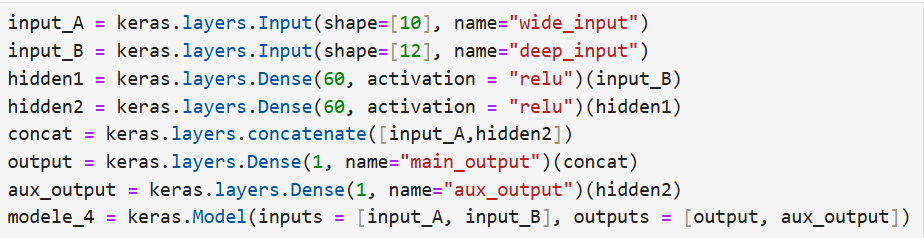
**Intelligence artificielle – Evaluation 2ème partie – réseaux profonds**

**Gaetan Corin**

**Exercice n°1 : Architecture**

Je vous propose le réseau de neurone suivant, implémenté avec Keras :

**

1. Représenter schématiquement ce réseau de neurone.

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

1. Indiquez, éventuellement au travers d’un exemple, le type de tâche qu’il pourrait être amené à accomplir.

Ce réseau de neurone pourrait servir à réaliser de la recommandation de contenu vidéo ou audio en prenant en compte par exemple

* Les caractéristiques de la personne visé(age, nationalité etc) en entré1(inputA)
* les caractéristiques de la vidéo en entré2(inputB).

La couche de sortie1 pourrait avoir comme résultat la probabilité que cette vidéo soit pertinente pour la personne visée

La couche de sortie2 pourrait avoir comme résultat la probabilité que cette vidéo soit pertinent sans prendre en compte de personne visée

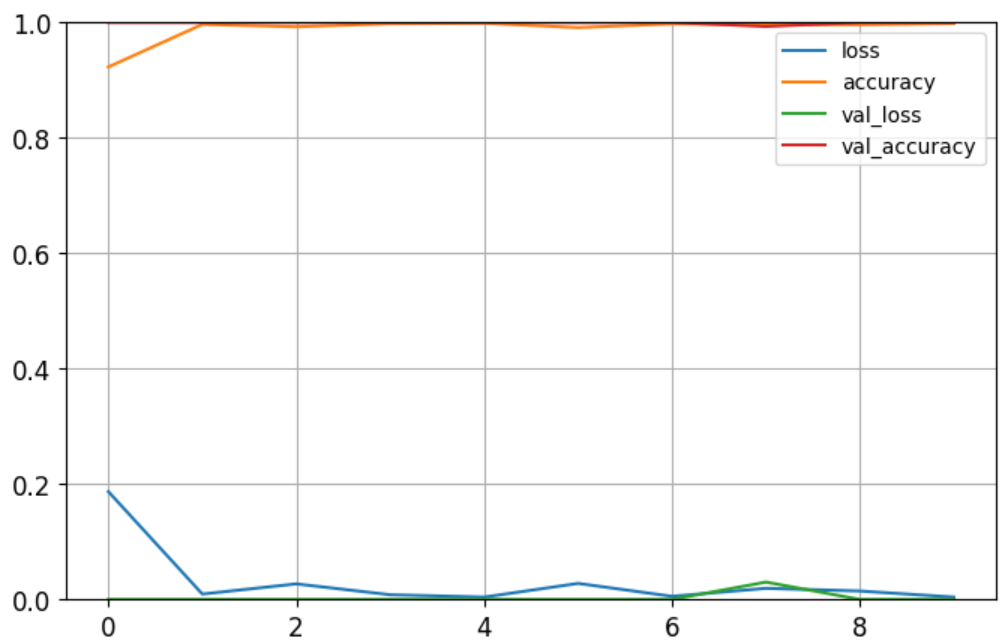
1. A votre avis, quelle pourrait être l’utilité de son architecture particulière ?

Cette architecture particulière permet de réaliser facilement un comparatif de résultat entre la couche de sortie1 et 2 en analysant rapidement l’impact que peux avoir la couche d’entré 1 sur le résultat final

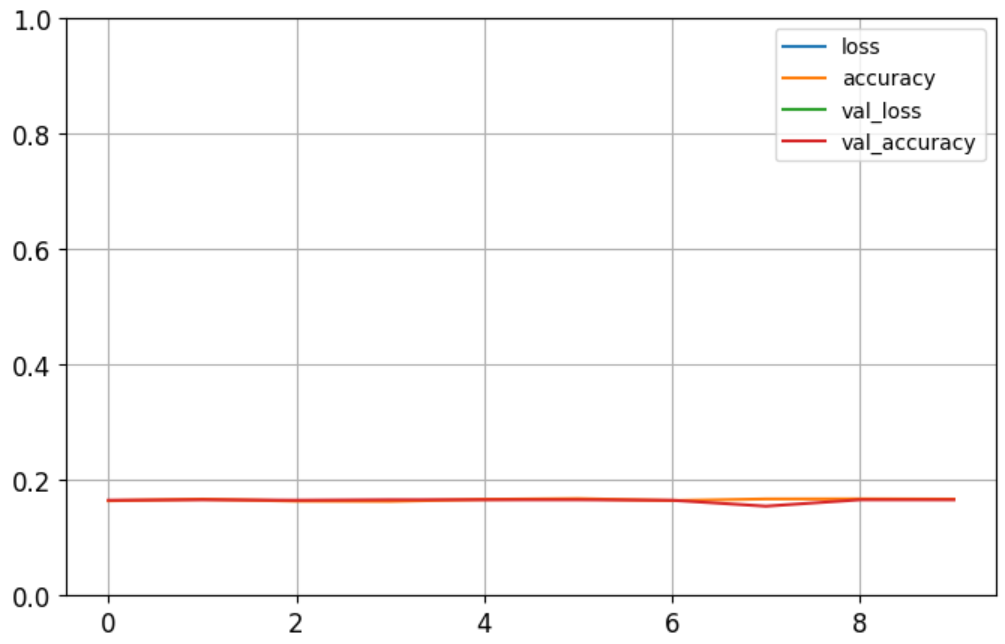
**Exercice n°2 : Apprentissage**

Vous disposez de deux programmes distincts, **pyt\_runes\_CNN\_1.ipynb** et **pyt\_runes\_CNN\_2.ipynb.** Ces programmes doivent permettre la classification de runes germaniques, tel qu’elle a été vu lors de l’exercice en autonomie. Après avoir exécuté les deux programmes, nous obtenons les courbes d’apprentissage suivantes :

**pyt\_runes\_CNN\_1**



**pyt\_runes\_CNN\_2**



1. Dans quel programme le réseau de neurone n’a-t-il pas réussi à apprendre ? En comparant les deux programmes, pouvez-vous identifier et expliquer pourquoi ?

C’est dans le programme pyt\_runes\_CNN\_2 que le programme n’a pas réussi à apprendre.  
Sur le programme pyt\_runes\_CNN\_1, les données d’entrée sont légèrement différentes car elles sont tournés sur un angles de +/- 20 degrés.  
Sur le programme pyt\_runes\_CNN\_2, les données d’entrée sont très différentes car elles sont tournés sur un angles de +/- 50 degrés.  
La fourchette d’angle qui permet de créer le jeux de données dans pyt\_runes\_CNN\_2 est trop grande, ce qui crée de la confusion dans l’apprentissage du modèle, qui n’arrive plus a faire de lien entre 2 images similaires transformés trop fortement.

1. Dans le programme ou l’apprentissage s’est passé correctement, estimez-vous qu’il y-a surapprentissage ? Argumentez brièvement votre réponse.

Si nous en jugeons uniquement par la courbe, nous pouvons effectivement en juger qu’il y a eu un surapprentissage et cela dès la première époque. En effet, après la première époque, et sur les 8 suivantes, les résultats du modèle entre les valeurs d’entrainement et de tests restent similaires à travers les époques. De plus, elles se touchent entre les valeurs d’entrainement et de tests et n’évoluent pas.  
Le fait que le modèle soit en surapprentissage dès l’époque 1 n’est pas normal pour un modèle. Cela peut venir d’un réseau de neurone surdimensionné par rapport à la tâche qu’il réalise.

1. Tel qu’il est configuré pour l’instant, le réseau de neurone est « surdimensionné » pour la tâche que l’on veut réaliser. Que modifieriez vous, dans son architecture, pour le rendre plus adapté et moins gourmand ?

J’arrêterais les filtres de defaultconv2D à 128 et je commencerais le réseau de neurone à 64.  
Je supprimerais donc les lignes suivantes :  
Une image contenant texte, capture d’écran, Police

Description générée automatiquement

**Exercice n°3 : classification de sons**

Vous disposez du programme **simple\_audio.ipynb** qui crée un réseau de neurone capable d’apprendre à classifier les signaux sonores. Il provient du site : [Reconnaissance audio simple : Reconnaître les mots-clés  |  TensorFlow Core](https://www.tensorflow.org/tutorials/audio/simple_audio?hl=fr)

Temps n°1 : Appropriation

1. Combien y’a-t-il de données respectivement dans le jeu de données d’entraînement et le jeu de données de validation ?

Il y a 6400 données dans le jeu d’entraînement et 1600 données dans le jeu de validation

1. De combien de couches est composé le réseau de neurones ?

Le réseau de neurones est composé de 11 couches

1. Combien y’a-t-il de connexions, au total, entre tous les neurones du réseau ?
2. Utilisez les courbes d’apprentissage pour décrire, en quelques lignes, comment s’est déroulé l’entraînement du réseau de neurone.

D’après les courbes d’apprentissages d’accuracy et de perte, le réseau de neurone a suivi un entrainement adapté a la tache qui lui a été confié. Nous pouvons voir les courbes loss et accuracy qui continue a progresser durant toute la durée des époques. Les courbes val\_loss et val\_accuracy, quand à elles, continue elle aussi a progresser dans le même temps.  
Le modèle, à la première époque, donne de meilleur résultat sur les données de validation que sur les données d’entrainement, ce qui est surprenant, car cela pourrait dire que les données de validation sont plus difficile que les données d’entrainement.

1. A partir de la dernière figure, pouvez-vous déterminer :
   1. Quel est le mot qui est le plus efficacement identifié

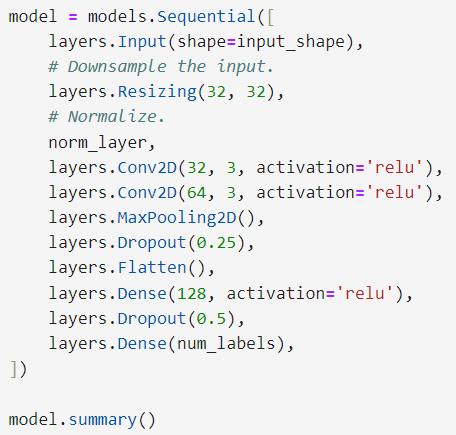
Le mot « Left » est celui le plus identifié avec une matrice de confusion à 101 sur son label

* 1. Quels sont les deux mots qui sont le plus souvent confondus

Les mots « Go » et « No » sont les plus confondus, principalement lorsque le mot est labélisé « Go »

Temps n°2 : Evolution

Le modèle définit étant le suivant, que pourriez-vous modifier pour obtenir directement la probabilité d’appartenance d’un mot à chacune des catégories ?



Nous pourrions utilisé la fonction d’activation softmax ce qui permet de modifier la dernière cuche du modèle en probabilité.  
(layers.Dense(num\_labels, activation= « softmax »)

Temps n°3 : Adaptation

Vous disposez d’un échantillon différent du jeu de données initial, dans le dossier **recordings.zip.** Cet échantillon est issu de l’open data et disponible à l’adresse : [datasets/tensorflow\_datasets/datasets/spoken\_digit/spoken\_digit\_dataset\_builder.py at master · tensorflow/datasets (github.com)](https://github.com/tensorflow/datasets/blob/master/tensorflow_datasets/datasets/spoken_digit/spoken_digit_dataset_builder.py)

Le jeu de données est l’équivalant audio du MNIST, et contient 2500 enregistrements de chiffres (‘zero’, ‘one’, ‘two’, ‘three’, ‘four’, ‘five’, ‘six’, ‘seven’, ‘height’, ‘nine’). Il s’agit de 50 enregistrements des chiffres de 0 à 9, prononcés par 5 personnes différentes.

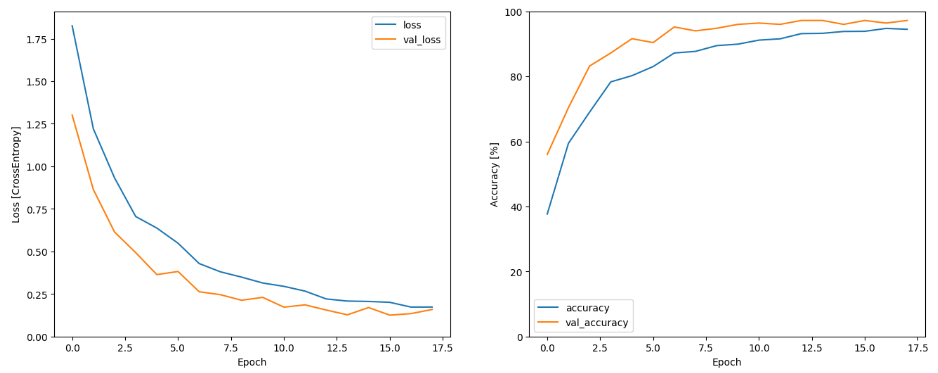
1. Adaptez le programme de manière à ce qu’il puisse apprendre à distinguer ces nouveaux mots. (Il est fortement déconseillé de modifier le nombre de données du nouveau jeu (2500 données). Un apprentissage sur cette base ne devrait pas poser de problème.)

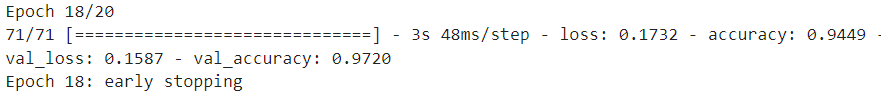
*Attention, c’est assez compliqué ! Si vous n’y arrivez pas, commencez par décrire, à l’écrit, ce que vous aviez l’intention de faire.*

*Je vais créer une boucle qui va récupérer les fichiers ainsi que les labelsnames.  
Les fichiers wav seront transformé en spectogramme pour réaliser la donnée d’entrée et les labelsname serviront a labélisé la donnée.*

*Je mélange la donnée.  
Sur ces 2500 données, 2000 seront des données d’entrainement et 500 seront des données de validation.  
Je fais tourner le même modèle que précédemment avec ces données et j’obtiens les courbes d’entrainement et de validation.*

1. Une fois le modèle adapté et ré-entrainé sur 20 époques, nous obtenons les courbes suivantes, ainsi que la dernière ligne de l’entraînement :





Que pensez-vous de l’apprentissage de notre modèle sur ces nouvelles données ?

L’apprentissage du modèle se comporte parfaitement bien sur les courbes d’accuracy et sur les courbes de pertes.  
Les courbes de données de validation commence sur l’époque 1 avec un résultat plus faible que sur les courbes de données d’entrainement, ce qui est normal.  
Par la suite, les courbes continuent à s’améliorer à travers les époques jusqu’au moment ou les données d’entrainement se stabilise et que les résultats se croisent. Le nombre d’époque est donc arrêter sur ce moment-là.