trained Networks for Image and Text Processing* 7](#_Toc178118509)

[*3.1 Pre-Trained Networks for Image Processing* 7](#_Toc178118510)

[*3.2 Pre-Trained Networks for Text Processing* 9](#_Toc178118511)

[*3.3 Concept of Embeddings* 10](#_Toc178118512)

[*4. Object Detection Module Using Pre-Trained Networks* 12](#_Toc178118513)

[*4.1 Using an ML Development Cycle* 12](#_Toc178118514)

[*4.2 Environment and Package Installation* 13](#_Toc178118515)

[*4.3 Training, Validating, and Testing Steps* 13](#_Toc178118516)

[*4.4 Pre-processing* 14](#_Toc178118517)

[*4.5 Object Classes the System Can Detect* 14](#_Toc178118518)

[*4.6 Performance Metrics* 14](#_Toc178118519)

[*4.7 Evaluation of Sub-System* 15](#_Toc178118520)

[*5. Addressing Common Challenges in Machine Learning Development* 16](#_Toc178118521)

[*5.1 Class Imbalance Problem* 16](#_Toc178118522)

[*5.2 Data Augmentation* 16](#_Toc178118523)

[*5.3 Handling Missing Data* 16](#_Toc178118524)

## *1. Types of Data Representation in Machine Learning*

Data representation plays a crucial role in the effectiveness of machine learning models. How data is structured varies depending on its type and the algorithms used. Below are several prevalent forms of data representations:

### *1.1 Tabular Data*

Tabular data is one of the most widely utilised structures in machine learning. It organises information into rows (instances) and columns (features or attributes). Each row corresponds to a unique instance, while each column represents a specific attribute related to that instance. **Example:** A dataset detailing house prices, where each row includes information such as the size of the house, the number of rooms, location, and the sale price. This format is commonly employed to predict outcomes like housing prices based on the provided features.

## *1.2* *Text Data*

Text data, commonly known as natural language data, necessitates specialised methods for machine learning models to interpret it. Typical representations include Bag-of-Words (BoW), Term Frequency-Inverse document Frequency (TF-IDF), and word embeddings (like Word2Vec and GloVe). Unstructured text must be transformed into a vector or matrix format for machine learning applications.

**Example:** A set of customer reviews, where each review is a text document. Models trained on this dataset determine each review's sentiment (e.g., positive or negative).

### *1.3 Image Data*

Image data comprises pixel values that can be represented as multidimensional arrays. Each pixel typically holds information regarding colour intensity (such as RGB values). Due to its complexity, deep learning models, particularly Convolutional Neural Networks (CNNs), are frequently utilised to handle image data.

**Example:** The MNIST dataset contains images of handwritten digits representing 28x28 pixel grids. This dataset is often used to train models for digit recognition tasks.

### *1.4 Time Series Data*

Time series data consists of sequences of data points arranged in chronological order. Machine learning models designed for time series data aim to identify trends, patterns, and temporal dependencies within the dataset. This type of data is prevalent in forecasting and anomaly detection scenarios.

**Example:** A dataset that includes daily closing prices of a stock. The temporal nature of the data enables models to predict future stock prices based on historical trends.

### *1.5 Graph Data*

Graph data is characterised by nodes and edges that signify entities and their interconnections. Graph-based models, such as Graph Neural Networks (GNNs), are employed to analyse this data type. Graph data is often relevant in areas where the relationships between entities are significant, such as social networks or recommendation systems.

**Example:** A social network graph where nodes represent users, and edges indicate friendships. Machine learning models can forecast future friendships based on the existing graph structure.

## *2. Common Methods for Scaling Data in Machine Learning*

Scaling data is an essential preprocessing step in many machine learning techniques, as it ensures that features contribute equally to the model’s effectiveness. Various scaling methods are applied based on the characteristics of the data and the specific machine-learning algorithm used. Below are three widely used scaling techniques, as well as the pros, cons, and examples.

### *2.1 Min-Max Scaling*

Min-Max scaling, called normalisation, rescales the data to a predetermined range, usually between 0 and 1. This is achieved using the following formula:

Figure 1. Min-Max Formula

Where and represent the minimum and maximum values of the features.

**Use:**

Min-Max scaling is beneficial when the bounds of your data are known, and you want to ensure that the scaled values fall within a specific range (e.g., 0 to 1). It is commonly used in algorithms that depend on distance calculations, such as k-nearest neighbours (KNN) and Support Vector Machines (SVM).

**Example:**

Consider a dataset of house prices ranging from $100,000 to $1,000,000. Using Min-Max scaling, a house priced at $500,000 would be transformed to 0.44 on a scale from 0 to 1.

**Advantages:**

* Maintains the relationships between features and ensures equal treatment for all features.
* Simple and straightforward to implement.

**Disadvantages:**

* Sensitive to outliers, as a single extreme value can significantly skew the scaling.

### *2.2 Standardisation*

Standardisation, also known as Z-score scaling, adjusts the data with a mean of 0 and a standard deviation of 1. The formula for standardisation is:

Figure 2. Standardisation Formula

Where is the mean of the feature, and is the standard deviation.

**Use:**

Standardisation is commonly employed in algorithms that assume a normal data distribution, such as Linear Regression, Logistic Regression, and Principal Component Analysis (PCA).

**Example:**

For a dataset of student test scores with a mean of 70 and a standard deviation of 10, a score of 80 would be standardised as follows:

Figure 3. Standardisation Example

This indicates that the score is 1 standard deviation above the mean.

**Advantages:**

* More robust against outliers since it standardises data based on its distribution rather than absolute min/max values.
* Help maintain the variance and importance of features over a wide range of values.

**Disadvantages:**

* If the data isn’t approximately normally distributed, standardisation may not be as effective
* Does not impose any fixed bounds (e.g., 0 to 1).

### *2.3 Robust Scaling*

Robust scaling is akin to standardisation but utilises the median and the interquartile range (IQR) instead of the mean and standard deviation. The formula is:

Figure 4. Robust Scaling Formula

Where IQR is the difference between the 75th and 25th percentiles.

**Use:**

Robust scaling is particularly useful when the dataset contains outliers, as it mitigates their impact on the scaled values.

**Example:**

Imagine a dataset of household incomes where most incomes hover around $50,000, but a few outliers exceed $1,000,000. Robust scaling will adjust the incomes while minimising the influence of these outliers.

**Advantages:**

* Less affected by outliers compared to Min-Max scaling and standardisation.
* Effectively scales the central portion of the data.

**Disadvantages:**

* May not perform optimally when most of the data is well-distributed without significant outliers.
* Since it focuses on medians, it might not always represent the data structure effectively for specific machine learning models.

|  |  |  |  |
| --- | --- | --- | --- |
| **Scaling Method** | **Key Advantages** | **Key Disadvantage** | **Use Case Example** |
| **Min-Max Scaling** | Simple, keep Data within fixed bounds | Sensitive to outliers | Knn, SVM, neural networks |
| **Standardisation** | Handles different distributions well | Doesn’t handle non-normal data as effectively | Linear Regression, PCA |
| **Robust Scaling** | Reduces the impact of outliers | Might not work well for well-distributed data | Datasets with significant outliers |

Figure 5. Summary of Scaling Methods

## *3. Pre-Trained Networks for Image and Text Processing*

Pre-trained networks are machine learning models trained on extensive datasets and can be adapted for specific tasks. These models save time and computational resources by providing a robust foundation rather than starting from scratch. Below are the two widely used pre-trained networks for image processing and two for text processing, followed by an explanation of embeddings.

### *3.1 Pre-Trained Networks for Image Processing*

#### *3.1.1 VGG16*

VGG16 is a convolutional neural network (CNN) model developed by the Visual Geometry Group (VGG) at Oxford University. It is recognised for its straightforward design and deep architecture, comprising 16 layers. VGG16 was trained on the ImageNet dataset, which includes millions of labelled images across thousands of categories.

**Use:**

VGG16 is frequently employed for image classification, object detection, and feature extraction. It can be fine-tuned for specific applications, such as identifying various types of plants, animals, or objects.

**Advantages:**

* Simple architecture that is easy to implement.
* Effective for transfer learning due to its depth.

**Disadvantages:**

* Requires substantial computational resources because of its large number of parameters.

#### *3.1.2 ResNet*

ResNet (Residual Networks) introduced skip connections or shortcuts, enabling models to train deeper architectures without encountering vanishing gradients. The most well-known variant, ResNet50, consists of 50 layers. Like VGG16, it was pre-trained on the ImageNet dataset.

**Use:**

ResNet is commonly used for image segmentation, object detection, and image recognition tasks.

**Advantages:**

* Facilitates the training of deeper networks by addressing the vanishing gradient issue.
* Performs complex tasks effectively due to its depth.

**Disadvantages:**

* More complex architecture compared to simpler CNN models.

### *3.2 Pre-Trained Networks for Text Processing*

#### *3.2.1 BERT (Bidirectional Encoder Representations from Transformers)*

BERT, created by Google, is a transformer-based model that processes text in a bidirectional manner, meaning it considers both the left and right context within a sentence. It is pre-trained on extensive corpora like Wikipedia and books, using tasks such as masked language modelling (predicting a masked word) and next-sentence prediction.

**Use:**

BERT can be fine-tuned for various tasks, including text classification, sentiment analysis, and question-answering systems.

**Advantages:**

* Captures bidirectional context, making it highly effective for natural language understanding tasks.
* Can be adapted for specific tasks with limited training data.

**Disadvantages:**

* Large model size and high computational demands for training and inference.

#### *3.2.2 GPT (Generative Pretrained Transformer)*

GPT, developed by OpenAI, is another transformer-based model designed to generate language. Unlike BERT, which focuses on understanding the text’s context, GPT is trained to produce coherent and human-like text by predicting the next word in a sequence.

**Use:**

GPT is widely applied in text generation, summarisation, and chatbot development tasks.

**Advantages:**

* Capable of producing high-quality text, making it suitable for creative and conversational applications.

**Disadvantages:**

* Similar to BERT, it has significant computational requirements and needs substantial resources for fine-tuning.

### *3.3 Concept of Embeddings*

Embeddings refer to dense, low-dimensional vector representations of data (such as words or images) that position similar data points closer together in vector space. Pre-trained networks like VGG16, ResNet, BERT, and GPT utilise embeddings to represent high-dimensional input data in a more meaningful and structured manner for machine learning models.

#### *3.3.1 Image Embeddings*

In image processing, pre-trained networks like VGG16 or ResNet are often used to create image embeddings. These embeddings serve as high-level feature representations of an image. For instance, a model like VGG16 can take an image as input, process it through its layers, and produce a feature vector (embedding) that summarises key characteristics of the image (e.g., shape, texture, and colour patterns).

**Example:** For a collection of images of cats and dogs, image embedding generated by ResNet would place similar images (e.g., different breeds of dogs) closer together in the feature space, facilitating the differentiation between categories by machine learning models.

#### *3.3.2 Text Embeddings*

In text processing, embeddings transform words or sentences into dense vectors. Pre-trained models like BERT or GPT generate contextual embeddings, where the meaning of a word varies depending on the context of the surrounding words.

**Example:** Consider the word “bank” in the phrases “I sat by the river bank” and “I deposited money in the bank.” BERT would produce different embedding for “bank” in these contexts, enabling the model to grasp the meaning based on usage.

**Advantages of Embeddings:**

* **Dimensionality Reduction:** Embeddings reduce the dimensionality of the data while preserving essential relationships, making it easier for machine learning models to process.
* **Semantic Understanding:** Embeddings capture semantic relationships, aiding models in generalisation. For example, word embeddings can recognise that “king” and “queen” are related, while image embeddings can cluster similar objects.

**Disadvantages of Embeddings:**

* **Computationally Intensive:** Generating high-quality embeddings with pre-trained models can be resource-heavy.
* **Context Dependence:** Some embeddings may struggle to capture nuances effectively, especially for words or images that are contextually ambiguous.

## *4. Object Detection Module Using Pre-Trained Networks*

### *4.1 Using an ML Development Cycle*

The development of an object detection module follows the standard Machine Learning (ML) cycle:

1. Problem Definition:

* The task is to detect hazardous objects (e.g., guns, knives) in images, which will be integrated into a security system.

2. Data Collection:

* Collect images of the object of interest (knives, guns) for both training and validation. Use datasets like Open images or manually collect images, then annotate them using tools like LabelImg.

3. Data Preparation:

* Split the dataset into training and validation sets.
* Preprocess the images (resizing to the model’s required input size, annotating with bounding boxes).

4. Model Selection:

* Use a pre-trained YOLOv8 model (since it’s lightweight, fast, and accurate), fine-tuned to detect hazardous objects.

5. Training:

* Train the YOLOv8 model on the custom dataset with hazardous objects to improve its detection capabilities.

6. Evaluation:

* Use performance metrics such as Precision, Recall, and mAP (Mean Average Precision) to evaluate the accuracy of the model.

7. Deployment:

* Deploy the model in a real-time environment for detecting hazardous object in images.

### *4.2 Environment and Package Installation*

Refer to the **README** file located inside the codes folder submitted along with this document.

### *4.3 Training, Validating, and Testing Steps*

1. Prepare Dataset:

* Use dataset for hazardous objects from Kaggle and label the images using the **labelling.ipynb** script, saving them to YOLO format (*.txt* files without bounding boxes).

2. Create a YAML File:

* Create a **fine-tune.yaml** file that defines paths to the dataset, the number of classes, and class names (e.g., ‘gun’, ‘knife’).

3. Fine-Tune the Model:

* Fine-tuning allows the pre-trained model to adjust to the specific tasks (hazardous object detection). Which is covered in the ***Fine-Tune the Model*** section in **odm.ipynb** file.

4. Validate the Model:

* After training, validate the model will take place in the ***Evaluate the Model*** section in the **odm.ipynb** file.

5. Test the Model:

* Finally, test the model on new images in the ***Perform Object Detection on an Image*** section in the **odm.ipynb** file.

### *4.4 Pre-processing*

Pre-processing mainly involves resizing the input images to fit the required input dimensions of the YOLO model (e.g., 640x640). This is important to maintain consistency across training, validation, and testing images. The annotation format must also be YOLO-compliant, with bounding boxes labelled in a specific way.

In this case, no advanced pre-processing like normalisation or augmentation is strictly required unless the aim tis to generalise the model across various lighting or background conditions.

### *4.5 Object Classes the System Can Detect*

With a fine-tuned model, the system can detect objects such as:

* **Knives** (used in crimes)
* **Guns** (handguns, pistols)

Additional hazardous objects can be added by updating the dataset and retraining the model with new object categories.

### *4.6 Performance Metrics*

For evaluating the performance of the object detection system, I use standard object detection metrics. These metrics help assess how well the model detects hazardous objects like guns or knives in the images.

1. Mean Average Precision (mAP):

* **mAP** is a primary performance metric used in object detection tasks. It calculates the area under the precision-recall curve for different *Intersection over Union (IoU)* thresholds. In this case, mAP is computed for *IoU=0.5* ([mAP@0.5](mailto:mAP@0.5)), which is a common threshold.
* In the **evaluate\_model** function, **results.box.maps** will output the mAP scores aafter validation.

2. Precision-Recall Curve:

* Precision and recall can also be visualised to understand the trade-off between detecting objects and minimising false positives. YOLO models use this internally for validation during fine-tuning.

### *4.7 Evaluation of Sub-System*

The system is evaluated through the following steps:

1. Validation Dataset:

* The system is evaluated on a validation dataset that contains labelled images with hazardous objects like knives and guns. This is configured in the ***fine-tune.yaml*** file using the custom dataset.
* The **evaluate\_model** function runs on the validation set specified in the YAML configuration file.

2. mAP Evaluation

* As shown in the code in **odm.ipynb**, after fine-tuning, the **evaluate\_model()** function calculates the mAP scores using *results.box.maps*. Higher mAP scores indicate better detection capability for hazardous objects.

## *5. Addressing Common Challenges in Machine Learning Development*

### *5.1 Class Imbalance Problem*

Class imbalance occurs when one class is significantly underrepresented. For instance, fewer images of hazardous objects like guns compared to benign images. This can cause the model to favour the majority class.

**Approaches:**

1. **Oversampling:** Increasing the number of minority class samples using techniques like ***SMOTE***, which generates synthetic examples.

2. **Undersampling:** Reducing the number of majority class samples to balance the dataset.

### *5.2 Data Augmentation*

Data augmentation involves creating modified versions of images to artificially expand the dataset, improving model robustness. Techniques like random rotation, flipping, and brightness adjustment can help balance and diversify the dataset.

### *5.3 Handling Missing Data*

To manage missing data:

1. **Imputation:** Replacing missing values with the mean, median, or mode.

2. **Dropping Missing values:** Removing samples or features with missing data if their presence is minimal.