Transferable Reinforcement Learning in Forex Trading

Cross-Currency Adaptation Techniques for EUR/USD and GBP/USD

Yavuz Hancer
Supervisor(s): Neil Yorke-Smith, Antonis Papapantoleon, Amin Kolarijani
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Yavuz Hancer
Final project course: CSE3000 Research Project
Thesis committee: Neil Yorke-Smith, Antonis Papapantoleon, Amin Sharifi Kolarijani, Julia Olkhovskaya

Abstract

This paper investigates the effectiveness of transfer learning techniques for accelerating the training of deep reinforcement learning (RL) agents in the foreign exchange (Forex) market. Specifically, the transfer of policies learned on the EUR/USD currency pair to the GBP/USD pair is the focus. Four transfer learning approaches are systematically compared: zero-shot transfer, full fine-tuning, partial fine-tuning, and reward-function transfer. A modular pipeline was developed, incorporating sinusoidal and trend/momentum-based market features, stepwise agent-specific metrics, and deep Q-network (DQN) architectures from the Stable Baselines 3 framework. Agent performance is evaluated through cumulative reward and Sharpe ratio metrics.

Experimental results demonstrate that partial fine-tuning accelerates initial learning by preserving generic market features acquired from the EUR/USD pair. However, our results indicate that directly training on the target currency pair yields superior ultimate performance, highlighting the nuanced limitations and potential benefits of cross-currency transfer learning in algorithmic trading.

1 Introduction

The Foreign Exchange (Forex) market is the world's largest and most liquid financial marketplace, with daily volumes exceeding \$6 trillion [2]. Its price series exhibit high noise, nonstationarity (statistical properties changing over time), and frequent regime shifts (trending vs. ranging periods), which make automated policy learning particularly challenging. Traditional supervised learning methods train on historical data but can not adapt online to new market dynamics.

Reinforcement learning (RL) offers a different paradigm: an agent interacts sequentially with an environment modeled as a Markov Decision Process (MDP) [15], observes state s_t, takes action a_t, receives reward r_t, and updates its policy to maximize cumulative return $\sum_t r_t$. Key deep RL algorithms include:

- **Deep Q-Networks (DQN)**: approximate the action-value function $Q(s, a)$ with a neural network and update via temporal-difference learning [10].

Applied to single-instrument trading, these methods have shown promise but often require thousands of episodes to converge and generalize poorly when deployed on a different currency pair or under novel volatility regimes.

1.1 Transfer Learning in Reinforcement Learning

Transfer learning in RL aims to reuse knowledge (network weights or learned representations) from a *source task* to accelerate learning in a *target task*. In robotics and gameplaying, transfer reduced sample complexity and improved performance across related tasks. In financial markets, a handful of studies explored cross-time or cross-asset transfer, but lacked a systematic comparison of different transfer modalities.

The most widely traded pairs (EUR/USD and GBP/USD) are the focus, and it was hypothesized that a DQN policy pretrained on EUR/USD could provide a strong initialization for GBP/USD. Four transfer techniques were examined:

1. **Zero-Shot Transfer**: directly applying the pretrained policy without further training.
2. **Full Fine-Tuning**: continuing gradient updates on the target data for all network layers.
3. **Partial Fine-Tuning**: freezing early layers and fine-tuning only deeper layers.
4. **Reward-Function Transfer**: retraining with a new custom reward (risk-adjusted return) while reusing the pretrained policy.

These were compared against a *From-Scratch* baseline that was trained only on GBP/USD.

To support reproducibility, a modular pipeline is to built that:

- engineered market features (sin-cos seasonal encodings, classic trend/momentum indicators) and agent features (cash-exposure, trade duration);
- instantiated discrete-action DQN agents with Stable Baseline 3;
- ran different transfer learning techniques on the same model on a different currency;
- logged cumulative reward, Sharpe ratio, and training time.

1.2 Research Question

This study seeks to answer the following primary question:

Primary Question:

How effectively can transfer learning techniques reduce training time and improve the performance of RL agents when applied to new currency pairs?

To break this down, these sub-questions are asked:

- **Sub-question 1:** How do these strategies compare in terms of final trading performance, as measured by cumulative reward and Sharpe ratio, relative to training an agent from scratch?
- **Sub-question 2:** What are the trade-offs between runtime efficiency and policy effectiveness when deploying transfer learning techniques in forex trading environments?

2 Background

2.1 Reinforcement Learning

Reinforcement Learning (RL) views trading as a sequence of decisions made by an *agent* interacting with a market *environment*. This is modeled as a Markov Decision Process (MDP) [16], defined by:

$$(\mathcal{S}, \mathcal{A}, P, R, \gamma),$$

where:

- $\mathcal{S}$ is the set of possible market states (recent price vectors, technical indicators, current position);
- $\mathcal{A}$ is the set of trading actions (`long`, `short`, `hold`);
- $P(s' \mid s, a)$ governs how the market transitions from state s to s' when action a is taken;
- $R(s, a)$ is the immediate reward—typically the profit or loss (P&L) resulting from that action;
- $\gamma \in [0, 1)$ is the discount factor, balancing immediate versus future P&L.

The agent's objective is to learn a policy $\pi(a \mid s)$ that maximizes the expected sum of discounted rewards:

$$J(\pi) \;=\; E_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \, r_t\Big],$$

where $r_t = R(s_t, a_t)$ is the P&L at step t. Rather than writing out full Bellman equations here, it is noted that the central insight is: *good* actions are those that not only yield immediate profit but also lead to future states where further profits are likely.

2.2 Deep-Q Networks

Deep Q-Networks (DQN) extend classic Q-learning to high-dimensional inputs (price time-series) by approximating the action-value function $Q(s, a)$ with a neural network $Q(s, a; \theta)$ [18]. Rather than maintaining a table of values, the agent learns parameters θ so that $Q(s, a; \theta)$ predicts the expected future P&L when taking action a in state s.

A key innovation in DQN is the use of a target network and experience replay:

- Target network: a separate copy $Q(s, a; \theta^-)$ of the Q-network whose weights θ^- are held fixed for several updates, stabilizing learning.
- Experience replay buffer $\mathcal{D}$: stores past transitions (s, a, r, s'). During training, minibatches are sampled uniformly from $\mathcal{D}$ to break temporal correlations and improve data efficiency [9].

At each training step, the DQN minimizes the temporal-difference (TD) error δ:

$$\delta = r + \gamma \max_{a'} Q(s', a'; \theta^-) \;-\; Q(s, a; \theta),$$

and updates its weights via stochastic gradient descent:

$$\theta \;\leftarrow\; \theta + \alpha \, \delta \, \nabla_\theta Q(s, a; \theta).$$

For exploration, DQN agents commonly use an ε-greedy policy: with probability ε select a random action to discover new market behaviors, and with probability $1 - \varepsilon$ select $\arg\max_a Q(s, a; \theta)$ to exploit current knowledge. In non-stationary markets like Forex, annealing ε over time or adopting Boltzmann (softmax) exploration, sampling actions with probability proportional to $\exp(Q(s, a)/\tau)$, can yield more robust performance.

By combining neural-network function approximation with these stability mechanisms, DQN provides a practical framework for learning trading policies directly from historical price data, balancing immediate profit (P&L) with the exploration of profitable patterns.

2.3 Transfer Techniques

In the remainder of this subsection, four principal paradigms for transferring across currency pairs are therefore reviewed. These principles range from direct, zero-shot deployment of a pretrained policy to various fine-tuning strategies that selectively adapt different parts of the network to new market dynamics.

1. **Zero-Shot Transfer.** Evaluate the source policy $\pi_S(s) = \arg\max_a Q_{\theta_S}(s, a)$ directly on M_T without further learning:

   $$\pi_T^{\mathrm{ZS}}(s) = \pi_S(s).$$

   Zero-shot transfer is included as the strictest test of cross-currency generalization: can a policy trained on EUR/USD be deployed immediately on GBP/USD without any further adjustment? By evaluating $\pi_S(s)$ directly in the target environment, it is probed whether the low-level feature representations and action preferences learned on one major FX pair capture universal market structure that transfers "out of the box." A near-zero or negative Sharpe ratio under zero-shot highlights domain mismatch and motivates more sophisticated adaptation, while any positive performance would signal surprisingly robust feature reuse.

2. **Full Fine-Tuning.** Initialize $\theta_T^{(0)} = \theta_S$ and continue updating all parameters on target samples:

   $$\theta_T^{(k+1)} = \theta_T^{(k)} - \alpha \nabla_\theta L_{M_T}(\theta_T^{(k)}).$$

   It simply continues gradient descent on all layers using GBP/USD data. This mode is chosen to quantify how much faster (if at all) it converges to a higher performing policy compared to training from scratch, and whether catastrophic forgetting[5] (the loss of previously learned representations when adapting to new data) of generic features undermines adaptation. In particular, full fine-tuning tests whether the inductive biases encoded in the entire network give us a head start, or whether they instead block learning by locking the agent into source specific characteristics.

3. **Partial Fine-Tuning.** Decompose $\theta = (\phi, \psi)$ into feature-extractor weights ϕ and decision-layer weights ψ, then freeze ϕ:

   $$\phi_T = \phi_S,$$
   $$\psi_T^{(k+1)} = \psi_T^{(k)} - \alpha \nabla_\psi L_{M_T}(\phi_S, \psi_T^{(k)}).$$

   Partial fine-tuning strikes a middle ground: freeze the lower "feature-extractor" layers (seasonal encodings, feature filters) learned on EUR/USD and retrain only the higher decision layers on GBP/USD. This technique is included to isolate the impact of preserving generic market representations while allowing adaptation where the target's volatility, liquidity, or price dynamics diverge. The goal is two-fold: reduce sample complexity and stabilize early learning by retaining broadly applicable encodings, yet still tailor the agent's positions and timing to the characteristics of the new currency pair.

4. **Reward-Function Transfer.** Reuse θ_S under a modified reward objective R'_T (e.g., risk-adjusted return) without changing representations:

$$\theta_T^{(0)} = \theta_S, \quad \theta_T \text{ learns under } R'_T.$$

   This shifts optimization from raw PL to metrics like $r'_t = \Delta\text{Equity}/\sigma(\Delta\text{Equity})$, testing robustness of the pretrained Q-mapping. This approach tests whether transferring incentives, the shape and scale of the reward signal that guided robust, risk-aware behavior on the source, can accelerate convergence even if the policy's weights are suboptimal. By reusing the pretrained Q-mapping under a new objective, it is sought to determine if aligning target-domain exploration with a calibrated Sharpe-ratio criterion reduces the need for per-pair hyperparameter tuning of the reward function.

5. **Training From Scratch.** As a baseline, initialize randomly $\theta_T^{(0)} \sim \mathcal{N}(0, I)$ and train solely on M_T:

$$\theta_T^{(k+1)} = \theta_T^{(k)} - \alpha \nabla_\theta L_{M_T}(\theta_T^{(k)}).$$

   This requires maximal sample complexity but involves no mismatch risk from source initialization. As a complementary baseline, this identical DQN architecture is trained from random initialization solely on GBP/USD. This "no-transfer" mode quantifies the full sample complexity and ultimate performance achievable without any source knowledge.

In our experiments, each mode: (i) cumulative reward $\sum_t r_t$, (ii) Sharpe ratio $E[r]/\text{std}(r)$, and (iii) training time were reported, thereby quantifying trade-offs between sample efficiency, performance, and computational cost.

3 Methods

In this section, the datasets, feature pipelines, trading environment, agent configuration, transfer methodologies, and evaluation protocol used in the experiments are described in detail.

3.1 Data Acquisition and Preprocessing

Fifteen-minute OHLCV data for two currency pairs were obtained from the Dukascopy repository.

- **Source Domain (EUR/USD):** Data spanning January 2, 2022 at 22:00 through June 30, 2023 at 20:45 (approximately 37 440 records) were collected.
- **Target Domain (GBP/USD):** Data spanning January 1, 2023 at 23:00 through December 29, 2023 at 21:45 (approximately 24 888 records) were collected.

Timestamps were aligned and missing intervals were forward-filled. Each series was then partitioned chronologically into a training set (70%) and an evaluation set (30%) to prevent look-ahead bias [3].

The EUR/USD pretraining period was intentionally capped at June 30, 2023 to enforce a strict causality boundary: all knowledge transferred to GBP/USD originates from data that entirely precedes the target training window. This cutoff also coincides with a marked volatility regime shift in mid-2023, allowing the assessment of transfer into a distinct market environment. Although more extensive history was available for both pairs, this span was chosen to simulate realistic constraints on pretraining data and to ensure that no future information from the target domain could influence the source model.

3.2 Feature Engineering

Two complementary feature sets were generated to characterize both market dynamics and the agent's own state at each decision point:

- **Market Features:**
  - *Seasonal Encodings*: Sine and cosine transforms of the intra-day (24 h) and intra-week (7 d) cycles were applied to capture periodic patterns in liquidity and volatility [8].
  - *Trend and Volatility Indicators*:
    - Average True Range (ATR) over a 14-bar window, quantifying recent volatility [19].
    - Moving Average Convergence Divergence (MACD) calculated with 12/26/9 settings, capturing trend momentum and signal crossovers [1].
    - Relative Strength Index (RSI) over 14 periods, measuring overbought/oversold conditions [19].
  - *Lagged Observations*: A 20-bar rolling look-back was computed over all raw and engineered columns, providing the agent with delayed signal histories and enabling the network to infer short-term temporal dependencies.

  Feature Selection: ATR, MACD, and RSI were selected due to their widespread empirical success in the FX markets [11], while sinusoidal time encodings follow recent reinforcement learning work on temporal awareness [17].

- **Agent-State Features:**
  Stepwise metrics were maintained to reflect the portfolio's instantaneous condition:
  - *Cash Exposure*: The fraction of total equity held in cash (uninvested), updated after each action to inform the agent about remaining buying power.

All feature transformations were applied in sequence and then standardized (zero mean, unit variance) based on the training partition. This ensured that inputs to the learning algorithm remained numerically stable and free of scale disparities across different feature types.

3.3 Trading Environment

The trading problem was modeled as a finite-horizon decision process in which an agent interacts with the market at 15-minute intervals [4]. At each step:

- **State ($\mathcal{S}$):** A feature vector comprising both market indicators (momentum, volatility, seasonal encodings, lagged history) and the agent's current uninvested capital (cash exposure). All features were normalized (zero mean, unit variance) based on the training data.

- **Actions ($\mathcal{A}$):** {-1, 0, +1}, corresponding to entering a short position, holding the current position, or entering a long position. At any time, the full available capital is deployed according to the chosen action (no partial positions or leverage).
- **Reward (R):** Under the default scheme, the instantaneous reward is the profit or loss realized over the 15-minute bar, computed as the action multiplied by the percentage price change of the base currency. For the risk-adjusted variant, this reward is divided by a short-term measure of volatility (an exponential moving standard deviation over recent bars) to encourage more stable performance.
- **Episode Horizon:** An episode begins at the first evaluation timestamp and proceeds step by step until the last available bar. No intermediate resets occur; the agent experiences the entire out-of-sample period in one continuous run.
- **Transaction Costs and Constraints:** Transaction costs were set to zero to focus purely on strategy efficacy. No margin or leverage was allowed, positions are always sized to consume at most the available cash.
- **Initial Conditions:** Each episode starts with \$10 000 in cash and zero open positions.

This environment design ensures a clear mapping between the agent's decisions and realized trading performance, while the optional risk-adjusted reward provides an alternative objective that explicitly penalizes volatility. By fixing transaction costs at zero and disallowing leverage, the results reflect the pure impact of the learned policy on profitability and risk.

3.4 Agent Architecture and Hyperparameters

A Deep Q-Network with a feed-forward policy network was employed in all experiments. The network consisted of two hidden layers containing 20 and 10 units, respectively. Hyperparameters were held constant across all training regimes:

- Learning rate: $\alpha = 1 \times 10^{-3}$
- Replay buffer size: 1 000 transitions (with a 1 000-step warm-up period)
- Target network update frequency: every 500 gradient steps
- Discount factor: $\gamma = 0.99$
- Exploration schedule: ϵ-greedy decayed from 1.0 to 0.05 over the first 30% of training
- Random seed: 42 (for reproducibility)

Hyperparameter Selection: All hyperparameters (learning rate, replay buffer size, network architecture) were chosen via preliminary grid searches over a small subset of the EUR/USD training data. Specifically, learning rates and batch sizes were varied to select the combination that maximized the early cumulative reward over training steps. This procedure yielded our default values like $\alpha = 1 \times 10^{-3}$.

Model Selection: While our experiments focus on DQN-based transfer, actor-critic methods such as Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) offer compelling advantages, particularly in continuous control and sample efficiency, and have shown strong performance in financial RL tasks [12]. We chose DQN to maintain a discrete-action framework aligned with the simple long/short/hold decision set and to leverage stable baselines implementations. However, exploring PPO or SAC in future work could enable smoother policy updates and more robust exploration, especially when extending to leverage or partial-position environments.

3.5 Transfer Learning Methodologies

Five training regimes were compared on the GBP/USD target domain:

1. **Zero-Shot Transfer:** The policy pre-trained on EUR/USD was directly evaluated on GBP/USD without further updates.
2. **Full Fine-Tuning:** All network weights were further trained for 100 000 steps on GBP/USD.
3. **Partial Fine-Tuning:** The first hidden layer was frozen to preserve low-level feature extractors, and the remaining parameters were trained for 100 000 steps.
4. **Reward-Function Transfer:** The EUR/USD pre-trained policy was fine-tuned on GBP/USD using the alternate risk-adjusted reward signal for 100 000 steps.
5. **From-Scratch Baseline:** A randomly initialized network was trained for 100 000 steps on GBP/USD.

3.6 Evaluation Protocol

After training or loading each model, sample evaluations were performed on the held-out GBP/USD data. The following metrics were recorded:

- **Cumulative Reward:** Sum of per-step returns during the evaluation period.

$$\sum_t r_t \text{ over all evaluation steps.}$$

- **Sharpe Ratio:** Mean per-step return divided by its standard deviation (with a small constant added to avoid division by zero). [13]

$$\text{Sharpe} = \frac{E[r_t]}{\sqrt{\text{Var}(r_t) + 10^{-8}}}.$$

- **Training Time:** Wall-clock duration required for the 100 000-step training run.

These metrics enabled comparison of sample efficiency, risk-adjusted performance, and computational cost across transfer learning strategies.

4 Results

This section presents the empirical evaluation of five reinforcement learning strategies applied to the GBP/USD trading task. All agents were trained for 100,000 steps and evaluated using three metrics: cumulative reward, Sharpe ratio (risk-adjusted return), and training time. Table 1 summarizes the numerical results, and Figures 1, 2, and 3 visualize final agent performance using each metric.

4.1 Quantitative Comparison

Mode	Sharpe	Reward	Train Time
Zero-Shot	-0.0061	-233.3	8.2
Full Fine-Tuning	-0.0151	-13.0	36.5
Partial Fine-Tuning	$+0.0057$	214.5	32.8
Reward-Function	-0.0162	-1.5	38.3
From Scratch	$+0.0268$	749.9	37.3

Table 1: Performance comparison of different transfer learning modes on GBP/USD after 100k steps.

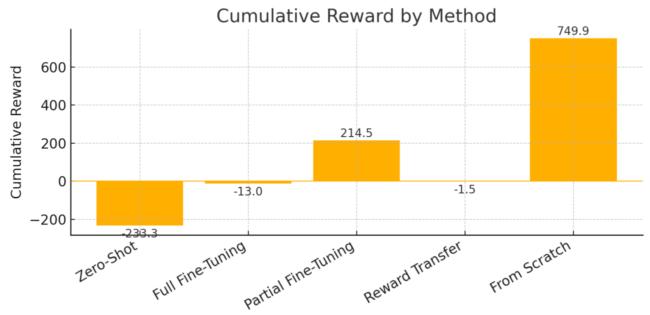

Figure 1: Final cumulative reward across 100k training steps for each method, as reported in Table 1. From-Scratch performs best. Partial Fine-Tuning provides moderate gains. All other transfer strategies yield negligible or negative reward.

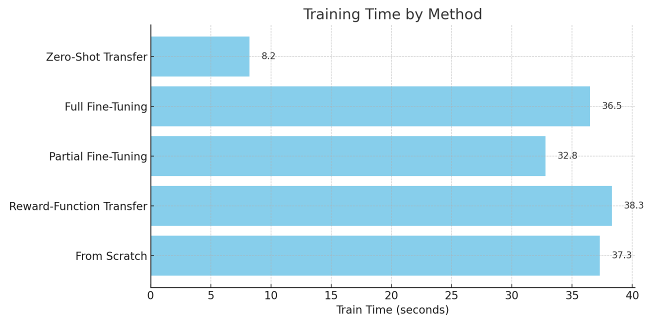

Figure 2: Final training runtime across 100k training steps for each method, as reported in Table 1. Zero-Shot Transfer is by far the fastest due to no additional training, while all other methods, regardless of transfer modality, require roughly 32–38 seconds to complete.

Figure 1 offers a visual summary of cumulative reward. The **From-Scratch** agent significantly outperforms all others, achieving a final reward of 749.9. In contrast, the **Zero-Shot Transfer** agent performs the worst, suggesting that unadapted source policies do not generalize to new currency regimes.

4.2 Performance Trends and Observations

From Table 1, Figure 1, Figure 2, and Figure 3, the following insights emerge:

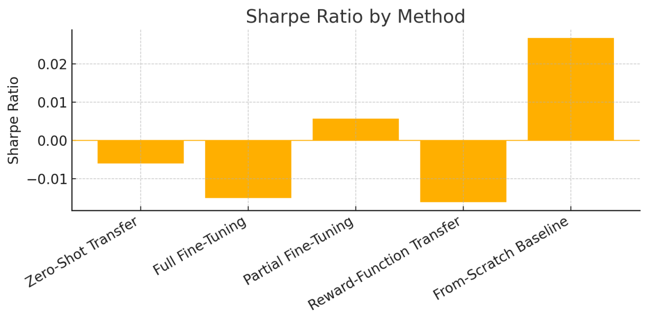

Figure 3: Final Sharpe ratios after 100k training steps for each method, as reported in Table 1. Only the From-Scratch baseline achieves a positive Sharpe, Partial Fine-Tuning yields a modest positive ratio, and all other transfer strategies finish negative.

- **From-Scratch learning achieves the best overall performance**, reaching a positive Sharpe ratio and the highest reward, despite requiring the most steps to converge. This suggests that a randomly initialized policy can eventually learn useful behavior on GBP/USD given enough exploration and gradient updates.
- **Partial Fine-Tuning provides a partial benefit**: the agent reaches a positive reward and Sharpe ratio, but substantially lags the from-scratch baseline. This suggests that freezing low-level layers may stabilize training, but at the cost of adaptability.
- **Full Fine-Tuning underperforms and is unstable**: while it benefits from reuse of a pre-trained model, it ends with a net negative reward and Sharpe, possibly due to catastrophic forgetting or overfitting to early random behaviors [6].
- **Reward-Function Transfer fails to guide learning effectively**: although it slightly improves over Full Fine-Tuning in raw reward, it still finishes with a negative Sharpe and negligible gains, indicating that reward shaping alone was insufficient.
- **Zero-Shot Transfer is consistently harmful**: lacking any domain-specific adaptation, it performs the worst across all metrics and illustrates that transferring a static policy across unrelated tasks can degrade returns.

4.3 Takeaways and Reflections

The combined analysis of performance and runtime suggests that not all transfer learning is efficient or beneficial in financial domains. While transfer methods often aim to reduce sample complexity or convergence time, this study finds that ineffective adaptation strategies may consume similar computational resources without yielding meaningful improvements.

These results reinforce the following lessons:

- Transfer learning strategies must be carefully matched to the domain and pretrained model quality.
- Training from scratch remains competitive in both runtime and performance when resources permit.

- Evaluating both reward and runtime reveals important trade-offs between efficiency and effectiveness.

The experimental results confirm that transfer learning must be applied with caution in financial RL. In this study, only one transfer method (Partial Fine-Tuning) offered modest benefits, and it did not even surpass the from-scratch baseline. More aggressive forms of transfer—like full fine-tuning or reward-only transfer, introduced volatility and instability, while zero-shot transfer performed worst.

Ultimately, the findings reinforce the following.

- Fine-tuning needs to be structured to avoid destructive overwriting of knowledge.
- Stable learning from scratch remains viable when time and data permit.
- Visualizing outcomes with cumulative metrics helps reveal true agent behavior beyond a single score.

5 Discussion

5.1 Summary of Findings

This research systematically evaluated the effectiveness of transfer learning techniques for reinforcement learning (RL) agents in forex trading, specifically when transferring knowledge from EUR/USD to GBP/USD. The empirical results highlight a clear hierarchy of effectiveness among the tested transfer methods. Contrary to initial expectations, training the agent from scratch on GBP/USD yielded the highest cumulative reward (749.9) and Sharpe ratio (0.0268), significantly outperforming all transfer-based methods. This strongly indicates that the GBP/USD-specific dynamics necessitate tailored training strategies, and generalized knowledge from other currency pairs is insufficient for achieving optimal performance without substantial adaptation.

5.2 Zero-Shot Transfer

The zero-shot transfer approach, involving direct application of a policy trained on EUR/USD to GBP/USD without additional tuning, showed particularly poor performance, resulting in negative cumulative reward (-233.3) and Sharpe ratio (-0.0061). These outcomes emphasize the substantial differences in underlying market behavior between currency pairs, indicating that even closely related forex markets possess distinct characteristics that cannot be ignored. Thus, direct policy transfer without adaptation appears severely inadequate.

5.3 Full Fine-Tuning Transfer

Interestingly, full fine-tuning, which adapts all network layers from the pre-trained model, failed to improve results significantly, yielding negative metrics (Sharpe ratio of -0.0151, cumulative reward of -13.0). The challenges observed likely stem from catastrophic forgetting, where previously acquired general features are overwritten during training, compromising the agent's ability to generalize effectively to the new currency pair. This result highlights the risks of extensive re-optimization without careful control over layer adaptation.

5.4 Partial Fine-Tuning Transfer

Partial fine-tuning, which involves freezing lower-level network layers and adapting only higher-level decision-making layers, provided modest but positive results (Sharpe ratio of 0.0057 and cumulative reward of 214.5). This suggests that early network layers successfully retain transferable representations of general market features (such as trends and momentum), thereby stabilizing training and enabling moderate improvements in learning efficiency. However, despite these benefits, partial fine-tuning did not surpass the performance of training from scratch, underscoring a trade-off between quick adaptation and ultimate policy effectiveness.

5.5 Reward-Function Transfer

Reward-function transfer, which reuses pre-trained policies with a modified reward objective, performed comparably poorly (Sharpe ratio of -0.0162, cumulative reward of -1.5). This outcome highlights a critical insight: simply adjusting incentives without addressing underlying representational mismatches between currency pairs does not substantially enhance performance. Thus, a more comprehensive approach involving structural adaptations of the policy appears essential.

5.6 Methodological Reflections

The study adopted a rigorous and controlled methodological approach, isolating transfer learning methods while keeping other variables constant, such as model architecture and training parameters. This control enabled clear insights into the relative effectiveness of each transfer learning technique. However, this approach might have constrained the adaptability and optimization potential of the agents.

5.7 Limitations

A key limitation of this study is the baseline model chosen for initial training on EUR/USD, which performed less effectively than anticipated. Consequently, the transfer of suboptimal policies inherently restricted the potential performance gains achievable by subsequent adaptation techniques. Additionally, the computational constraints limited the complexity and optimization of the models employed. Future studies should therefore investigate more robust baseline models with enhanced computational resources, potentially allowing deeper neural networks, sophisticated architectures, and richer feature sets. This could lead to improved baseline performance and consequently more meaningful assessments of transfer learning effectiveness.

Despite careful methodological control, this study was constrained by limited computing resources and a tight timeline, which led to running experiments with a fixed budget and logging only the end metrics (final cumulative reward, Sharpe ratio, runtime). The design choice prioritized clear state comparisons of transfer vs. scratch performance over capturing noisy intermediate steps. Consequently, the sample efficiency or precise convergence behavior of each transfer method can not be spoken under study, a gap that may obscure differences in how quickly or stably agents learn.

5.8 Practical Implications

The findings illustrate nuanced limitations of transfer learning for forex RL scenarios, particularly emphasizing that transfer learning should be applied cautiously and selectively. While partial fine-tuning holds potential benefits by preserving generic market knowledge, robust results ultimately depend on extensive domain-specific training. Future researchers should carefully consider the trade-offs between faster convergence offered by transfer learning and the higher absolute performance attainable through complete retraining on target markets.

6 Responsible Research

6.1 Ethical Considerations

This research was conducted solely for academic purposes and does not constitute investment advice or propose deployment of live trading strategies. All experiments were performed on historical, publicly available data from Dukascopy, specifically the EUR/USD and GBP/USD currency pairs, at 15-minute resolution. No personal, sensitive, or proprietary data was used.

While reinforcement learning in financial domains has exciting potential, the associated risks, including algorithmic bias, market instability, and overfitting to historical conditions were acknowledged. This work remains strictly within a simulated environment and is not intended for direct financial use. Readers and future researchers are advised to exercise caution and implement appropriate risk management and regulatory safeguards if applying these methods in production settings.

6.2 Use of Generative AI

OpenAI's ChatGPT was used as an assistive tool throughout the writing process, in accordance with TU Delft's guidelines for responsible academic use of generative AI. The model supported tasks such as improving academic tone, rephrasing informal sentences, suggesting synonyms to reduce repetition, generating LaTeX equations and tables, and providing outline structures for some sections.

Specifically, ChatGPT was used for:

- Rewriting sentences to align with academic style (turning "the agent did kinda bad" into "the agent underperformed relative to baseline").
- Offering alternatives to repetitive phrases and improving clarity.
- Generating LaTeX syntax for tables, math expressions, and formatting.
- Suggesting section headers and improving document structure.
- Checking tone and consistency in complex or reflective passages.

Example prompts used during development include:

- `Can you tell me which words to rephrase to keep the academic tone in this paragraph?`
- `Write a LaTeX table comparing Sharpe ratios across five strategies with these given data in hand.`
- `Suggest a more concise way to say the reward function is changed but the policy is fixed.`
- `Give me a professional way to say zero-shot is the worst.`

Importantly, ChatGPT was **not** used to generate scientific content, design experiments, or interpret data. All technical contributions, model implementations, and evaluations were performed by the author. Generative AI was limited to editorial and formatting assistance only.

6.3 Reproducibility

To promote transparency and repeatability, this project was designed with reproducibility as a key principle. The complete pipeline—including feature engineering, environment setup, training, and evaluation—was implemented in a modular Python codebase using Stable Baselines3 and custom Gym environments. Reproducibility was ensured through:

- Explicit reporting of all hyperparameters (learning rate, gamma, exploration schedule) in Section 3.
- Fixed random seed (42) across training environments and PyTorch for determinism.
- Chronological train-test data splits to avoid look-ahead bias.
- Consistent evaluation metrics: Sharpe ratio, cumulative reward, and runtime.
- Public availability of the scripts and the project structure is available in our project repository.

This setup enables easy replication and extension by future researchers.

7 Conclusions and Future Work

7.1 Conclusions

This research explored the potential of transfer learning in reinforcement learning (RL) for algorithmic trading, specifically by transferring a DQN agent trained on EUR/USD to a new market: GBP/USD. The study systematically compared four transfer techniques—zero-shot transfer, full fine-tuning, partial fine-tuning, and reward-function transfer—against a baseline trained from scratch on the target domain.

The results show that training directly on GBP/USD yielded the best outcomes in terms of cumulative reward and Sharpe ratio. This underscores that financial markets, even closely related ones, often possess distinct statistical properties that limit the direct generalizability of learned policies. Among the transfer methods, partial fine-tuning offered the most encouraging performance, preserving some useful low-level features while allowing higher-level adaptation. In contrast, both zero-shot and full fine-tuning approaches failed to deliver meaningful improvements, likely due to poor cross-market alignment and catastrophic forgetting. Reward-function transfer, while conceptually appealing, proved insufficient in practice without concurrent policy adaptation.

Overall, the findings highlight that while transfer learning remains a promising paradigm for improving training efficiency, its practical success in forex trading hinges on careful method selection, the quality of the source model, and the degree of structural similarity between source and target environments.

7.2 Limitations

Despite the careful design, several limitations constrain the generality of this study. The source model trained on EUR/USD was relatively simple and underperformed, likely restricting the potential benefits of transfer. The environment design also involved idealized assumptions—such as no transaction costs or slippage—that may limit real-world applicability. Moreover, the analysis was restricted to only two currency pairs and did not explore volatile regimes or macroeconomic shocks.

Computational constraints further limited exploration of larger architectures or longer training horizons, which may be essential to fully unlock the advantages of transfer learning. These limitations call for caution in extrapolating the results and open space for more ambitious experimental setups in future work.

7.3 Contributions

This thesis presents a reproducible and extensible RL pipeline tailored to forex trading tasks. The implemented environment is fully compatible with the Gym API and includes engineered market and agent-state features. The experimental framework allows for modular experimentation and clear benchmarking of transfer learning strategies. Importantly, the study offers a transparent comparison of transfer methods under controlled conditions, contributing both conceptual insights and practical tools to the financial RL research community.

7.4 Future Work

Several promising directions arise from this work. First, future research could benefit from a stronger source model, trained with deeper architectures, larger datasets, or more rigorous tuning, which may yield more transferable representations. Additionally, integrating domain adaptation techniques could improve feature generalization across markets.

Critical Step: Extending the logging pipeline to record per-episode (or per-step) metrics—cumulative reward, loss, Q-value estimates, ϵ, and run each transfer mode across multiple random seeds. Then, plotting average learning curves to illustrate exactly how fast each method reaches, say, 50% of its final reward and how stably it converges. This will expose sample-efficiency trade-offs hidden by end-state tables.

Additionally, our fine-tuning strategy uses a fixed 100,000-step schedule; adaptive schedules (progressively unfreezing layers or using cyclical learning rates) may better balance retention of source-domain features with adaptation to the target market [7][14]. Investigating these alternative algorithms and dynamic fine-tuning schedules stands as a promising direction for improving both convergence speed and final policy performance.

Expanding the experimental scope to include a wider array of assets—such as commodities, indices, or even multi-asset portfolios—would allow testing transferability under broader conditions. Introducing transaction costs, leverage, partial observability, and real-time constraints would also enhance realism and test the robustness of strategies. Moreover, investigating meta-learning frameworks or fine-tuning methods designed for few-shot adaptation could enable more flexible and sample-efficient agents.

In conclusion, while transfer learning in algorithmic trading presents non-trivial challenges, this thesis establishes a practical foundation for future exploration and highlights key considerations for building more scalable, adaptable, and intelligent trading systems.

References

[1] Gerald Appel. *Technical Analysis: Power Tools for Active Investors*. Financial Times Prentice Hall, 2005.

[2] Bank for International Settlements. Examination of foreign exchange and otc derivatives markets. BIS Triennial Central Bank Survey, 2019. https://www.bis.org.

[3] Christoph Bergmeir and José M. Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. In *Proceedings of the 35th International Conference on Machine Learning (ICML) — Gym Workshop*, 2016.

[5] Robert M. French. Catastrophic forgetting in connectionist networks. In *Trends in Cognitive Sciences*, volume 3, page 128–135, 1999.

[6] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[7] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proc. of the 56th Annual Meeting of the ACL (ACL)*, pages 328–339, 2018.

[8] Rob J Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2018.

[9] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321, 1993.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[11] Cheol-Ho Park and Scott H. Irwin. What do we know about the profitability of technical analysis? *Journal of Economic Surveys*, 21(4):786–826, 2007.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*, 2017.

[13] William F. Sharpe. Mutual fund performance. *The Journal of Business*, 39(1):119–138, 1966.

[14] Leslie N. Smith. Cyclical learning rates for training neural networks. *arXiv preprint arXiv:1506.01186*, 2017.

[15] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[16] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[18] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[19] J. Welles Wilder. *New Concepts in Technical Trading Systems*. Trend Research, 1978.