○

# ML HACKATHON
# Analysis_Report: PESU_EC_030_032_035_045

**CONTRIBUTORS:**
Devanshi Shukla (PES2UG23AM030)
Dhriti Rai (PES2UG23AM032)
Farhaan Arshad (PES2UG23AM035)
Ishaan Tandon (PES2UG23AM045)

## 1) Key Observations

- **Hardest parts**
    - **State representation design.** Encoding a variable-length masked word into a fixed-size numeric vector required a careful choice.
    - **Reward shaping.** Getting a sparse but meaningful reward signal in Hangman was a little tough. We needed to guess correctly while also penalizing wasted guesses and repeats.
    - **Combining probabilistic model with RL.** Using the HMM to provide prior letter probabilities helped guide the agent's action preferences, but integrating those probabilities properly into the observation vector and into the mask logic required iteration.

- **Insights gained**
    - A simple char-level HMM over the corpus dataset gives a useful prior that directs early exploration toward letters likely given the pattern.
    - Small positive reward per correct letter and a larger terminal reward for finishing the word accelerates learning more than a purely terminal-only reward.
    - Masking invalid actions and using a target network stabilizes learning.
    - Curriculum or more corpus coverage would likely improve generalization.

---

○

# 2) Strategies (design choices & why)

## HMM design

- **Class & training:** We implemented a character-level HMM. Training is counting first-letter frequencies and letter-to-letter transitions (start_counts, trans_counts), with Laplace smoothing. Log-probabilities (log_start, log_trans) are used for numerical stability.

- **Scoring & letter distribution:** For a given pattern (masked word), letter_distribution(pattern, guessed, words):

    ○ Filters corpus words by pattern length and matching observed letters.

    ○ Scores each candidate word with logprob_word (sum of start + transition log-probs).

    ○ Converts log-probs to normalized weights and accumulates probabilities for letters at blank positions.

    ○ Zeros out probabilities for already-guessed letters and normalizes the final letter probability vector.

- A char-level HMM is compact, fast, and gives a statistically grounded prior for letter likelihoods conditioned on the masked pattern. It complements RL by providing informed priors early in training and for unseen states.

## RL (environment & agent)

- **Environment:** HangmanEnv models the game. step(letter):

    ○ If letter already guessed → reward = -2.

    ○ If letter in word → reveal positions, reward = +1.

    ○ If letter not in word → reward = -1 and increment wrong.

    ○ If the word becomes fully revealed → reward += 10 and done = True (win).

    ○ If wrong >= max_wrong → reward -= 5 and done = True (lose).

- **Observation / State (obs_to_vec)**

    ○ pattern encoded as one-hot-like per position: MAX_LEN * 27 (each position encoded across 27 dims — 26 letters + blank). This gives a positional encoding of which letter is present or blank.

    - guessed vector of length 26 (binary hot-vector for which letters have been guessed).
    - hmm_probs vector of length 26 (the HMM's letter probabilities for the current pattern).
    - lives normalized scalar (lives/6).
    - Total STATE_DIM = MAX_LEN * 27 + 26 + 26 + 1.
- **Action space:** 26 discrete actions (letters a–z). Actions masked via mask_actions(obs) which sets guessed letters' mask to 0. During learning the Q-values for masked actions are penalized by adding a large negative term (-1e6) so they are never chosen.
- **Agent architecture:** DQN (PyTorch) feed-forward network:
    - DQN(s_dim, a_dim, hidden=256) with layers:
        - Linear(s_dim → 256) → ReLU
        - Linear(256 → 128) → ReLU
        - Linear(128 → a_dim)
    - policy_net and target_net used, with MSE loss and Adam optimizer (lr 1e-4).
- **Replay & training details**
    - ReplayBuffer capacity = 50,000.
    - learn_start = 2000, learn_every = 4.
    - Batch size = 64.
    - Gamma = 0.99.

---

○

# 3) Exploration

- **Exponential decay:** Smoothly reduces randomness so the agent has broad experience early (important to populate replay buffer with varied transitions) and then focuses on exploiting learned Q-values.

- **Action masking:** Masking guessed letters both in the action-selection step and in the training target (big negative Q for masked actions) reduces wasted exploration and makes exploration more meaningful (explore untried letters rather than repeats).

---

# 4) Future Improvements

- **More careful reward shaping & ablation**
    - Experiment with differing magnitudes for per-letter reward vs terminal reward (e.g., smaller per-letter reward but larger terminal win reward), or use a shaped reward proportional to letters revealed at the guess.

- **Better language model**
    - Replace or augment HMM with a word-level language model or an n-gram / masked language model to get stronger prior distributions over candidate words.

- **State representation upgrades**
    - Use embeddings for letters/positions (learnable) rather than fixed 27-dim position one-hot, or use a small recurrent encoder (LSTM) over the pattern to compress variable-length pattern information more effectively.

- **Policy improvements**

    Try actor-critic (A2C) or PPO style policy gradient methods which sometimes handle sparse/episodic rewards better than DQN.

    - Alternatively, treat the problem as ranking candidate letters (use cross-entropy / supervised pretraining from corpus-derived heuristics).

- **Data & training**
    - Augment corpus (more words) and create curriculum training (start with short words, then gradually increase length).
    - Increase episodes and tune hyperparameters (batch, lr, hidden sizes).

- **Evaluation & metrics**

- 
    - Log and compare metrics: success rate, avg wrong guesses, avg repeated guesses, distribution of wins/loses by word length.
    - Evaluate generalization to out-of-corpus words and to different word lengths.

- **Safety & robustness**
    - Ensure masking is airtight; add checks so that the policy is never able to select masked actions at eval time.

- **Architecture tweaks**
    - Try deeper/wider nets, or small attention layers to focus on pattern positions most informative for current HMM probs.