مكتب التكوين المهني و إنعاش الشغل

Office de la formation professionnelle et de la promotion du travail

Direction Régionale : Casablanca Settat *Cité des métiers et des compétences*

Contrôle continu $N°3$: APPRENTISSAGE PROFOND (M108)
Année 2024/2025

Filière : Intelligence Artificielle
Niveau : TS
Groupe : IA103
Nom et Prénom : .....

Durée : 2H
Variante : 2
Barème : /20

QCM Transfert Learning et LSTM (5 x 1 pt = 5 points)

Cochez une seule réponse correcte par question.

Question 1 : Quelle étape est généralement réalisée avant d'entraîner un modèle en transfert learning ?

- [ ] Supprimer les poids du modèle pré-entraîné
- [ ] Remplacer le dataset ImageNet par CIFAR-10
- [X] Ajouter une ou plusieurs couches spécifiques à la nouvelle tâche
- [ ] Réduire la profondeur du réseau en supprimant des couches

Question 2 : Quel est l'intérêt principal d'utiliser une architecture LSTM plutôt qu'un simple RNN ?

- [ ] Réduire le besoin de GPU pour l'entraînement
- [ ] Accélérer l'inférence sur des images
- [X] Mieux gérer les longues dépendances temporelles dans les séquences
- [ ] Réduire le nombre de paramètres

Question 3 : Quelle méthode permet de mettre à jour seulement les nouvelles couches ajoutées au modèle lors du fine-tuning ?

- [ ] Supprimer les anciennes couches
- [X] Geler les couches du modèle de base
- [ ] Réduire le taux d'apprentissage à zéro
- [ ] Convertir le modèle en format ONNX

Question 4 : Lequel des éléments suivants fait partie du mécanisme interne d'une cellule LSTM ?

- [ ] Une couche de convolution
- [X] Une porte d'oubli (forget gate)
- [ ] Un filtre de pooling
- [ ] Un vecteur de bruit aléatoire

Question 5 : Quelle stratégie décrit le mieux l'approche "fine-tuning" après un premier entraînement en transfert learning ?

- [ ] Réentraîner toutes les couches d'un modèle pré-entraîné depuis zéro
- [ ] Ajouter uniquement une couche de sortie
- [X] Réentraîner certaines couches supérieures du modèle de base en plus des nouvelles
- [ ] Appliquer un filtre de bruit sur les données d'entrée

Exercice 1 : Prédiction de la température corporelle par LSTM (7 points)

Vous travaillez au sein d'un centre hospitalier spécialisé dans le suivi de patients en soins intensifs. On vous confie la tâche de développer un modèle d'apprentissage profond de type **LSTM** capable de prédire la `température corporelle` des patients à partir de mesures vitales collectées quotidiennement.

Extrait du fichier utilisé : `patient_vitals.csv`

Date	Fréquence_Cardiaque	Saturation_O2	CRP	Température
01/01/2023	92	96	18.2	37.8
02/01/2023	100	95	20.5	38.1
03/01/2023	105	93	22.8	38.4
...	...	...	...	...

Créez un modèle LSTM qui prédit la `Température` du jour à partir des données des **4 jours précédents** concernant :

- la `Fréquence_Cardiaque`,
- la `Saturation_O2`,
- la `CRP` (protéine C-réactive).

Chargement et Prétraitement des données (2 points)

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Chargement des données
df = pd.read_csv("patient_vitals.csv")
df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)
df.set_index('Date', inplace=True)

# Normalisation
scaler = MinMaxScaler(feature_range=(0, 1))
#autre reponse: data = scaler.fit_transform(df[['Fréquence_Cardiaque', 'Saturation_O2', 'CRP', '
    Température']])

data = scaler.fit_transform(df)

# Séparation 75% / 25%
train_size = int(len(data) * 0.75)
train_data = data[:train_size]
test_data = data[train_size:]
```

–

On souhaite créer des séquences de 4 jours glissants. Complétez le code suivant : (3 points)

```
def create_dataset(dataset, time_step=1, input_cols=[0], target_col=0):
    X, Y = [], []
    for i in range(len(dataset) - time_step - 1):
        window = dataset[i : i + time_step, input_cols]
        target = dataset[i + time_step, target_col]
        X.append(window)
        Y.append(target)
    return np.array(X), np.array(Y)

#je compte en consédiration input_cols = [0,1,2]
X_train, y_train = create_dataset(train_data, time_step=4, input_cols=[0,1,2,3], target_col=3)
X_test, y_test   = create_dataset(test_data, time_step=4, input_cols=[0,1,2,3], target_col=3)
```

—

Complétez la suite du code pour construire, compiler et entraîner le modèle : (2 points)

```
model = Sequential()
#je compte en consédiration input_shape=(time_step=4,n_features=3)
model.add(LSTM(50, return_sequences=False, input_shape=(4, 4)))
model.add(Dense(50))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
Histo = model.fit(X_train, y_train,
          validation_data=(X_test, y_test),
          epochs=10,
          batch_size=64)

# Prédiction
y_pred = model.predict(X_test)
```

Exercice 2 : Classification d'images industrielles (8 points)

Construire un modèle de classification à 3 classes à l'aide de `DenseNet169` en transfert learning.

1. Importation du modèle pré-entraîné (3 points)

```
# Import des bibliothèques nécessaires
import tensorflow as tf
from tensorflow.keras.applications import DenseNet169
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout, Input
from tensorflow.keras.optimizers import Adam

# Définir la taille d'entrée des images (ex: 224x224 pixels RGB)
input_shape = (224, 224, 3)

# Charger DenseNet169 sans la tête (include_top=False)
base_model = DenseNet169(weights='imagenet', include_top=False, input_shape=input_shape)

# Geler les couches du modèle de base
base_model.trainable = False
```

2. Construction et compilation du modèle (3 points)

```
# Création du modèle séquentiel
model = Sequential()

# Ajouter la base DenseNet comme première couche (modèle fonctionnel encapsulé dans une couche
    Keras)
model.add(base_model)

# Ajouter les couches personnalisées
model.add(GlobalAveragePooling2D())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(3, activation='softmax'))


# Compilation du modèle
model.compile(optimizer=Adam(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

3. Entraînement du modèle **(2 points)**

```
# Entraînement pendant 20 époques avec batch_size de 32
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=20,
    batch_size=32
)
```

Remarque : Cette approche utilise un gel des couches de `DenseNet169`. Pour aller plus loin, on pourrait ensuite "dégeler" progressivement certaines couches pour un *fine-tuning* plus précis.