Reinforcement Learning for FX trading

Yuqin Dai, Chris Wang, Iris Wang, Yilun Xu

A brief intro to our strategy…

RL Strategy on High-frequency Forex (1/2)

How is Forex traditionally traded?

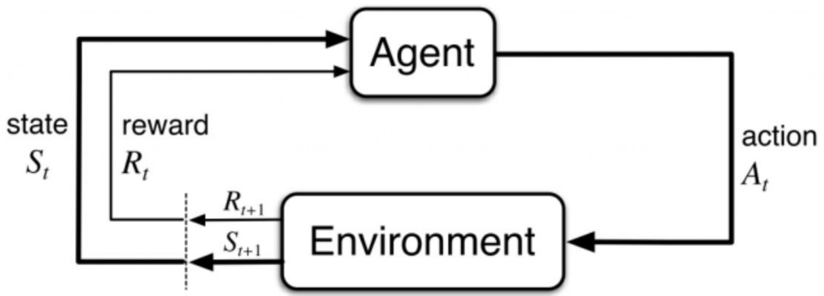
- A few key decisions:
    - Currency pair to trade
    - Position size
    - When to enter/exit
    - Which dealer to use/how to execute the trade
    - Bid-ask spread

- Traditional strategies use Momentum, Mean Reversion, Pivots, Fundamental Strategy, Stop-loss orders
    - Trend-based -> machine learning?
    - Scalping, Day trading, Longer time frames

RL Strategy on High-frequency Forex (2/2)

Reinforcement learning for forex trading

- Reinforcement Learning (RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.
- Trading is an “iterative” process, and past decisions affect future, long-term rewards in indirect ways
    - Compared to supervised learning, we are not making or losing money at a single time step…
- Traditional “up/down” prediction models do not provide an actionable trading strategy
- Incorporate longer time horizon
- Give us more autonomy in trading policy, regularize the model from trading too frequently



Since midterm presentation...

- ➔ A larger dataset: 6 days -> 1 month (25 trading days)
- ➔ We found a data processing bug in our previous result, which indicates our previous PnL result is no longer valid
- ➔ More models: linear direct RL -> deep direct RL and DQN
- ➔ More currency pairs: AUDUSD -> AUDUSD, EURUSD, GBPUSD
- ➔ Hyperparameter tuning and error analysis
- ➔ Migrate to cloud

Data Processing and Training Pipeline

Data Processing

- Clean raw dataset and sample it into a second-level one
- Pad the data for each liquidity provider to the same time frame
- Build an order book by picking the best bid/ask prices
- Extract features using bid/ask/mid price returns from all 8 currency pairs
- Train one target currency at a time
- Choose model structure based on AUDUSD while train the same model for EURUSD and GBPUSD

Training Pipeline

	2/1	2/3	2/4	2/5	2/6	Train Week 1
Eval/test Week 1	2/7	2/8	2/10	2/11	2/12	Train Week 2
Eval/test Week 2	2/13	2/14	2/15	2/17	2/18	Train Week 3
Eval/test Week 3	2/19	2/20	2/21	2/22	2/24	
	2/25	2/26	2/27	2/28	3/1	

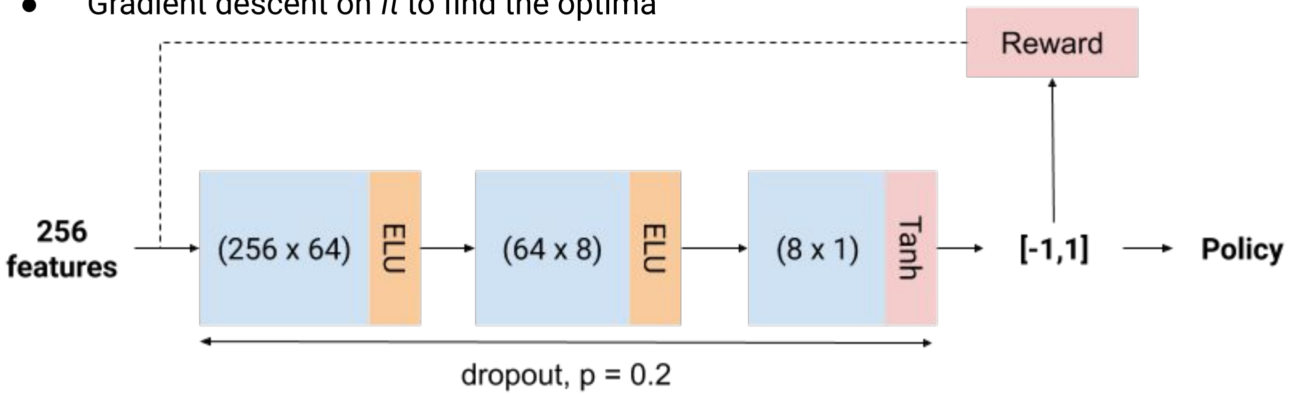
AUDUSD, EURUSD, GBPUSD

Deep direct reinforcement learning model…

Deep Direct RL Model (1/2)

Goal	Maximize total (undiscounted) return over **1-hour horizon** by making short/long trading decisions for target currency per second
Input	**Per second** bid-ask prices for target currency and other available currency pairs; include the recent **16-second returns** as features
Action	Float between -1 (short the currency with all cash) and 1 (long the currency with all cash)
Method	Policy Gradient ● Maximize the “expected” reward when following a policy π ● Actions are chosen by ‘actor’, i.e. mapping current features to next action ● Gradient descent on π to find the optima

$$J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{\tau} r_t]$$



Deep Direct RL Model (2/2)

In detail

$$a_t = Tanh(< w, x_{t-1} > + < w', a_{t-1} > +b)$$
$$r_t = f(a_t, a_{t-1})$$
$$R = r_1 + ... + r_\tau$$

Rewards incorporating bid-ask spreads

	-1	0	1
-1	0	-Ask[t]	-2*Ask[t]
0	Bid[t]	0	-Ask[t]
1	2*Bid[t]	Bid[t]	0

REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, \ldots, T-1$:
    $G \leftarrow$ return from step t (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta})$

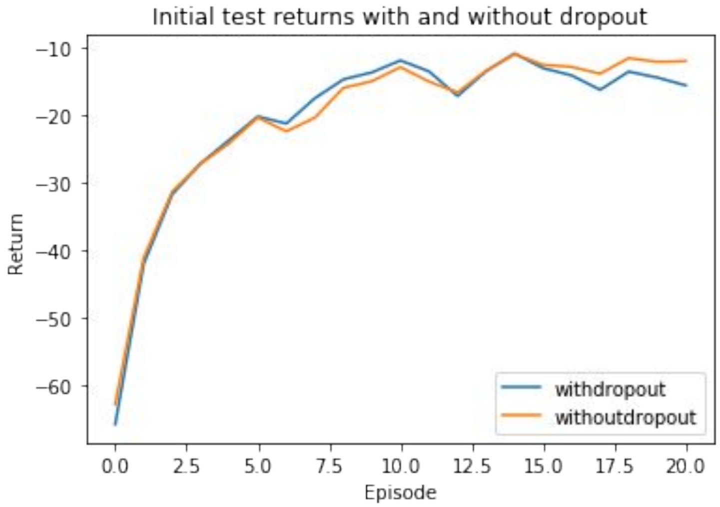
Hyperparameter Tuning and Experimentation

	Eval Reward, epoch 20
Adam	
SGD	✅

Number of Features	Eval Reward, epoch 20
256	✅
512	

Bias	Eval Reward, epoch 20
Bias in last layer	✅
No bias in last layer	

Number of hidden layers	Eval Reward, epoch 20
(64, 8)	✅
(50, 50)	



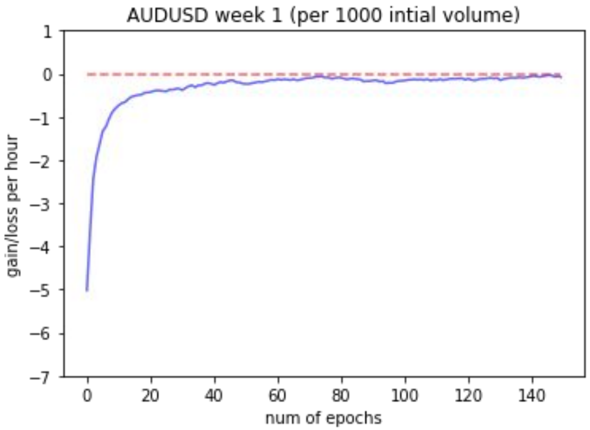
Reward with dropout vs no dropout

Hyperparameter Analysis

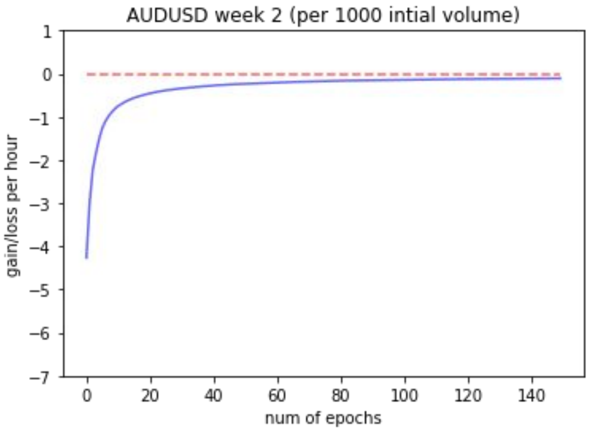
- **Dropout**
  - Prevents our model from overfitting as the epoch number increases
- **Number of features**
  - Too many features can be harmful because they impart noise and make it harder for the model to converge
- **Bias**
  - An additional parameter in the last layer can be helpful in allowing the model to learn new relationships
- **Number of hidden layers**
  - The number of hidden layers should decrease gradually
- **Adam vs. SGD**
  - Adam is faster at first in learning, but SGD generalizes better long-term

Result Analysis: AUDUSD (1/2)

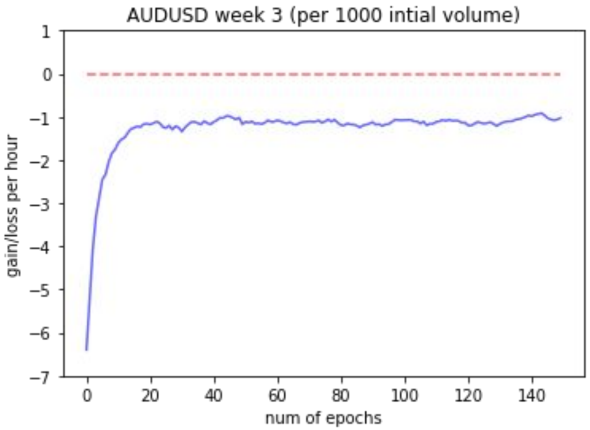
Deep Direct Reinforcement Learning model performance on eval set



12 hours' training
Breakeven per hour with per AUD 1,000 initial capital
Yield ~ 0.000%



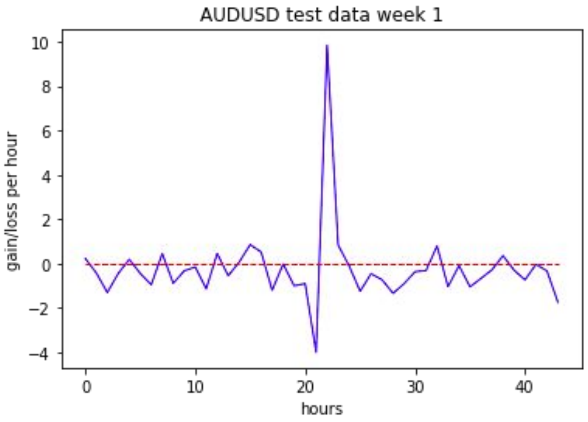
12 hours' training
Breakeven per hour with per AUD 1,000 initial capital
Yield ~ 0.000%



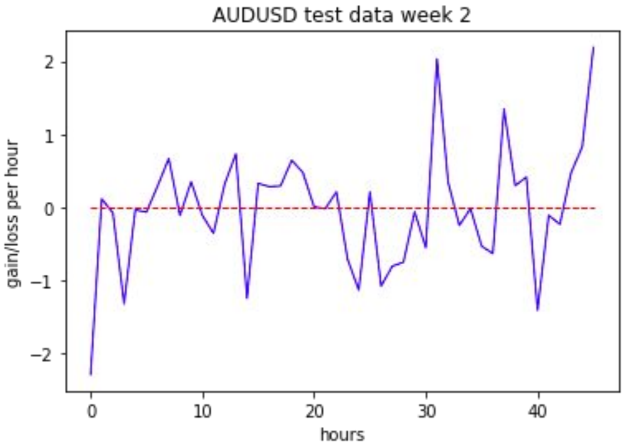
12 hours' training
Lose USD 1.03 per hour with per AUD 1,000 initial capital
Yield ~ -0.147%

Result Analysis: AUDUSD (2/2)

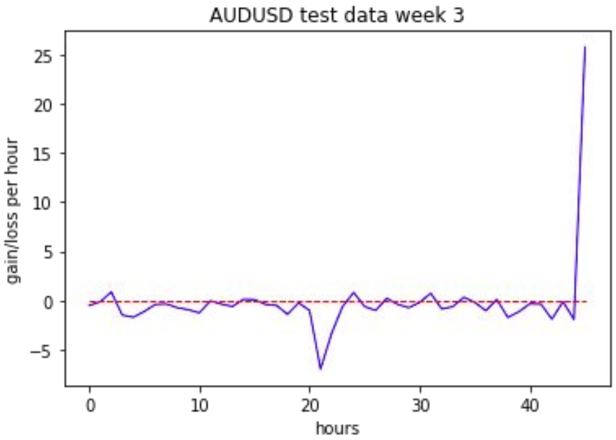
Deep Direct Reinforcement Learning model performance on eval & test set



Lose USD 0.25 per hour with per AUD 1,000 initial capital, with std of USD 1.745
Yield ~ -0.036%



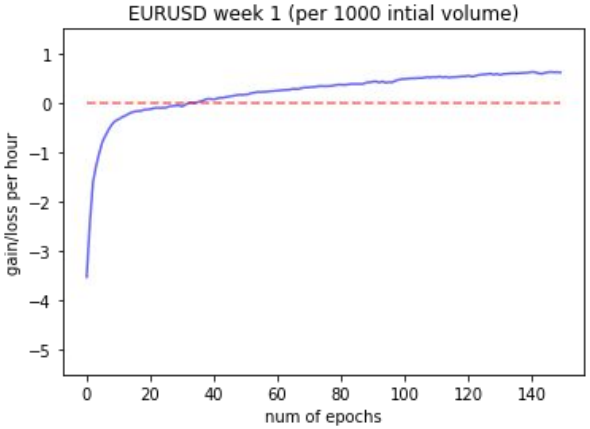
Breakeven per hour with per AUD 1,000 initial capital, with std of USD 0.811
Yield ~ 0.000%



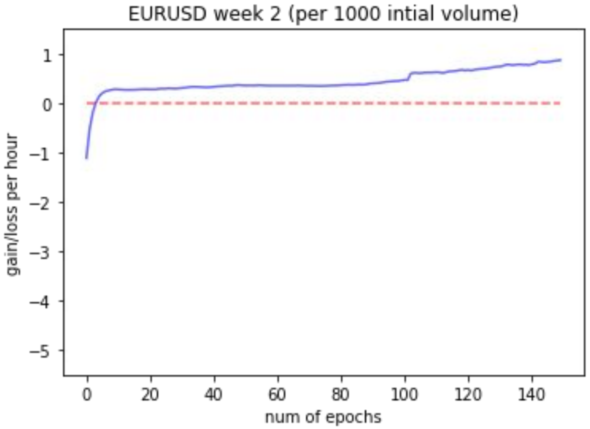
Lose USD 0.12 per hour with per AUD 1,000 initial capital, with std of USD 4.06
Yield ~ -0.017%

Result Analysis: EURUSD (1/2)

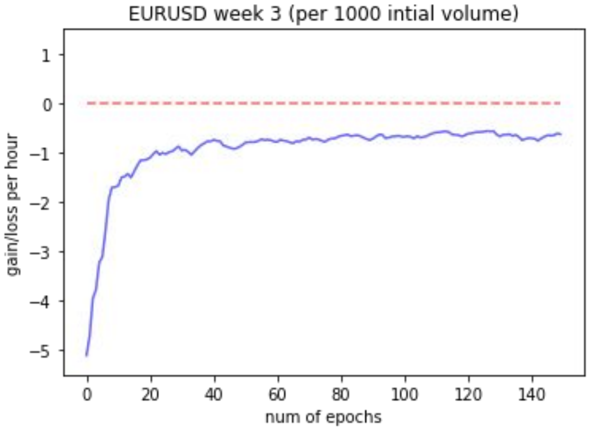
Deep Direct Reinforcement Learning model performance on eval set



12 hours' training
Gain USD 0.62 per hour with per EUR 1,000 initial capital
Yield ~ 0.055%



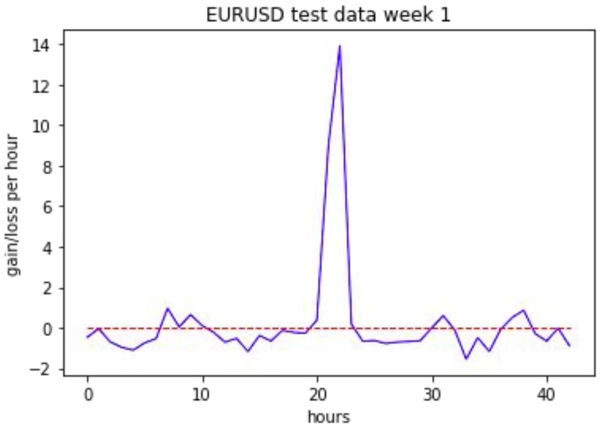
12 hours' training
Gain USD 0.87 per hour with per EUR 1,000 initial capital
Yield ~ 0.078%



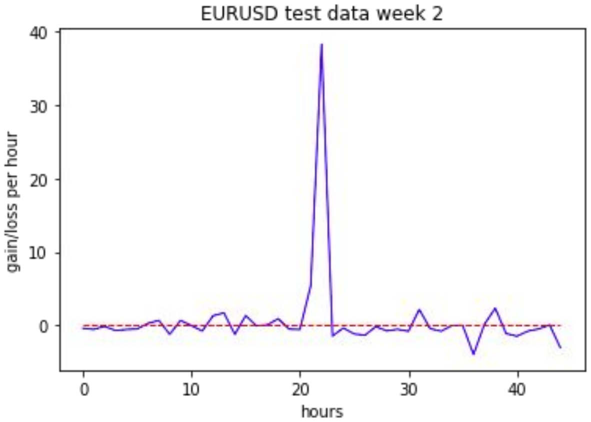
12 hours' training
Lose USD 0.63 per hour with per EUR 1,000 initial capital
Yield ~ -0.056%

Result Analysis: EURUSD (2/2)

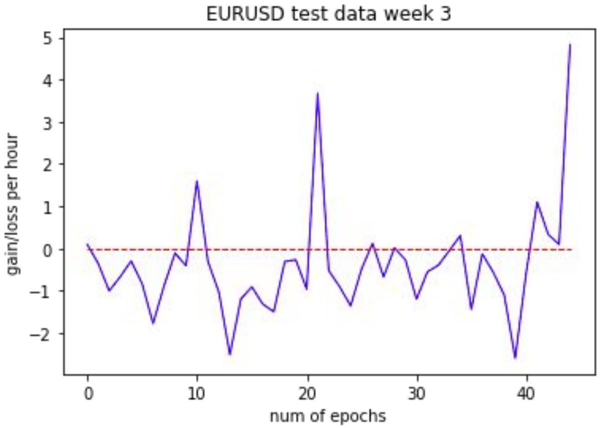
Deep Direct Reinforcement Learning model performance on eval & test set



Gain USD 0.18 per hour with per EUR 1,000 initial capital, with std of USD 2.758
Yield ~ 0.014%



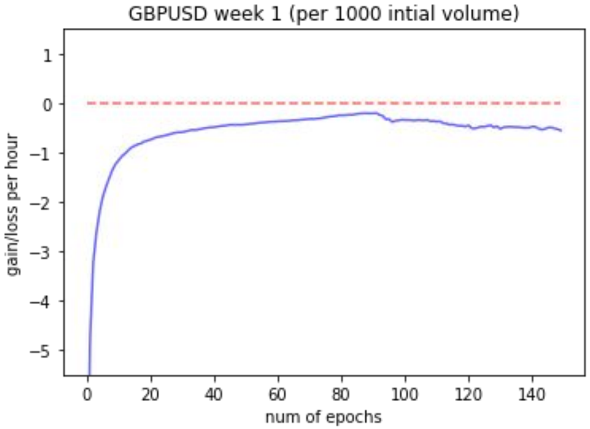
Gain USD 0.30 per hour with per EUR 1,000 initial capital, with std of USD 2.593
Yield ~ 0.023%

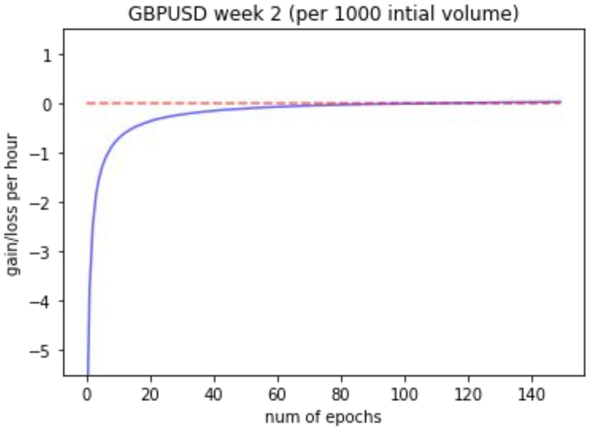


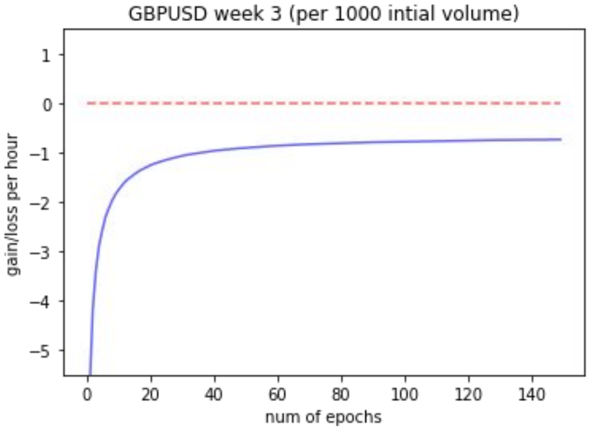
Gain USD 0.69 per hour with per EUR 1,000 initial capital, with std of USD 7.291
Yield ~ 0.052%

Result Analysis: GBPUSD (1/2)

Deep Direct Reinforcement Learning model performance on eval set







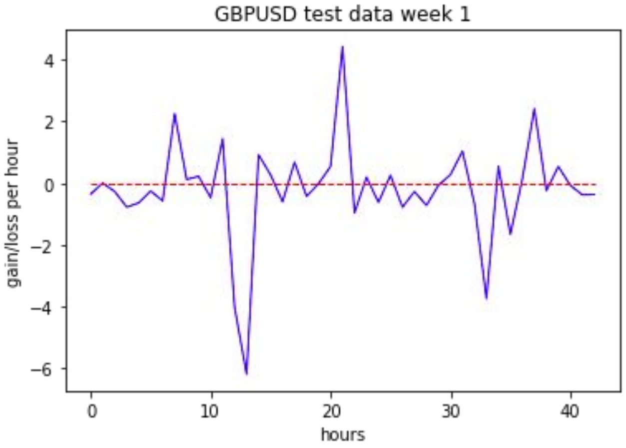
12 hours' training
Lose USD 0.55 per hour with per GBP 1,000 initial capital
Yield ~ -0.044%

12 hours' training
Breakeven per hour with per GBP 1,000 initial capital
Yield ~ 0.000%

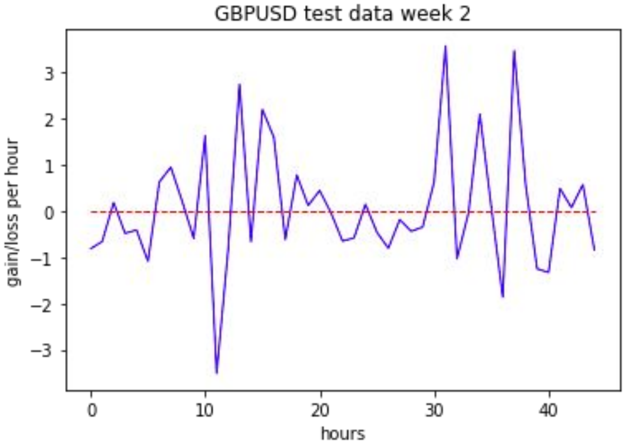
12 hours' training
Lose USD 0.74 per hour with per EUR 1,000 initial capital
Yield ~ -0.058%

Result Analysis: GBPUSD (2/2)

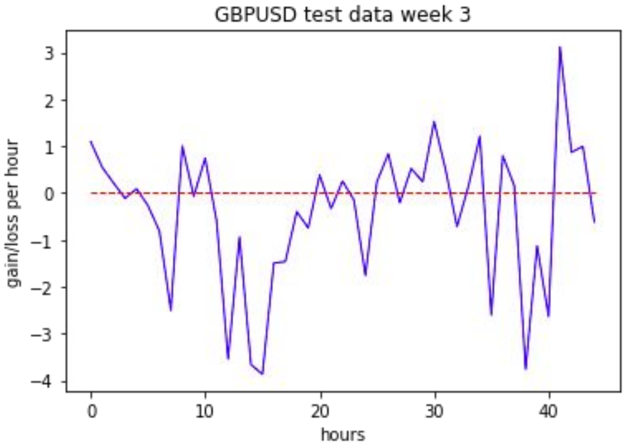
Deep Direct Reinforcement Learning model performance on eval & test set



Lose USD 0.21 per hour with per GBP 1,000 initial capital, with std of USD 1.589
Yield ~ -0.016%



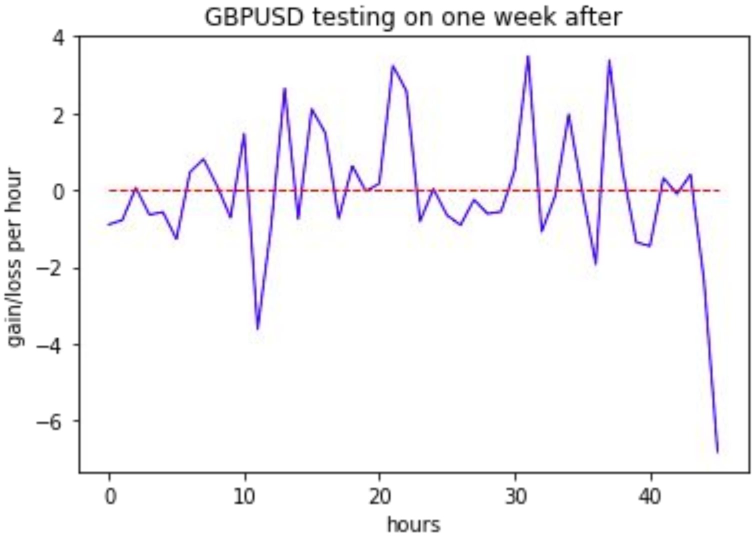
Gain USD 0.072 per hour with per GBP 1,000 initial capital, with std of USD 1.298
Yield ~ 0.0056%



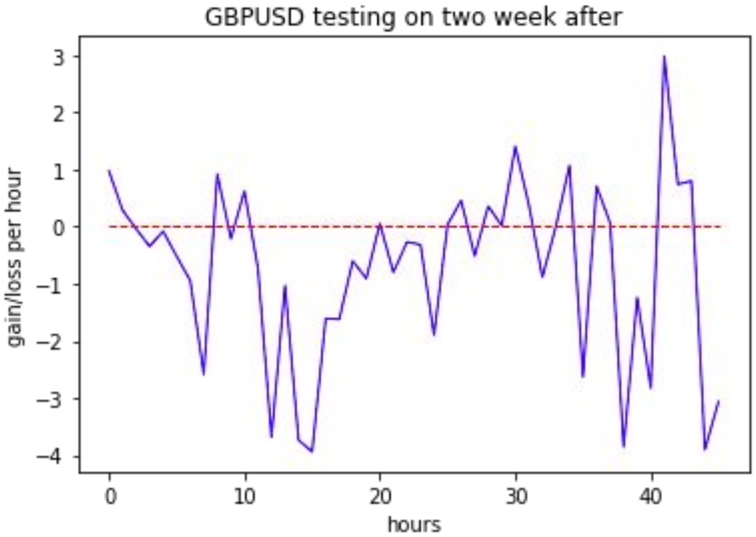
Lose USD 0.413 per hour with per GBP 1,000 initial capital, with std of USD 1.478
Yield ~ -0.032%

Across-time/ Currency Analysis (1/2)

Deep Direct Reinforcement Learning model performance on lag data



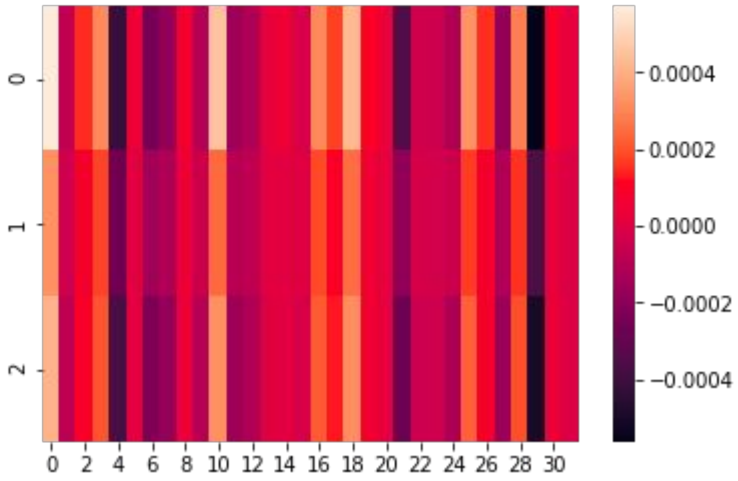
GBPUSD model trained on train week 1 and tested on eval week 2



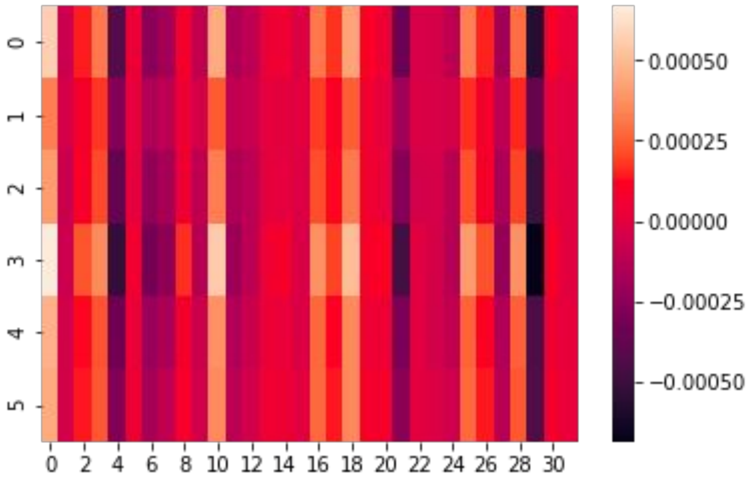
GBPUSD model trained on train week 1 and tested on eval week 3

Across-Time/ Currency Analysis (2/2)

Deep Direct Reinforcement Learning model gradient w.r.t. first 32 features



GBPUSD model gradients on eval week 1



GBPUSD model gradients on eval week 1, 2 and 3

<u>Deep DRL model keeps looking for the same "patterns" across different time horizons</u>

Deep Q-Network RL model…

Deep Q-Network Model (1/3)

Goal

Estimate long-term discounted state-action pair values by Q network, and train an optimal policy based on the estimation

Input

Per second bid-ask prices for target currency and mid price of other available currency pairs; include the recent **16-second log returns, timestamp and previous position** as features;

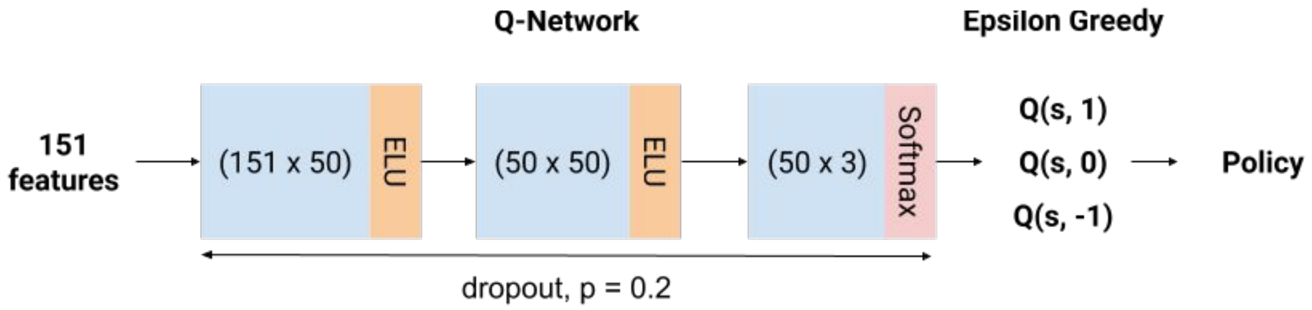
Action

-1 (short), 0 (neutral) or 1 (long)

Method

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,\boldsymbol{a},\boldsymbol{r},\boldsymbol{s}')\sim\mathcal{D}}\left[\|\mathbf{r} + \gamma Q_{\theta^-}(\boldsymbol{s}', \arg\max_{a'} Q_\theta(\boldsymbol{s}', a')) - Q_\theta(s, \boldsymbol{a})\|^2\right]$$

$$\theta \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}(\theta)$$



Deep Q-Network Model (2/3)

Initialize $T \in \mathbb{N}$, recurrent Q-network Q_θ, target network Q_{θ^-} with $\theta^- = \theta$, dataset $\mathcal{D}$ and environment E, $steps = 1$

Simulate env E from dataset $\mathcal{D}$

Observe initial state s from env E

for each step **do**

  $steps \leftarrow steps + 1$

  Select greedy action w.r.t. $Q_\theta(s, a)$ and apply to env E

  Receive reward r and next state s' from env E

  Augment actions to form $\mathcal{T} = (s, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s'})$ and store $\mathcal{T}$ to memory $\mathcal{D}$

  if $\mathcal{D}$ is filled and $steps \mod T = 0$ **then**

    Sample a sequence of length T from $\mathcal{D}$

    Train network Q_θ

  end if

  Soft update target network

end for

Deep Q-Network Model (3/3)

Customize 1: environment

- Self-defined environment which can draw training data point (bid-ask price with features) in order, without leaking the future price

Customize 2: memory replay

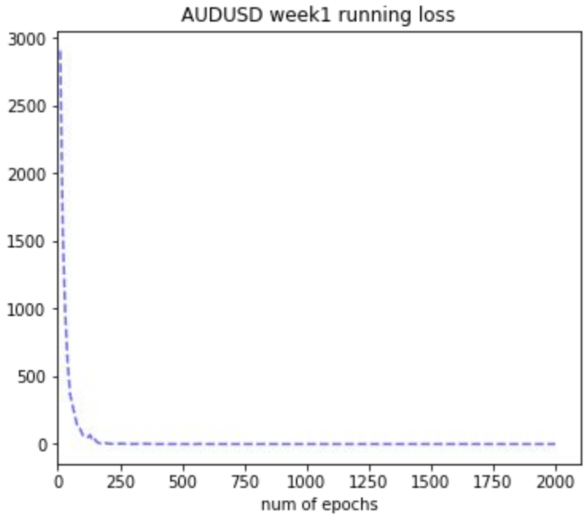
- Choose a small buffer size to conduct memory replay, which incorporates our belief that the most recent data points are more relevant in the market

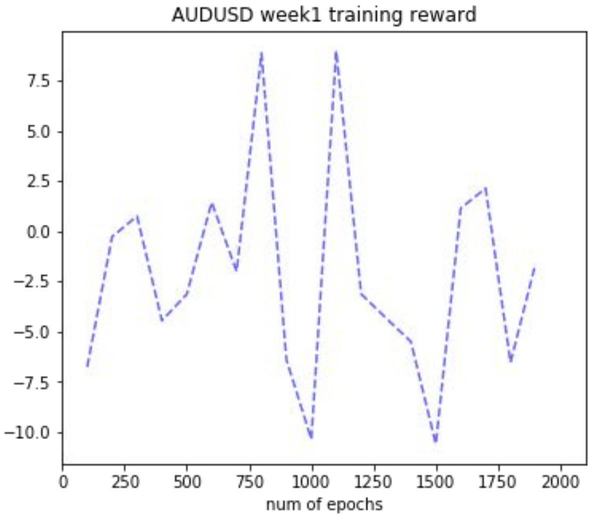
Customize 3: exploration strategy

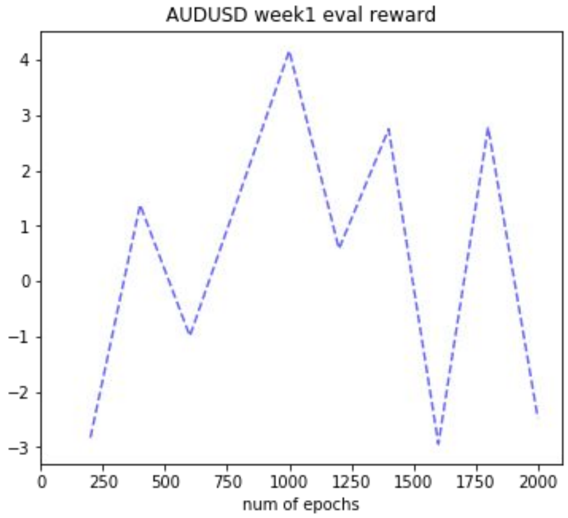
- Use standard epsilon greedy to encourage exploration during policy training
- Furthermore, use action augmentation to encourage deep exploration. For example, our policy chooses action 1 at time step t, with reward r. Then we add **(s, 1, r), (s, -1, r) and (s, 0, 0)** to the buffer

Result Analysis: AUDUSD (1/2)

Deep Q-Network Reinforcement Learning model performance







Running loss of the model decreases monotonically, while the training and eval reward fail to increase over time, accordingly

Result Analysis: AUDUSD (2/2)

Test result

- RL Agent learns to take neutral positions only (action = 0) and breaks even on the test set

Conclusion and explanation

- Running loss decreases monotonically while training and eval reward diverge
    - The Q-Network can successfully model the infinite discounted state-action value
    - The Q-Network may not represent 1-hour trading returns well
    - Epsilon greedy + action augmentation are not sufficient to train the optimal policy

- Agent decides to make almost no trade on test set
    - Limited flexibility: we only allow the agent to choose from {-1, 0, 1}
    - Confusing signals: we give a bunch of signals to the model without delicate feature engineering. The agent may learn to keep neutral only after seeing large amount of data points with close features

Our key takeaways…

Conclusions

- **Why Forex RL trading works**
  - trend-based; resembles factor model
- **DRL vs. DQN**
  - DRL is more interesting to explore
- **Out-of-sample performance varies with time periods**
  - performs the best when test period is 1 week after training period
- **Performance largely depends on feature selection**
  - 16 features perform better than 32
- **Deep models work better**
  - able to capture more complex inter-feature relations

Potential Next Steps

- **Incorporate better features**
  - Feature engineering (e.g. Time-series analysis)

- **Build a better architecture**
  - Add residual blocks
  - LSTM

- **More Training and Hyperparameter tuning**
  - Train with data of a longer time span
  - Regularization, optimizer

- **Add an Online Learning Scheme**
  - Update with incoming data

Thank you for listening!
Any questions?

Reference

1. Y. Deng, F. Bao, Y. Kong, Z. Ren and Q. Dai, "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653-664, March 2017.

2. Huang, Chien Yi. "Financial Trading as a Game: A Deep Reinforcement Learning Approach." *arXiv preprint arXiv:1807.02787* (2018).

3. J. Moody and M. Saffell, "Learning to trade via direct reinforcement," in *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 875-889, July 2001.