This project employs custom recurrent neural networks (RNNs) to predict the nationality of a person based on their name’s character sequence. Techniques such as padding, class weighting, and F1-score evaluation are used to handle imbalanced data and improve model performance.

**Project: Nationality Prediction with RNNs**

**Part 1 – Data Loading & Preprocessing**

* **Dataset:** ~44,500 names from 41 different countries (from nam\_dict.txt).
* **Imbalance:** Dataset is highly imbalanced (e.g. China, U.S.A., Italy have far more names than others).
* **Character Set:** 57 unique characters (letters + diacritics + special characters).
* **Encoding:** Names were lowercased and converted to one-hot vectors (name2tensor).
* **Padding:** Since names have variable length, shorter names were left-padded with zeros to match the max length (22). This allowed batching into a (batch\_size, 22, 57) tensor.
* **Splits:** Data was divided into train (70%), validation (15%), and test (15%).

**Part 2 – Model Design**

* **Architecture:**
  + Custom **single-layer RNN** implemented in PyTorch (nn.RNN).
  + Input size = 57 (characters), hidden size = tunable, output size = 41 (countries).
  + Hidden state updated per character.
  + Final hidden state passed to fully connected layer (Linear) → LogSoftmax.
* **Equation:**
  + ht=tanh⁡(Whxt+Uhht−1+bh)h\_t = \tanh(W\_h x\_t + U\_h h\_{t-1} + b\_h)ht​=tanh(Wh​xt​+Uh​ht−1​+bh​)
  + yt=logSoftmax(Wyht+by)y\_t = \text{logSoftmax}(W\_y h\_t + b\_y)yt​=logSoftmax(Wy​ht​+by​)

**Part 3 – Training Setup**

* **Loss Function:** Negative Log-Likelihood Loss (NLLLoss).
* **Class Weights:** Applied inverse-frequency weighting to address class imbalance.
* **Optimizer:** Adam.
* **Gradient Clipping:** Clipped norm to 2 to avoid exploding gradients.
* **Metrics:** Used **F1-score** (macro) in addition to accuracy (more meaningful for imbalanced datasets).

**Part 4 – Training Process**

* **Batch Forward Propagation:** Initialized hidden states with zeros, ran full sequence, collected last output.
* **Validation:** Computed F1-score, precision, recall, and loss at intervals.
* **Monitoring:** Tracked training/validation loss & F1 in real-time with matplotlib.
* **Results:**
  + Model learned meaningful country-character patterns.
  + Validation F1-score improved steadily, though imbalance made some minority classes harder to predict.

**Part 5 – Key Insights**

* Padding enabled batch training (otherwise only one sample at a time).
* Weighting classes was essential; otherwise the model would trivially predict majority classes like "China" and still get deceptively high accuracy.
* F1-score gave a much clearer picture of model quality compared to raw accuracy.

**Interview Preparation: Key Concepts**

**Data & Preprocessing**

1. **Why one-hot encoding?**  
   It’s a simple, sparse representation where each character is uniquely identifiable. Unlike embeddings, this avoids introducing learned bias at the preprocessing stage.
2. **Why pad names to equal length?**  
   RNNs require consistent sequence lengths in a batch. Padding standardizes input sizes.
3. **What issues arise from class imbalance?**  
   The model may over-predict majority classes, giving misleadingly high accuracy but poor generalization.

**RNN Model & Training**

1. **Difference between RNN and CNN for text tasks?**
   * RNNs process sequentially, capturing order and dependencies.
   * CNNs capture local patterns but don’t inherently model sequence order.
2. **Why gradient clipping in RNNs?**  
   Prevents exploding gradients, common when backpropagating through long sequences.
3. **Why use LogSoftmax + NLLLoss?**  
   LogSoftmax ensures numerical stability, and NLLLoss matches classification over multiple classes.
4. **What is the vanishing gradient problem?**  
   Gradients shrink as they backpropagate through time, making it hard for vanilla RNNs to learn long-term dependencies. (Motivation for LSTMs/GRUs).

**Evaluation & Metrics**

1. **Why is F1-score more useful than accuracy here?**  
   Accuracy is dominated by majority classes. F1 balances precision and recall, ensuring minority classes are considered.
2. **What’s macro vs. weighted F1?**
   * Macro: unweighted average over all classes.
   * Weighted: accounts for class support. Macro is more appropriate here to evaluate minority classes.