La Librairie Keras

La Librairie Keras : La librairie du Deep learning

keras makes Deep Learning simple

http://keras.io/

Keras : une librairie de haut niveau pour concevoir vos modèles de Deep Learning

- TensorFlow, PyTorch : librairies de plus bas niveau
  - Utiles pour des chercheurs en informatique
- Keras : une librairie pour praticiens

METEO FRANCE

Quand utiliser Keras ?

- Différences entre Keras et scikit-learn:
  - Keras :
    - flexibilité dans la création d'architecture des réseaux de neurones
    - utilisation du GPU
- Quand utiliser Keras au lieu de scikit-learn?
  - Dès qu’on veut faire du Deep Learning (réseaux de neurones profonds)

METEO FRANCE

Le modèle linéaire sur Keras

```
from keras.layers import Dense
from keras.models import Sequential


def linear_model_keras():
    model=Sequential() #création d'un modèle séquentiel
    model.add(Dense(1,input_shape=(9,))) # j'ajoute le Layer "Dense" au modèle
    return model # je renvoie le résultat du layer Dense
model=linear_model_keras()
```

- Le paramètre 1 de “Dense” correspond au nombre de modèles linéaires que je vais créer.
- Le nombre de paramètres en sortie de la couche Dense est égal au nombre de modèles linéaires que je vais créer.
- Ici, pour obtenir un modèle linéaire, je choisis simplement le paramètre 1.

Exemples de Layers (couches) Keras

- Convolution : Conv2D
- Fonction d’activation : Activation (relu, sigmoid , tanh, etc..)
- Dropout : Dropout
- Combinaison linéaire : Dense

METEO FRANCE

Le modèle linéaire sur Keras

- Rappel : sous scikit-learn, le modèle linéaire est déjà implémenté
- Sur Keras, on peut implémenter le modèle linéaire comme cas particulier du Layer “Dense”.
- En effet, le modèle linéaire est équivalent à un neurone sans fonction d'activation.

$$f(x)=Ax+b$$

METEO FRANCE

Régression linéaire avec le code Keras

```python
from keras.layers import Dense
from keras.models import Sequential

def linear_model_keras():
    model=Sequential() #création d'un modèle séquentiel
    model.add(Dense(1,input_shape=(9,))) # j'ajoute le Layer "Dens
    return model # je renvoie le résultat du layer Dense
model=linear_model_keras()
model.compile(optimizer="SGD",loss="mean_squared_error")
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_108 (Dense)            (None, 1)                 10
=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
```

METEO FRANCE

Régression logistique avec le code Keras

```python
from keras.layers import Dense
from keras.models import Sequential

def linear_model_keras_classification():
    model=Sequential() #création d'un modèle séquentiel
    model.add(Dense(1,input_shape=(9,))) # j'ajoute le Layer "Dense" au modèle
    model.add(Activation("sigmoid")) # Je renvoie une probabilité
    return model # je renvoie le résultat du layer Dense
model=linear_model_keras()
model.compile(optimizer="SGD",loss="binary_crossentropy")
model.summary()
```

nombre entre 0 et 1

fonction de perte pour classification

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_111 (Dense)            (None, 1)                 10
=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
```

Librairie scikit-learn comparée à la Librairie Keras

```
model_sklearn=sklearn.linear_model.LogisticRegression()
model_sklearn.fit(X,Y)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

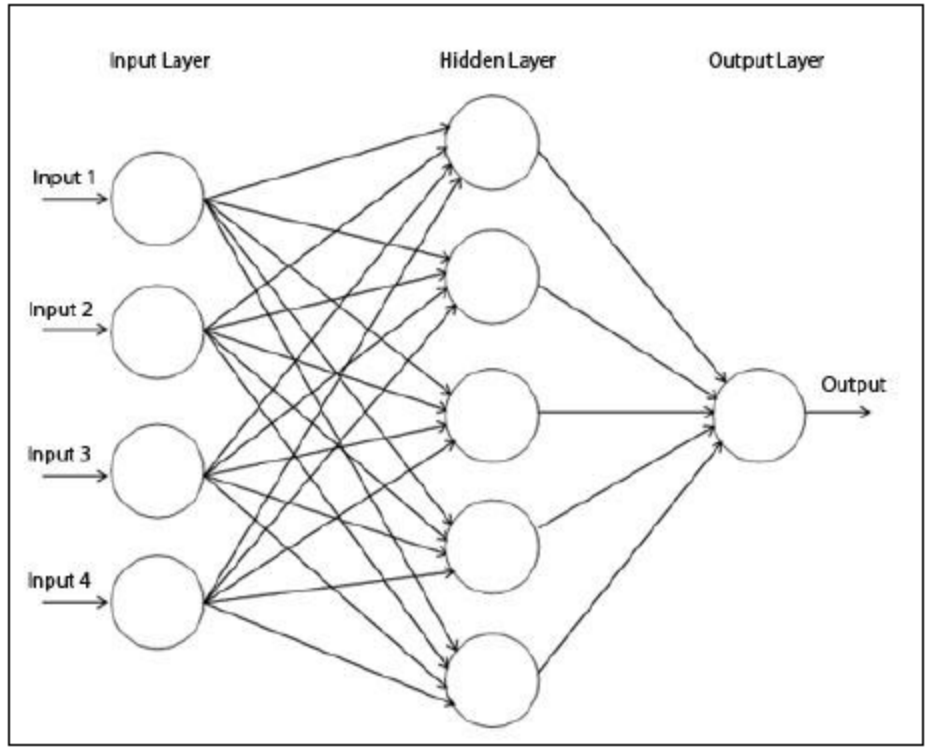
```
model_keras=nn_classification()
model_keras.fit(X,Y,epochs=3,batch_size=32)
```

```
Epoch 1/3
20640/20640 [==============================] - 2s 113us/step - loss: 1.2904 - acc: 0.4720
Epoch 2/3
20640/20640 [==============================] - 1s 69us/step - loss: 0.9831 - acc: 0.6365
Epoch 3/3
20640/20640 [==============================] - 1s 71us/step - loss: 0.8706 - acc: 0.6811
<keras.callbacks.History at 0x7f92b419def0>
```

nombre de fois où le modèle voit les données

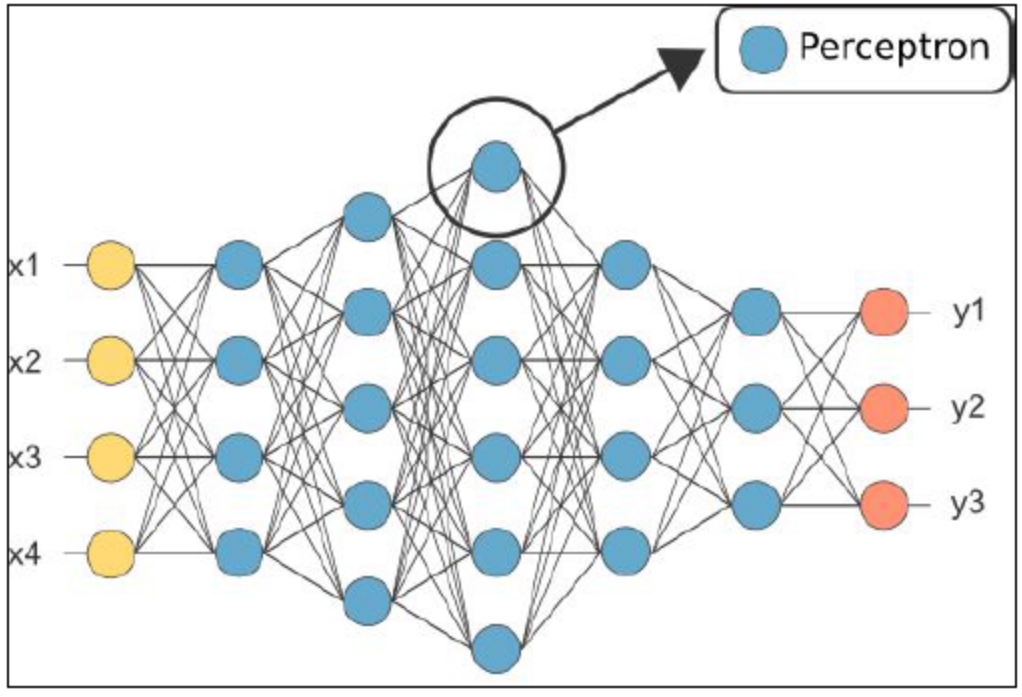
nombre d'observations envoyés en même temps

Exercice 1



Ecrivez le code Keras correspondant à ce réseau de neurones.

Exercice 2



Ecrivez le code Keras correspondant à ce réseau de neurones.

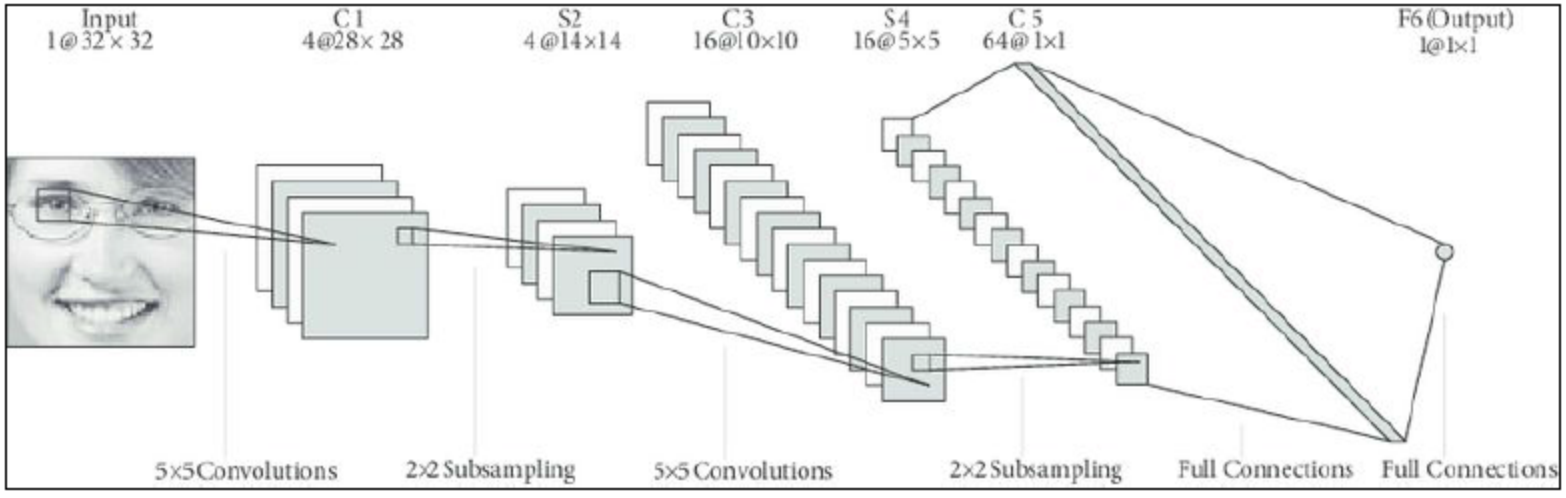
Exercice 3

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Dessinez l'architecture correspondante

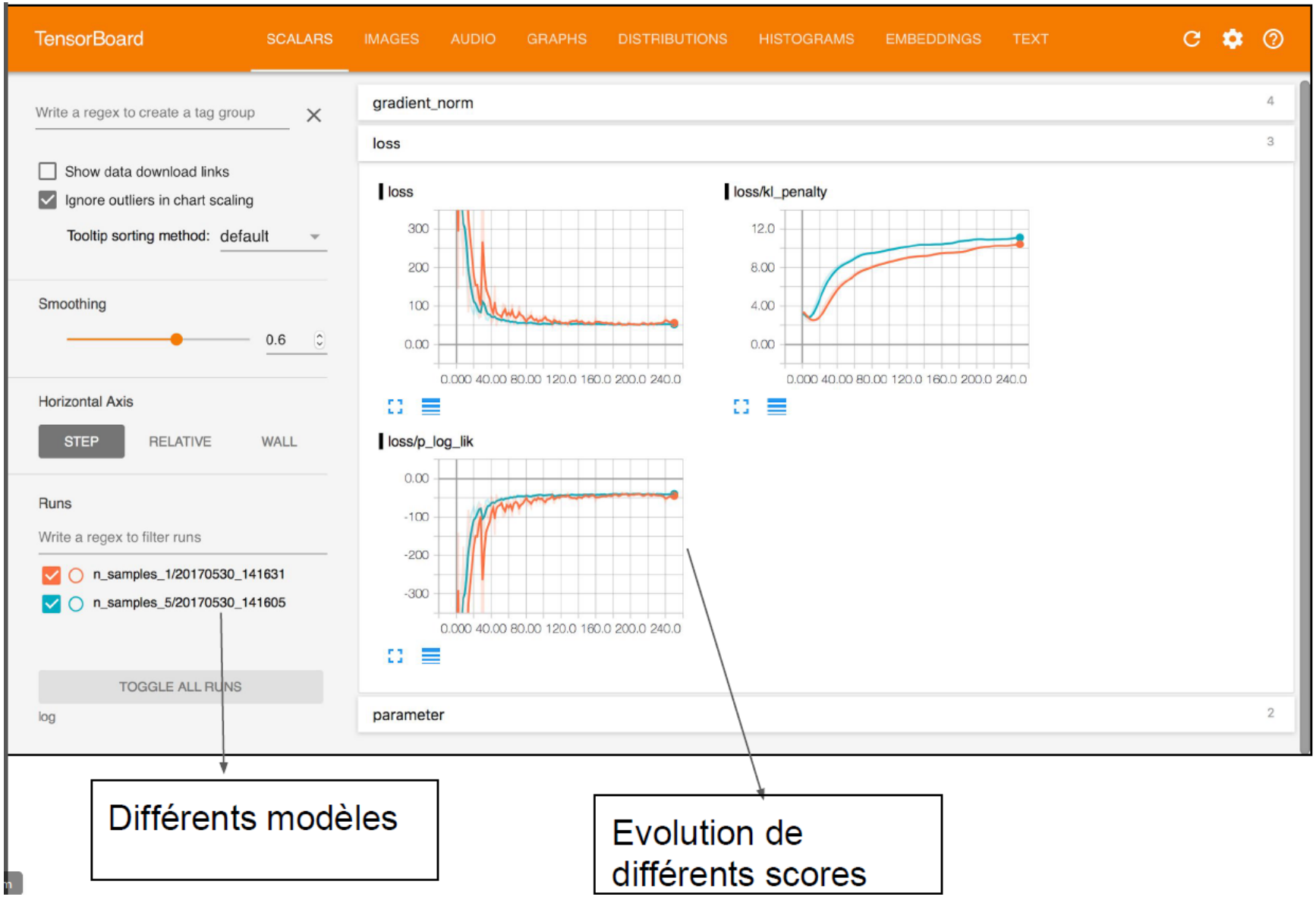
METEO FRANCE

Exercice 4



Donnez le code Keras correspondant à ce réseau de neurones.

Visualisation de l'entrainement :



Questions ?