Fiche solution CNN - Detaillee

Fiche solution detaillee - Reseaux convolutifs (CNN)

1. **Pretraitement des donnees :**

- **Pourquoi normaliser les images ?**

  Les pixels des images MNIST ont des valeurs entre 0 et 255. Normaliser ces valeurs entre 0 et 1

  permet d accelerer l apprentissage et de rendre le modele plus stable.

- **Pourquoi reformater les donnees ?**

  Le CNN attend des images en format [N, largeur, hauteur, canaux], ici [N, 28, 28, 1].

  Le dernier "1" indique qu il s agit d images en niveaux de gris (un seul canal).

Code :

```python
X_train = X_train_data.reshape(-1, 28, 28, 1) / 255.0
X_test = X_test_data.reshape(-1, 28, 28, 1) / 255.0
```

2. **Construction du modele CNN :**

- **Pourquoi utiliser des couches convolutives ?**

  Les couches convolutives extraient automatiquement des caracteristiques importantes (bords, motifs)

  des images sans necessiter d ingenierie manuelle. Elles utilisent des filtres pour detecter ces motifs.

Fiche solution CNN - Detaillee

- **Role de chaque composant :**

  - **Conv2D (Convolution 2D) :** Identifie des motifs dans les images.

    - 1ere couche : 32 filtres (large capacite pour capturer les motifs initiaux).

    - 2eme couche : 16 filtres (reduite progressivement la complexite).

  - **Flatten :** Aplatit les donnees en un vecteur pour les passer a la couche Dense.

  - **Dense :** Effectue la classification. Ici, 10 neurones pour 10 classes (0 a 9).

Code :

```python
model = Sequential([
    Conv2D(32, kernel_size=3, padding='same', activation='relu', input_shape=(28, 28, 1)),
    Conv2D(16, kernel_size=3, padding='same', activation='relu'),
    Flatten(),
    Dense(10, activation='softmax')
])
```

3. **Compilation et entrainement :**

- **Pourquoi Adam comme optimiseur ?**

  Adam combine les avantages de deux methodes populaires (momentum et RMSprop) pour ajuster les poids rapidement et de maniere stable.

- **Pourquoi categorical_crossentropy ?**

  C est la fonction de perte adaptee a la classification multiclasse.

- **Impact du nombre d epoques et de la taille des lots :**
  - **Epoques :** Plus le nombre d epoques est eleve, plus le modele s ameliore (jusqu a un certain point). Trop d epoques peuvent entrainer un surapprentissage.
  - **Taille des lots :** Petite taille : mise a jour frequente des poids (rapide mais bruitee). Grande taille : apprentissage stable mais plus lent.

Code :

```python
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, Y_train, epochs=5, batch_size=32, validation_split=0.2)
```

4. **Evaluation et analyse des resultats :**
- **Perte et precision :**
  - **Perte :** Mesure l erreur commise par le modele. Une perte faible indique que le modele predit bien.
  - **Precision :** Proportion des predictions correctes.
- **Interpretation :**
  - Si precision d entrainement > precision de validation : possible surapprentissage.
  - Si perte de validation reste elevee : ajustez les hyperparametres.

Code :

```python
test_loss, test_accuracy = model.evaluate(X_test, Y_test)
```

print(f"Test Loss: {test_loss}")

print(f"Test Accuracy: {test_accuracy}")

```

5. **Exercices et observations :**

1. **Troisieme couche convolutive :**

- Ajoutez une couche avec 8 filtres :

```python

model.add(Conv2D(8, kernel_size=3, activation='relu'))

```

- Observation : Plus de couches permettent au modele d apprendre des caracteristiques plus complexes.

2. **Modification des epoques :**

- Passez de 5 a 10 epoques :

```python

history = model.fit(X_train, Y_train, epochs=10, batch_size=32, validation_split=0.2)

```

- Observation : La precision peut augmenter mais attention au surapprentissage (ecart entre validation et entrainement).

3. **Changement d optimiseur :**

- Essayez RMSprop :

```python

```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

- Observation : RMSprop est souvent performant pour des donnees avec des gradients irreguliers.

4. **Tracer les courbes :**

- Code pour tracer les pertes :

```python
import matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
```

- Observation : Si les courbes divergent, ajustez le modele (reduisez les epoques, changez le taux d apprentissage).

En resume, ce modele CNN utilise des outils puissants pour detecter des motifs dans les images. L ajustement des hyperparametres est crucial pour obtenir de bons resultats.