Foundations of AI, Machine Learning, Deep Learning, and Computer Vision

A Mathematical and Conceptual Guide for MSc Students

Author: Bishal Pandey

University of Southampton

Version 1.0 — August 15, 2025

Contents

1 Linear Algebra

1.1 Linear Algebra for Machine Learning

Linear Algebra forms the mathematical foundation for many Machine Learning algorithms, especially in handling data as vectors, matrices, and tensors. Understanding these concepts enables efficient representation and computation in models.

1.1.1 Vectors

A **vector** is an ordered list of numbers representing a point or direction in space. Example:

$$\mathbf{v} = \begin{bmatrix} 2 \\ 5 \\ -1 \end{bmatrix}$$

Here, $\mathbf{v}$ is a 3-dimensional vector.

Key Operations:

- **Addition:** $\mathbf{a} + \mathbf{b}$
- **Scalar Multiplication:** $c \cdot \mathbf{a}$
- **Dot Product:** $\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$

1.1.2 Matrices

A **matrix** is a 2D array of numbers, often used to represent datasets or transformations.

Example of a 2×3 matrix:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Common Matrix Operations:

- **Transpose:** A^T
- **Multiplication:** AB
- **Inverse:** A^{-1} (when it exists)

1.1.3 Application in Machine Learning

In ML, data is often stored in a matrix X of shape $(n_{\text{samples}}, n_{\text{features}})$, where each row is a sample and each column is a feature. Operations like multiplying X with a weight vector $\mathbf{w}$ form the basis of linear models:

$$\hat{y} = X\mathbf{w} + b$$

2 Probability and Statistics

2.1 Probability and Statistics for Machine Learning

Probability and statistics provide the tools to model uncertainty, estimate patterns, and make predictions from data. In Machine Learning, these concepts are used for algorithm design, model evaluation, and understanding data distributions.

2.1.1 Basic Probability Concepts

Probability

The probability of an event A is defined as:

$$P(A) = \frac{\text{Number of favorable outcomes}}{\text{Total number of outcomes}}, \quad 0 \leq P(A) \leq 1$$

Conditional Probability

Measures the probability of event A given event B:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad P(B) > 0$$

Bayes' Theorem

Used for updating probabilities based on new evidence:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

This is widely used in algorithms like Naive Bayes for classification.

—

2.1.2 Random Variables

A **random variable** assigns numerical values to outcomes of a random experiment.

- **Discrete RV:** Takes countable values (e.g., number of clicks on an ad).
- **Continuous RV:** Takes any value within a range (e.g., height, temperature).

—

2.1.3 Key Statistical Measures

- **Mean:** $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$
- **Variance:** $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$
- **Standard Deviation:** $\sigma = \sqrt{\sigma^2}$
- **Median:** Middle value when data is sorted
- **Mode:** Most frequently occurring value

—

2.1.4 Common Probability Distributions

Bernoulli Distribution

Models binary outcomes (0 or 1):

$$P(X = x) = p^x(1-p)^{1-x}, \quad x \in \{0, 1\}$$

Normal (Gaussian) Distribution

Commonly seen in natural phenomena:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Uniform Distribution

All outcomes have equal probability:

$$f(x) = \frac{1}{b-a}, \quad a \leq x \leq b$$

—

2.1.5 Application in Machine Learning

- Estimating class probabilities in classification tasks
- Measuring uncertainty in model predictions
- Initializing weights in neural networks (often sampled from Gaussian or uniform distributions)
- Evaluating statistical significance of model results

3 Optimization and Calculus

3.1 Optimization and Calculus for Machine Learning

Optimization is the process of finding the best parameters for a model to minimize or maximize an objective function. In Machine Learning, this usually means minimizing a loss function using calculus-based methods.

3.1.1 Derivatives and Gradients

Derivative

Measures how a function changes as its input changes:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Gradient

The gradient is a vector of partial derivatives, indicating the direction of steepest ascent for multivariable functions:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

—

3.1.2 Gradient Descent Algorithm

Gradient Descent is the most common optimization algorithm in ML. It updates parameters in the direction opposite to the gradient to minimize the loss function.

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \cdot \nabla_\theta J(\theta) \tag{3.1}$$

Where:

- θ – Model parameters
- η – Learning rate

- $J(\theta)$ – Loss function

Pseudocode:

1. Initialize parameters θ randomly.
2. Compute gradient $\nabla_\theta J(\theta)$.
3. Update $\theta \leftarrow \theta - \eta \cdot \nabla_\theta J(\theta)$.
4. Repeat until convergence.

—

3.1.3 Variants of Gradient Descent

- **Batch Gradient Descent** – Uses all training data for each update.
- **Stochastic Gradient Descent (SGD)** – Uses one sample at a time for faster but noisier updates.
- **Mini-batch Gradient Descent** – Compromise between batch and SGD.
- **Adam Optimizer** – Adaptive learning rates with momentum.

—

3.1.4 Applications in Machine Learning

- Training neural networks
- Logistic regression parameter updates
- SVM optimization
- Matrix factorization in recommender systems

4 Machine Learning Algorithms

4.1 Machine Learning Algorithms

Machine Learning algorithms can be broadly categorized into supervised, unsupervised, and reinforcement learning. This section covers the most common algorithms and their core principles.

4.1.1 Supervised Learning

Trains a model on labeled data to predict outputs for new inputs.

Linear Regression

Models the relationship between input X and output y using:

$$\hat{y} = X\mathbf{w} + b$$

Optimized by minimizing Mean Squared Error (MSE).

Logistic Regression

Used for binary classification, modeling the probability of the positive class:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\mathbf{w}^\top x + b)}}$$

Decision Trees

Recursive partitioning of data into homogeneous subsets based on features. Key concepts: Gini impurity, information gain.

Random Forest

Ensemble of decision trees with bootstrapping and feature randomness.

Support Vector Machines (SVM)

Finds the hyperplane that maximizes the margin between classes.

—

4.1.2 Unsupervised Learning

Trains models on unlabeled data to find hidden patterns.

K-Means Clustering

Partitions data into k clusters by minimizing intra-cluster variance.

Principal Component Analysis (PCA)

Reduces dimensionality by projecting data to directions of maximum variance.

—

4.1.3 Model Evaluation

- Classification: Accuracy, Precision, Recall, F1-score, ROC-AUC
- Regression: MAE, RMSE, R^2 Score

—

4.1.4 Applications in Machine Learning

- Spam detection
- Stock price prediction
- Customer segmentation
- Image classification

5 Deep Learning Foundations

5.1 Deep Learning

Deep Learning uses multi-layer neural networks to learn hierarchical representations from data.

5.1.1 Artificial Neural Networks (ANNs)

An ANN consists of layers of interconnected neurons:

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}, \quad a^{[l]} = f(z^{[l]})$$

Forward Pass

Input propagates through the network to produce an output.

Backward Pass

Gradients are computed using backpropagation to update weights.

—

5.1.2 Activation Functions

Function	Formula	Notes
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	Outputs in (0,1)
ReLU	$\max(0, x)$	Speeds up convergence
Tanh	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	Outputs in (-1,1)

—

5.1.3 Training Neural Networks

1. Initialize weights
2. Forward pass to compute predictions
3. Compute loss (e.g., cross-entropy, MSE)
4. Backpropagate gradients
5. Update weights using optimizers (SGD, Adam)

—

5.1.4 Regularization Techniques

- Dropout
- L1/L2 regularization
- Early stopping

6 Computer Vision

6.1 Computer Vision

Computer Vision enables machines to interpret and process visual data such as images and videos.

6.1.1 Image Representation

Images are stored as pixel grids (height × width × channels). For grayscale, channels = 1; for RGB, channels = 3.

—

6.1.2 Convolutional Neural Networks (CNNs)

CNNs are designed to extract spatial features from images using convolutional layers.

Convolution Operation

Applies filters (kernels) to input images to detect patterns like edges and textures.

Pooling Layers

Reduce spatial dimensions to lower computation and increase robustness.

- Max pooling – keeps the maximum value in a patch
- Average pooling – keeps the average value

—

6.1.3 Data Augmentation

Improves generalization by modifying training images:

- Rotation, flipping, scaling
- Color jittering
- Random cropping

—

6.1.4 Applications in ML

- Object detection
- Facial recognition
- Medical image analysis
- Self-driving cars

7 Advanced Mathematical Topics

7.1 Advanced Topics in Machine Learning

7.1.1 Principal Component Analysis (PCA)

Projects high-dimensional data into fewer dimensions while retaining variance.

7.1.2 Singular Value Decomposition (SVD)

Matrix factorization technique used in dimensionality reduction and recommender systems.

7.1.3 Kullback-Leibler (KL) Divergence

Measures the difference between two probability distributions:

$$D_{KL}(P\|Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}$$

7.1.4 Ensemble Learning

Combines multiple models to improve performance.

- Bagging (e.g., Random Forest)
- Boosting (e.g., XGBoost)
- Stacking

7.1.5 Transfer Learning

Uses pretrained models and fine-tunes them for a new task.

7.1.6 Explainable AI (XAI)

Techniques like LIME and SHAP to interpret model predictions.

8 Extended Topics

8.1 Extended Topics

8.1.1 Transformers

Architecture using self-attention for NLP and beyond. Example: BERT, GPT models.

8.1.2 Reinforcement Learning

Training agents to make sequential decisions using rewards and penalties.

8.1.3 Generative Adversarial Networks (GANs)

Two networks (generator and discriminator) compete to produce realistic data.

8.1.4 Self-Supervised Learning

Learning useful representations from unlabeled data.

8.1.5 Ethics in AI

Responsible AI practices:

- Fairness and bias mitigation
- Privacy preservation
- Transparency and accountability

9 Real ML

9.1 Level 1: Beginner Machine Learning Projects

9.1.1 Overview

Level 1 focuses on structured datasets and classical ML models. Each project follows a complete ML pipeline: data loading, preprocessing, model training, evaluation, and interpretation.

9.1.2 Projects Completed

1. **Iris Flower Classification**
    - **Task:** Multi-class classification (3 flower species)
    - **Model:** Logistic Regression
    - **Accuracy:** 100%
2. **MNIST Digit Classifier (8×8)**
    - **Task:** Multi-class classification (digits 0–9)
    - **Model:** Logistic Regression
    - **Accuracy:** ∼97.2%
    - **Flattened shape:** (1797, 64), each 8×8 image converted to 64-length vector
3. **Breast Cancer Detection**
    - **Task:** Binary classification (malignant vs. benign)
    - **Dataset imbalance:** handled using `class_weight='balanced'`
    - **Model:** Logistic Regression
    - **Accuracy:** ∼95.6%
    - **Key focus:** High recall on malignant class (class 0)

9.1.3 Core Concepts Covered

- Train-Test Split (`train_test_split()`) — with stratification for imbalanced datasets.
- Feature Scaling — `StandardScaler` normalizes features to zero mean and unit variance.
- Logistic Regression — Used for both binary and multiclass classification.

- Model Evaluation — accuracy, precision, recall, F1-score, confusion matrix.
- Flattening Images — from (8, 8) to 64-dim feature vector.
- Class Imbalance — tackled via `class_weight='balanced'`.

9.1.4 Q&A and Deep Concepts

Why do we use `fit()` on training data and only `transform()` on test data?

To prevent **data leakage**. `fit()` learns the scaling parameters (mean, std) from the training set, and the same parameters are applied to test data using `transform()` to simulate unseen real-world data.

What is `class_weight='balanced'`?

Automatically adjusts weights inversely proportional to class frequency:

$$w_i = \frac{n_{\text{samples}}}{n_{\text{classes}} \cdot n_i}$$

Where n_i is the number of samples in class i. Used to improve model learning when data is imbalanced (e.g., more benign than malignant cases).

Why use Logistic Regression here instead of TensorFlow or CNNs?

Level 1 focuses on small datasets and structured/tabular data — classical ML models are more efficient and interpretable. TensorFlow and CNNs are better suited for large image or sequence datasets.

Can I use KNN or similar models in small projects like plant disease detection?

Yes. For simple image classification or structured datasets:

- KNN can work, but may not scale well.
- CNNs perform better on real images (e.g., plant leaves).

What is SVM with Kernels?

Support Vector Machines (SVMs) can create non-linear decision boundaries using **kernels**:

- Linear Kernel — straight-line separation.
- RBF (Radial Basis Function) — for non-linear, complex boundaries.

Supervised vs. Unsupervised Models — with Examples

- **Supervised Learning:** Data has input-output labels.
  - Logistic Regression — Tumor classification.
  - KNN — Iris flower classification.
  - SVM — Email spam detection.

- **Unsupervised Learning:** No output labels, patterns inferred.
  - KMeans Clustering — Customer segmentation.
  - PCA — Dimensionality reduction.

9.1.5 Flattening Explanation (MNIST 8×8)

Image shape: `(1797, 8, 8)` → flattened using `digits.data` → `(1797, 64)`. Each 8×8 pixel image becomes a 64-length feature vector for the ML model.

9.1.6 Final Notes

- Level 1 complete ✓
- All notebooks saved under: `Real ML/Level 1/`
- GitHub repository created and updated

9.2 Types of Machine Learning Models and Practical Applications

9.2.1 Overview

Machine Learning models are typically categorized into:

- **Supervised Learning** — trained on labeled data.
- **Unsupervised Learning** — trained on unlabeled data.
- **Semi-Supervised Learning** — combination of labeled and unlabeled data.
- **Reinforcement Learning** — learns by interacting with an environment via rewards.

9.2.2 Supervised Learning Models

Logistic Regression Binary/multiclass classification.
Example: Tumor classification (malignant vs. benign).

Linear Regression Predicts continuous values.
Example: House price prediction.

K-Nearest Neighbors (KNN) Classification and regression.
Example: Plant species recognition.

Support Vector Machine (SVM) Classifier with kernel support.
Example: Email spam detection.

Decision Tree Simple tree-based classifier.
Example: Customer churn prediction.

Random Forest Ensemble of decision trees.
Example: Loan approval prediction.

XGBoost / LightGBM Gradient boosting models.
Example: Fraud detection.

Naive Bayes Probabilistic model using Bayes' theorem.
Example: Text classification (spam, sentiment).

9.2.3 Unsupervised Learning Models

K-Means Clustering Groups data into clusters.
Example: Customer segmentation.

Hierarchical Clustering Builds nested clusters.
Example: Biological taxonomy grouping.

Principal Component Analysis (PCA) Dimensionality reduction.
Example: Visualizing high-dimensional data.

Autoencoders Neural networks for compression and reconstruction.
Example: Anomaly detection, denoising.

9.2.4 Other Categories

Semi-Supervised Learning Combines few labeled and many unlabeled data.
Example: Large-scale text classification with limited labels.

Reinforcement Learning Agent learns via rewards in an environment.
Example: Game playing (e.g., AlphaGo), robotics.

9.3 Level 2: Real-World Tabular Regression

9.3.1 Overview

This section includes three real-world regression projects using structured/tabular data. These projects reinforce essential machine learning concepts such as feature scaling, model selection, evaluation metrics, hyperparameter tuning, and handling real-world datasets.

9.3.2 Projects Covered

1. California Housing Price Prediction
2. Bike Sharing Demand Prediction
3. Medical Insurance Cost Prediction

9.3.3 Project 1: California Housing Price Prediction

Objective: Predict the median house prices using demographic and geographic data from California.

Dataset: `sklearn.datasets.fetch_california_housing` **Samples:** 20,640 **Features:** 8 numerical

Workflow:

- Data loading and exploratory data analysis (EDA)
- Train-test split
- Feature scaling using `StandardScaler`
- Model training and comparison:
  - Linear Regression
  - Decision Tree (with hyperparameter tuning)
  - Random Forest (with `GridSearchCV`)

Final Performance (Tuned Random Forest):

- RMSE = 0.50
- MAE = 0.33
- $R^2 = 0.8058$

9.3.4 Project 2: Bike Sharing Demand Prediction

Objective: Predict daily bike rental counts based on weather, season, holidays, temperature, and other variables.

Dataset: `train.csv` from the Bike Sharing Demand dataset **Samples:** 10,886 **Target:** Count of bikes rented

Key Concepts:

- Feature engineering: extracting temporal features (hour, day, month, year)
- One-hot encoding for categorical variables
- Handling skewed distributions
- Feature scaling using `StandardScaler`

Final Performance (Tuned Random Forest):

- RMSE = 73.27
- MAE = 44.70
- $R^2 = 0.8374$

9.3.5 Project 3: Medical Insurance Cost Prediction

Objective: Predict insurance charges based on customer characteristics such as age, BMI, smoking status, and region.

Dataset: `insurance.csv` **Target:** `charges` (continuous value)

Features Used:

- Numerical: `age`, `bmi`, `children`
- Categorical: `sex`, `smoker`, `region` (encoded using `pd.get_dummies`)

Key Concepts:

- Scaling numerical features with `StandardScaler`
- Handling categorical variables using dummy encoding
- Model tuning using `GridSearchCV`

Final Performance (Tuned Random Forest):

- RMSE = 4076.57
- MAE = 2642.62
- $R^2 = 0.8742$

9.3.6 ML Concepts and Questions Covered

Supervised vs. Unsupervised Learning:

- Linear Regression → Supervised
- Decision Tree → Supervised
- Random Forest → Supervised
- Support Vector Machine (SVM) → Supervised
- KMeans → Unsupervised

Evaluation Metrics for Regression:

- **MAE (Mean Absolute Error):** Average of absolute differences.
- **RMSE (Root Mean Squared Error):** Penalizes large errors more heavily.
- R^2 **Score:** Proportion of variance explained by the model.

Other Key Concepts:

- `StandardScaler`: Scales features to zero mean and unit variance.
- `random_state=42`: Ensures reproducibility.
- `fit()` vs `transform()`:
  - `fit()`: Learns scaling parameters (on training data only).
  - `transform()`: Applies learned parameters (on both train and test sets).
- `GridSearchCV`: Automates hyperparameter search with cross-validation.
- **Decision Tree Overfitting:** Deep trees may memorize training data.
- **Random Forest:** Ensemble of decision trees to reduce variance.
- **CNN vs Traditional ML:** CNNs for image data, ML models for structured/tabular data.

9.3.7 Conclusion

These Level 2 projects provided hands-on experience with end-to-end machine learning pipelines for real-world regression tasks. Key takeaways:

- Feature engineering and data preprocessing.
- Training and evaluating multiple ML models.
- Interpreting model performance using key metrics.
- Performing hyperparameter tuning with `GridSearchCV`.

These projects form a strong foundation for more advanced work in Levels 3 to 5.

9.4 Level 3: Classification on Tabular Data

9.4.1 Overview

Level 3 focuses on solving classification problems using structured/tabular datasets. The primary goal is to develop, compare, and evaluate machine learning models through a well-defined pipeline involving preprocessing, model training, hyperparameter tuning, and performance evaluation.

9.4.2 Projects Completed

Project 1: Titanic Survival Prediction

- **Objective:** Predict whether a passenger survived the Titanic disaster.
- **Target Variable:** `Survived` (0 = No, 1 = Yes)
- **Best Model:** Random Forest Classifier (Accuracy $\approx$ 82.68%)
- **Key Techniques:**
  - Feature engineering: extracting `Title` from passenger names.
  - One-hot encoding for categorical variables.
  - Feature scaling with `StandardScaler`.
  - Ensemble methods including soft and hard voting classifiers.

Project 2: Heart Disease Prediction

- **Objective:** Predict the presence of heart disease from patient medical records.
- **Target Variable:** `target` (1 = Disease Present, 0 = Absent)
- **Best Model:** Random Forest (Accuracy = 87.5%)
- **Key Techniques:**
  - Feature correlation analysis using heatmaps.
  - Standardizing numerical features with `StandardScaler`.
  - Ensemble methods and voting classifiers for improved accuracy.
  - Evaluation using accuracy, precision, recall, F1-score, and confusion matrix.

Project 3: Loan Approval Prediction

- **Objective:** Predict loan approval status based on applicant details.
- **Target Variable:** `Loan_Status` (Y = Approved, N = Rejected)
- **Best Model:** Voting Classifier (Soft Voting Accuracy $\approx$ 84.4%)
- **Key Techniques:**

- Missing value imputation using median (numerical) and mode (categorical).
- Label encoding for categorical variables.
- Feature scaling integrated into a pipeline.
- Model persistence using `joblib` for saving/loading.

9.4.3 Skills Strengthened

- End-to-end preprocessing for classification tasks.
- Model comparison and selection.
- Practical use of scaling, encoding, and imputation.
- Cross-validation and hyperparameter tuning for improved performance.

9.4.4 Conclusion

Level 3 projects provided in-depth experience in binary and multiclass classification, feature engineering, and ensemble methods. These projects form the bridge between basic ML tasks and advanced deep learning use cases in later levels.

9.5 Level 4: Image Classification with CNNs

9.5.1 Overview

Level 4 focuses on **Computer Vision** tasks using image datasets. The main goal is to build models that can accurately classify images using **Convolutional Neural Networks (CNNs)**. Projects at this stage include binary classification, multiclass classification, and small-scale vision tasks.

Learning Objectives:

- Preprocessing and handling image datasets.
- Understanding CNN architecture and intuition.
- Applying CNNs for both binary and multiclass classification.
- Using visual metrics (accuracy/loss curves, confusion matrices) for evaluation.

Due to CPU-only limitations, projects avoid heavy transfer learning or complex augmentation, but still aim to achieve strong results.

9.5.2 Project 1: Cat vs Dog Image Classification

Description

Binary classification task distinguishing between cats and dogs using CNN.

Dataset: Kaggle `Dogs vs. Cats`

- Training: 5604 images
- Validation: 2401 images
- Test: 2000 images (1000 cats, 1000 dogs)
- Images resized to 150×150 pixels.

Model Architecture

- `Conv2D` + `MaxPooling2D` (3 blocks)
- Flatten layer
- Dense (ReLU) + Dropout
- Output: Dense (Sigmoid)

Training Details

- Loss: `binary_crossentropy`
- Optimizer: Adam
- Epochs: 20
- Batch Size: 32

Performance

- Train Accuracy: 81.35%
- Validation Accuracy: 80.97%
- Test Accuracy: 81%

Observations

- Good baseline performance without transfer learning.
- Some errors due to noisy/ambiguous images.
- Accuracy plateaued around 81% without overfitting.

9.5.3 Project 2: CIFAR-10 Multiclass Classification

Description

Classifying images into 10 categories (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck).

Dataset: CIFAR-10

- Training: 50,000 images
- Test: 10,000 images
- Each image: 32×32 pixels, RGB

Model Architecture

- `Conv2D(32)` + `ReLU` + `MaxPooling2D`
- `Conv2D(64)` + `ReLU` + `MaxPooling2D`
- `Conv2D(128)` + `ReLU` + `MaxPooling2D`
- Flatten
- Dense(128) + ReLU + Dropout(0.5)
- Output: Dense(10) + Softmax

Training Details

- Loss: `categorical_crossentropy`
- Optimizer: Adam
- Epochs: 50
- Batch Size: 64
- Early Stopping + Basic Augmentation (flip, shift)

Performance

- Train Accuracy: 78.2%
- Validation Accuracy: 74.1%
- Test Accuracy: 73.7%

Observations

- Data augmentation helped reduce overfitting.
- Misclassifications mainly between visually similar classes.

9.5.4 Project 3: Flower Classification (5 Classes)

Description

Multiclass classification for flowers: daisy, dandelion, rose, sunflower, tulip.

Dataset: Kaggle `Flowers Recognition`

- ∼4,000 images split into train/validation/test (70/15/15)
- Images resized to 150×150 and normalized.

Model Architecture

- 3 `Conv2D` + `MaxPooling2D` blocks
- Flatten + Dense layers
- Output: Dense(5) + Softmax

Techniques Used

- `ImageDataGenerator` with rotation, zoom, and flip.
- EarlyStopping (patience = 5)
- Batch size: 32, Epochs: 50

Performance

- Train Accuracy: ∼98%
- Validation Accuracy: ∼92%
- Test Accuracy: Similar to validation.

Observations

- High accuracy on small dataset due to augmentation.
- Some confusion between visually similar flowers.

Limitations

Advanced transfer learning not used due to CPU-only constraints.

9.5.5 Conclusion

Level 4 provided practical experience in:

- Handling and preprocessing image data.
- Designing and training CNNs for binary/multiclass tasks.
- Applying augmentation and regularization to prevent overfitting.

These projects form the foundation for moving into advanced deep learning tasks in Level 5.

10 Level 5: Audio & Image Projects

10.1 Project 1 — Speech Emotion Recognition (CSV Features)

Objective

Classify emotions from speech using a CSV dataset containing pre-extracted audio features.

Dataset

- Rows: individual audio samples.
- Columns: engineered features (e.g., MFCC aggregates, chroma, spectral).
- Target: discrete emotion labels (happy, sad, angry, neutral, etc.).

Preprocessing

- Train/test split with fixed `random_state`.
- Label encoding for targets.
- Feature scaling via `StandardScaler`.
- Optional outlier detection and basic EDA.

Models

- Logistic Regression, Random Forest, SVM as baselines.
- Model selection via cross-validation.

Training Setup

- Metrics: Accuracy, Precision, Recall, F1-Score.
- Confusion Matrix for error analysis.
- Cross-validation for robust generalization estimates.

Results

- Consistent performance across folds; strong baseline for tabular audio emotion recognition.

Conclusion

- Feature scaling and validation strategies were key to stable results.
- Classical ML performs competitively on structured audio features.

Beginner Questions

- **Why scale features?** Many ML algorithms assume features are on similar scales.
- **Why cross-validation?** It provides a more reliable estimate of model performance than a single train/test split.

10.2 Project 2 — Speech Emotion Recognition (Actors' Audio)

Objective

Recognize emotions directly from raw audio recordings of actors.

Dataset

- Public speech dataset (e.g., RAVDESS) in WAV format.
- Emotions: neutral, happy, sad, angry, fearful, disgust, surprised.

Preprocessing

- Load and trim audio; resample to a consistent rate.
- Extract MFCCs (padded/truncated for fixed length).
- Train/test split ensuring speaker diversity.

Models

- 2D CNN over MFCC spectrograms.
- Hybrid CNN+RNN for capturing temporal patterns.

Training Setup

- Loss: Categorical Cross-Entropy; Optimizer: Adam.
- Dropout regularization; Early stopping on validation loss.
- Metrics: Accuracy, Precision, Recall, F1-Score; Confusion Matrix.

Results

- Good generalization; confusion between acoustically similar emotions.

Conclusion

- MFCCs + CNNs capture emotion-related audio patterns effectively.
- Data balancing and augmentation (time-stretch, pitch-shift) improve robustness.

Beginner Questions

- **Why MFCCs?** They mimic human auditory perception while reducing noise.
- **Why pad/truncate?** Deep models require fixed-length inputs for consistent training.

10.3 Project 3 — Image Classification with Transfer Learning (ResNet50)

Objective

Classify natural scene images into six categories using transfer learning.

Dataset

- Intel Image Classification dataset:
  - `seg_train/` (training set)
  - `seg_test/` (test set)
  - `seg_pred/` (prediction set)
- Classes: buildings, forest, glacier, mountain, sea, street.

Preprocessing

- Resize to 224×224; normalize pixel values.
- Augment training data (rotation, shift, shear, zoom, flip).
- Validation: 20% split from training set.

Model

- Base: ResNet50 pretrained on ImageNet (`include_top=False`).
- Frozen convolutional backbone.
- Head: GlobalAveragePooling2D $\rightarrow$ Dense(256, ReLU) $\rightarrow$ Dense(6, Softmax).

Training Setup

- Optimizer: Adam (lr $= 1 \times 10^{-4}$).
- Callbacks: EarlyStopping, ReduceLROnPlateau.
- Epochs: up to 20; Batch Size: 32.

Metrics & Results

- Test Accuracy: $\approx 62.9\%$; Test Loss: ≈ 0.93.
- Mild overfitting detected; validation plateaued at 60–62%.

Conclusion

- Transfer learning provided a strong starting point despite CPU-only limits.
- Further improvements possible via fine-tuning and heavier augmentation.

Beginner Questions

- **Why freeze backbone?** Stabilizes initial training using pretrained weights.
- **When unfreeze?** After top layers converge—use a smaller learning rate.
- **Why 224×224?** Matches ImageNet training size for ResNet50.

10.4 Project 4 — Fake News Detection

Objective

Classify news articles as **Fake** or **True**.

Dataset

- Two CSVs: `fake.csv` and `true.csv`.
- Labels: 0 = Fake, 1 = True.

Workflow

1. Merge datasets and assign labels.
2. EDA: class distribution, text length, frequent words.
3. Clean text: remove punctuation, stopwords, normalize.
4. Extract features via TF-IDF vectorization.
5. Train Logistic Regression model.
6. Evaluate: Accuracy, Precision, Recall, F1-score.

Performance

- Accuracy: 0.9854

Macro Avg: 0.99 Precision, 0.99 Recall, 0.99 F1-score
Weighted Avg: 0.99 Precision, 0.99 Recall, 0.99 F1-score

Table 10.1: Classification Report

Class	Precision	Recall	F1-score	Support
0 (Fake)	0.99	0.98	0.99	4733
1 (True)	0.98	0.99	0.98	4247
Overall	0.99	0.99	0.99	8980

Conclusion

The model is highly effective for fake news detection and demonstrates strong potential for real-world deployment.

Conclusion

This handbook documents a complete journey through practical machine learning, following a progressive roadmap from foundational regression and classification on tabular data to advanced projects in computer vision and audio processing.

Throughout the five levels, each project reinforced not only the technical skills of model building, evaluation, and tuning, but also the broader competencies of problem-solving, experimentation, and critical analysis of results. By structuring the learning path in stages, the progression allowed for:

- Mastery of fundamental preprocessing, feature engineering, and evaluation techniques.
- Gradual introduction to more complex data types, from structured tables to unstructured images and audio.
- Exposure to a variety of algorithms — linear models, tree-based methods, ensemble techniques, convolutional networks, recurrent networks, and transfer learning.
- Development of reproducible workflows using Python, popular ML libraries, and best practices in model selection.

Key takeaways from this journey include:

1. The importance of understanding the problem and the dataset before choosing a model.
2. The role of preprocessing and feature engineering in boosting model performance.
3. The need for evaluation beyond accuracy — using metrics suited to the task and dataset.
4. Awareness of hardware constraints and how to adapt approaches accordingly.
5. Continuous iteration: refining features, tuning hyperparameters, and testing new architectures.

By completing this roadmap, the foundation is set for tackling more complex, large-scale, and domain-specific machine learning challenges. The skills gained here can be applied to research, industry projects, or competitive machine learning, and serve as a launchpad for lifelong learning in artificial intelligence.

Next Steps:

- Explore larger datasets and advanced architectures with GPU/TPU acceleration.
- Engage in Kaggle competitions or open-source contributions.

- Dive deeper into specialized subfields such as natural language processing, reinforcement learning, or generative AI.
- Keep refining documentation and reproducibility — essential skills for both research and industry.

This handbook is not the end of the journey, but the beginning of a deeper engagement with the exciting and ever-evolving field of machine learning.