Réseaux de neurones convolutifs avec des noyaux spécifiques pour le jeu d'échecs

PFIA 2025

Raphaël Mathiot [2] Olivier Goudet[1] Tristan Cazenave[2]

[1] LERIA, Université d'Angers, 2 Boulevard Lavoisier, Angers 49045, France

[2] LAMSADE, Université Paris Dauphine - PSL, CNRS, Paris, France

Juin 2025

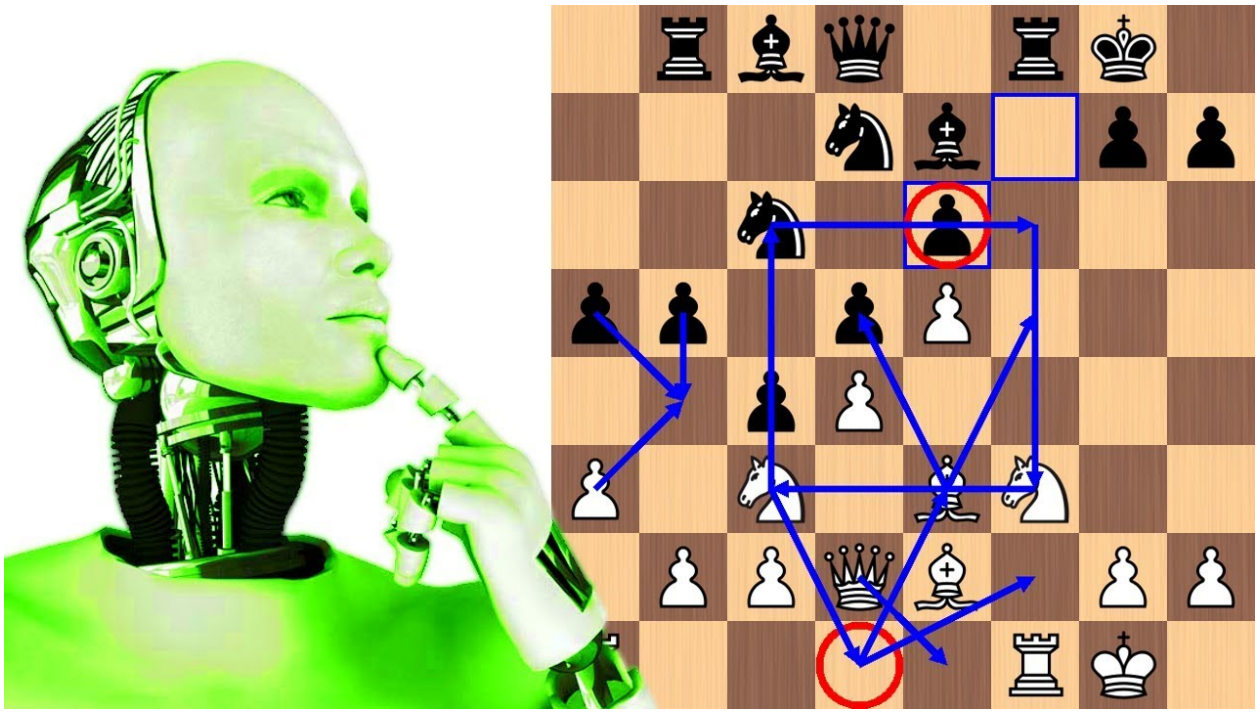
Plan

Section 1

Contexte - réseaux de neurones dans les programmes d'échecs

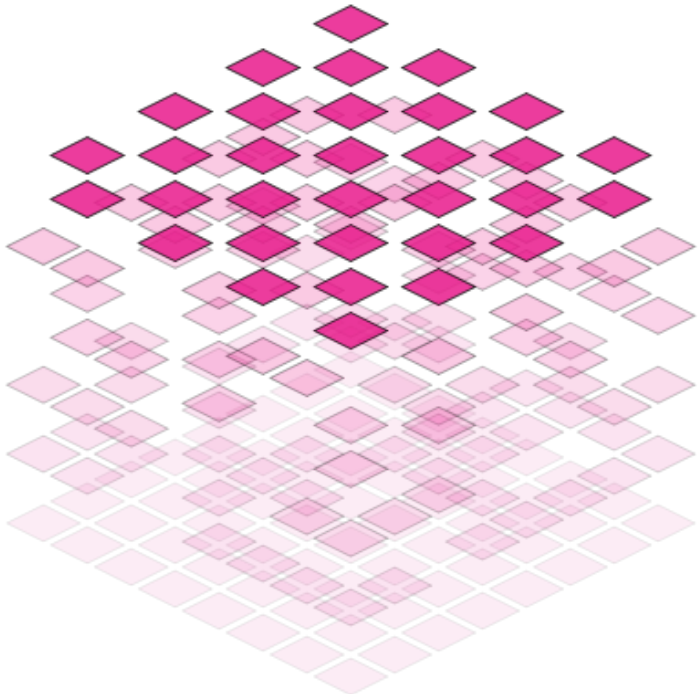
Le programme d'échecs AlphaZero

- AlphaZero a atteint un niveau dépassant largement celui des meilleurs grands maîtres - Silver et al. (2017, 2018).
- Il a appris à jouer seul, sans utiliser de connaissances d'experts.



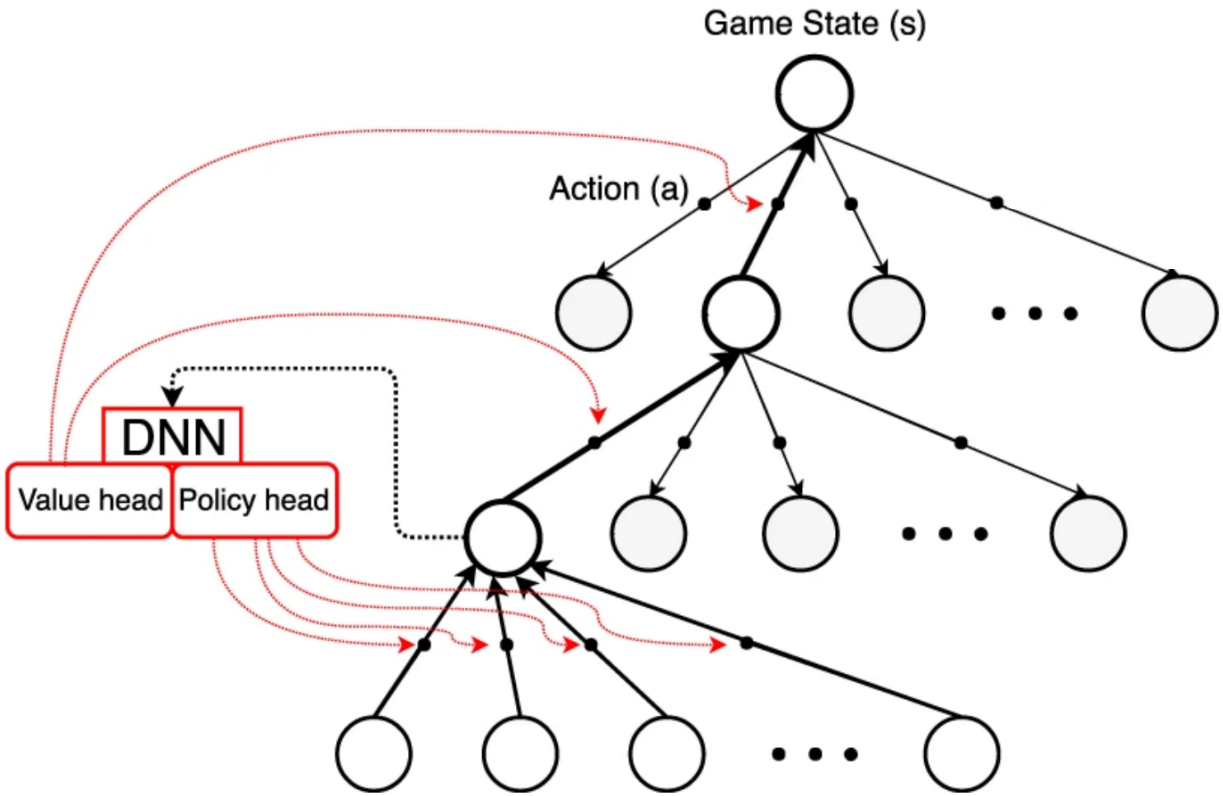
LeelaChess Zero (2018)

- Une adaptation gratuite et open-source d'AlphaZero pour le jeu d'échecs.
- Auteurs principaux : Gian-Carlo Pascutto, Gary Linscott.
- `https://lczero.org/`.



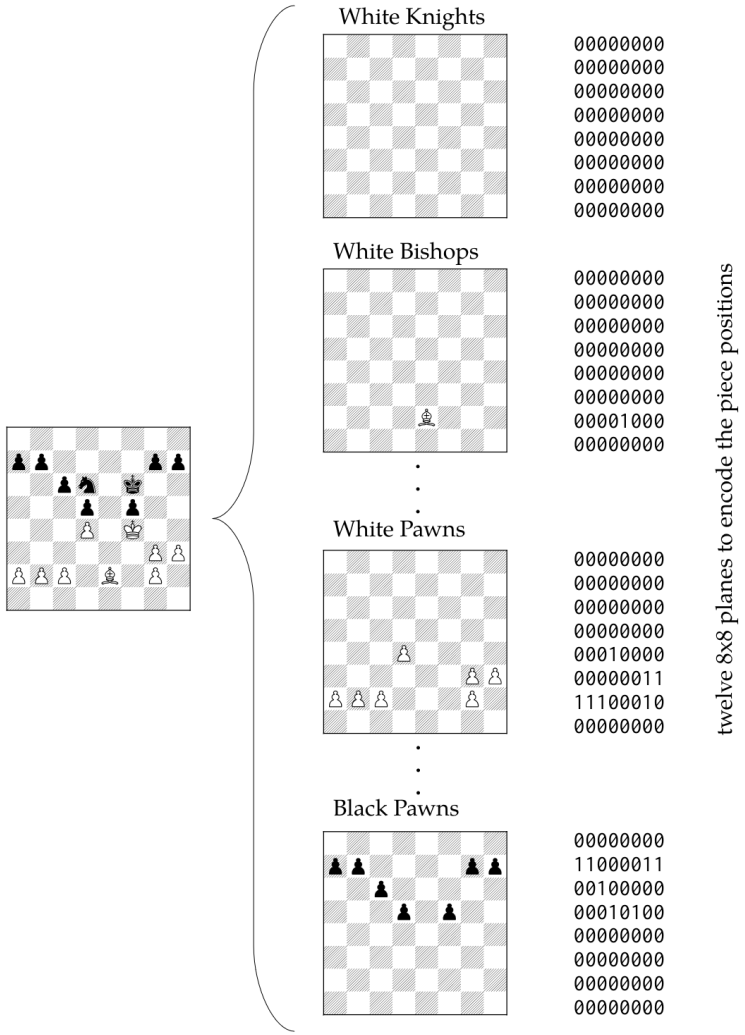
AlphaZero et Lc0

- Une méthode d'apprentissage profond par renforcement, combinant les réseaux de neurones avec la recherche arborescente Monte-Carlo (Monte-Carlo Tree Search).

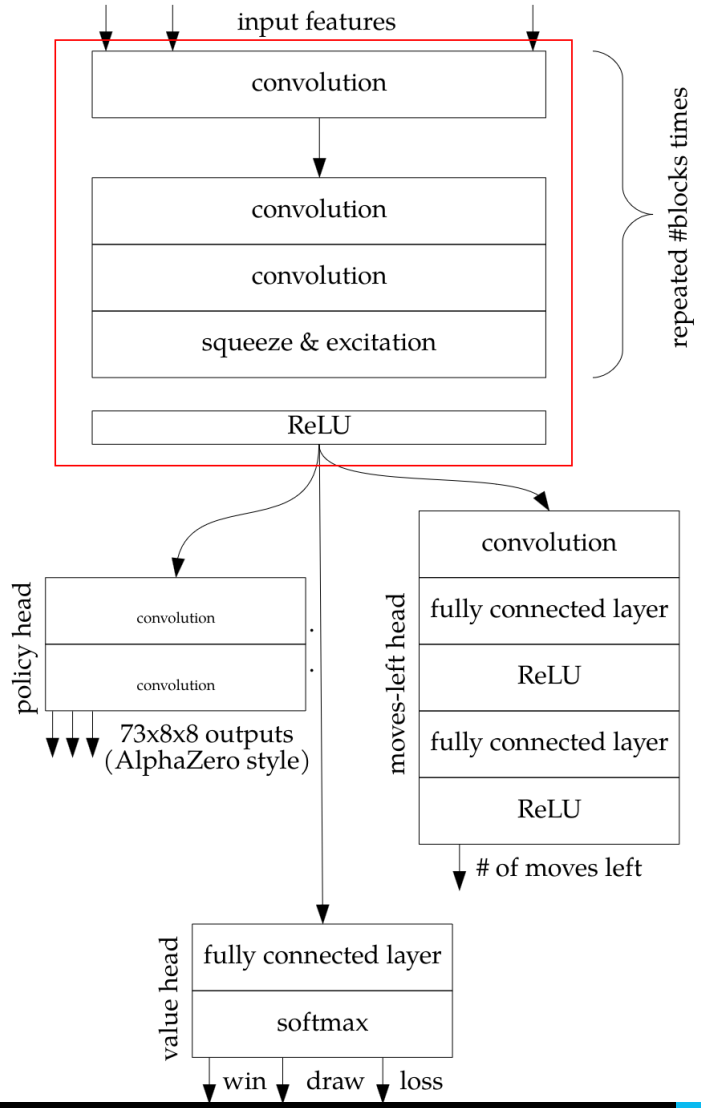


Input du réseau de neurones

Une image de taille 8×8 avec 112 canaux.



Architecture du réseau de neurones résiduel utilisé par Lc0

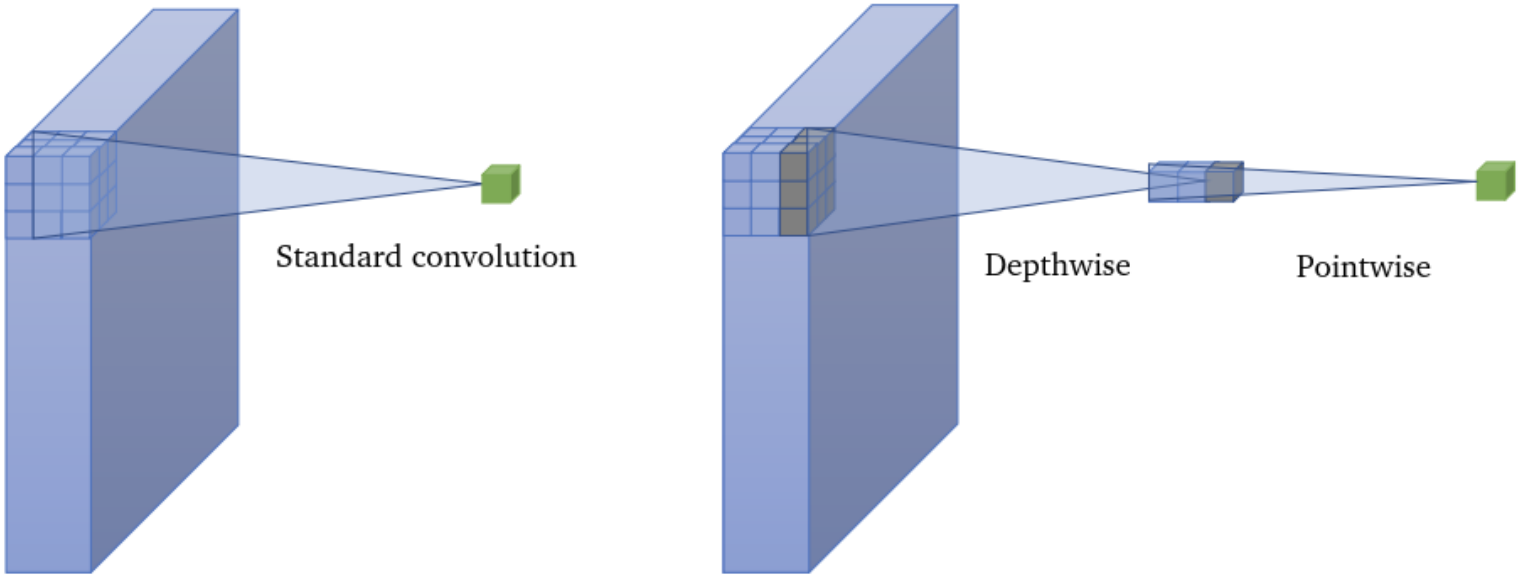


Section 2

Plusieurs architectures de réseaux de neurones

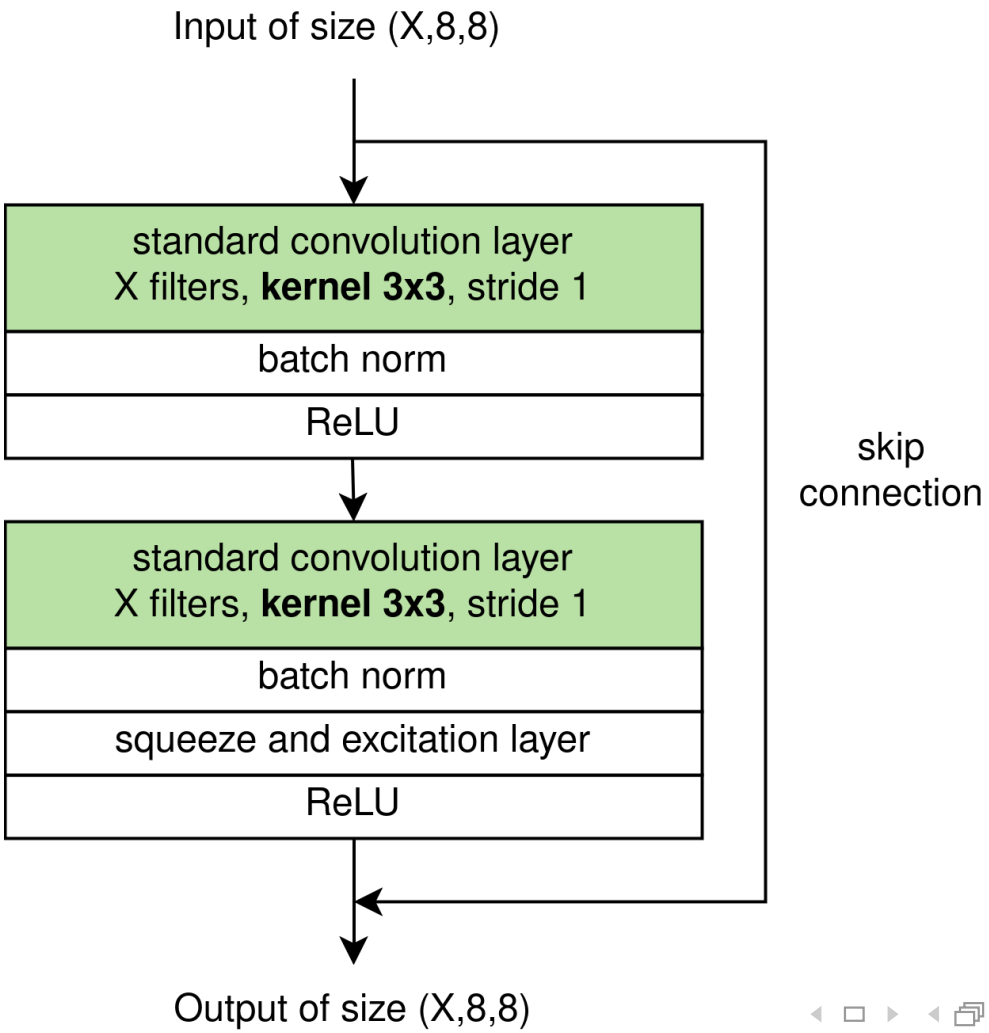
Convolutions standard et séparable en profondeur

- Lc0 utilise des convolutions standard avec noyau de taille 3×3.
- Alternative : la convolution séparable en profondeur (Sifre and Mallat, 2014), combinée à une convolution point par point.



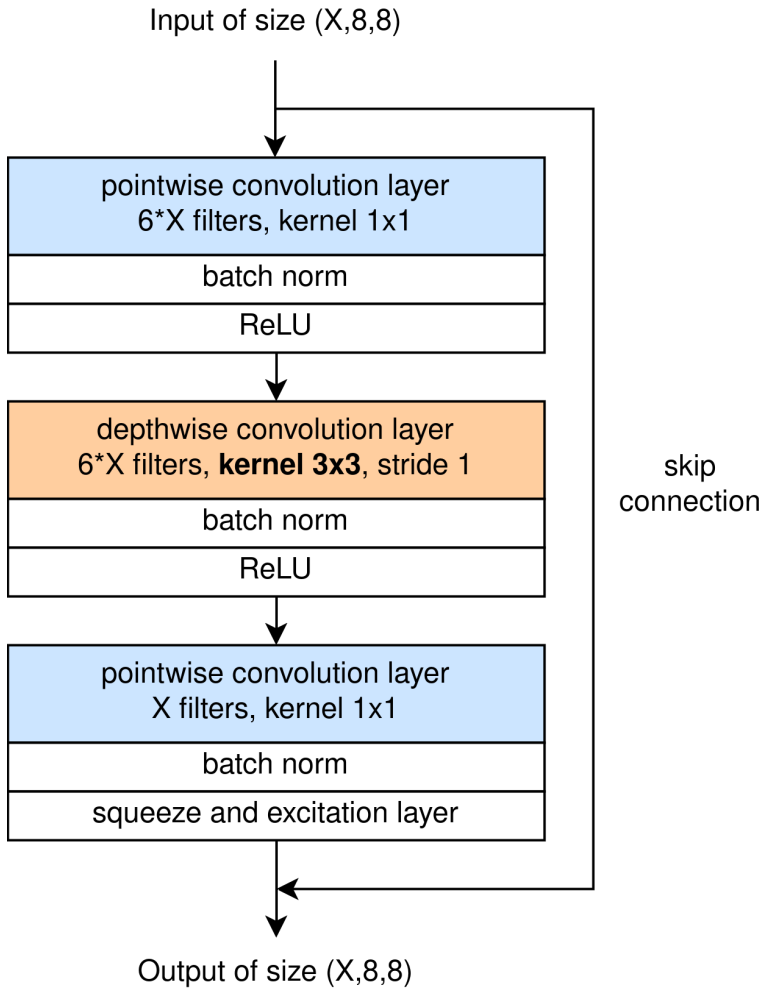
Bloc résiduel utilisé par Lc0

Dans la suite de ce travail, nous avons utilisé des noyaux de convolution de taille 3×3 ou 5×5.



Bloc Mobile Net

L'architecture Mobile Net (Sandler et al., 2018) a été appliquée au jeu de Go avec de très bons résultats par (Cazenave, 2020).

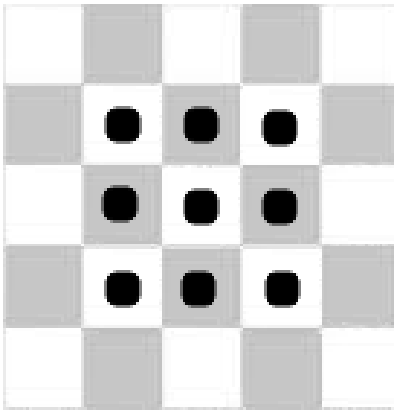


Section 3

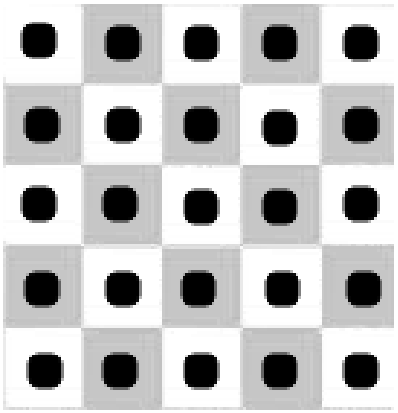
Noyaux spécifiques aux échecs

Noyaux classiques de taille 3×3 et 5×5 utilisés dans les couches de convolution

Noyau de convolution 3×3 classique

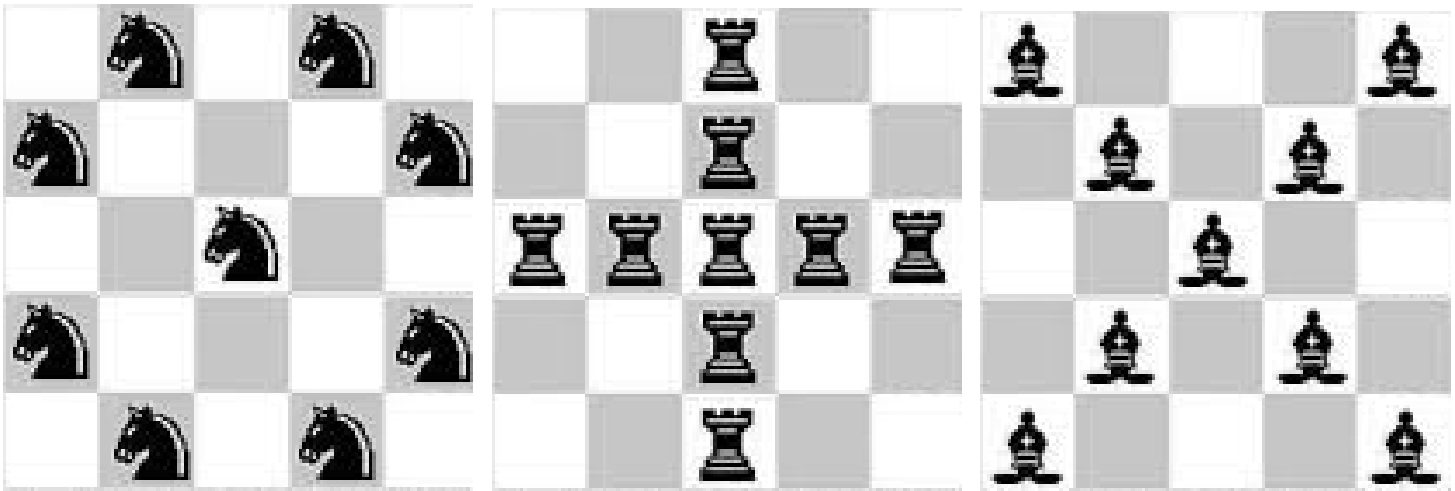


Noyau de convolution 5×5 classique



Nouveaux noyaux de convolution pour le jeu d'échecs

- Noyaux d'échecs de taille 5×5 : masques inspirés des mouvements des cavaliers, des tours et des fous.
- **Chaque noyau d'échecs a autant de paramètres non masqués qu'un noyau 3x3 classique.**



Trois types de filtres combinés dans une couche de convolution en profondeur

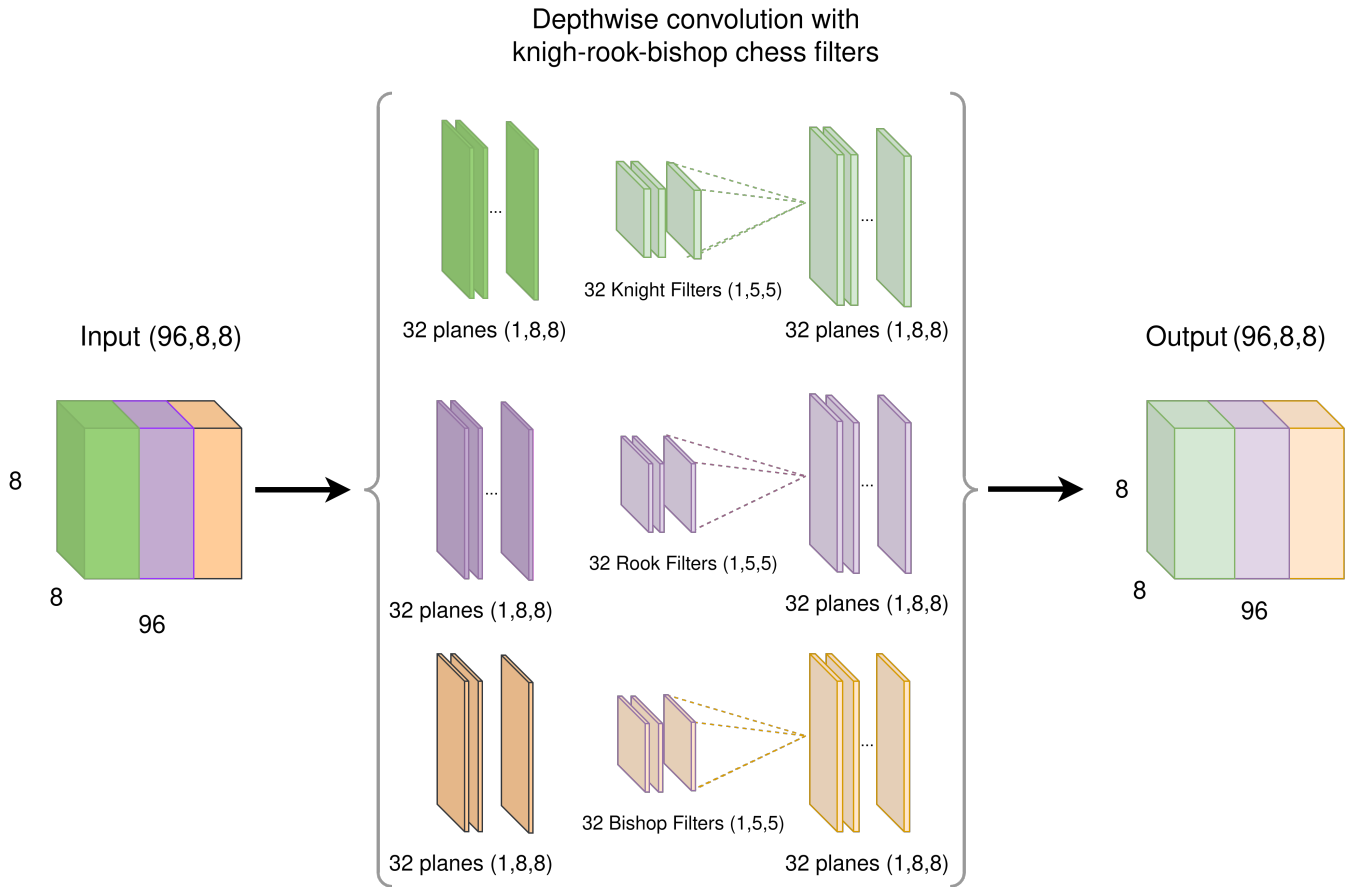


Figure 1 – Convolution en profondeur avec des filtres cavalier-tour-fou appliqués au tenseur en input de taille $(96, 8, 8)$.

Section 4

Caractéristiques des réseaux de neurones

Nombre de paramètres des configurations testées avec kernel 5x5

		Réseaux résiduels		**Mobile nets**	
Nb blocs	Nb filtres	Nb params (M.)		Nb params (M.)	
		Standard	Filtres d'échecs	Standard	Filtres d'échecs
6	96	7.195	**5.426**	4.921	**4.865**
12	96	10.046	**6.507**	5.755	**5.645**
18	96	12.897	**7.589**	6.590	**6.424**
6	192	16.017	**8.939**	7.265	**7.154**
12	192	27.414	**13.258**	10.427	**10.206**
18	192	38.811	**17.578**	13.589	**13.257**

Section 5

Entraînement des réseaux de neurones

Jeu de données d'entraînement

- Dataset : un million de parties récentes jouées par lc0.
- 80% pour l'entraînement et les 20 derniers % pour la validation.
- Entraînement d'un réseau de neurones sur des parties déjà jouées par lc0 en essayant de prédire directement :
  - Le meilleur coup joué par lc0 (après exploration de l'arbre).
  - La valeur de l'état courant.
  - Le nombre de coups restant jusqu'à la fin de la partie.
- Pour chaque configuration du réseau, 5 modèles sont entraînés et leurs résultats agrégés.

Résultats

Figure 2 – Précision des coups joués et perte MSE sur l'ensemble de validation pour les architectures avec 18 blocs, 96 filtres et des noyaux de taille 5×5.

Section 6

Comparaison du niveau de jeu de Lc0 avec les différentes architectures de réseau de neurones

Motivation et hypothèses

- En observant l'amélioration des différentes métriques avec l'introduction des architectures Mobile Nets, et notamment avec l'ajout des noyaux d'échecs, on peut espérer que ces résultats se traduisent par une augmentation du niveau de jeu pour un **nombre égal de nœuds évalués**.
- Lorsqu'on introduit un temps de jeu limité, un autre paramètre entre en compte : le **temps d'inférence**. En moyenne, une architecture mobile net évalue deux à trois fois moins de nœuds qu'une architecture avec bloc résiduels. Cela procurera donc un net avantage au bloc résiduel dans ce type de match.
- La question serait alors de savoir comment **s'équilibrent** ces deux effets inverses : la qualité de la fonction d'évaluation du Mobile Net suffit-il à compenser le plus faible nombre de nœuds évalués ?

Objectifs et cadre du tournoi

- Deux formats de tournoi différents
  - **Tournoi 1** : taille de l'arbre de recherche limitée à 20000 nœuds, afin de comparer uniquement la qualité de la fonction d'évaluation, sans prendre en compte les différences de temps d'inférence.
  - **Tournoi 2** : temps limité à 60 secondes pour l'intégralité de la partie, avec 0.6 seconde d'incrément par coup, afin de prendre en compte la rapidité des réseaux.
- Nous allons faire s'affronter des réseaux de tailles semblables, en termes de nombre de blocs et de taille de chaque bloc. Pour les résultats qui suivent, les architectures sélectionnées ont 18 blocs avec 96 filtres.
- Les premiers coups sont imposés afin d'entraîner des parties déséquilibrées, et chaque rencontre comporte un match aller et un match retour avec échange de couleurs.

Modèles en compétition : 96x18

- Résiduel 3x3 : modèle résiduel classique avec noyau standard 3x3.
- Résiduel 5x5 : modèle résiduel avec noyau standard 5x5.
- Résiduel BR : modèle résiduel avec noyau d'échecs tour-fou 5x5.
- Résiduel BRK : modèle résiduel avec noyau d'échecs cavalier-tour-fou 5x5.
- Mobile Net : modèle Mobile Net avec noyau standard 5x5.
- Mobile Net BRK : modèle Mobile Net avec noyau d'échecs cavalier-tour-fou 5x5.

Résultats du premier tournoi : 20.000 noeuds maximum

Classement	Nom	Score	Elo relatif	Intervalle de confiance
1	Mobile Net BRK	74.2%	183.90	22.12
2	Résiduel BR	52.6%	17.91	20.52
3	Mobile Net	50.6%	4.21	20.76
4	Résiduel 3x3	44.8%	-35.92	20.99
5	Résiduel 5x5	41.4%	-60.08	21.03
6	Résiduel BRK	36.3%	-97.78	20.47

- Domination de la part du réseau Mobile Net avec filtres d'échecs, qui gagne 74,2% de ses parties, et a un classement Elo estimé à 165 points de plus que celui du second.
- On retrouve l'ordre observé pour les métriques précédentes : le noyau d'échecs cavalier-tour-fou offre les meilleures performances parmi les Mobile Nets, tandis que le noyau tour-fou est le meilleur parmi les résiduels.

Résultats du second tournoi : 60 secondes + 0.6 seconde d'incrément

Classement	Nom	Score	Elo relatif	Intervalle de confiance
1	Mobile Net BRK	70.8%	154.14	46.13
2	Résiduel 3x3	58.3%	58.45	47.50
3	Résiduel BR	55.0%	34.86	45.25
4	Mobile Net 3x3	45.4%	-31.94	51.24
5	Résiduel 5x5	37.1%	-91.83	55.27
6	Résiduel BRK	33.3%	-120.41	50.47

- Comme attendu, l'écart entre Mobile Net et résiduel diminue par rapport au tournoi précédent (100 elo de différence contre 160 avant).
- Mais le réseau Mobile Net garde l'avantage : il semblerait que sa meilleure fonction d'évaluation compense sa latence importante.

Exemple de partie jouée entre Mobile Net et résiduel

- https://lichess.org/YEE8ChZA#0

Conclusion

- Remplacer les blocs résiduels standard par des blocs Mobile Net :
  - Améliore la précision des coups et de l'évaluation.
  - Augmente la latence et la mémoire requise pour l'entraînement et l'inférence.
- Les noyaux d'échecs réduisent le nombre de paramètres tout en améliorant les résultats.
- Imposer une structure tirée du mouvement des pièces dans les couches de convolution semble être utile pour le programme d'échecs.
- Cette augmentation de la précision se traduit directement par une amélioration conséquente du niveau de jeu.
- Mais la faiblesse du Mobile Net réside dans sa latence, qui pourrait être améliorée par une implémentation bas-niveau d'un kernel CUDA fusionnant la convolution en profondeur et la convolution point par point.

Réferences

Cazenave, T. (2020). Mobile networks for computer go. *IEEE Transactions on Games*, 14(1) :76–84.

Qararyah, F., Azhar, M. W., Maleki, M. A., and Trancoso, P. (2024). Fusing depthwise and pointwise convolutions for efficient inference on gpus. page 58–67.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2 : Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Sifre, L. and Mallat, S. (2014). Rigid-motion scattering for texture classification. *arXiv preprint arXiv :1403.1687*.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv :1712.01815*.

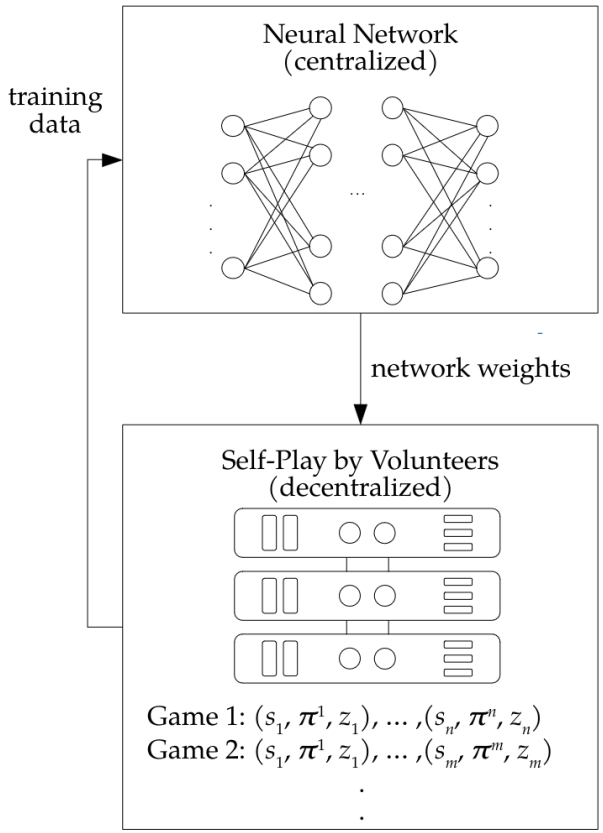
Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419) :1140–1144.

Input du réseau de neurones

- Une position d'échecs est convertie en un tenseur avec 112 plans de taille 8×8 (encodage classique de Lc0) :
  1. Les 6 premiers plans sont dédiés aux pièces du joueur dont c'est le tour, et les 6 suivants à celles de son adversaire.
  2. Le treizième plan contient des 1 si une ou plusieurs répétitions ont eu lieu.
  3. Ces 13 plans sont répétés 8 fois pour encoder la position courante, ainsi que les 7 dernières positions.
  4. Les 8 derniers plans encodent des informations supplémentaires, comme les droits de roque de chaque côté.

Structure du programme Lc0

- Deux programmes principaux :
  1. Un code en python avec la librairie Tensorflow pour l'entraînement du réseau de neurones.
  2. Un code en C++ avec la librairie Cuda, pour faire jouer le programme.



Caractéristiques des réseaux résiduels

Réseau résiduel avec des noyaux de taille 5×5					
Nb blocs	Nb filtres	Nb params (M.)		Mémoire (GiB)	Latence (s)
		Classique	Filtres d'échecs		
6	96	7.195	**5.426**	2.357	0.0940
12	96	10.046	**6.507**	2.357	0.108
18	96	12.897	**7.589**	2.357	0.126
6	192	16.017	**8.939**	4.405	0.121
12	192	27.414	**13.258**	4.405	0.165
18	192	38.811	**17.578**	4.405	0.211

- L'implémentation actuelle des noyaux d'échecs est suboptimale : calcul d'un noyau 5×5 classique en fixant à 0 les paramètres n'appartenant pas au masque d'échecs.
- Avec une implémentation bas niveau du noyau d'échecs, une couche de convolution utilisant un noyau d'échecs de taille 5×5 devrait être presque aussi rapide qu'avec un noyau classique de taille 3×3.

Caractéristiques des réseaux Mobile nets

Réseau Mobile net avec des filtres de taille 5×5					
Nb blocs	Nb filtres	Nb params (M.)		Mémoire (GiB)	Latence (s)
		Classique	Filtres d'échecs		
6	96	4.921	4.865	3.637	0.103
12	96	5.755	5.645	3.637	0.133
18	96	6.590	6.424	3.637	0.172
6	192	7.265	7.154	6.965	0.144
12	192	10.427	10.206	6.965	0.218
18	192	13.589	13.257	6.965	0.294

- Beaucoup moins de paramètres que dans les blocs résiduels grâce à la convolution séparable en profondeur, mais la latence n'est pas meilleure.
- Une implémentation bas niveau de la convolution séparable en profondeur améliorerait ce temps d'inférence.

Détails des métriques utilisées

- Précision moyenne de la *policy* : pourcentage de fois où le coup avec la plus grande probabilité calculée par la *policy head* correspond exactement à celui sélectionné en appliquant la recherche Monte-Carlo.
- Perte MSE moyenne : erreur des moindres carrés moyenne entre l'output z_i de la *value head* et la valeur cible v_i, pour chaque position de l'ensemble de validation. Pour une position i donnée, à partir du vecteur de probabilités $(p_i^{win}, p_i^{draw}, p_i^{loss})$ émis par la *value head*, la valeur z_i est calculée de la manière suivante : $z_i = p_i^{win} - p_i^{loss}$.

Résultats supplémentaires : impact des noyaux d'échecs pour les réseaux résiduels

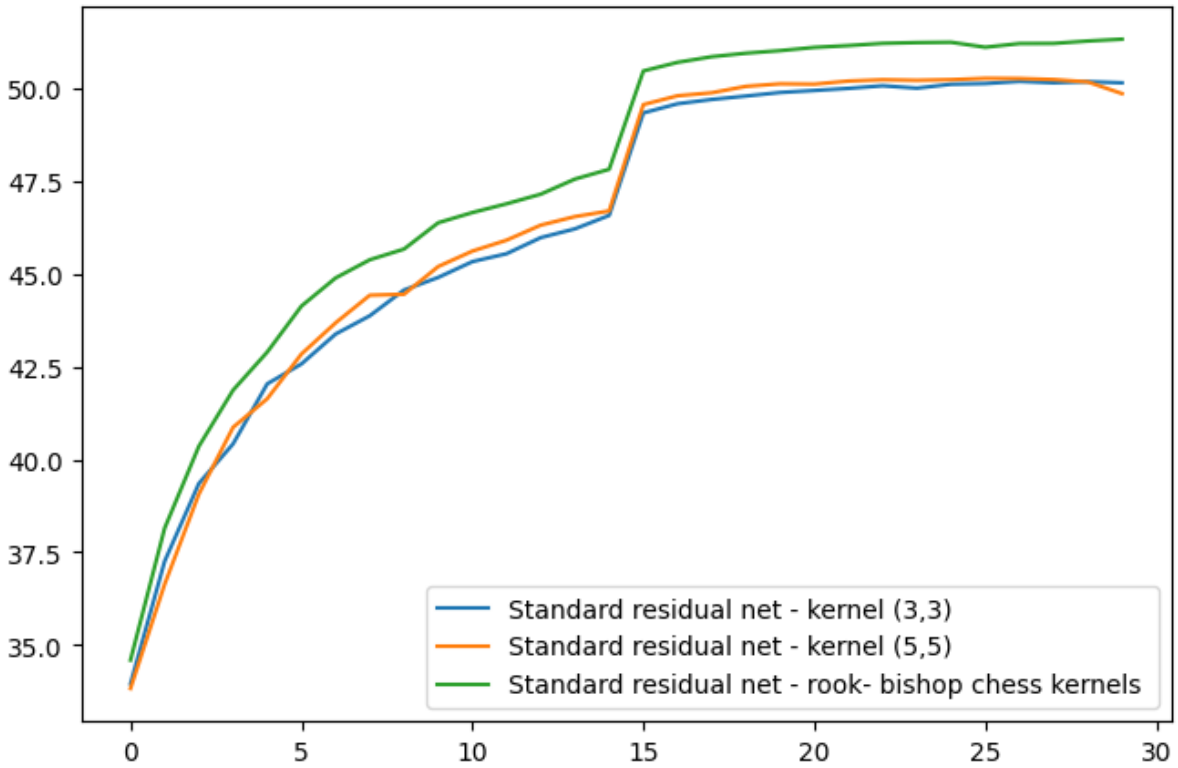


Figure 3 – Evolution de la précision de la *policy* sur l'ensemble de validation pour un réseau résiduel avec 12 blocs et 384 filtres, en fonction du type de noyau.

Résultats supplémentaires : impact des noyaux d'échecs pour les réseaux Mobile Nets

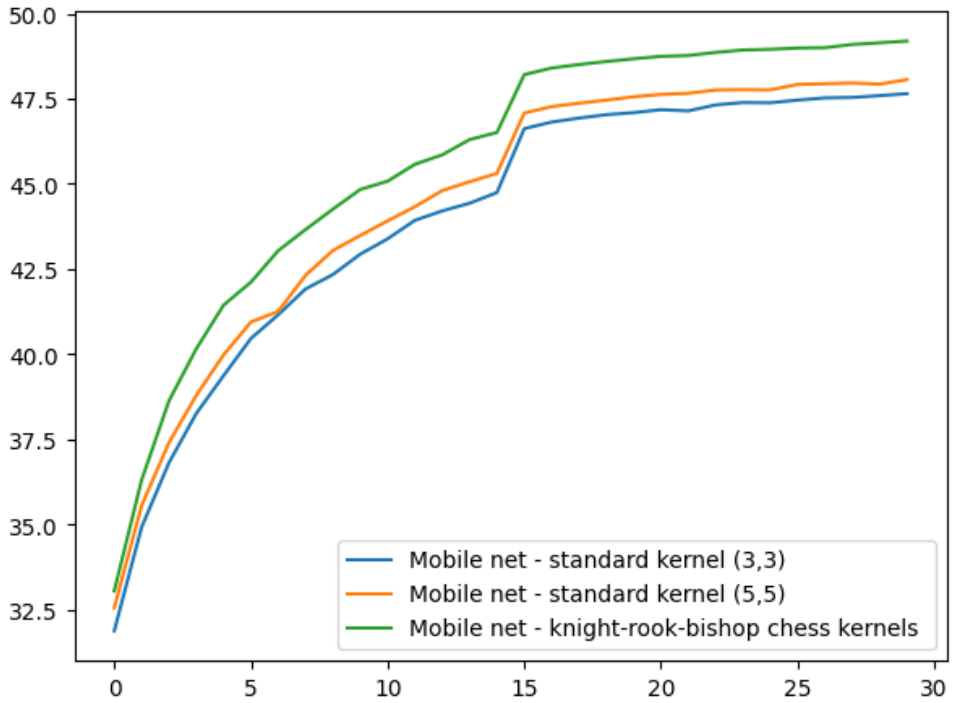


Figure 4 – Evolution de la précision de la *policy* sur l'ensemble de validation pour un réseau Mobile Net avec 12 blocs et 128 filtres (728 en expansion), en fonction du type de noyau.

Impact des différents noyaux d'échecs

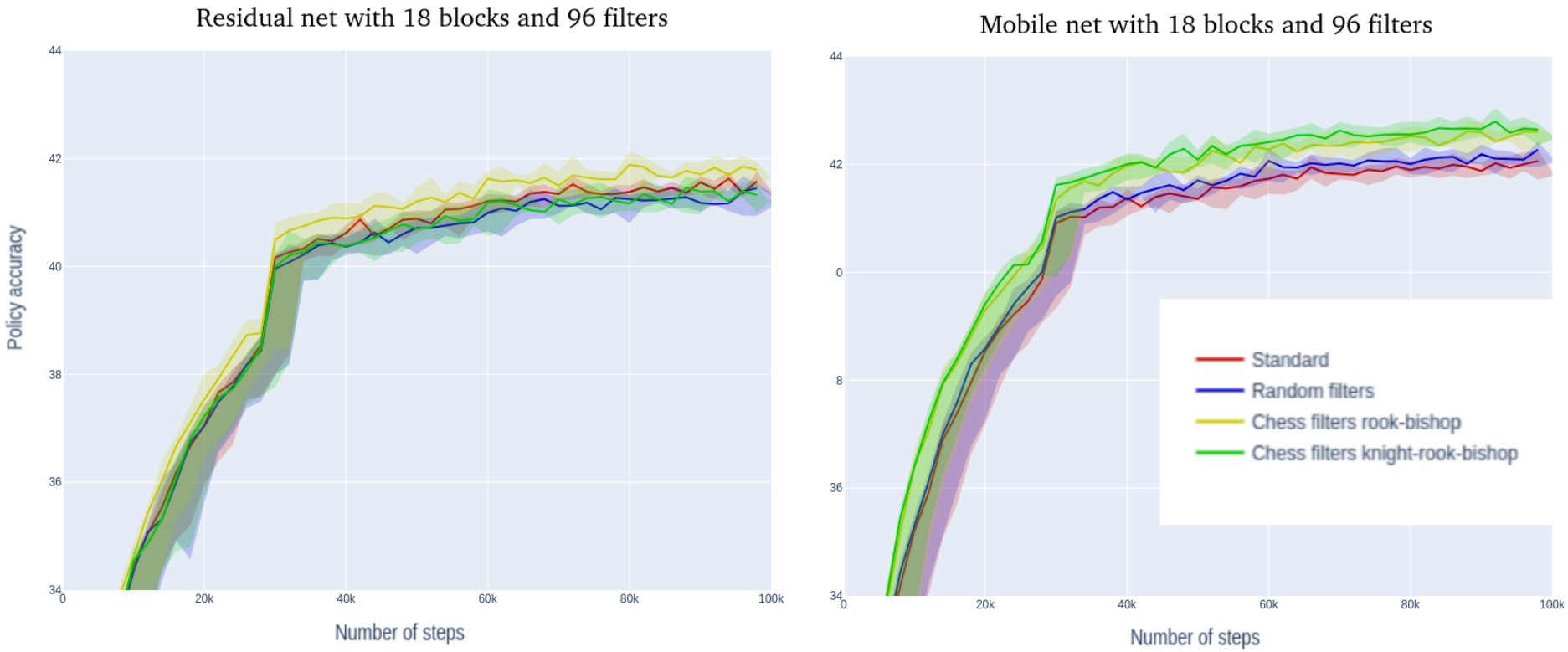


Figure 5 – Évolution de la précision des coups joués sur l'ensemble de validation pour le réseau résiduel (à gauche) et le Mobile Net (à droite), avec 18 blocs et 96 filtres, pour différents types de noyaux.

Motivation pour une implémentation bas-niveau

- Ces résultats montrent que le Mobile Net peut surpasser un réseau résiduel beaucoup plus rapide.
- La prochaine étape de notre travail consiste donc à essayer de réduire la latence du Mobile Net afin d'améliorer encore ses performances dans des matches avec temps limité.
- Même si ce n'est pas le facteur principal de cette lenteur, cela permettrait de ne pas stocker ni faire de calculs inutiles avec les 16 poids sur 25 qui sont masqués par les noyaux d'échecs.

Problème de l'implémentation actuelle

Pourquoi le Mobile Net a-t-il une si grande latence ? Le problème vient des nombreux accès à la mémoire globale, qui est très lente. Pour 96 filtres :

- Convolution en profondeur ($576 \longrightarrow 576$) : $576 \times 8 \times 8 = 36.864$ éléments à écrire en mémoire globale.
- Convolution point par point ($576 \longrightarrow 96$) : ces 36.864 éléments doivent être lus en mémoire globale par les 96 filtres. Cela fait $96 \times 576 \times 8 \times 8 = 3.538.944$ lectures en mémoire globale !
- Pour une position uniquement ! En pratique, Lc0 évalue plusieurs dizaines de positions simultanément, donc ces valeurs peuvent être multipliées par 10, voire 100.

Implémentation bas-niveau : fusion de la convolution en profondeur et point par point

- L'objectif est d'écrire un kernel CUDA, c'est-à-dire une fonction exécutée en parallèles par les milliers de threads composant le GPU.
- Enchaîner les deux opérations dans le même kernel (Qararyah et al., 2024) : les résultats intermédiaires de la convolution en profondeur sont stockés dans la **mémoire partagée** par un bloc (groupe de threads), qui est beaucoup plus rapide que la mémoire globale.
- Problème : la mémoire partagée a une capacité autour de 48kB par bloc, ce qui ne suffit pas à stocker les 36.864 éléments.
- Tiling ! Chaque bloc de threads s'occupe d'une portion de l'échiquier. Par exemple, en divisant l'échiquier en 4 carrés de côté 4, on passe à $576 \times 4 \times 4 = 9216$ éléments, ce qui rentre dans la mémoire partagée.