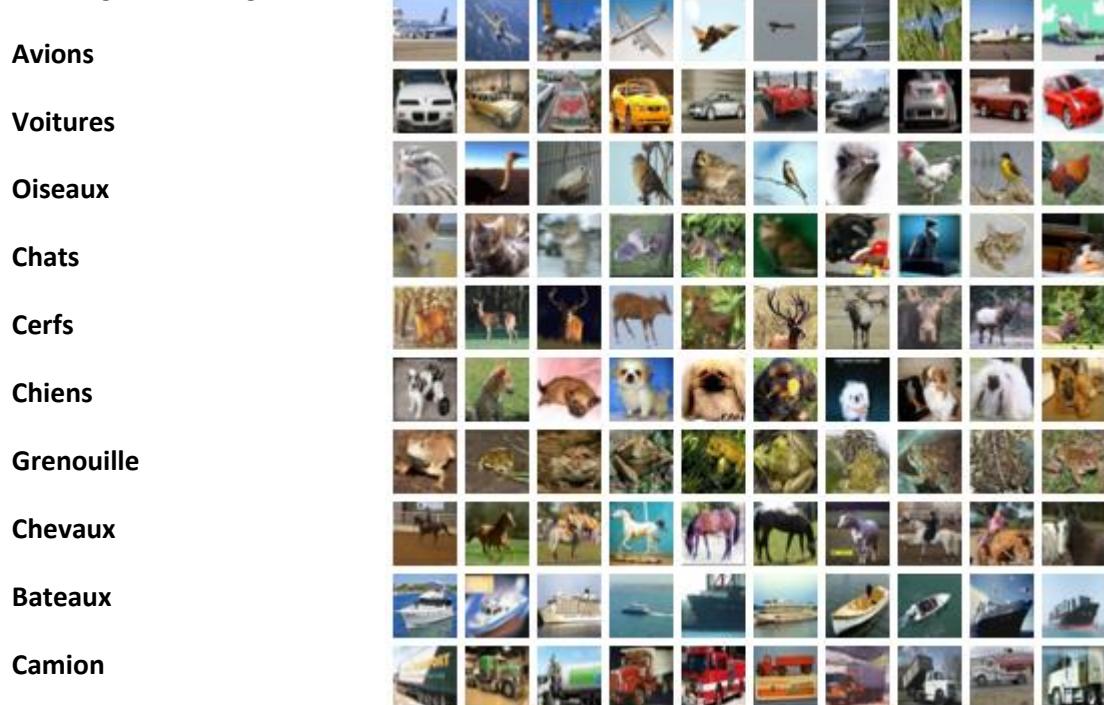


Deep Learning

Projet CIFAR10

Le CIFAR 10:

C'est un Dataset contenant 60 000 images en 32 pixels par 32 en couleurs. Nous avons pour celui-ci 10 catégories d'image :



Notre but est donc de lancer un algorithme d'apprentissage permettant de catégoriser ces images correctement grâce au Deep Learning.

Nous avons donc testé différents codes en python sous l'API Keras regroupant notamment les backends Tensorflow et Theano.

La première étape de notre code était d'importer le dataset cifar10 que nous avons fait à l'aide de la commande load_data().

Une fois ceux-ci importés dans notre base d'apprentissage et de test, il nous fallut utiliser un reshape dans le but de transformer le dataset en un vecteur pour les entrées des deux bases.

```
x_train = np.reshape(x_train, (-1, 32, 32, 3))/255.0  
x_test = np.reshape(x_test, (-1, 32, 32, 3))/255.0
```

Les options 32, 32, 3 correspondent à la longueur, la largeur de l'image ainsi que 3 tensors étant la couleur.

Ensuite nous transformons les vecteurs de la base d'apprentissage et de test pour qu'ils nous retournent une matrice de classe binaire.

```
y_train = keras.utils.to_categorical(y_train)  
y_test = keras.utils.to_categorical(y_test)
```

La deuxième étape fut de préparer le modèle que nous allions utiliser. Nous avons pris le modèle Sequential qui est un groupe de couches linéaires.

Dans ce modèle nous devons donc ajouter des couches de neurones puis les compilées à l'aide d'un optimizer pour enfin à l'aide de la commande fit() lancer l'apprentissage en précisant donc :

- **Le batchsize** = le nombre de test pouvant être réalisés en une fois durant une epoch (choisi en fonction de notre mémoire allouée ou bien de la puissance de notre carte graphique (pour tensorflow-gpu)).
- **Les epochs** = Le nombre d'itération d'apprentissage
- **Les callbacks** = type et emplacement des logs qui seront créés sur le run.
- **Validation_data** = précise quelle est la base de test/ de validations

Tableau de tous les runs effectués

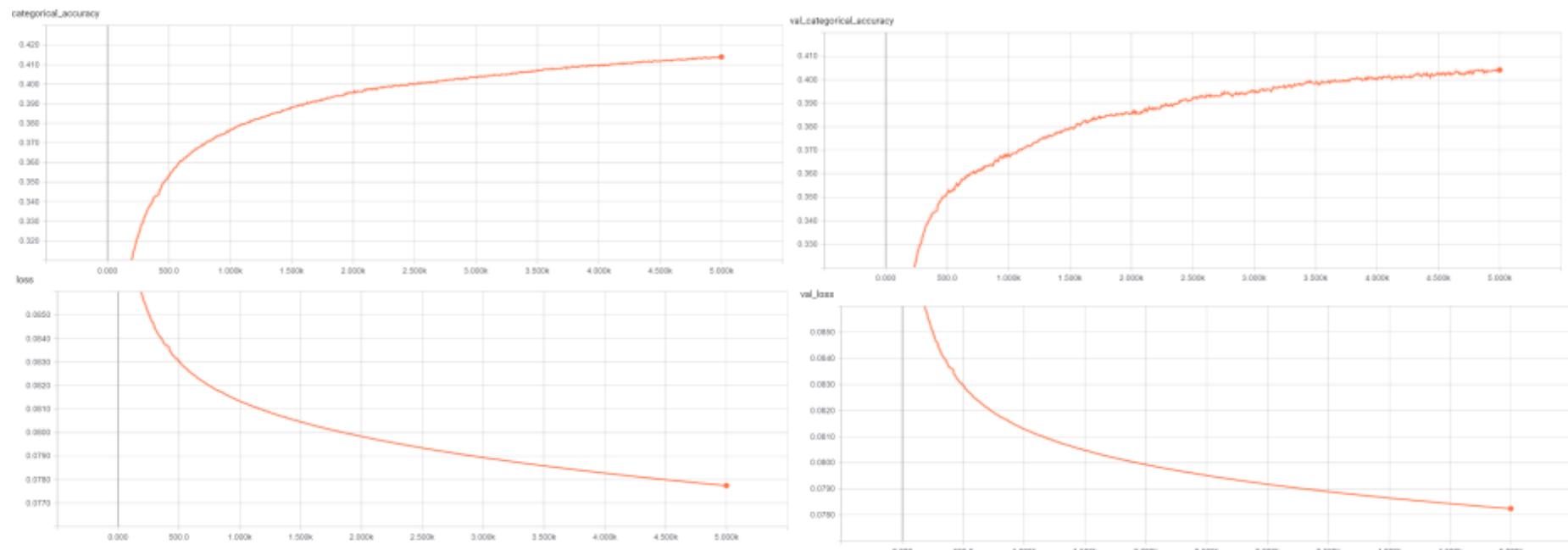
Ordre	Nom	Categ	Val_Categ	Epochs	Surraprentissage Observé	Pourcentage Gardé
1	CIFAR10_LINEAR_S10_LR_0.5	45	41	5000	vers 2322 epochs	41%
2	CIFAR10_LINEAR_S10_LR_0.05	41	40	5000	non	40%
3	CIFAR10_LINEAR_S10_LR_1	47	43.70	2000	non	47%
4	CIFAR10_LINEAR_1S16_S10_LR_0.5	42	40	1000	vers 789 epochs	40%
5	CIFAR10_LINEAR_1S32_S10_LR_0.5	41	40	1000	vers 900 epochs	40%
6	CIFAR10_LINEAR_T32_S10_LR_1	46	42	2000	vers 920 epochs	42%
7	CIFAR10_CONVNET1_16_SIGMOID_MAXPOOL_3_DROPOUT_S10_LR_1	25	23	30	non	23%
8	CIFAR10_CONVNET_16_RELU_MAXPOOL_2_DROPOUT_S10_LR_0.5	31	34	2900	non	34%
9	CIFAR10_CONVNET_16_RELU_MAXPOOL_2_DROPOUT_S10_LR_1	57	60	2000	non	60%
10	CIFAR10_CONVNET_32_RELU_MAXPOOL_3_DROPOUT_S10_LR_1	68	64	2000	vers 1214 epochs	62%
11	CIFAR10_2CONVNET1_16_16_SIGMOID_MAXPOOL_3_DROPOUT_S10_LR_0.5	55	57	273	non	57%
12	CIFAR10_2CONVNET2_16_32_RELU_MAXPOOL_3_DROPOUT_S10_LR_1	11	16	33	non	16%
13	CIFAR10_2CONVNET2_32_64_RELU_MAXPOOL_2_DROPOUT_S10_LR_1	74	72	2000	vers 1527 epochs	70%
14	CIFAR10_2CONVNET2_64_64_RELU_2MAXPOOL2_T64_S10_LR_0,5	45	46	27	non	46%
15	CIFAR10_2CONVNET1_16_16_RELU_MAXPOOL_3_DROPOUT_SM10_LR_1	27	36	60	non	36%
16	CIFAR10_2CONVNET1_16_16_RELU_MAXPOOL_3_DROPOUT_SP10_LR_1	19	25	20	non	25%
17	CIFAR10_2CONVNET2_32_64_RELU_MAXPOOL_3_DROPOUT_1SP25_S10_LR_1	77	75	800	vers 701 epochs	74%
18	CIFAR10_3CONVNET2_32_64_32_RELU_MAXPOOL_DROPOUT_S10_LR_1	72	72	662	non	72%
19	CIFAR10_3CONVNET3_64_64_64_RELU_MAXPOOL_DROPOUT_S10_LR_0.05	64	65	10000	non	65%
20	CIFAR10_3CONVNET3_64_64_64_RELU_MAXPOOL_DROPOUT_S10_LR_1	76	75	3577	non	75%
21	CIFAR10_2CONVNET2_32_64_RELU_MAXPOOL_DROPOUT_T64_S10_LR_1	89	76	2000	vers 625 epochs	71%
22	CIFAR10_2CONVNET2_32_64_RELU_MAXPOOL_DROPOUT_R64_S10_LR_1	81	74	860	vers 521 epochs	71%
23	CIRFAR10_2CONVNET2_64_64_RELU_2MAXPOOL2_2_DROPOUT_2_T64_S10_LR_1	93	78	488	vers 80 epochs	74%
24	CIFAR10_CONVNET2_64_RELU_MAXPOOL_DROPOUT_T64_S10_LR_1	64	63	73	non	63%
25	CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_2R64_S10_LR_1	95	78	2000	vers 708 epochs	76.5%
26	CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_3T64_S10_LR_1	94	79.01	2000	vers 320 epochs	72%
27	CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_3E64_S10_LR_1	95	79.46	2000	vers 450 epochs	76.85%
28	CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_3R64_SM10_LR_1	95	76	1200	vers 418 epochs	74%
29	CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_3E64_SM10_LR_1	97	78	2000	vers 364 epochs	76.33%
30	CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_3E64_S10_LR_1_RMSPROP	96.80	78	2000	vers 118 epochs	76.12%
31	CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT50_3E64_S10_LR_1_RMSPROP	94	78	2000	vers 519 epochs	80.07%

Phase d'apprentissage

Perceptron Multi-couche (MLP)

Pour commencer nos tests nous sommes partis comme pour le MNIST à tester l'apprentissage tout d'abord sur un simple perceptron multicouche pour avoir une référence d'apprentissage.

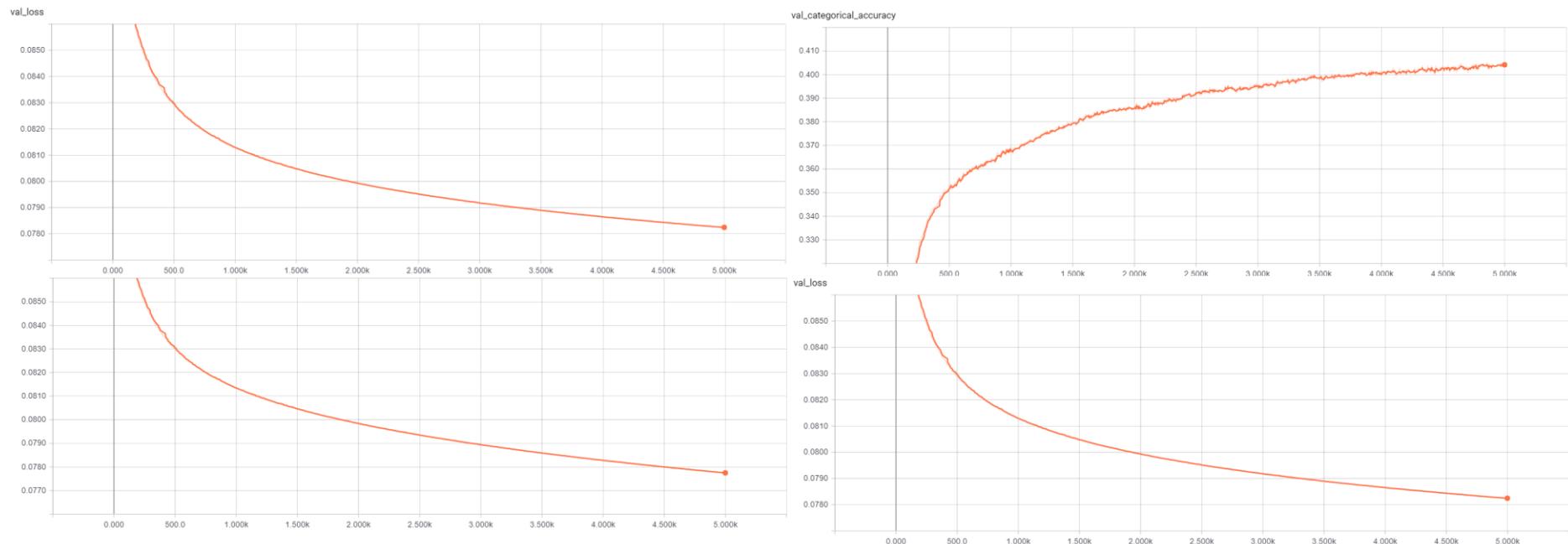
Test 1: CIFAR10_LINEAR_S10_LR_0.5



Nous avons donc commencé avec une seule couche :

Ici nous avons donc une seule couche sigmoid de 10 neurones avec un learning rate de 0.5 sur 5k d'epochs. Ce run nous a donné 41% sur la base de test.

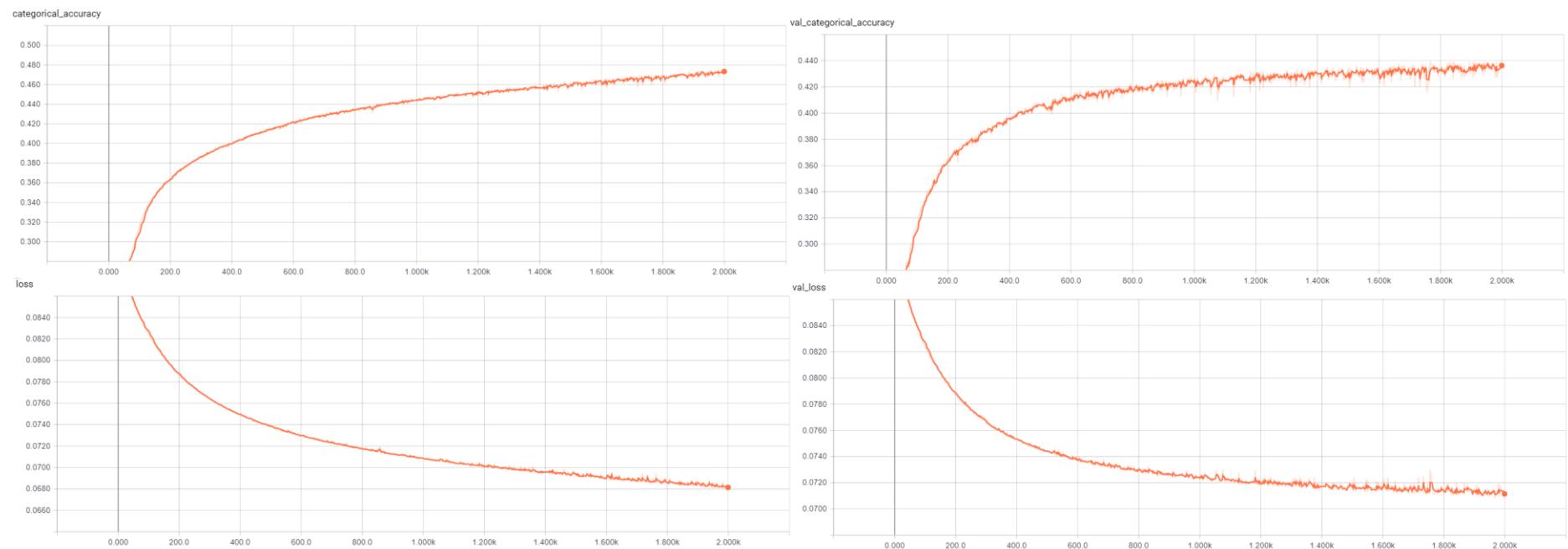
Test 2: CIFAR10_LINEAR_S10_LR_0.05



Nous avons alors modifié le learning rate pour le mettre à 0.05 sur 5k d'epochs.

Ce run a donné un résultat évidemment inférieur (40%) puisque pour rendre le learning rate intéressant et plus important, il aurait fallu aussi augmenter significativement le nombre d'epochs.

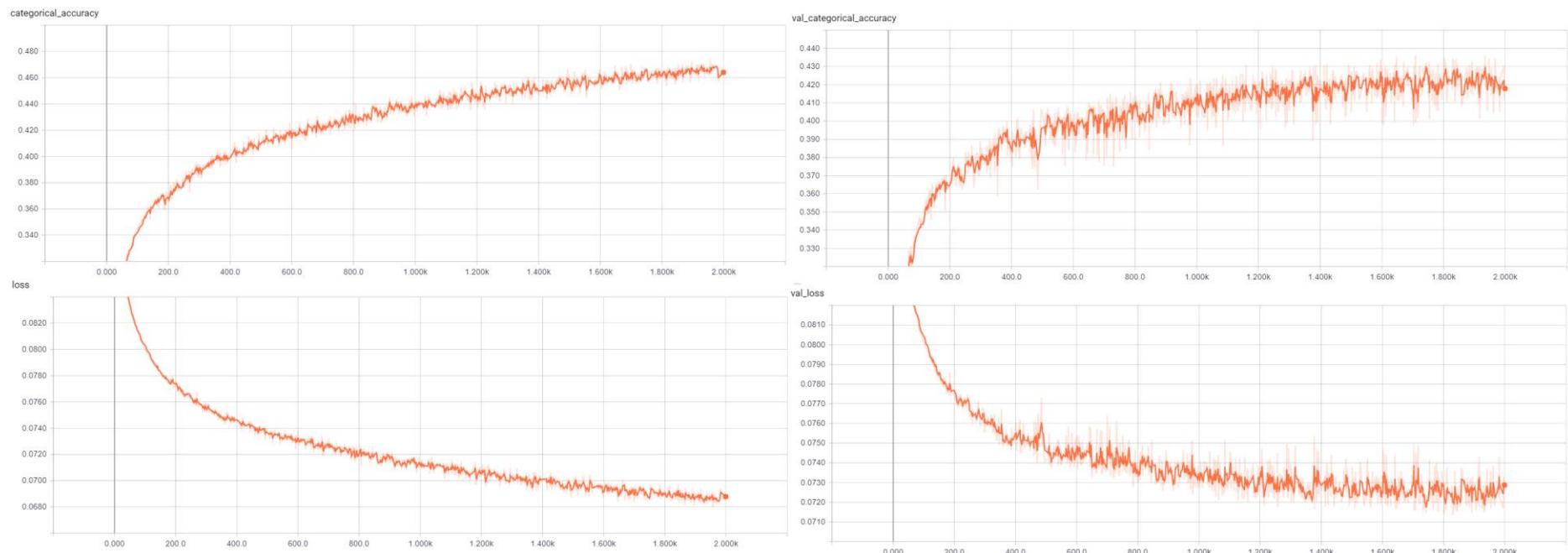
Test 3: CIFAR10_LINEAR_S10_LR_1



Ayant fait l'erreur de mettre un learning rate trop bas et sur deux run d'affilé, nous avons rectifié le tir en plaçant le learning rate à 1 sur 2K d'epochs

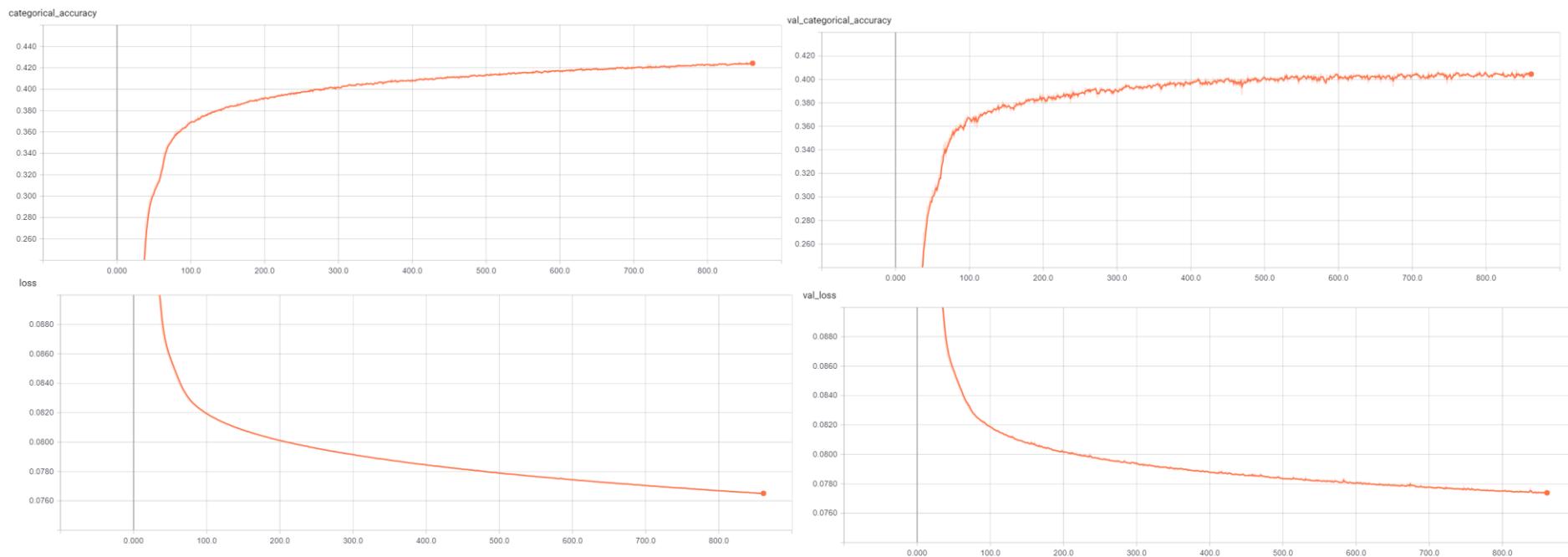
Ce run a apporté 43.7% ce qui est nettement préférable.

Test 4: CIFAR10_LINEAR_1S16_S10_LR_0.5



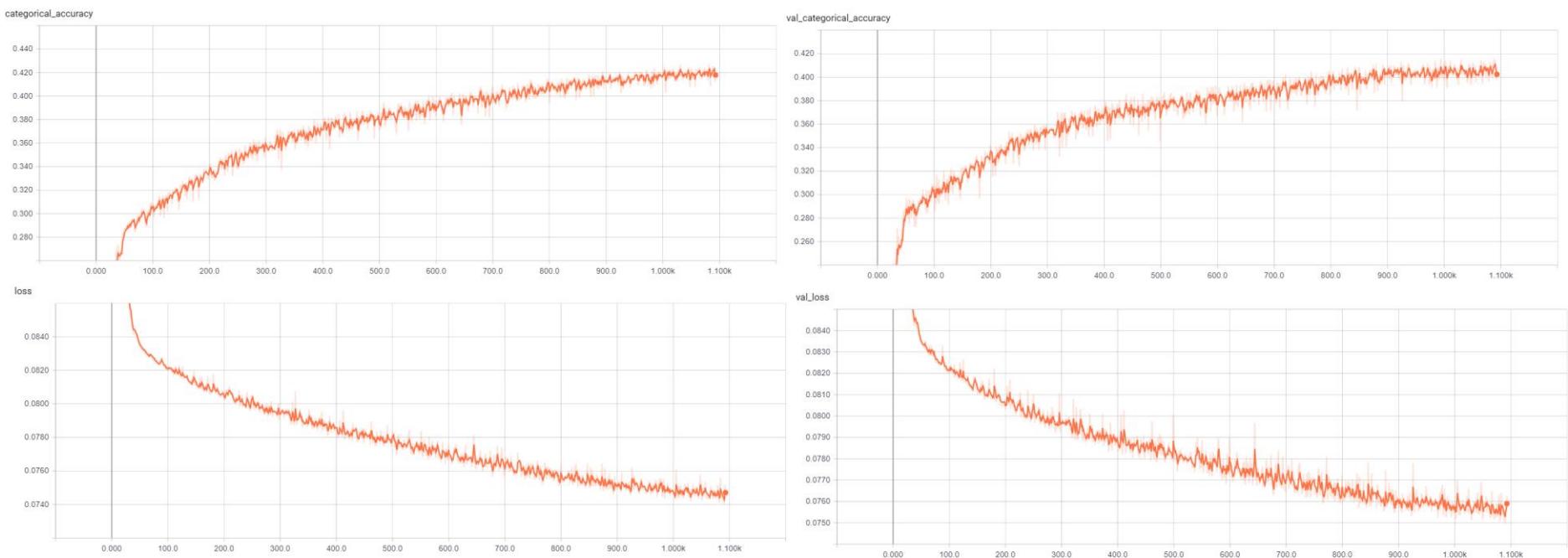
Nous avons ensuite testé d'ajouter des couches à ce modèle en commençant par une couche de sigmoid de dense 16 avec 0.5 de LR. Ce run a atteint 40%.

Test 5: CIFAR10_LINEAR_1S32_S10_LR_0.5



Nous avons alors testé d'augmenter le nombre de neurones de la couche à 32 pour la sigmoid ajoutée. Ce run a atteint encore une fois 40% ce qui n'est pas une avancée.

Test 6: CIFAR10_LINEAR_T32_S10_LR_1



Le prochain run a donc été testé avec une tangeante hyperbolique à la place de la sigmoid toujours avec 32 de dense. Ce run a atteint 42% ce qui est mieux mais nous avons pu observer un surapprentissage vers l'epoch 920 sur 2000 donc nous avons gardé le modèle avant cette epoch. Cela ne nous donna seulement 40% de réussite.

Réseau neuronal convolutif (CNN)

Comme nous avons comme tâche de travailler sur des images, il est intéressant d'utiliser les réseaux de convolutions pour notre apprentissage.

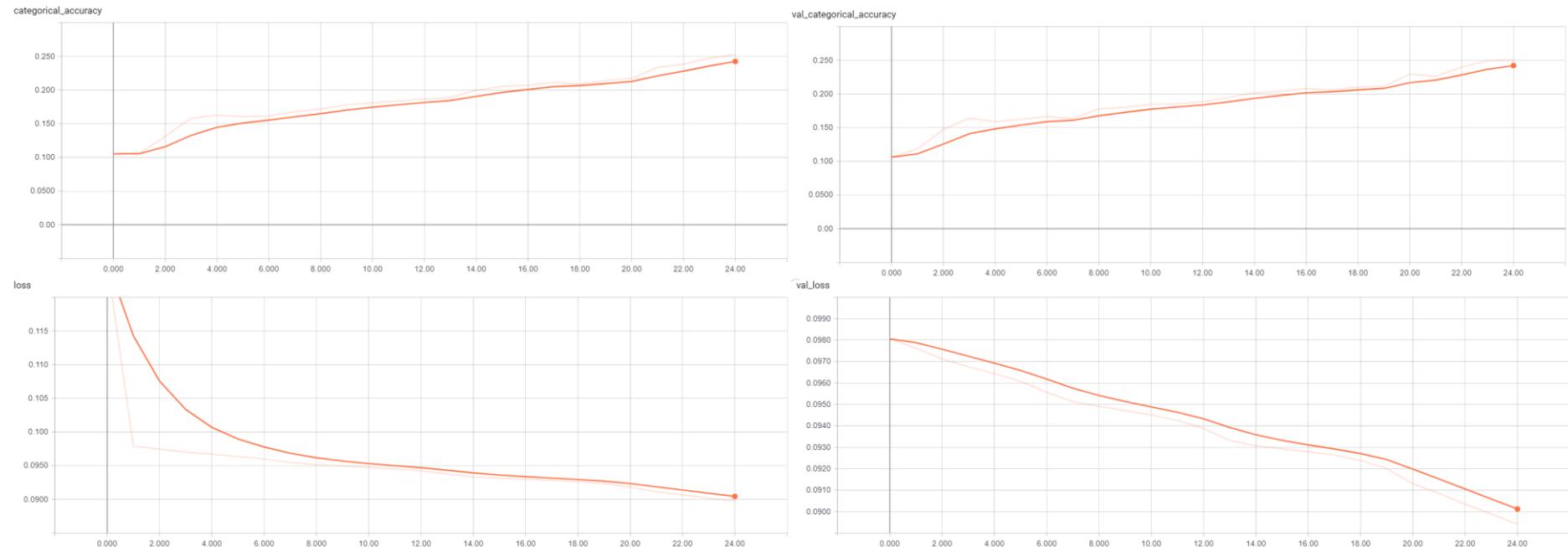
Nous avons commencé par une seule couche de convolution de 16 filtres (nombre de neurones avec leurs poids associés) avec des matrices 3×3 toujours de la même forme*.

Ensuite nous avons ajouté une couche de maxpool2D qui consiste donc à diviser l'image en plusieurs images réduites (ici de 3×3) en gardant les plus hautes valeurs de chaque matrice. (Particulièrement utile dans la réduction du coût de calcul mais aussi dans la prévention contre l'overfitting).

Enfin le dernier ajout est le dropout qui a pour fonction de supprimer aléatoirement des liens entre les neurones en mettant leurs poids à zéro. Cela permet donc au réseau de convolution d'apprendre même s'il manque de l'information. Un des plus de cette fonction est de pouvoir limiter le surapprentissage. Le chiffre que nous précisons dans le code est le pourcentage de neurones qui seront déconnectés.

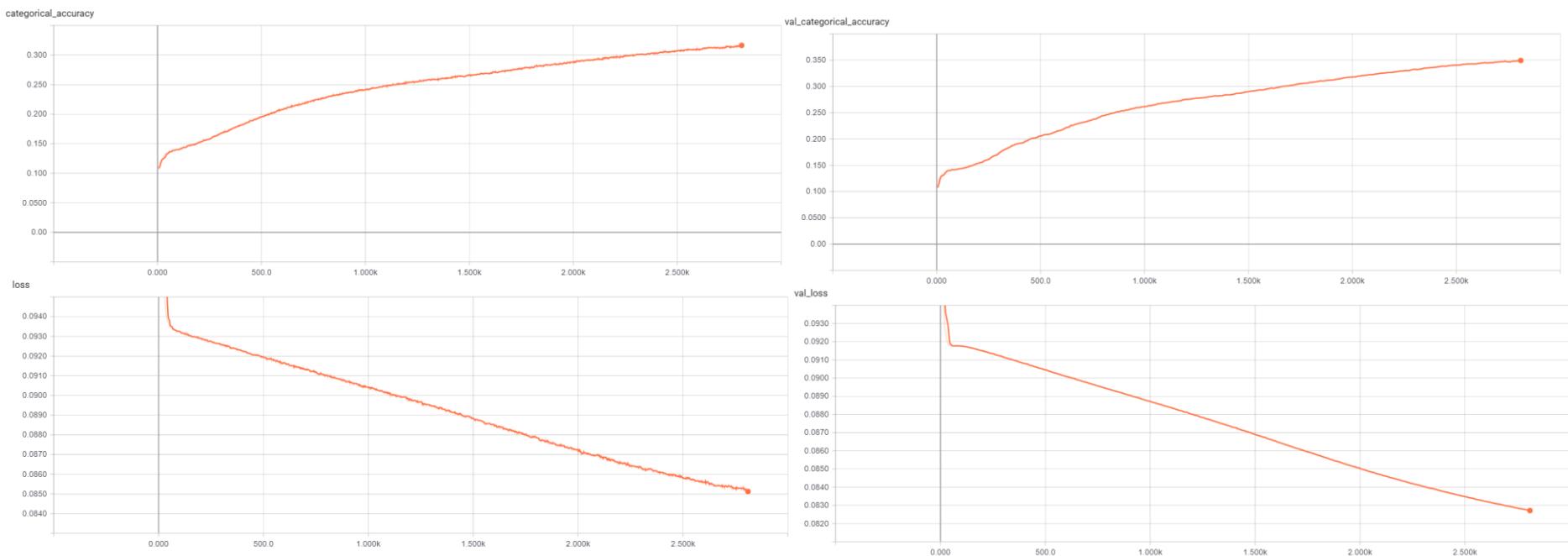
*A noté que l'`input_shape` n'est précisée que sur la première couche car les couches suivantes utiliseront automatiquement la même que la première.

Test 7: CIFAR10_CONVNET1_16_SIGMOID_MAXPOOL_3_DROPOUT_S10_LR_1



Ce run a duré 30 epochs et a atteint 23% ce qui n'est vraiment pas une amélioration donc nous avons voulu tester une autre fonction d'activation dans notre bloc de convolution. En revanche nous aurions effectivement due le laisser tourner sur plus d'epochs pour voir une réelle amélioration.

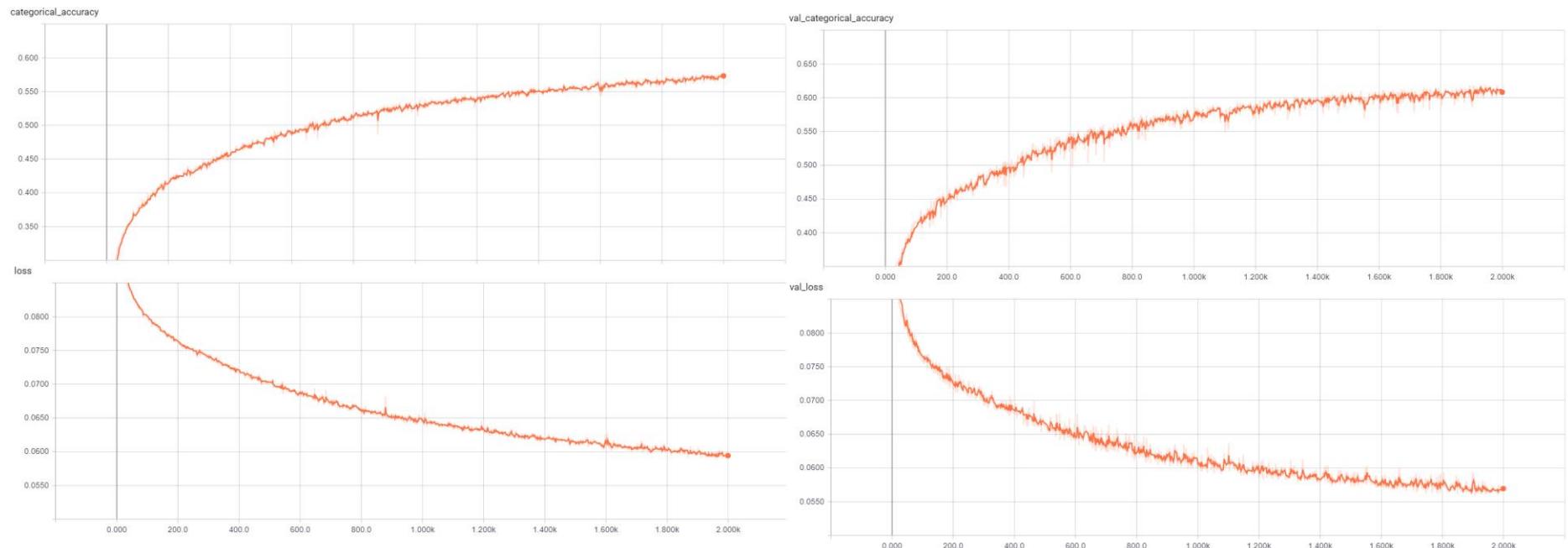
Test 8: CIFAR10_CONVNET_16_RELU_MAXPOOL_2_DROPOUT_S10_LR_0.5



Ce Run nous a permis d'atteindre 34% sur 2k d'epochs ce qui est ce coup-ci une nette amélioration.

Dans ce run nous avons utilisé la fonction d'activation relu pour notre bloc de convolution car elle permet la formation d'un réseau neuronnal bien plus rapide qu'avec la tangente hyperbolique ou la sigmoïde. Elle a pour avantage d'augmenter les propriétés non linéaires de la fonction de décision ce qui est pratique pour résoudre donc un problème comme le nôtre.

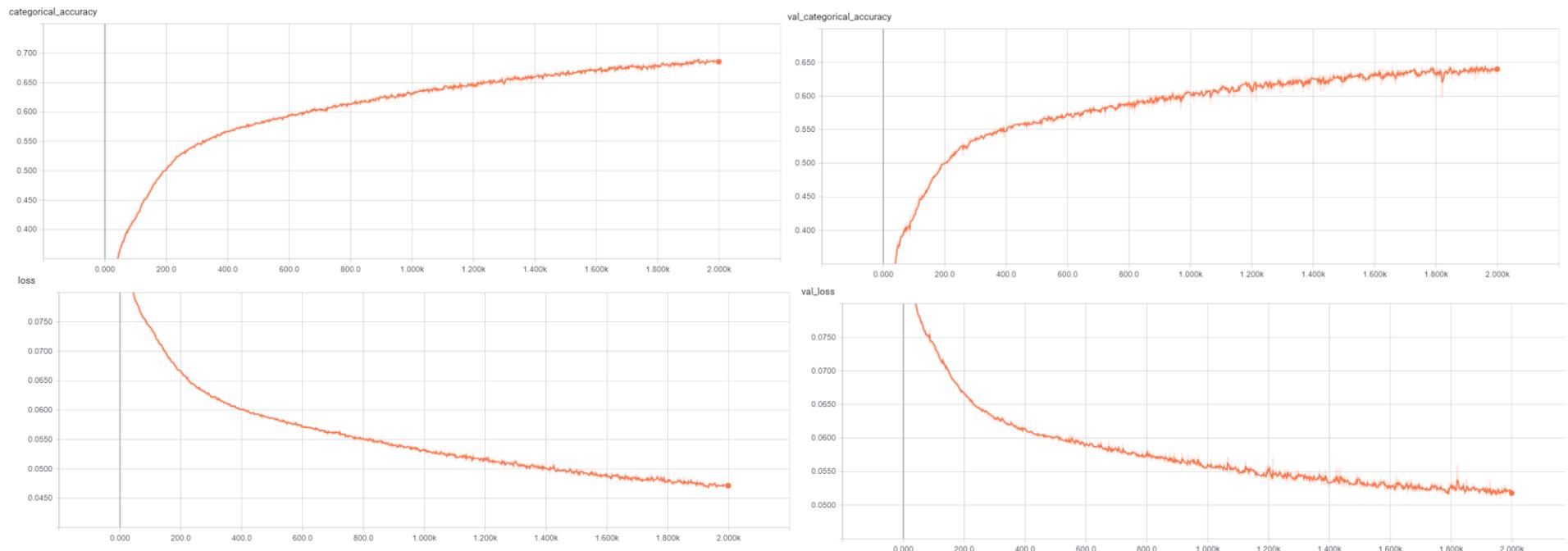
Test 9 : CIFAR10_CONVNET_16_RELU_MAXPOOL_2_DROPOUT_S10_LR_1



Ensuite nous avons testé le changement du LR qui une fois de plus était trop bas pour observer une réelle avancée sur le premier modèle.

Ce run nous a permis d'atteindre 60% sur 2k d'epoch ce qui est enfin une réelle amélioration

Test 10: CIFAR10_CONVNET_32__RELU_MAXPOOL_3_DROPOUT_S10_LR_1

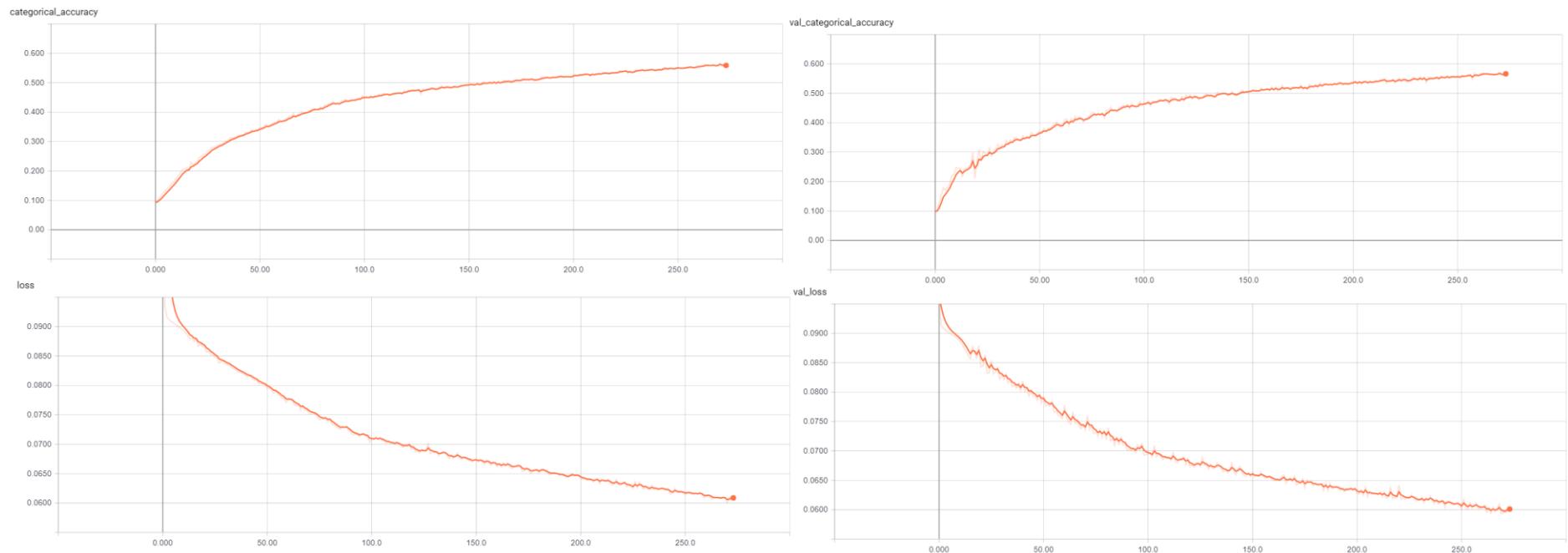


Pour le test suivant nous avons monté le nombre de filtres à 32. Ce run a atteint 64% ce qui est une nette amélioration mais en revanche nous pouvons noté un début de surapprentissage vers l'epoch 1214 ce qui nous fait garder le modèle à 62%.

Ajout d'un bloc de convolution:

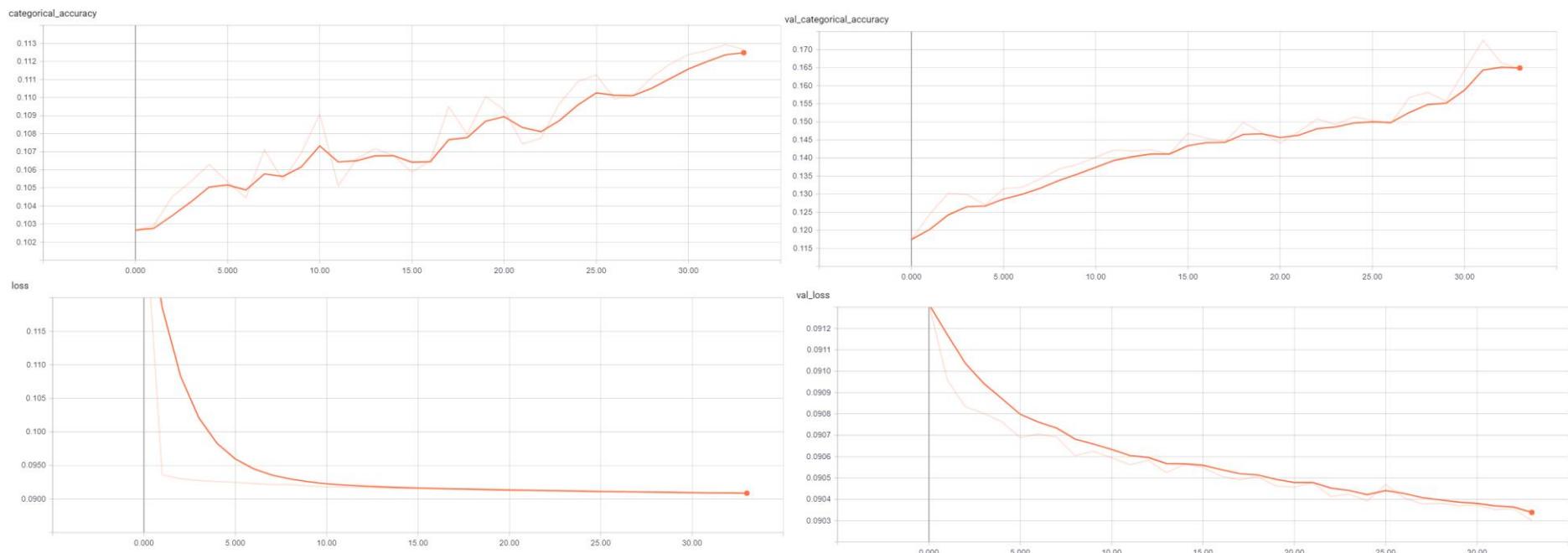
Pour la suite de nos tests nous avons un bloc de convolution (soit la copie conforme de la première partie du code (convnet+activation'relu'+maxpool+dropout) dans le but de complexifier le modèle et de voir si nous pouvons obtenir de meilleurs résultats.

Test 11: CIFAR10_2CONVNET1_16_16_SIGMOID_MAXPOOL_3_DROPOUT_S10_LR_0.5



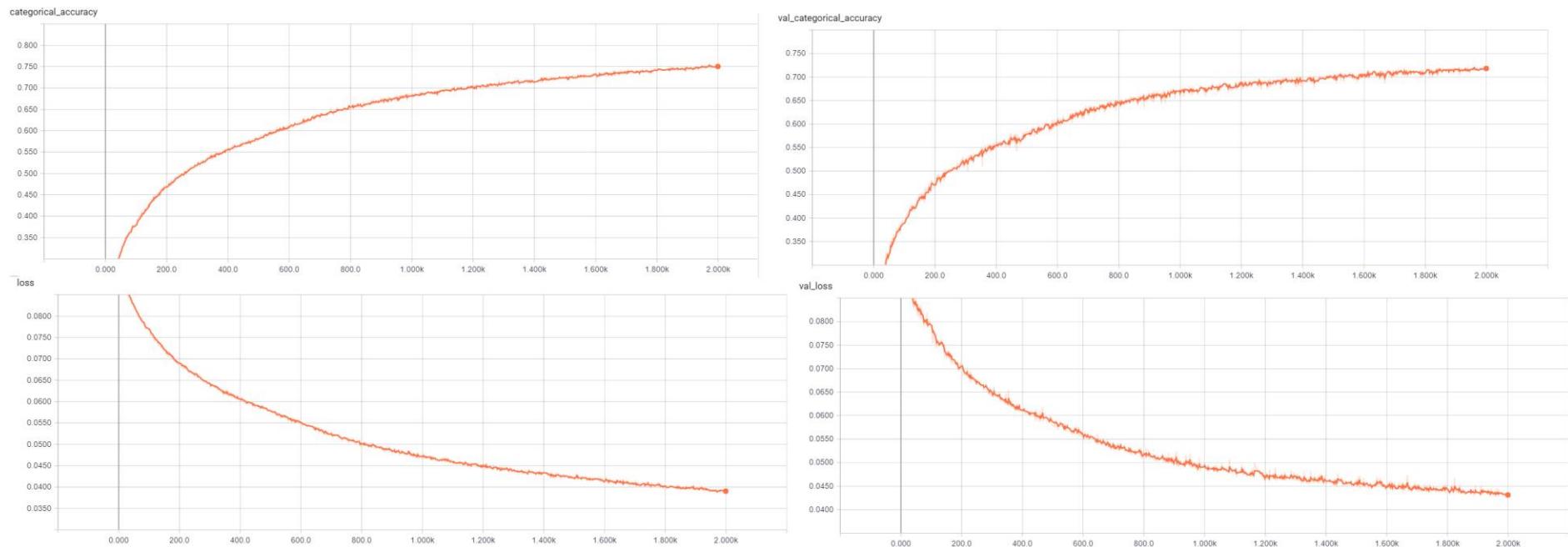
Après l'ajout de ce bloc nous avons pu obtenir 57% en 273 epochs ce qui est une bonne amélioration. En revanche il faut noter que dans ce test la fonction d'activation relu n'a pas été utilisé mais la sigmoid car nous l'avions faite en parallèle.

Test 12: CIFAR10_2CONVNET2_16_32_RELU_MAXPOOL_3_DROPOUT_S10_LR_1



Dans ce test-ci nous avions justement voulu faire l'ajout d'un bloc mais avec la fonction 'relu'. Cela nous a permis d'atteindre 16% sur 33 epoch. Ce run a été coupé cours pour être testé plutôt avec des filtres plus grands.

Test 13: CIFAR10_2CONVNET2_32_64_RELU_MAXPOOL_2_DROPOUT_S10_LR_1

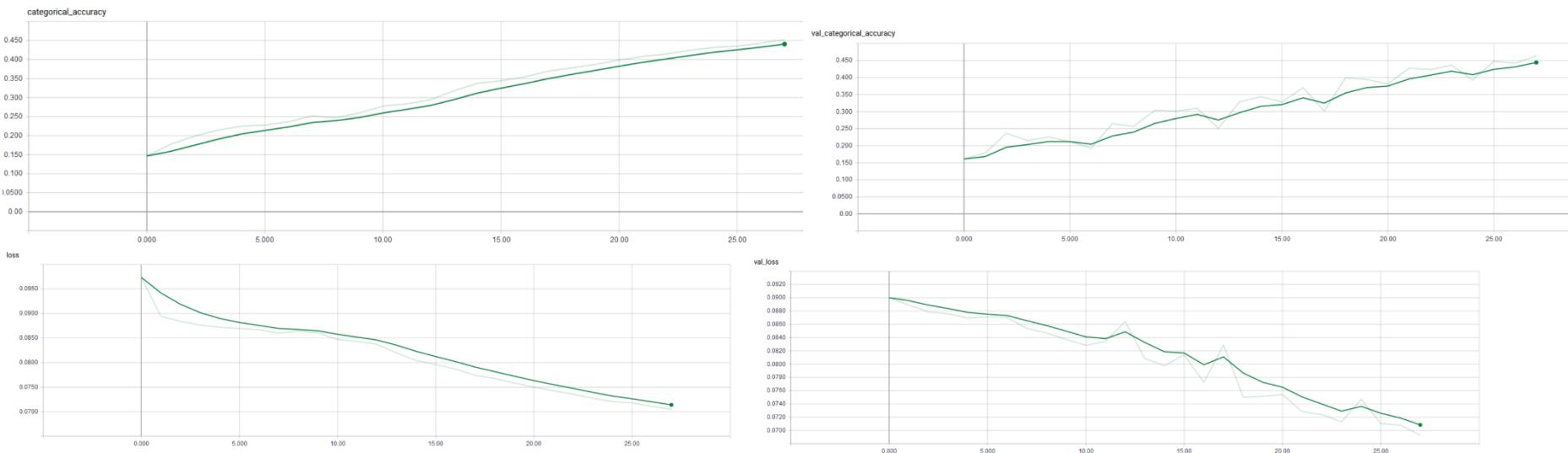


Dans ce test nous avons donc augmenter les filtres dans nos blocs de convolution mais pas seulement. En effet nous avons placé deux couches de convolution par bloc car cela permettrait d'avoir de meilleures performances sur le modèle.

Ce run nous a permis d'atteindre 72%. En enlevant le surapprentissage vers l'epoch 1527 nous obtenons un modèle à 70% de réussites.

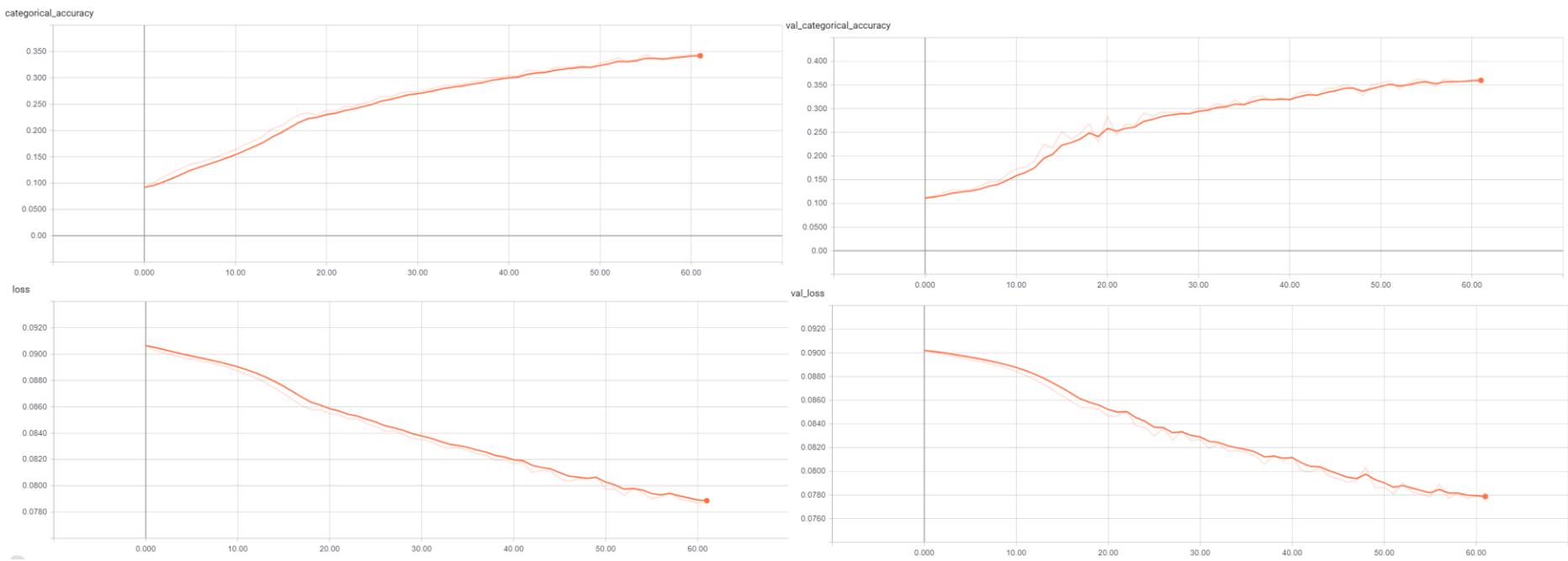
Nous en avons aussi profité pour réduire le maxpool à des matrices 2*2 pour pouvoir supprimer les pixels artificiels ajouté par la convolution et notamment le padding.

Test 14: CIFAR10_2CONVNET2_64_64_RELU_2MAXPOOL2_T64_S10_LR_0.5



Ce test avait pour but de tester l'ajout d'une couche de tangente hyperbolique pour voir si nous notions une amélioration. Ce test n'a pas abouti.

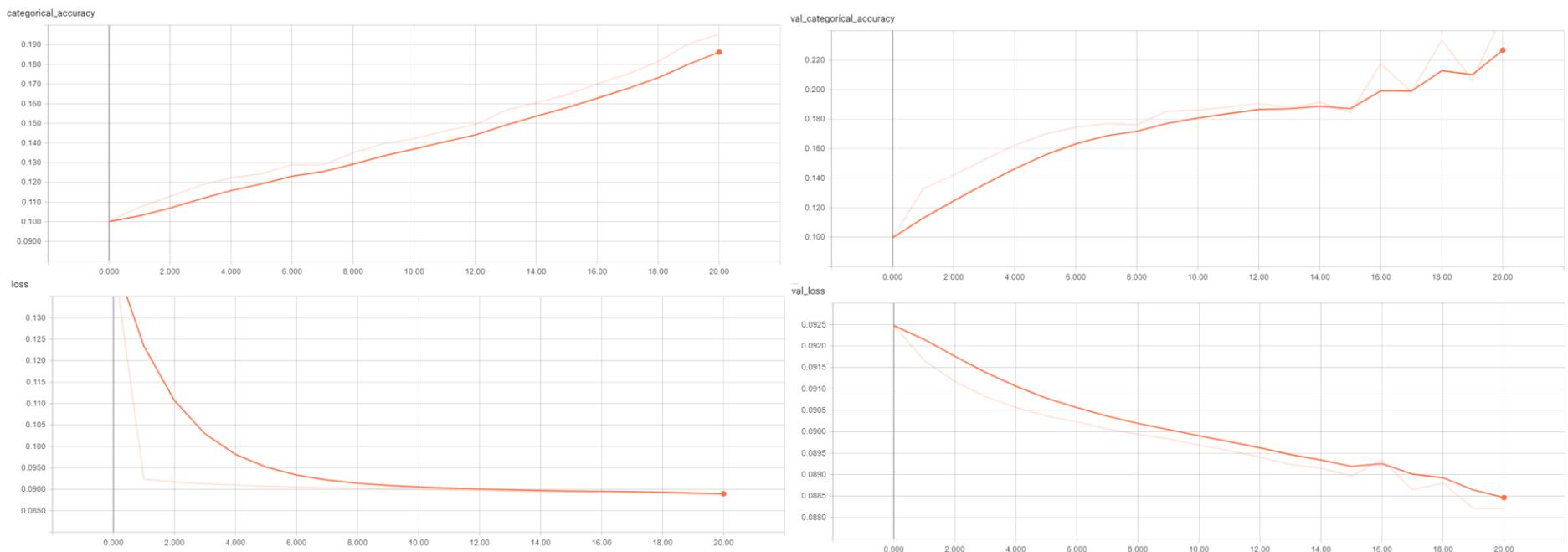
Test 15: CIFAR10_2CONVNET1_16_16_RELU_MAXPOOL_3_DROPOUT_SM10_LR_1



Ce test avait pour but de tester le changement de la fonction d'activation finale (à l'origine la sigmoid(10)) par une fonction softmax car tout comme la sigmoïde ses résultats sont compris entre 0 et 1. Cependant la fonction softmax calcul la probabilité qu'un objet soit bien celui que nous cherchons.

Ce test ayant été fait après le test 11 a révélé être moins performant qu'avec la sigmoïde et a donc été stoppé à l'epoch 60 où il avait atteint 36%

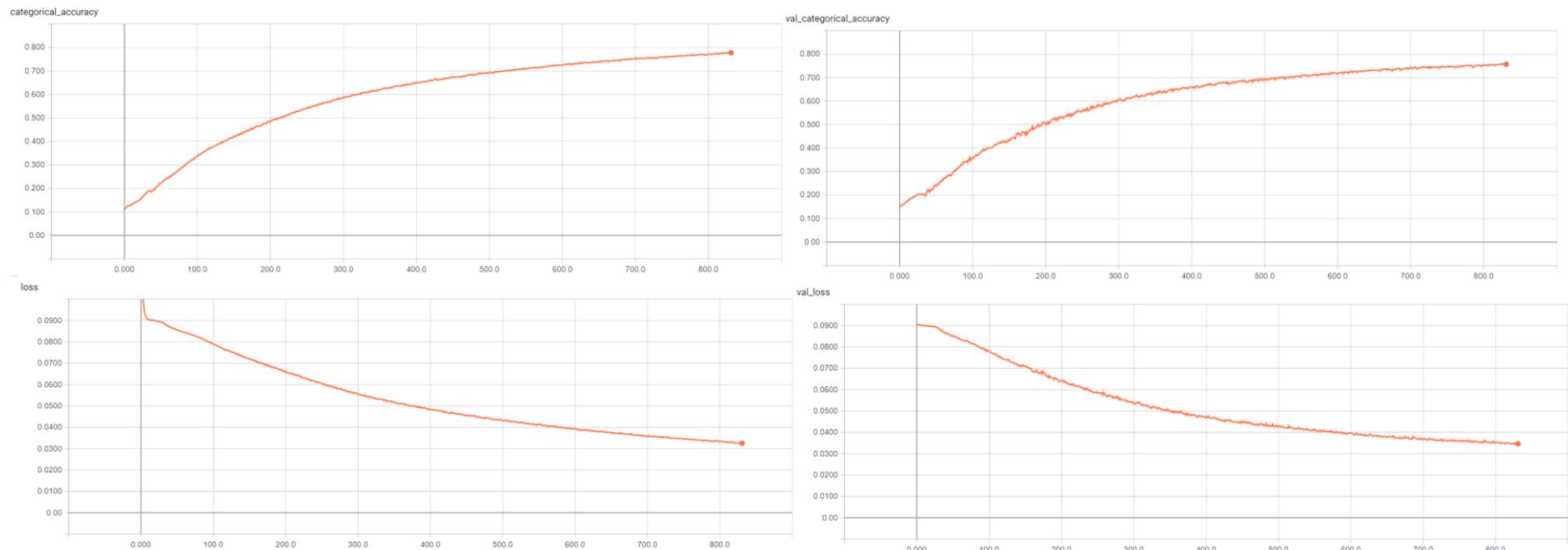
Test 16: CIFAR10_2CONVNET1_16_16_RELU_MAXPOOL_3_DROPOUT_SP10_LR_1



Ce test, tout comme le test 15 a permis de tester cette fois-ci la fonction softplus qui est comme une fonction relu mais en adouci, son avantage est que autour de 0 nous avons un peu plus de marge qui laisse la possibilité de ne pas tuer certains neurones. En revanche ce run était aussi moins performant qu'avec la sigmoide donc il fut stoppé.

Nous avions atteint 25% en 20 epoch.

Test 17: CIFAR10_2CONVNET2_32_64_RELU_MAXPOOL_3_DROPOUT_1SP25_S10_LR_1

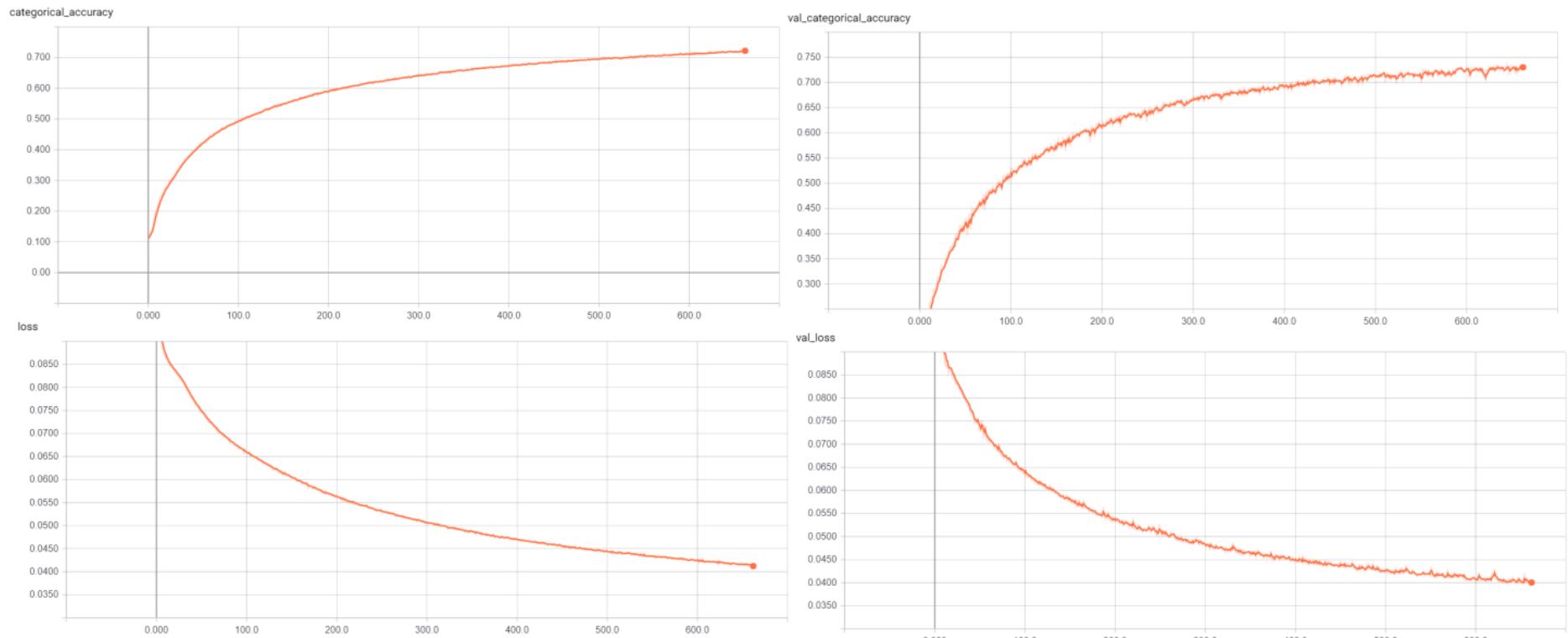


Dans ce test nous avions replacé la sigmoïde(10) mais rajouté une fonction softplus juste avant celle-ci avec 25 neurones. Ce run nous a permis d'atteindre 75% en 800 epochs. Un surraprentissage est notable en revanche vers 701 epoch et donc nous avons gardé le modèle à 74%. Ceci est un résultat intéressant qui nous encouragera par la suite à tester l'ajout de couches.

Ajout d'un bloc de convolution supplémentaire.

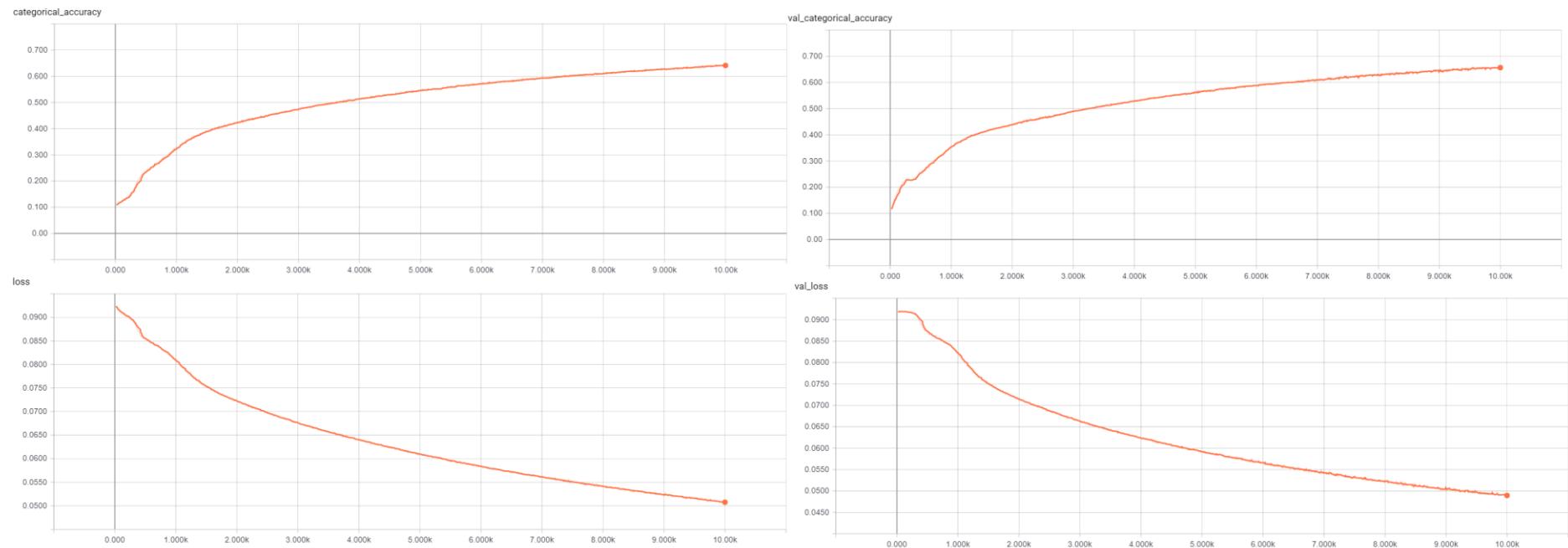
Ayant eu des résultats probants pour l'ajout d'un deuxième bloc, nous avions donc voulu tester avec 3.

Test 18: CIFAR10_3CONVNET2_32_64_32_RELU_MAXPOOL_DROPOUT_S10_LR_1



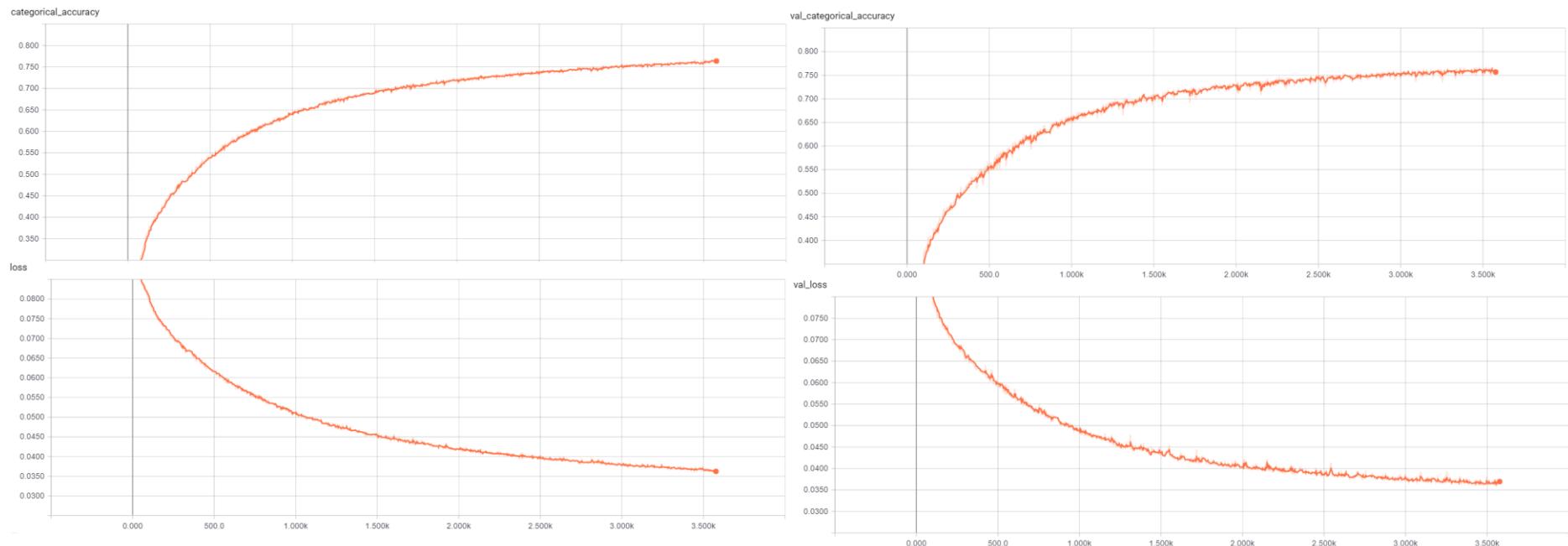
Dans ce test nous avons commencé avec seulement une couche 'relu' par bloc. Ce run nous a apporté une précision de 73%.

Test 19: CIFAR10_3CONVNET3_64_64_64_RELU_MAXPOOL_DROPOUT_S10_LR_0.05



Ce test a été réalisé avec 3 blocs de convolutions mais avec 64 filtres. Le learning rate a été baissé à 0.05 et les epoch ont été monté à 10k pour voir si nous aurions de bons résultats. Ce run nous a donné 65% ce qui est inférieur à ce que nous espérions, en revanche nous n'avons pas noté de surapprentissage sur nos 10k d'epochs.

Test 20: CIFAR10_3CONVNET3_64_64_64_RELU_MAXPOOL_DROPOUT_S10_LR_1



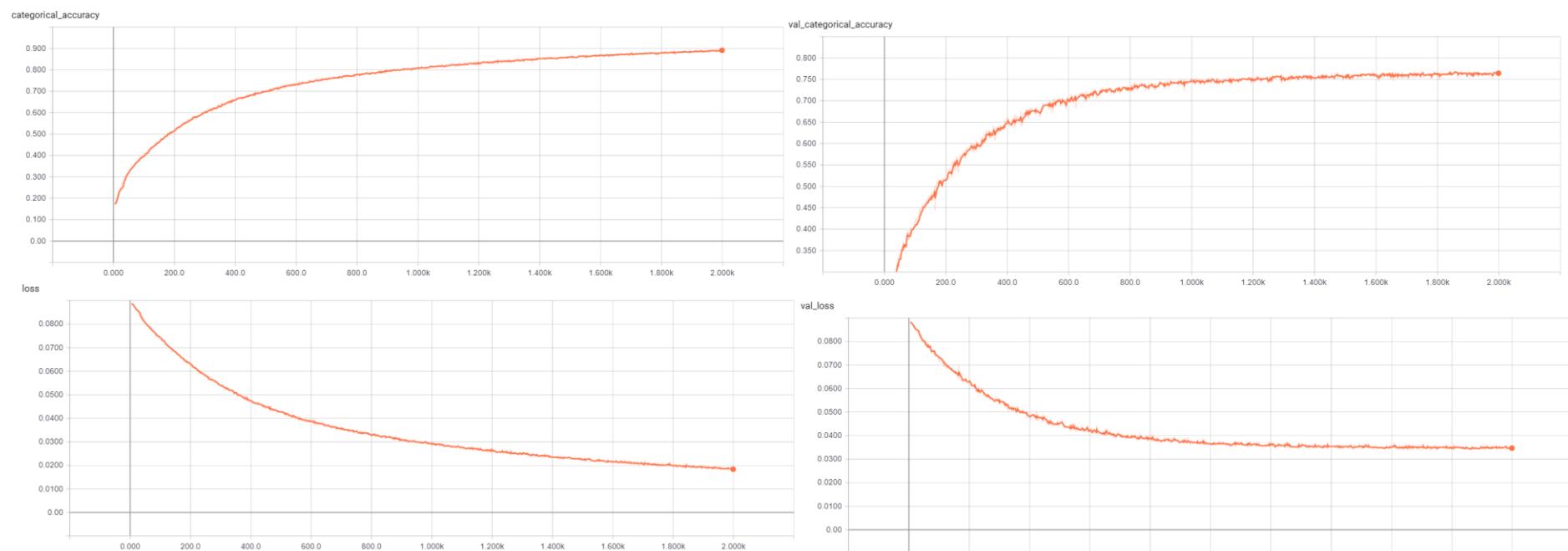
Le résultat précédent, étant décevant nous avons décidé de la refaire mais avec un learning rate plus élevé et donc sur moins d'epochs.

Nous avons atteint 75% en 3577 epochs ce qui est beaucoup mieux.

Retrait d'un bloc de convolution et ajout de couches cachées.

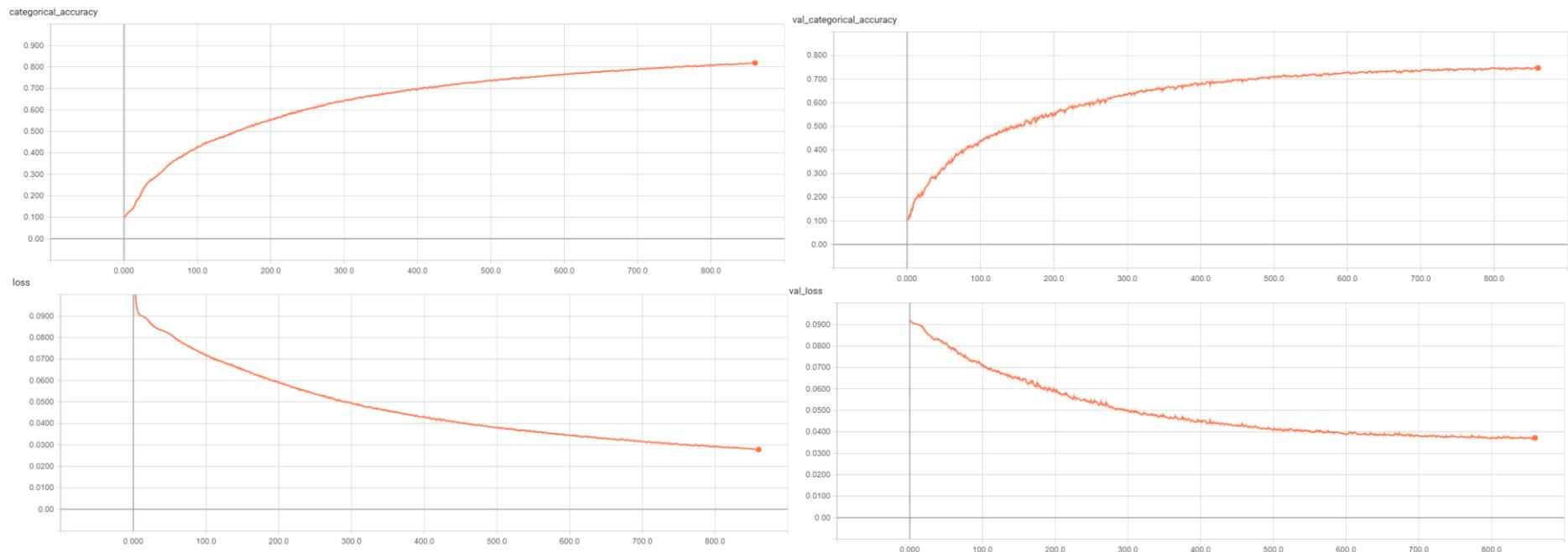
Les résultats avec 3 blocs de convolution ne nous semblaient pas changer beaucoup par rapport aux 2 blocs et c'est pourquoi nous avons choisi de garder le modèle le plus simple et de revenir donc à deux blocs. Nous avons ensuite tenté d'améliorer le modèle en ajoutant des couches cachées pour voir si nous avions une amélioration.

Test 21: CIFAR10_2CONVNET2_32_64_RELU_MAXPOOL_DROPOUT_T64_S10_LR_1



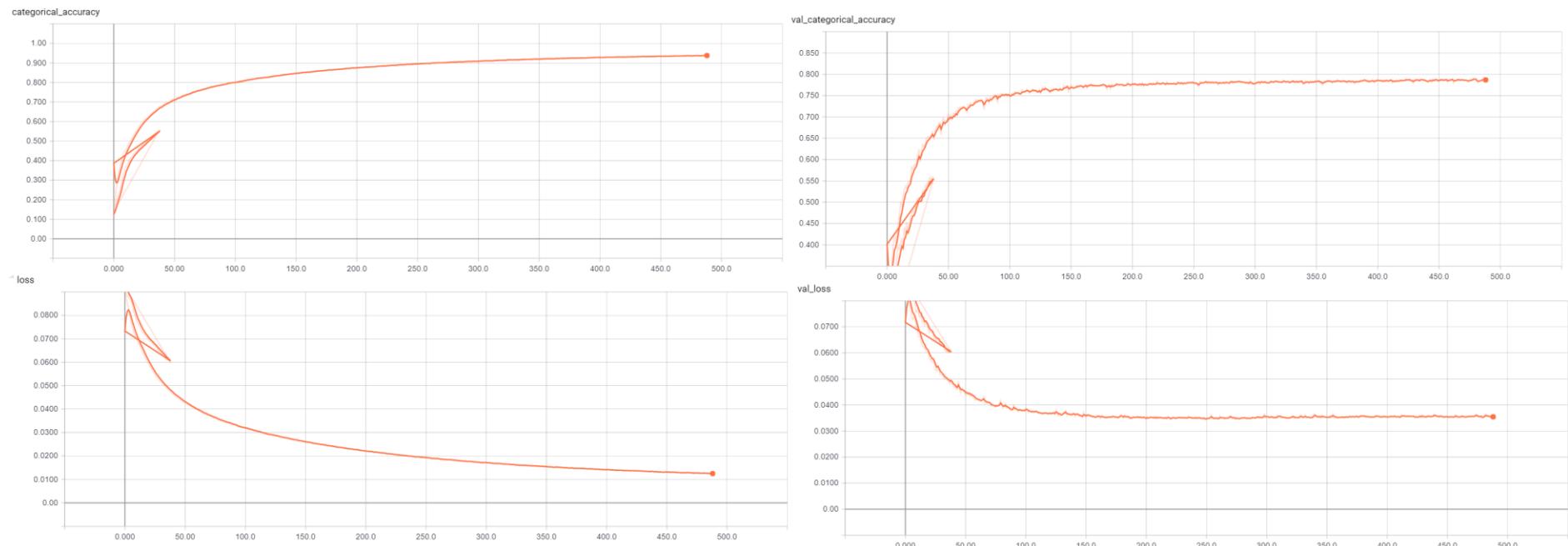
Dans ce test nous avons commencé par ajouté une couche de tanh(64) pour noter une amélioration. Nous avons pu atteindre 76% ce qui nous avions jugé de bonne nouvelle. En revanche nous n'avions remarqué que plus tard (ainsi que pour les courbes suivantes) que le surapprentissage nous faussait les résultats. En effet vers l'epoch 625 nous commençons le surapprentissage ce qui nous a ramené au modèle nous donnant 71%.

Test 22: CIFAR10_2CONVNET2_32_64_RELU_MAXPOOL_DROPOUT_R64_S10_LR_1



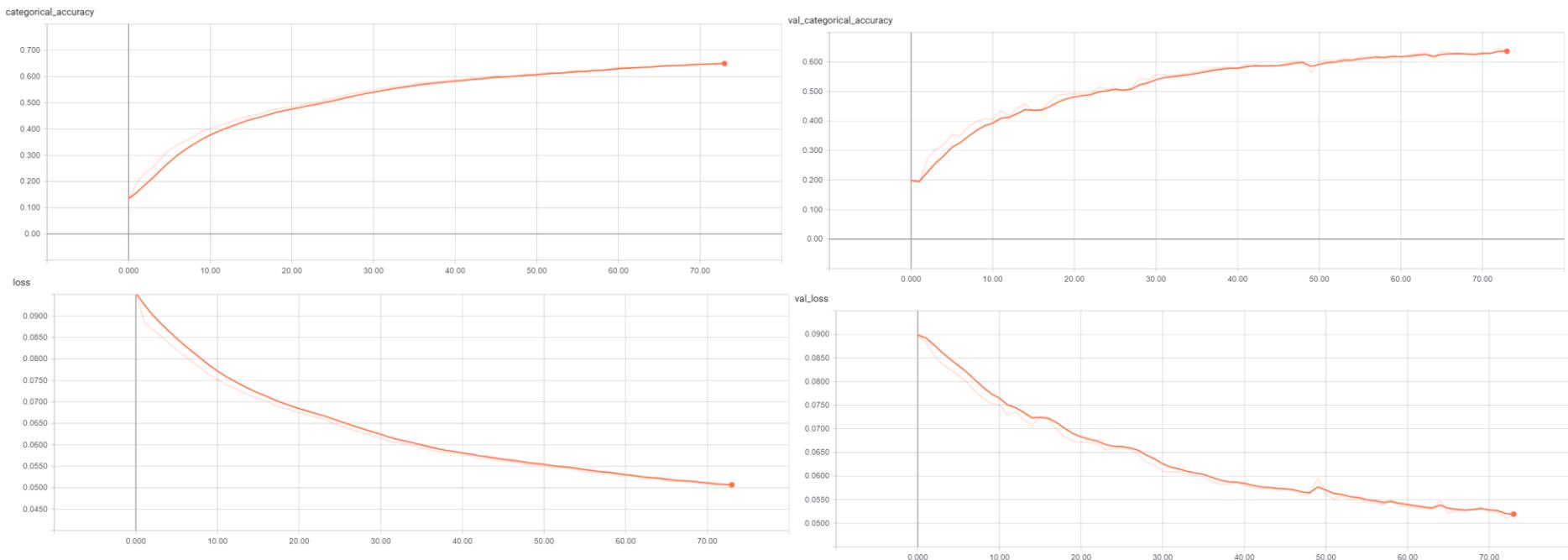
Dans ce test-ci nous avions remplacé la tanh par une fonction 'relu' dans le but de déterminer laquelle serait la plus efficace. Ce run nous a permis 71% de réussite avant le surapprentissage vers 521 epochs

Test 23: CIRFAR10_2CONVNET2_64_64_RELU_2MAXPOOL2_2_DROPOUT_2_2T64_S10_LR_1



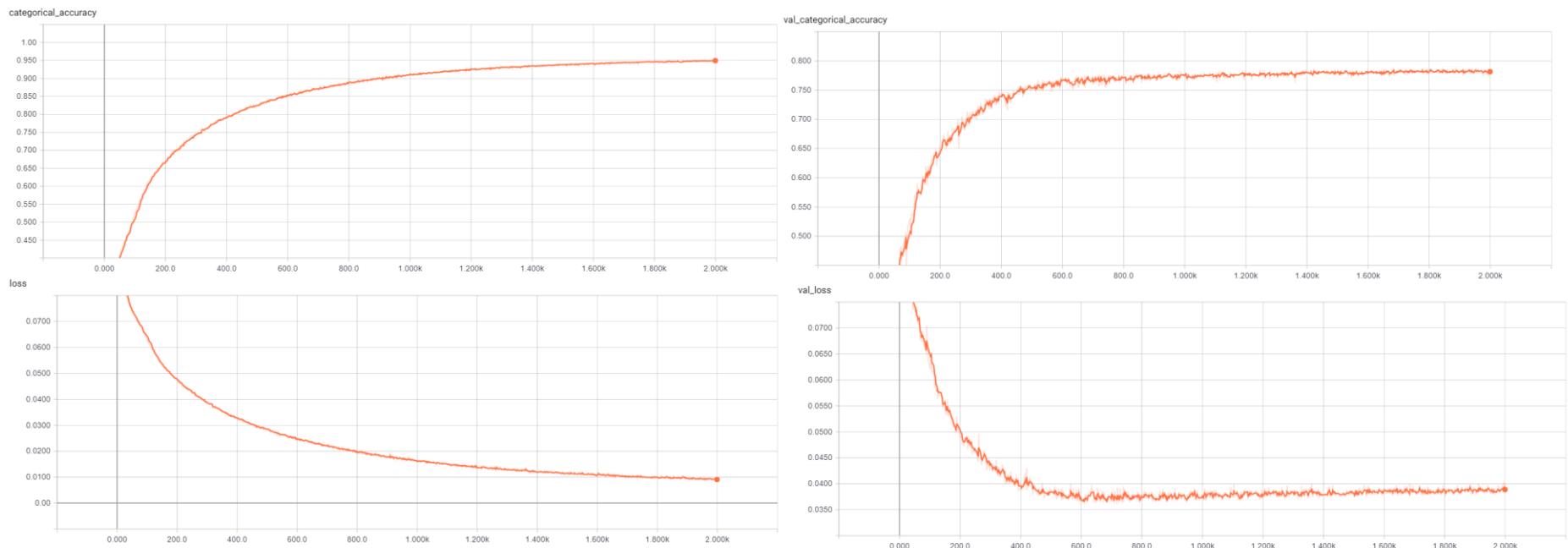
N'ayant pas eu de résultat permettant réellement quelle fonction d'activation fût la plus performante nous décidâmes de rajouter une couche supplémentaire (en tanh cette fois-ci). Le run a rapporté 74% avant de tomber dans le surapprentissage vers l'epoch 80 ce qui nous permet clairement de dire que l'ajout de neurones en couches cachées augmente la précision du modèle.

Test 24: CIFAR10_CONVNET2_64_RELU_MAXPOOL_DROPOUT_T64_S10_LR_1



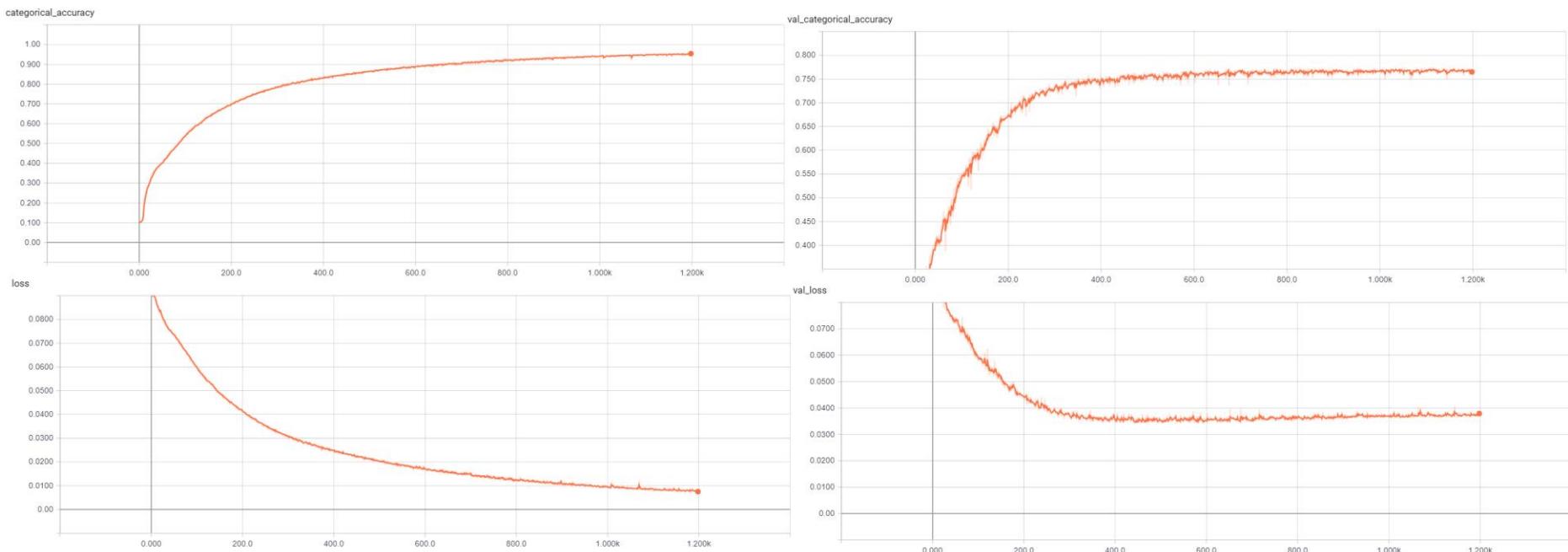
Ce test a été fait pour vérifier que nous avions bien besoin des deux blocs de convolutions dans notre modèle. Il a donc été réalisé avec une couche cachée de tanh à comparer avec le test 21. Ce run a atteint 63% ce qui est inférieur au test 21 donc l'ajout de bloc est justifié.

Test 25: CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_2R64_S10_LR_1



Ce test a été fait en parallèle du test 23 pour tenter une fois de plus de déterminer si la 'relu' ou la 'tanh' est plus performante avec deux couches cachées ajoutées. Ce run a atteint 76.5% avant le surapprentissage vers l'epoch 708, ce qui, cette fois est mieux, il semblerait que le relu soit donc plus performant que la tanh.

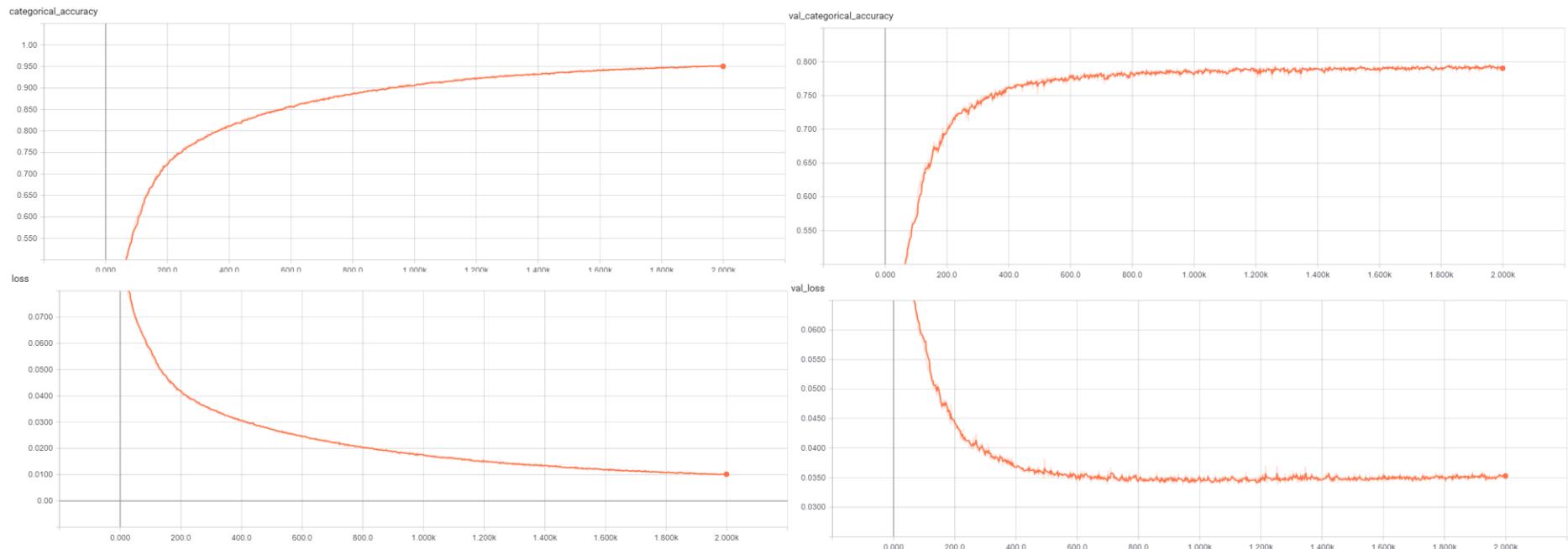
Test 26: CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_3T64_S10_LR_1



Toujours dans la démarche de tester si la relu est plus performante que la tanh.

Ce test a été fait cette fois-ci avec 3 couches cachées de tanh et nous a permis d'atteindre 72% avant le surapprentissage vers l'epoch 320 ce qui n'est pas beaucoup plus qu'avec seulement 2 couches.

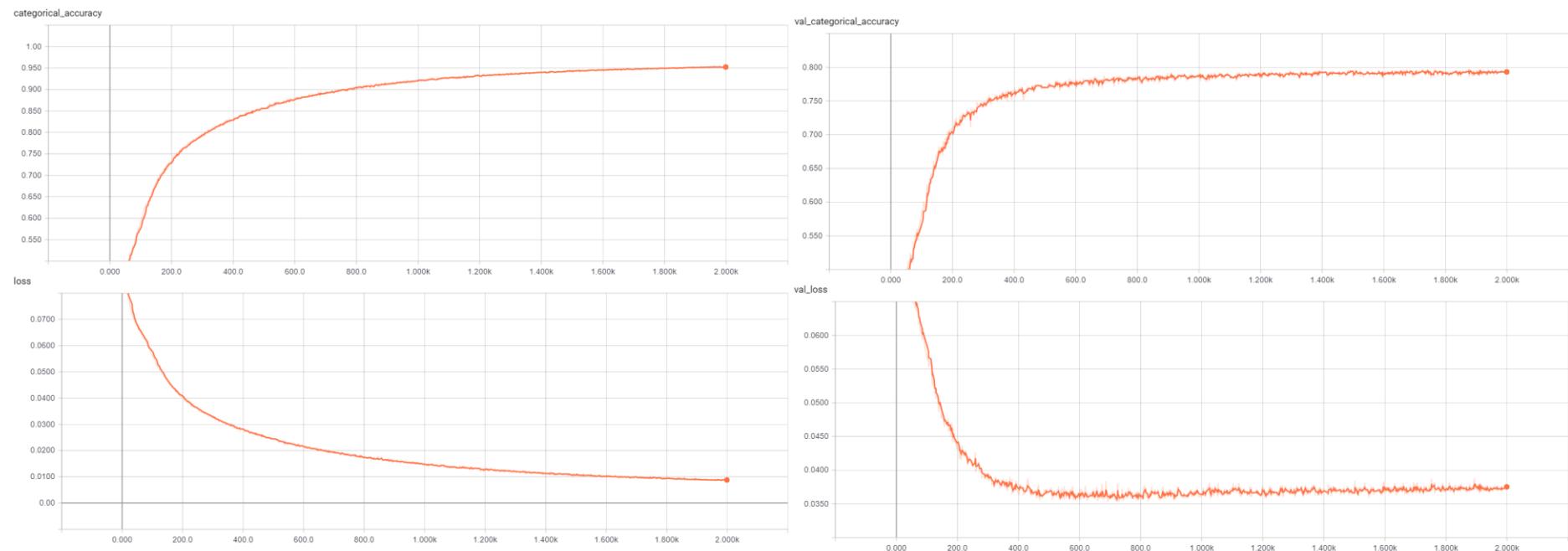
Test 27: CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_3E64_S10_LR_1



Ce test nous a permis de tester une nouvelle fonction d'activation qui est la 'elu'. Cette fonction ressemble à la fonction relu mais comporte les avantages de prendre en charge les valeurs négatives lorsque $x < 0$ et donc donne un gradient différent de 0 pour $x < 0$, cela permet d'éviter d'avoir des neurones morts. Le seul inconvénient est que la fonction est bien plus longue à calculer du fait de son exponentielle mais cela est compensé par sa vitesse de convergence.

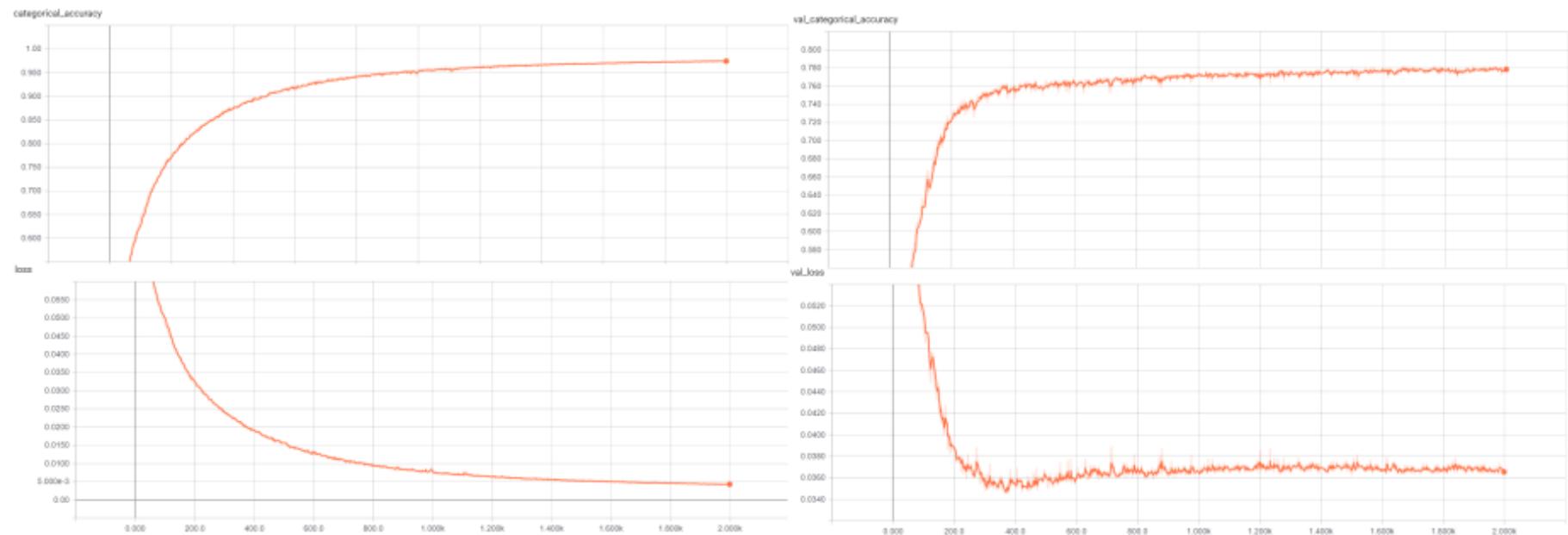
Ce run nous a permis d'atteindre 76.85% avant un surapprentissage vers l'epoch 450 ce qui est une amélioration pas très haute par rapport au test 23 avec 2 couches relu qui avait obtenu 76.5% de précision.

Test 28: CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_3R64_SM10_LR_1



Pour les 2 tests suivant nous avons voulu rechanger la fonction d'activation finale (sigmoid(10)) en softmax pour avoir un véritable run et de vrais résultats à exploiter. Et donc nous avons commencé avec 3 'relu' de dense 64 en couche cachées. Ce un a apporté 74% avant son surapprentissage vers 418 epoch ce qui est nettement inférieur à la sigmoïde.

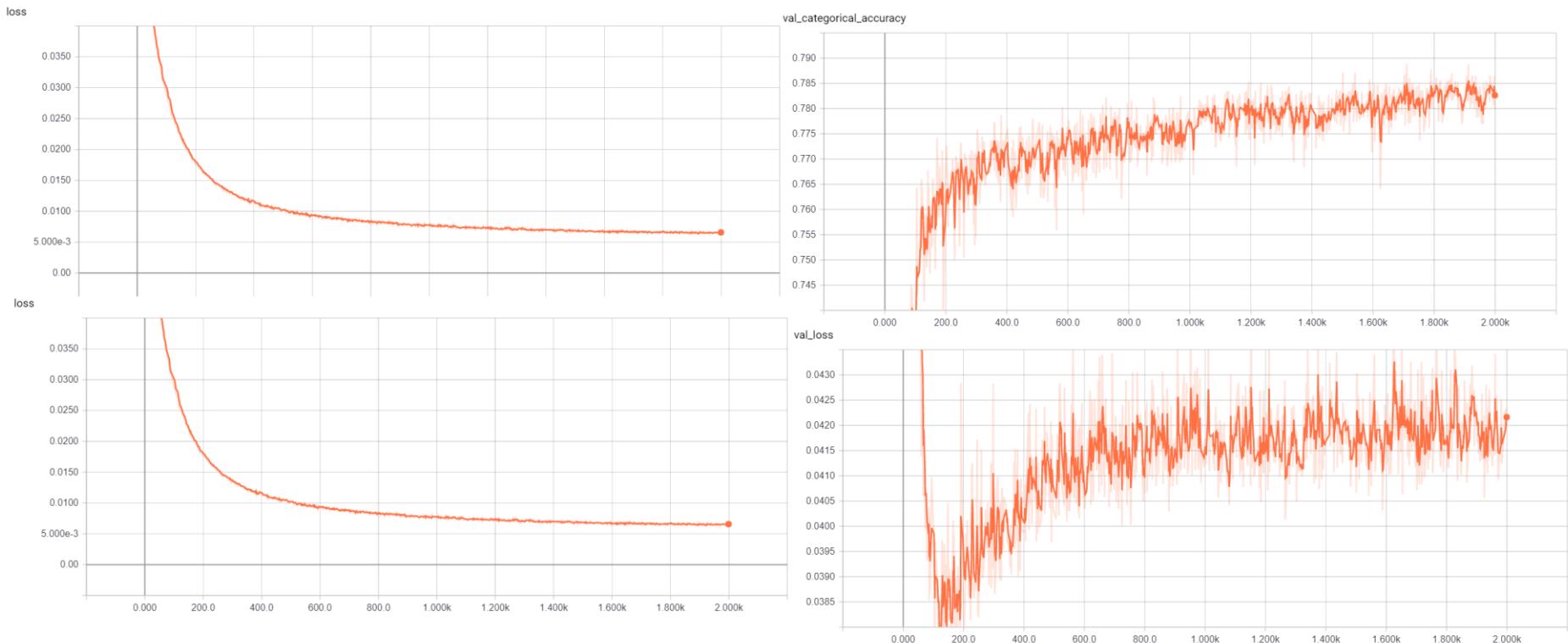
Test 29: CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_3E64_SM10_LR_1



Néanmoins il semblait intéressant de le tester avec la fonction 'elu' pour tenter d'observer une différence. Un run qui a obtenu 76.33% de précision avant son surapprentissage vers l'epoch 364 ce qui reste une bonne amélioration. Il semblerait que la fonction softmax donne de meilleurs résultats couplés à la fonction 'elu'.

En revanche utiliser une fonction relu avec sigmoid semble donner a peu de choses près les même résultat donc nous continuerons nos test plutôt avec la celle-ci.

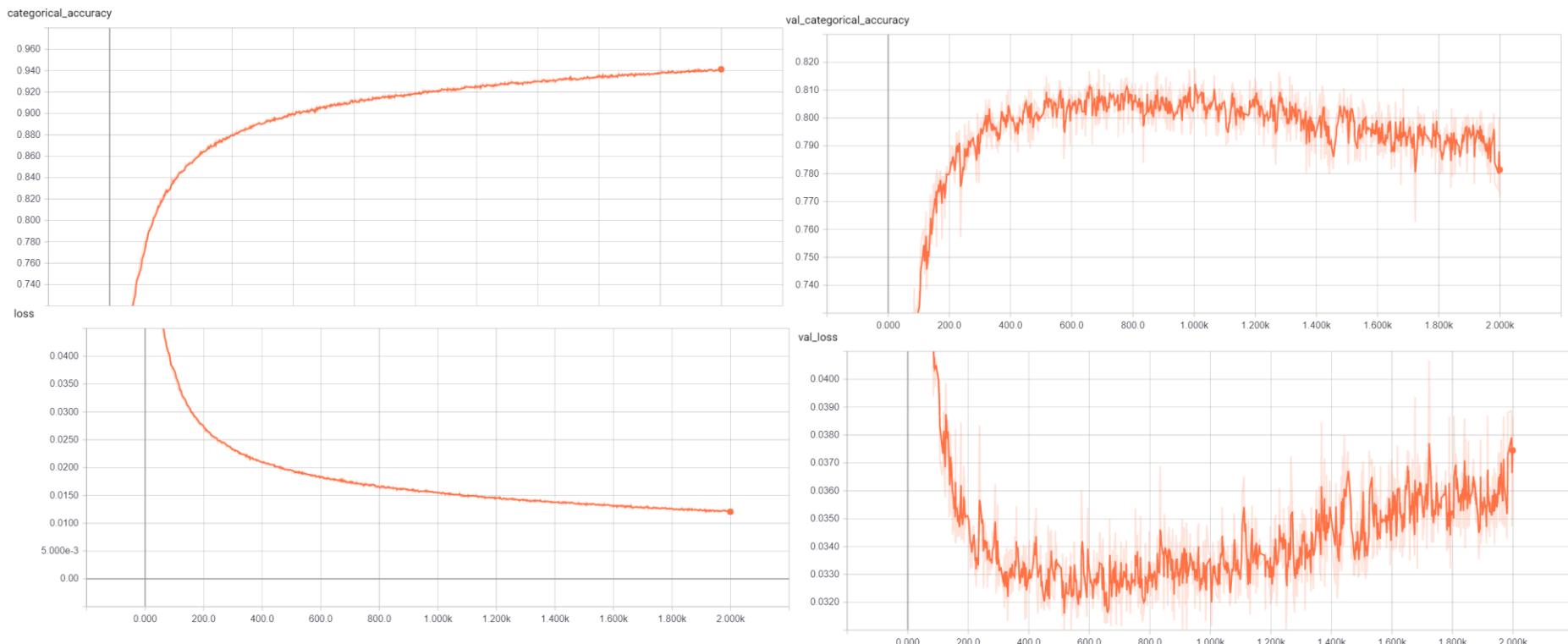
Test 30: CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT_3E64_S10_LR_1_RMSPROP



Dans ce test nous nous sommes servis d'un nouvel optimizer pour remplacer SGD qui est le RMSPROP. Cet optimizer est un algorithme à taux d'apprentissage adaptif. Il permet à l'algorithme de mieux se stabiliser vers l'optimum et aussi d'avoir un entraînement bien plus rapide.

Nous avons donc pu observer que le run un des plus rapides à obtenir des résultats avant le surapprentissage (450 contre 118 epochs pour le RMSPROP) et nous a apporter une précision de 76.12%.

Test 31: CIFAR10_2CONVNET2_64_64_RELU_MAXPOOL_2_DROPOUT50_3E64_S10_LR_1_RMSPROP



Dans ce dernier test nous avons décidé de monter les pourcentages du dropout car c'est quelque chose que nous avions laissé de côté durant les tests mais qui sert notamment à limiter le surapprentissage et le dernier test nous a permis d'en observer un très important. C'est pourquoi donc nous avons monté à 0.30 le dropout du premier bloc et à 0.50 celui du deuxième. Les résultats furent très bien accueillis car le surapprentissage fut donc détecter bien plus tard que sur le test 30 et nous a permis d'atteindre une précision de 80.07% sur notre modèle ce qui est notre meilleur résultat.