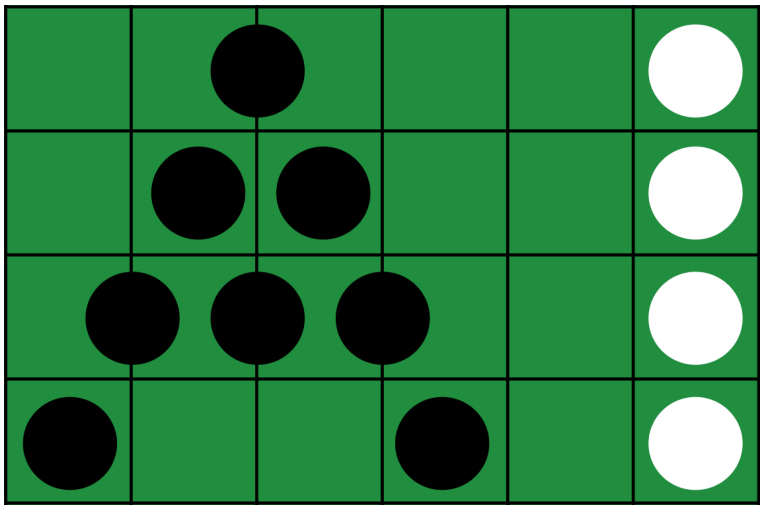
ENSIM
École d'ingénieurs
Le Mans Université

Deep Learning Project

Neural Network Models as Othello Player

Groupe TP2 :
Amburtsev Andrei



4A INFO ENSIM
Professeur :
Shamsi Meysam

Provided MLP and LSTM architectures

Description of MLP

- Input layer: 64 (8*8 – board size)
- Two hidden layers: 128 and 128 Linear layers
- Output layer: 64 (8*8 – board size)
- Optimizer: Adam
- Learning rate: 0.001
- Dropout: 0.1
- Epoch: 200 (Early stopping – 20)
- Batch size: 1000
- Len Samples: 1 (one to one)
- Number of parameters: 33088
- The best accuracy on DEV: 14.64%

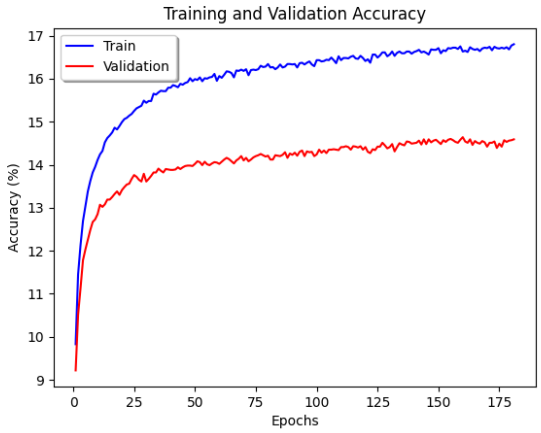
Description of LSTM

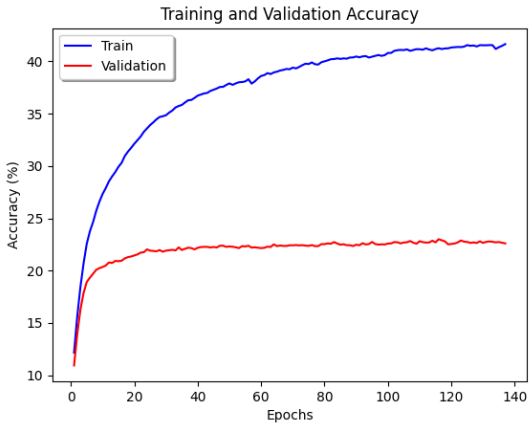
- Input layer: 64 (8*8 – board size)
- One hidden layer: 128 Linear layer
- Output layer: 64 (8*8 – board size)
- Optimizer: Adam
- Learning rate: 0.005
- Dropout: 0.1
- Epoch: 200 (Early stopping – 20)
- Batch size: 1000
- Len Samples: 5 (sequence to one)
- Number of parameters: 107584
- The best accuracy on DEV: 23%

Architecture	**Nombre de paramètres**	**Précision de Validation Dataset**
MLP	33088	14.64%
LSTM	107584	23%

Entre deux architectures, il y a une différence importante. MLP a besoin de moins du temps de calcul comme le nombre de paramètres est égal à 33088. Ce qui est trois fois moins par rapport au nombre de paramètres de LSTM équivalent à 107584. Alors que LSTM est plus coûteux à calculer, cela donne la meilleure précision : 23% contre 14.64% de MLP, ce qui montre que ce modèle est plus performant.

MLP **LSTM**





Dans les graphes suivants, on peut voir la précision de Train et Validation datasets en fonction de Epochs. Le contraste entre ces deux architectures reste crucial. En MLP, la différence d'accuracy entre training dataset et validation dataset est moins dramatique qu'en LSTM. C'est-à-dire, le comportement d'overfitting est plus présent chez LSTM proposé. Donc, dès qu'il rencontre les nouvelles données, LSTM est moins adapté pour se comporter efficacement. Néanmoins, LSTM se comporte mieux en général, comme la précision de training set a été élevée, plus de 40% en comparaison de 17% pour MLP.

En comparant, on obtient le tableau suivant :

Architecture	**Nombre de paramètres**	**Précision de Validation Dataset**	**Précision de Training Dataset**
MLP	33088	14.64%	16.8%
LSTM	107584	23%	41.64%

MLP et LSTM optimizing

Afin d'optimiser les architectures de MLP et LSTM, on change le nombre de couches, ainsi que la dimension de chaque couche sans changer les autres hyperparamètres. Cette approche a une influence directe sur la complexité du modèle, si on augmente le nombre de couches et/ou les dimensions, on augmente aussi le nombre de paramètres, donc le temps de calcul. Mais parfois si on diminue le nombre de paramètres, la complexité et donc le temps de calcul, on peut quand même améliorer la précision du modèle, cela bien évidemment est la situation plus préférable.

Alors, on ne change que les couches et ces dimensions.

MLP

Hidden Layer 1	Hidden Layer 2	Hidden Layer 3	Hidden Layer 4	Hidden Layer 5	Parameters	Accuracy (DEV)
96	-	-	-	-	12448	15.63%
96	96	-	-	-	21760	14.96%
96	96	96	96	96	49696	8.95%
96	256	256	256	256	244896	7.99%
128	-	-	-	-	16576	15.98%
128	96	96	96	96	54848	9.52%
128	128	-	-	-	33088	14.64%
128	192	192	192	192	156608	7.11%
192	-	-	-	-	24832	15.77%
192	96	-	-	-	37216	15.99%
192	96	96	-	-	46528	14.37%
192	96	96	96	-	55840	11.5%
192	96	96	96	96	65152	8.72%
192	96	96	96	128	70304	8.17%
192	96	96	96	192	80608	8.17%
192	96	96	96	256	90912	7.49%
192	96	96	128	256	102208	8.21%
192	96	96	192	96	83680	7.83%
192	96	96	192	128	91904	8.88%
192	96	96	192	192	108352	7.96%
192	128	-	-	-	45440	15.55%
192	128	256	192	128	152512	7.43%
192	256	128	192	96	144288	8.19%
192	256	128	256	96	158688	7.51%
256	-	-	-	-	33088	16.44%
256	96	-	-	-	47520	16.64%
256	128	-	-	-	57792	16.23%
256	192	192	192	192	189504	7.74%

On a les résultats bien intéressants. Par exemple, le modèle avec presque trois fois moins de paramètres (hidden layer 1 – 96) a meilleur accuracy de 15.63% par rapport de 14.64% pour le modèle fourni au début. Et on obtient expérimentalement que si on augmente le nombre de couche de paramètres à traiter, on n'a pas forcement la tendance de l'amélioration. Le meilleur modelé qu'on trouve en changeant de manière différente le nombre de couche, c'est celui de 256-96 avec 16.63%.

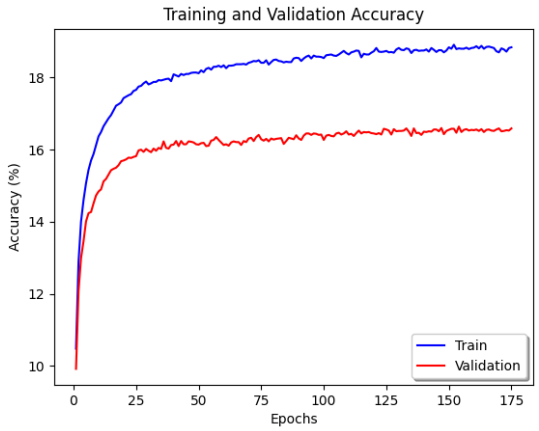
Ensuite, l'autre approche d'optimiser le modèle, c'est ajouter non-linéarité avec la fonction d'activation. On va expérimenter avec le modèle le plus performant qu'on a trouvé pour le moment (hidden layer 1 est égal 256, hidden layer 2 est égal 96) :

Fonction d'activation	Sans fonction (linear)	Sigmoid	Tanh	Leaky ReLU	ReLU
Precision	16.64%	16.17%	18.76%	22.01%	22.25%

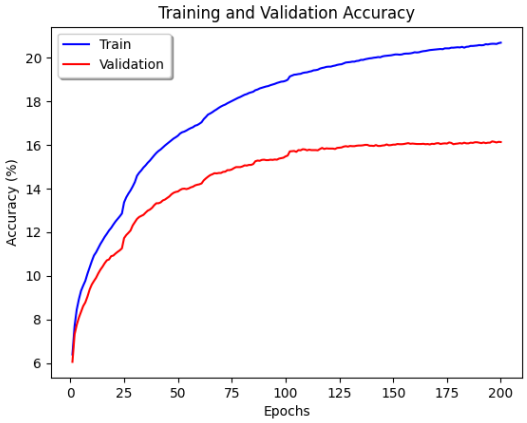
Le nombre de paramètres reste inchangeable. La fonction a été ajoute après la deuxième couche (de dimension 96 ici) et avant la sortie.

On peut voir que lorsqu'on ajoute la non-linéarité, on arrive à optimiser le modèle assez fortement en passant de 16.64% à 22.25% avec le meilleur choix de fonction.

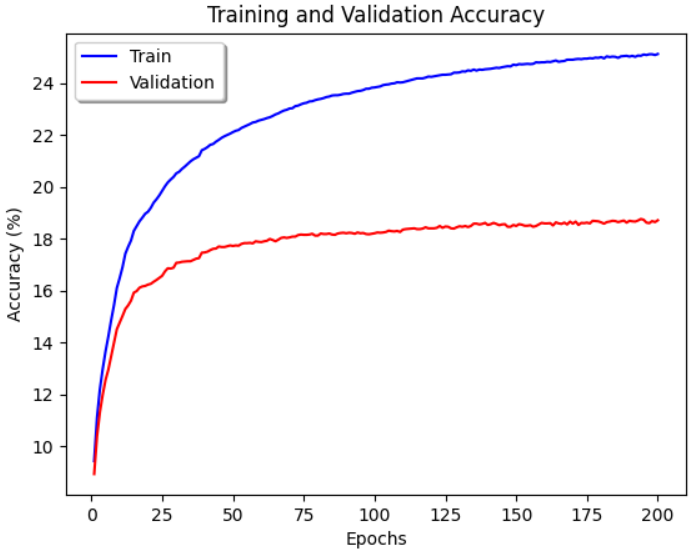
C'est aussi intéressant de regarder l'évolution et l'influence sur le modèle de différentes fonctions d'activations dans les learning curves sur la page suivante :



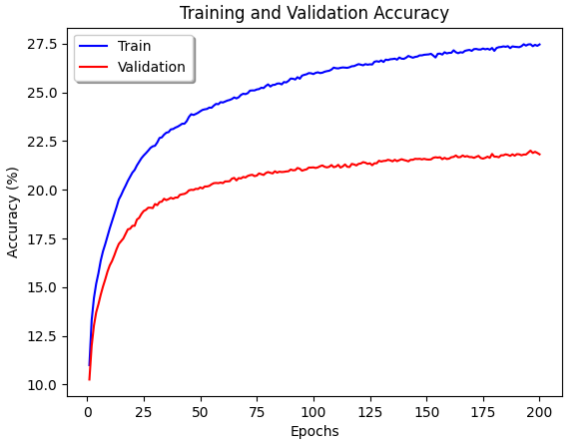
256 Linear 96 Linear



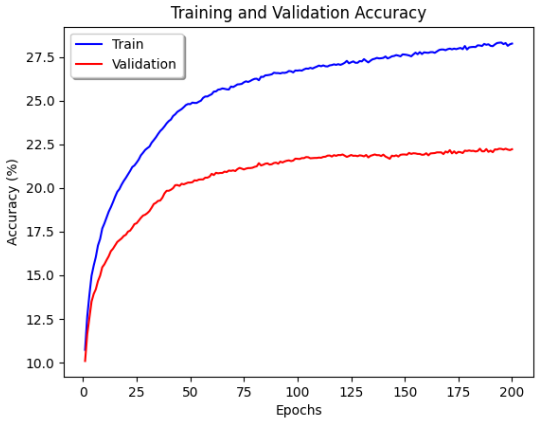
256 Linear 96 Sigmoid



256 Linear 96 Tanh



256 Linear 96 Leaky ReLU



256 Linear 96 ReLU

LSTM

Hidden Layer 1	Hidden Layer 2	Hidden Layer 3	Hidden Layer 4	Hidden Layer 5	Parameters	Accuracy (DEV)
96	-	-	-	-	68416	20.57%
96	96	-	-	-	77728	21.36%
96	96	96	96	96	105664	4.87%
96	192	192	192	192	204352	2.01%
96	256	256	256	256	300864	2.06%
128	-	-	-	-	107584	23%
128	96	96	-	-	127232	12.17%
128	96	96	96	-	136544	5.33%
128	96	96	128	-	141696	5.66%
128	96	96	192	-	152000	3.56%
128	96	96	256	-	162304	1.99%
128	96	96	96	96	145856	2.39%
128	96	96	96	128	151008	3.49%
128	96	96	96	192	161312	3.48%
128	96	96	96	256	171616	2.07%
128	96	96	128	96	152032	4.27%
128	96	96	128	128	158208	3.35%
128	96	96	128	192	170560	2.38%
128	96	96	128	256	182912	2.52%
128	96	96	192	96	164384	2.44%
128	96	96	192	128	172608	3.18%
128	96	96	192	192	189056	2.05%
128	96	96	192	256	205504	3.29%
128	96	96	256	96	176736	2.53%
128	96	96	256	128	187008	1.93%
128	96	96	256	256	228096	2.18%
128	96	128	96	96	152032	3.18%
128	192	-	-	-	136448	22.12%
128	256	-	-	-	148800	20.42%
128	256	256	-	-	214592	5.96%
128	256	256	256	-	346176	2.33%
192	-	-	-	-	210496	22.38%
256	-	-	-	-	346176	23%
256	96	-	-	-	360608	21.53%
256	128	-	-	-	370880	21.49%
256	128	128	128	128	420416	1.76%
256	256	-	-	-	411968	20.45%

Ici les modèles de 256 et 128 dimensions dans la couche première couche cachée ont été identiques par rapport à la précision, pour la suite, on va s'appuyer sur celui de 128 qui a besoin moins de paramètres et de temps de calcul.

On prend le modèle de base avec une seule couche cachée de dimension de 128 et on compare les fonctions d'activation :

Fonction d'activation	Sans fonction (Linear)	Sigmoid	Tanh	Leaky ReLU	ReLU
Précision	23%	14.45%	21.31%	20.1%	21.52%

Testing different optimizers

Pour tester et comparer les différents optimizers, on choisit les mêmes hyperparamètres que dans le modèle de base :

MLP :

- Input layer: 64 (8*8 – board size)
- Two hidden layers: 128 and 128 Linear layers
- Output layer: 64 (8*8 – board size)
- Learning rate: 0.001
- Dropout: 0.1
- Epoch: 200 (Early stopping – 20)
- Batch size: 1000
- Len Samples: 1 (one to one)
- Number of parameters: 33088

LSTM:

- Input layer: 64 (8*8 – board size)
- One hidden layer: 128 Linear layer
- Output layer: 64 (8*8 – board size)
- Learning rate: 0.005
- Dropout: 0.1
- Epoch: 200 (Early stopping – 20)
- Batch size: 1000
- Len Samples: 5 (sequence to one)
- Number of parameters: 107584

Ensuite, on obtient le tableau suivant :

Architecture	Precision de **Adam**	Precision de **SGD**	Precision de **RMSprop**	Precision de **Adagrad**	Precision de **Adadelta**
MLP	14.64%	1.58%	15.73%	11.31%	1.76%
LSTM	23%	2.36%	22.29%	14.54%	1.64%

Pour les deux architectures, les optimizers Adam et RMSprop sont plus intéressants que les autres. Mais bien évidemment avec les autres hyperparamètres on ne verra pas les résultats identiques, surtout le changement de learning rate peut altérer les résultats de manière assez important. De tel façon que les autres optimizers peuvent être plus performant que maintenant.

Optimizing learning rate

Pour tester et comparer les différents learning rate, on choisit les mêmes hyperparamètres que dans le modèle de base :

MLP :

- Input layer: 64 (8*8 – board size)
- Two hidden layers: 128 and 128 Linear layers
- Output layer: 64 (8*8 – board size)
- Optimizer: Adam
- Dropout: 0.1
- Epoch: 200 (Early stopping – 20)
- Batch size: 1000
- Len Samples: 1 (one to one)
- Number of parameters: 33088

LSTM:

- Input layer: 64 (8*8 – board size)
- One hidden layer: 128 Linear layer
- Output layer: 64 (8*8 – board size)
- Optimizer: Adam
- Dropout: 0.1
- Epoch: 200 (Early stopping – 20)
- Batch size: 1000
- Len Samples: 5 (sequence to one)
- Number of parameters: 107584

Architecture	Precision de **0.0001**	Precision de **0.001**	Precision de **0.005**	Precision de **0.01**	Precision de **0.1**	Precision de **1**
MLP	13.13%	14.64%	11.82%	7.69%	2.12%	1.76%
LSTM	14.85%	18.95%	23%	24.39%	6.65%	3.53%

The impact of different epochs and batch size

Epochs

Architecture	Precision de **50 epochs**	Precision de **100 epochs**	Precision de **200 epochs**	Precision de **300 epochs**	Precision de **500 epochs**
MLP	15.24%	16.12%	14.64%	15.98%	15.94%
LSTM	21.19%	20.91	23%	22.15%	20.93%

Batch size

Architecture	Precision de **100 batch size**	Precision de **1000 batch size**	Precision de **5000 batch size**	Precision de **15000 batch size**	Precision de **30000 batch size**
MLP	14.83%	14.64%	14.6%	13.74%	13.38%
LSTM	28.15%	23%	19.15%	17.52%	16.53%

Pour les epochs, les résultats sont difficilement interprétables, l'augmentation ou diminution des epochs ont les tendances différentes par rapport à l'accuracy. Alors que pour batch size, lorsqu'on les diminue, c'est-à-dire quand on renouvelle les weights plus fréquemment en traitant la quantité des samples moins importante, pour les hyperparamètres données, on obtient le meilleur résultat

Evaluation metric and New data generation

Parmi les meilleurs modèles, on a choisi ceux dont l'accuracy on DEV est supérieur à 14%. Chaque modèle choisi a confronté deux à deux tous les autres modèles performants en deux parties : un pour les pièces blanches et un pour les pièces noires afin d'avoir les résultats plus cohérents possible. Au total on a obtenu plus de 25.000 parties avec les conditions et deux adversaires divers.
On a stocké en format de fichier h5 les logs des parties et également, on a compté le taux de gagnant (win rate) de chaque modèle. C'est notre deuxième métrique d'évaluation. Les deux façons d'évaluer les modèles se différent de manière important, voici les tableaux comparatifs sur la page suivante :

Top 10 Win Rate

Model description	Accuracy	Win Rate
Dropout 0.1/LSTM/Adam/Learnings rate 0.001/Batch size 1000/Epoch 200/128 Linear 128 Linear 96 Linear 256 Linear	15.65%	75.89%
Dropout 0.1/MLP/Adam/Learnings rate 0.001/Batch size 1000/Epoch 500/128 Linear 128 Linear	15.94%	70.98%
Dropout 0.1/MLP/Adam/Learnings rate 0.001/Batch size 1000/Epoch 200/192 Linear 96 Linear 96 Linear	14.37%	70.54%
Dropout 0.1/LSTM/Adam/Learnings rate 0.001/Batch size 1000/Epoch 200/128 Linear 256 Linear 128 Linear 128 Linear	14.40%	70.54%
Dropout 0.1/LSTM/Adam/Learnings rate 0.001/Batch size 1000/Epoch 200/128 Linear 128 Linear 128 Linear 256 Linear	14.37%	70.09%
Dropout 0.1/MLP/Adam/Learnings rate 0.001/Batch size 1000/Epoch 200/192 Linear 96 Linear 128 Linear	14.21%	69.64%
Dropout 0.1/LSTM/Adam/Learnings rate 0.001/Batch size 1000/Epoch 200/128 Linear 128 Linear 128 Linear 192 Linear	16.20%	69.20%
Dropout 0.1/LSTM/Adam/Learnings rate 0.001/Batch size 1000/Epoch 200/128 Linear 128 Linear 192 Linear 96 Linear	14.76%	68.30%
Dropout 0.1/LSTM/Adam/Learnings rate 0.001/Batch size 1000/Epoch 200/128 Linear 192 Linear 128 Linear 96 Linear	16.23%	68.30%

Top 10 Accuracy

Model description	Accuracy	Win Rate
Dropout 0.1/LSTM/Adam/Learnings rate 0.005/Batch size 100/Epoch 200/128 Linear	28.15%	59.38%
Dropout 0.1/LSTM/Adam/Learnings rate 0.005/Batch size 100/Epoch 200/96 Linear	26.14%	51.79%
Dropout 0.1/LSTM/Adam/Learnings rate 0.01/Batch size 1000/Epoch 200/128 Linear	24.39%	57.14%
Dropout 0.1/LSTM/Adam/Learnings rate 0.005/Batch size 1000/Epoch 200/192 Linear	22.38%	58.93%
Dropout 0.1/LSTM/RMSprop/Learnings rate 0.005/Batch size 1000/Epoch 200/128 Linear	22.29%	46.43%
Dropout 0.1/LSTM/Adam/Learnings rate 0.005/Batch size 1000/Epoch 200/128 Linear 256 Linear	20.42%	58.04%
Dropout 0.1/LSTM/Adam/Learnings rate 0.01/Batch size 5000/Epoch 100/256 ReLU	20.36%	41.96%
Dropout 0.1/MLP/Adam/Learnings rate 0.0001/Batch size 1000/Epoch 200/256 Linear 192 ReLU	19.85%	34.38%
Dropout 0.1/LSTM/Adam/Learnings rate 0.005/Batch size 5000/Epoch 200/192 Linear	19.81%	53.13%

Dans les tableaux en haut on voit que l'accuracy et win rate sont deux métriques d'évaluations différents qui déterminent de manière différentes le classement de modèles.

On a extrait les meilleurs modèles de LSTM et MLP selon les métriques différentes dans le tableau suivant :

Model description	Accuracy	Win Rate
Dropout 0.1/LSTM/Adam/Learnings rate 0.005/Batch size 100/Epoch 200/128 Linear	**28.15%**	**59.38%**
Dropout 0.1/MLP/Adam/Learnings rate 0.0001/Batch size 1000/Epoch 200/256 Linear 192 ReLU	**19.85%**	**34.38%**
Dropout 0.1/MLP/Adam/Learnings rate 0.001/Batch size 1000/Epoch 500/128 Linear 128 Linear	**15.94%**	**70.98%**
Dropout 0.1/LSTM/Adam/Learnings rate 0.001/Batch size 1000/Epoch 200/128 Linear 128 Linear 96 Linear 256 Linear	**15.65%**	**75.89%**

***Color of the best models regarding Accuracy**
***Color of the best models regarding Win Rate**

Pour le dataset fourni on prend en considération ces 4 modèles afin de les entrainer avec data généré par les logs de joueurs AI. On a ajouté donc plus de 25.000 fichiers h5 qui stockent les données sous format de matrice (2, 60, 8, 8) :

Since we have only complete games, The shape of game array is (2,60,8,8).

- **8,8 :** 8×8 array represent a board status or a move
- **60 :** represent series of board status or moves. The first move is done by black and next by white, and it turns alternate one by one. Since the game board contain 8×8 or 64 cells and the game starts with 4 occupied cells, it requires 60 moves (30 moves by each player).
- **2 :** represent one series of array for board status and one for moves.

Enfin on compare ces 4 modèles avec le même dev en tant que dataset de validation pour évaluer accuracy (on n'a changé que training set) et on simule plus de 200 jeux pour chaque modèle avec les mêmes adversaires AI qu'avant. On obtient les résultats suivants.

Model description	Training dataset			
	Provided		Provided & Generated	
	Accuracy	Win Rate	Accuracy	Win Rate
Dropout 0.1/LSTM/Adam/Learnings rate 0.005/Batch size 100/Epoch 200/128 Linear	**28.15%**	**59.38%**	**17.83%**	**71.81%**
Dropout 0.1/MLP/Adam/Learnings rate 0.0001/Batch size 1000/Epoch 200/256 Linear 192 ReLU	**19.85%**	**34.38%**	**10.59%**	**62.08%**
Dropout 0.1/MLP/Adam/Learnings rate 0.001/Batch size 1000/Epoch 500/128 Linear 128 Linear	**15.94%**	**70.98%**	**5.14%**	**70.13%**
Dropout 0.1/LSTM/Adam/Learnings rate 0.001/Batch size 1000/Epoch 200/128 Linear 128 Linear 96 Linear 256 Linear	**15.65%**	**75.89%**	**9.69%**	**73.15%**

Dans le tableau comparatif on voit qu'il y a une tendance d'amélioration de win ratio pour les modèles qui a eu la meilleure précision si on applique une nouvelle dataset. Pour les leaders de win rate d'avant, cette règle ne s'applique pas et le win rate a diminuée mais pas dramatiquement.

Conclusion

On a effectué le travail assez complet et complexe de développement des modèles d'intelligence artificiel en changeant les hyperparamètres et surtout en rapportant pleins d'informations et logs qu'on a analysés et utilisés. Neanmoins, il reste plein de choses qu'on peut encore faire, par exemple, implémenter et tester, CNN, la gamme plus complète de hyperparamètres pour LSTM et MLP. Pendant ce TP on a développé compétences et connaissances concernant l'intelligence artificiel, il est aussi intéressant de commencer à comprendre que dans l'IA il y a toujours un moyen d'améliorer les modèles en appliquant les méthodes divers et variés aux différents niveaux : soit au modèle, soit à dataset.

Annexe

https://docs.google.com/spreadsheets/d/1eaxj0kavYpnIbLGCjsHDVBAZVa2LOzPb5vifCtBMcVQ/edit?usp=sharing – Tableau de tous les modèles qui ont participé en compétition.