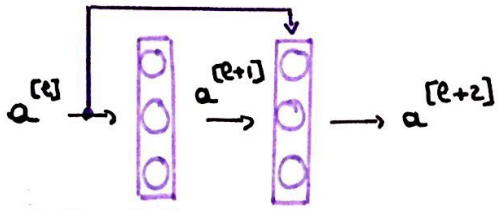
ResNet

Deep neural network
=> difficile à entrainer
=> Exploding et Vanishing gradient
=> Solution : skipping connections

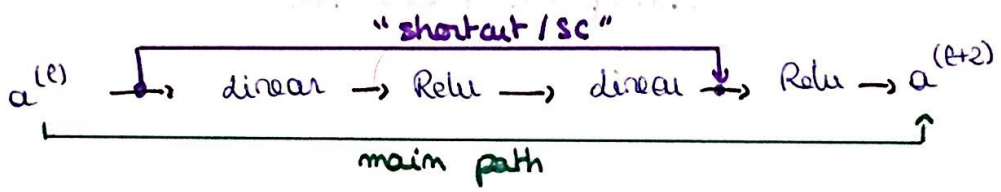
Residual block



N'impacte pas les performances

On utilise souvent des "same" conv.

$$z^{[l+1]} = w^{[l+1]} a^{[l]} + b^{[l+1]}$$
$$a^{[l+1]} = g(z^{[l+1]}) \quad \text{Relu}$$
$$z^{[l+2]} = w^{[l+2]} a^{[l+1]} + b^{[l+2]}$$
$$a^{[l+2]} = g(z^{[l+2]}) \quad \text{Relu}$$

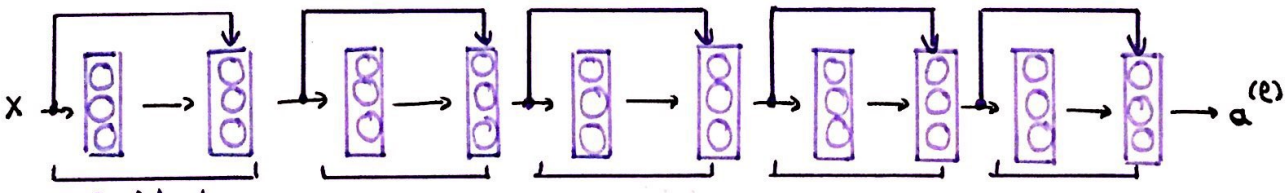


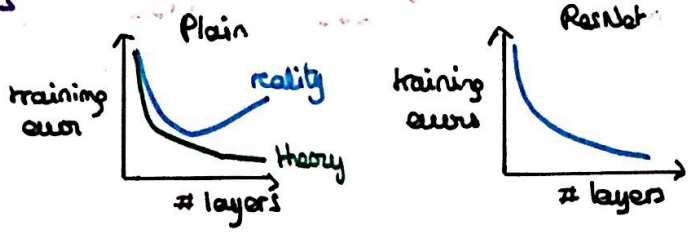
$$a^{[l+2]} = g(z^{(l+2)} + a^{(l)})$$

on ajoute $a^{(l)}$ avant la Relu

=> permet d'entrainer des réseaux très profonds
=> Stack de plein de blocs résiduels.

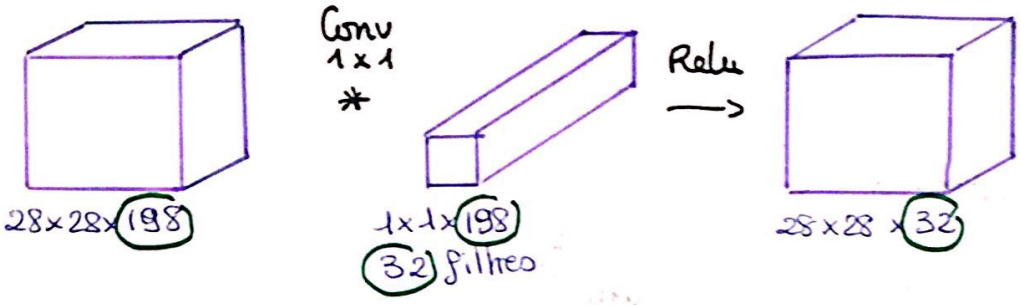
"Plain Network"





permet d'avoir des réseaux plus profonds avec bonne perf
⊕ aide pour vanish gdt

1 x 1 Convolution



=> Utile si on veut réduire le nombre de channels.
≠ Pooling utilisé pour réduire n_H et n_W

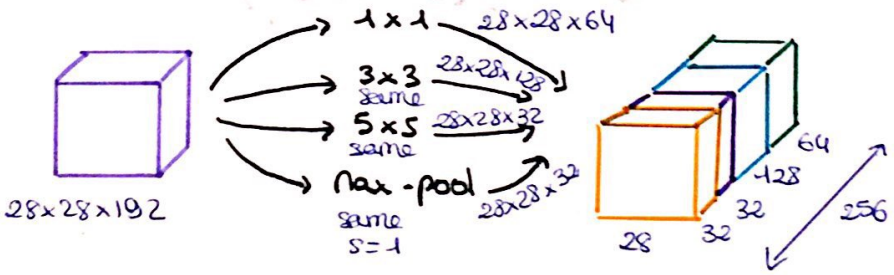
=> On peut aussi garder le même nombre de channels ou l'augmenter.

=> Sert pour les Inception Network.

Inception Network

=> Pour éviter de choisir entre Pool et CONV ou la taille du filtre.

=> Idée = concaténer toutes les sorties puis les mettre dans un réseau global.



=> Pb de coût en calcul.

=> On peut passer par des 1 x 1 convolution pour obtenir les sorties (couche bottleneck) pour ↓ coût de calcul.

exemple :

28x28 x192 — CONV 5x5 same 32 → 28x28 x32

= 120 M parameters

28x28 x192 — CONV 1x1 16 — CONV 5x5 32 → 28x28 x32

= 12.4 M parameters

=> Les couches peuvent être utilisées en réseaux

=> On peut aussi avoir des branches de sorties intermédiaires. Aide à s'assurer que les features calculés ne sont pas trop mauvaise pour protéger la sortie (régularisation, ↓ overfitting).