Below is a **reduced set of five main categories** (instead of eight) and a **generic three-tiered approach** that can be applied to the problems within each category. The same *basic → traditional ML → advanced AI* strategy is adaptable to nearly all the tasks in that category. Each category includes a quick listing of relevant tasks (by number and name), followed by a **generic** set of three possible solutions, along with notes on inputs/outputs, advantages, architecture, and domain constraints.

Category 1. NLP & Educational Tools

Tasks

- **(1) Language Translation App**
- **(3) Basic Chatbot for FAQs**
- **(6) Automated Text Summarization**
- **(7) Simple Optical Character Recognition (OCR)**
- **(9) Automatic Handwriting Recognition**
- **(15) AI-based Spell Checker**
- **(17) Simple Email Auto-reply**
- **(18) Predictive Text Typing**
- **(19) Simple Handwritten Digit Recognition**
- **(20) Simple Language Understanding Chatbot**
- **(24) Simple Emoticon Recognition in Text**
- **(25) Simple Document Summarization Tool**
- **(27) Basic Time Management Assistant**
- **(28) Simple Job Application Screening**
- **(30) AI-powered Language Translation Keyboard**
- **(35) AI for Simple Sign Language Recognition**
- **(37) AI for Simple Barcode Scanner**
- **(50) Creating subjects for FIA exams using AI**
- **(51) Generating multiple choice problems on a specific topic, having as input a pdf file**
- **(52) Creating a list of topics for the exam, having as input the course notes**
- **(53) Recognizing hand-written exam papers with AI**
- *(Also includes tasks that mix text, scanning, or educational screening.)*

Generic 3-Tier Solutions

1. **Heuristic / Rule-based Approach**
    - **Idea**: Use simple dictionaries, regular expressions, or pattern matching.
    - **Input/Output**:
        - *Input*: Short text, PDF files, or typed queries.
        - *Output*: Identified keywords, basic translations, or direct text replies.
    - **Advantages**:
        - Easy to implement, minimal data requirements.
        - Highly interpretable and fast to run on basic hardware.
    - **Architecture**:
        1. Ingest text input
        2. Apply rule-based or dictionary lookups
        3. Return result (translation, classification, or short answer)
    - **Domain Constraints**:
        - Limited scalability and accuracy.
        - Doesn't handle ambiguous or complex sentences well.
2. **Traditional Machine Learning**
    - **Idea**: Use classical NLP pipelines (e.g., bag-of-words, TF-IDF, SVM, or random forest) for classification or text generation tasks. For OCR/handwriting recognition, use standard image processing + an SVM/MLP for character classification.
    - **Input/Output**:
        - *Input*: Labeled text data, possibly annotated images (for OCR tasks).
        - *Output*: Predicted category, recognized text, summarized text, or next-word suggestions.
    - **Advantages**:
        - More robust than rule-based; can handle moderate complexity.
        - Generally requires less data and computational power than deep learning.
    - **Architecture**:
        1. Data pre-processing (tokenization, cleaning, feature extraction)
        2. ML classification/regression model (e.g., SVM, logistic regression)
        3. Output text or predictions
    - **Domain Constraints**:
        - Requires representative training data for best results.
        - May struggle with highly nuanced text or messy handwriting without specialized feature engineering.
3. **Advanced / Deep Learning**
    - **Idea**: Use neural networks (CNNs for OCR, LSTM/Transformers for language tasks) for complex classification, translation, summarization, or handwriting recognition.
    - **Input/Output**:
        - *Input*: Large corpora of text or large sets of labeled images/scanned documents.
        - *Output*: High-quality translations, accurate text classification, or recognized handwriting at scale.
    - **Advantages**:
        - Superior handling of complex linguistic structures (Transformers, GPT-based).
        - End-to-end training often yields higher accuracy.
    - **Architecture**:
        1. Neural network (RNN/LSTM/Transformers for text, CNN for OCR)
        2. Attention mechanisms for summarization or translation
        3. Output: text predictions, translations, or recognized text
    - **Domain Constraints**:

- Requires large training datasets.
- Computationally expensive (GPU/TPU).
- Proper hyperparameter tuning and domain adaptation are often needed.

Category 2. Computer Vision & Multimedia Analysis

Tasks

- **(2) Image Recognition for Dog Breeds**
- **(8) Basic Emotion Recognition**
- **(10) Smartphone Camera Object Recognition**
- **(13) Simple Object Counting in Images**
- **(16) Plant Disease Identification**
- **(22) Basic Handwritten Equation Solver**
- **(23) AI for Parking Space Detection**
- **(26) AI for Noise Classification in Environmental Sounds** *(audio, but included in "multimedia")*
- **(29) Basic Facial Expression Emoji Generator**
- **(31) Simple Facial Recognition Attendance System**
- **(32) AI for Plant Identification**
- **(33) Basic Hand Gesture Control for Smart Devices**
- **(34) AI for Social Distancing Monitoring**
- **(36) Basic Traffic Sign Recognition**
- **(38) Basic Face Recognition for Photo Album Organization**

Generic 3-Tier Solutions

1. **Heuristic / Traditional Computer Vision**

    - **Idea**: Rely on feature extraction (e.g., edges, color histograms) + rule-based or classical CV algorithms. Possibly use thresholding for object counting or detection.
    - **Input/Output**:
        - *Input*: Single images, video frames, or audio signals (for noise classification).
        - *Output*: Simple detection of shapes, categories, or approximate counts.
    - **Advantages**:
        - Low computational overhead.
        - Understandable and interpretable.
    - **Architecture**:
        1. Image pre-processing (grayscale, edge detection)
        2. Feature extraction (SIFT, SURF, or color histograms)
        3. Simple rule-based or distance-based classification
    - **Domain Constraints**:
        - Struggles with complex variations (lighting, angles).
        - Not ideal for large-scale variability in images or audio signals.

2. **Traditional Machine Learning (Classical ML + Feature Vectors)**

    - **Idea**: Use an SVM, decision tree, or random forest on top of extracted features (HOG, LBP, MFCC for audio).
    - **Input/Output**:
        - *Input*: Pre-labeled images or audio clips with extracted feature vectors.
        - *Output*: Predicted label (dog breed, object type), or numeric value (object count).
    - **Advantages**:
        - More robust than heuristic approaches for moderate complexities.
        - Less data-hungry than deep learning.
    - **Architecture**:
        1. Data collection & labeling
        2. Feature extraction (e.g., HOG for images, MFCC for audio)
        3. ML model for classification or regression
    - **Domain Constraints**:
        - Requires careful feature engineering.
        - Limited performance if the environment changes drastically (e.g., new lighting).

3. **Deep Learning (CNNs / Transfer Learning)**

    - **Idea**: Use convolutional neural networks (CNNs) or pre-trained models (e.g., ResNet, VGG, MobileNet) and fine-tune them on domain-specific data. For audio, use spectrogram-based CNNs.
    - **Input/Output**:
        - *Input*: Raw images/audio frames.
        - *Output*: Detailed classification (breed, emotion, disease, traffic sign), bounding boxes, or counts.
    - **Advantages**:
        - High accuracy on complex tasks.
        - Automatically learns features, reducing manual engineering.
    - **Architecture**:
        1. Preprocessing (resize images, convert audio to spectrograms)
        2. CNN with several layers or a pre-trained model for feature extraction
        3. Classification/regression layer for final output
    - **Domain Constraints**:
        - Requires large labeled datasets.
        - High computational cost (GPUs).
        - Overfitting risk if training data is limited.

Category 3. Predictive Analytics & Forecasting

Tasks

- **(4) Predicting Weather Conditions**
- **(14) Weather Forecast for Outdoor Activities**
- **(21) AI-based Plant Watering Reminder**
- **(45) Giving loans in banks** (credit risk assessment)
- **(49) Predicting the price of a house based on textual description**

Generic 3-Tier Solutions

1. **Heuristic / Simple Statistical Methods**

    - **Idea**: Moving averages, simple regression, or threshold-based triggers.
    - **Input/Output**:
        - *Input*: Historical data (weather, loan defaults, house listings).
        - *Output*: Basic forecast (next day's temperature, yes/no precipitation, loan approval decision).
    - **Advantages**:
        - Highly interpretable.
        - Minimal data requirements and quick deployment.
    - **Architecture**:
        1. Collect historical data
        2. Compute average or regression line
        3. Produce short-term predictions
    - **Domain Constraints**:
        - Quickly becomes inaccurate with complex patterns.
        - Not adaptive to abrupt changes.

2. **Machine Learning Models (Regression / Classification)**

    - **Idea**: Random Forest, Gradient Boosting, or SVM for numeric or binary outcome predictions (e.g., temperature or default vs. no-default).
    - **Input/Output**:
        - *Input*: Features from historical data, e.g., meteorological measurements, financial metrics.
        - *Output*: Numerical forecasts (temperature, price) or probability (likelihood of loan default).
    - **Advantages**:
        - Balances complexity and interpretability (with partial dependence plots).
        - Generally higher accuracy than simple heuristics.
    - **Architecture**:
        1. Data pre-processing & feature engineering (e.g., date/time features, exogenous variables)
        2. ML model training (regression or classification)
        3. Output predicted values or probabilities
    - **Domain Constraints**:
        - Needs consistent, good-quality historical data.
        - May overfit if hyperparameters are not tuned.

3. **Advanced / Deep Learning (Time Series + Large Data)**

    - **Idea**: LSTM, Transformer-based time series, or deep neural networks.
    - **Input/Output**:
        - *Input*: Large multi-year datasets, possibly external data (satellite images, economic indicators).
        - *Output*: More refined multi-step forecasts, confidence intervals, or classification.
    - **Advantages**:
        - Can capture complex seasonality and patterns.
        - Potentially very accurate with sufficient data.
    - **Architecture**:
        1. Sequence modeling (LSTM, 1D CNN, or temporal attention)
        2. Dense layers for final regression/classification
        3. Optionally produce multi-day or multi-output predictions
    - **Domain Constraints**:
        - High computational needs (training on GPUs).
        - Requires large, clean datasets.
        - Model interpretability can be challenging.

Category 4. AI in Control & Interactive Systems

Tasks

- **(5) Gesture-based Game Control**
- **(11) Simple Puzzle Solver**
- **(12) Automated Home Lighting System**
- **(41) Automation of warehouse workers**
- **(43) Moving a non-playable character (NPC)**
- **(44) Detecting the game illustrated in a video**
- **(46) Creating a player in the game Go**
- **(47) Programming a bot in an FPS game**
- **(48) A humanoid robot that learns to play tennis with AI**

(Note that some tasks like #33 "Basic Hand Gesture Control for Smart Devices" were grouped under Computer Vision, but the concepts overlap.)

Generic 3-Tier Solutions

1. **Heuristic / Scripted Rules**

    - **Idea**: Hard-code sequences or rules (e.g., if NPC sees an obstacle, turn left).
    - **Input/Output**:
        - *Input*: Sensor data (camera feed, game state, user gestures).
        - *Output*: Specific actions (move left, turn on lights, pick up item).
    - **Advantages**:
        - Predictable behavior, straightforward debugging.
        - Minimal data needed.
    - **Architecture**:
        1. Sensor or environment reading
        2. Rule-based logic (e.g., finite state machines)
        3. Action command (turn on light, move to tile)
    - **Domain Constraints**:
        - Cannot adapt well to new or unexpected scenarios.
        - Lacks intelligence once environment changes.

2. **Traditional ML / Planning**

    - **Idea**: For robots or NPCs, use pathfinding (A*), decision trees, or Q-learning with discrete states.
    - **Input/Output**:
        - *Input*: Simplified world representation, labeled training data for gestures or puzzle states.
        - *Output*: Next best move or predicted action.
    - **Advantages**:
        - More robust than purely scripted; can learn from small datasets.
        - Good for simpler puzzles (Sudoku) or well-defined tasks.
    - **Architecture**:
        1. State modeling (graph of possible moves, sensor readings)
        2. Planning or ML-based policy (Q-learning, BFS, decision trees)
        3. Action output (e.g., path step, next puzzle move)
    - **Domain Constraints**:
        - Might still be limited to known states or pre-labeled data.
        - Tuning hyperparameters for specific tasks can be time-consuming.

3. **Advanced / Deep Reinforcement Learning**

    - **Idea**: Use DQN, PPO, or policy gradients to learn optimal policies in dynamic environments (e.g., controlling a robot playing tennis, advanced game bots).
    - **Input/Output**:
        - *Input*: Raw environment data (visual input from cameras or game screens, sensor streams).
        - *Output*: Continuous or discrete actions (move joystick, rotate servo).
    - **Advantages**:
        - Can discover complex strategies over time.
        - Adapts to changing conditions if trained or fine-tuned properly.
    - **Architecture**:
        1. Neural network processes states/frames
        2. Reinforcement learning algorithm updates a policy or Q-function
        3. Action selection based on policy output
    - **Domain Constraints**:
        - Requires substantial training episodes and possibly simulation.
        - Risk of overfitting or convergence on suboptimal policies if not carefully managed.

Category 5. Optimization & Resource Allocation

Tasks

- **(39) Aircraft parking at airports**
- **(40) Packing purchases**
- **(42) Determining the size of the aircraft fleet to serve a set of flights**

Generic 3-Tier Solutions

1. **Heuristic / Rule-based Optimization**

    - **Idea**: Use a set of if-then rules, “first-fit” or “greedy” strategies to allocate resources quickly.
    - **Input/Output**:
        - *Input*: Constraints (e.g., flight schedules, packing sizes).
        - *Output*: Assignments (which plane goes to which gate, how to pack items).
    - **Advantages**:
        - Fast to implement, good for smaller or simpler cases.
        - Easy to understand.
    - **Architecture**:
        1. Gather constraints (size, weight, schedules)
        2. Apply rule-based or greedy algorithm (fill from smallest to largest, etc.)
        3. Produce assignment or packing plan
    - **Domain Constraints**:
        - Not guaranteed to find the optimal solution.
        - Might fail in complex real-world scenarios.

2. **Classical Optimization Methods**

    - **Idea**: Formulate as an Integer Linear Program (ILP) or Mixed Integer Programming (MIP). Use solvers like CPLEX or Gurobi.
    - **Input/Output**:
        - *Input*: Mathematical formulation (objective function, constraints).

        - *Output*: Optimal or near-optimal solution (parking allocation, packing arrangement).
    - **Advantages**:
        - Well-studied methods with proven performance for small-medium scale.
        - Many optimization libraries readily available.
    - **Architecture**:
        1. Define decision variables (e.g., x_ij = 1 if plane i uses gate j)
        2. Define constraints (capacity, scheduling)
        3. Solve using a standard optimizer to get best solution
    - **Domain Constraints**:
        - May become computationally expensive at large scale.
        - Requires careful constraint definitions.

3. **Metaheuristics / AI-based Search**

    - **Idea**: Use evolutionary algorithms, simulated annealing, or a custom heuristic guided by ML.
    - **Input/Output**:
        - *Input*: Same data as classical optimization (flight times, item dimensions).
        - *Output*: Near-optimal or best-known allocation in a timeframe.
    - **Advantages**:
        - Scalable to large, complex problems.
        - Flexible about constraints and objective changes.
    - **Architecture**:
        1. Initialize random solutions
        2. Iteratively improve solutions (mutation, crossover, acceptance criteria)
        3. Return best solution found under resource/time constraints
    - **Domain Constraints**:
        - No guaranteed global optimum.
        - Performance depends on parameter tuning (mutation rates, population size).

How to Use These Three Solutions

- Each **category** has different **domains** (NLP vs. computer vision vs. optimization), but the **3-tier approach** remains the same:

    1. Start with **basic / heuristic** methods.
    2. Move to **classical machine learning or formal optimization**.
    3. Adopt **advanced approaches (deep learning, advanced solvers, reinforcement learning)** if data, computational resources, and complexity demand it.

- **Input & Output**: Usually revolve around collecting domain-specific data (text, images, sensor data, scheduling constraints) and producing a classification, forecast, or plan.

- **Advantages of Proposed Solutions**:

    - Tiered approach allows projects to begin small and scale up.
    - Balances interpretability, data requirements, and accuracy.
    - Addresses constraints in each domain (e.g., limited data, time-critical tasks).

- **Architecture Considerations**:

    - Pipeline design (data ingestion → feature engineering → model/solver → output).
    - Potential for continuous deployment or online learning (especially in dynamic environments).

- **Domain Constraints**:

    - Data availability and quality.
    - Real-time or large-scale demands.
    - Ethical or regulatory constraints (especially for finance or sensitive image data).

This **generic framework** can be tailored to each specific problem in the list, ensuring that whether you are **translating text, recognizing objects, forecasting weather, automating robots**, or **optimizing resources**, you have a clear **progression from simple to advanced AI methods** that address **input/output requirements, domain constraints, and architectural considerations.**