CNN – Classification Intel Image Classification

Objectif : entraîner un CNN pour une classification binaire et analyser l'impact du nombre de convolutions.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropou
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import os
from glob import glob
```

In [10]:
```
IMG_SIZE = (150, 150)
BATCH_SIZE = 32
EPOCHS = 20
```

In [11]:
```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    zoom_range=0.2,
    shear_range=0.2,
    horizontal_flip=True
)

validation_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    "../dataset/seg_train",
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode="categorical"  # MULTICLASSE
)

validation_generator = validation_datagen.flow_from_directory(
    "../dataset/seg_test",
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode="categorical"  # MULTICLASSE
)
```

```
Found 14034 images belonging to 6 classes.
Found 3000 images belonging to 6 classes.
```

In [12]:
```
def build_cnn(nb_conv):
    model = Sequential()

    # 1ère convolution
    model.add(Conv2D(32, (3,3), activation='relu', input_shape=(150,150,3)))
    model.add(MaxPooling2D((2,2)))

    # 2ème convolution
    model.add(Conv2D(32, (3,3), activation='relu'))
    model.add(MaxPooling2D((2,2)))
```

```
    # 3ème convolution OPTIONNELLE
    if nb_conv == 3:
        model.add(Conv2D(64, (3,3), activation='relu'))
        model.add(MaxPooling2D((2,2)))

    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))           # régularisation pour éviter overfitting
    model.add(Dense(6, activation='softmax'))  # 6 classes pour Intel

    model.compile(
        optimizer='adam',
        loss='categorical_crossentropy',  # multiclasse
        metrics=['accuracy']
    )


    return model
```

In [13]:
```
def get_callbacks(nb_conv):
    ckpt_dir = f"checkpoints_cnn_{nb_conv}"
    os.makedirs(ckpt_dir, exist_ok=True)

    checkpoint = ModelCheckpoint(
        filepath=os.path.join(ckpt_dir, "weights_epoch_{epoch:02d}.h5"),
        monitor="val_accuracy",
        save_weights_only=True,
        save_best_only=True,
        verbose=1
    )

    early_stop = EarlyStopping(
        monitor="val_accuracy",
        patience=5,
        restore_best_weights=True,
        verbose=1
    )

    return checkpoint, early_stop, ckpt_dir
```

In [14]:
```
model_2 = build_cnn(nb_conv=2)

checkpoint, early_stop, ckpt_dir = get_callbacks(2)

history_2 = model_2.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    epochs=20,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // BATCH_SIZE,
    callbacks=[checkpoint, early_stop],
    verbose=1
)
```

```
Epoch 1/20
438/438 [==============================] - ETA: 0s - loss: 1.1739 - accuracy: 0.5
390
Epoch 1: val_accuracy improved from -inf to 0.67641, saving model to checkpoints_
cnn_2\weights_epoch_01.h5
438/438 [==============================] - 292s 666ms/step - loss: 1.1739 - accur
acy: 0.5390 - val_loss: 0.8306 - val_accuracy: 0.6764
Epoch 2/20
438/438 [==============================] - ETA: 0s - loss: 0.9267 - accuracy: 0.6
525
Epoch 2: val_accuracy did not improve from 0.67641
438/438 [==============================] - 174s 398ms/step - loss: 0.9267 - accur
acy: 0.6525 - val_loss: 0.9686 - val_accuracy: 0.6378
Epoch 3/20
438/438 [==============================] - ETA: 0s - loss: 0.8216 - accuracy: 0.6
948
Epoch 3: val_accuracy improved from 0.67641 to 0.75571, saving model to checkpoin
ts_cnn_2\weights_epoch_03.h5
438/438 [==============================] - 165s 376ms/step - loss: 0.8216 - accur
acy: 0.6948 - val_loss: 0.6753 - val_accuracy: 0.7557
Epoch 4/20
438/438 [==============================] - ETA: 0s - loss: 0.7598 - accuracy: 0.7
258
Epoch 4: val_accuracy improved from 0.75571 to 0.77991, saving model to checkpoin
ts_cnn_2\weights_epoch_04.h5
438/438 [==============================] - 155s 354ms/step - loss: 0.7598 - accur
acy: 0.7258 - val_loss: 0.6280 - val_accuracy: 0.7799
Epoch 5/20
438/438 [==============================] - ETA: 0s - loss: 0.7020 - accuracy: 0.7
500
Epoch 5: val_accuracy improved from 0.77991 to 0.78931, saving model to checkpoin
ts_cnn_2\weights_epoch_05.h5
438/438 [==============================] - 157s 357ms/step - loss: 0.7020 - accur
acy: 0.7500 - val_loss: 0.6087 - val_accuracy: 0.7893
Epoch 6/20
438/438 [==============================] - ETA: 0s - loss: 0.6572 - accuracy: 0.7
659
Epoch 6: val_accuracy did not improve from 0.78931
438/438 [==============================] - 159s 363ms/step - loss: 0.6572 - accur
acy: 0.7659 - val_loss: 0.8201 - val_accuracy: 0.7151
Epoch 7/20
438/438 [==============================] - ETA: 0s - loss: 0.6300 - accuracy: 0.7
778
Epoch 7: val_accuracy improved from 0.78931 to 0.79301, saving model to checkpoin
ts_cnn_2\weights_epoch_07.h5
438/438 [==============================] - 157s 358ms/step - loss: 0.6300 - accur
acy: 0.7778 - val_loss: 0.5815 - val_accuracy: 0.7930
Epoch 8/20
438/438 [==============================] - ETA: 0s - loss: 0.6163 - accuracy: 0.7
810
Epoch 8: val_accuracy improved from 0.79301 to 0.81183, saving model to checkpoin
ts_cnn_2\weights_epoch_08.h5
438/438 [==============================] - 156s 355ms/step - loss: 0.6163 - accur
acy: 0.7810 - val_loss: 0.5401 - val_accuracy: 0.8118
Epoch 9/20
438/438 [==============================] - ETA: 0s - loss: 0.5906 - accuracy: 0.7
908
Epoch 9: val_accuracy improved from 0.81183 to 0.82124, saving model to checkpoin
ts_cnn_2\weights_epoch_09.h5
438/438 [==============================] - 156s 355ms/step - loss: 0.5906 - accur
```

```
acy: 0.7908 - val_loss: 0.5113 - val_accuracy: 0.8212
Epoch 10/20
438/438 [==============================] - ETA: 0s - loss: 0.5695 - accuracy: 0.7
970
Epoch 10: val_accuracy did not improve from 0.82124
438/438 [==============================] - 155s 354ms/step - loss: 0.5695 - accur
acy: 0.7970 - val_loss: 0.5159 - val_accuracy: 0.8172
Epoch 11/20
438/438 [==============================] - ETA: 0s - loss: 0.5509 - accuracy: 0.8
081
Epoch 11: val_accuracy did not improve from 0.82124
438/438 [==============================] - 157s 357ms/step - loss: 0.5509 - accur
acy: 0.8081 - val_loss: 0.5159 - val_accuracy: 0.8196
Epoch 12/20
438/438 [==============================] - ETA: 0s - loss: 0.5374 - accuracy: 0.8
087
Epoch 12: val_accuracy did not improve from 0.82124
438/438 [==============================] - 155s 354ms/step - loss: 0.5374 - accur
acy: 0.8087 - val_loss: 0.5692 - val_accuracy: 0.8007
Epoch 13/20
438/438 [==============================] - ETA: 0s - loss: 0.5241 - accuracy: 0.8
164
Epoch 13: val_accuracy did not improve from 0.82124
438/438 [==============================] - 156s 355ms/step - loss: 0.5241 - accur
acy: 0.8164 - val_loss: 0.5641 - val_accuracy: 0.7997
Epoch 14/20
438/438 [==============================] - ETA: 0s - loss: 0.5191 - accuracy: 0.8
137
Epoch 14: val_accuracy did not improve from 0.82124
Restoring model weights from the end of the best epoch: 9.
438/438 [==============================] - 155s 354ms/step - loss: 0.5191 - accur
acy: 0.8137 - val_loss: 0.5950 - val_accuracy: 0.8004
Epoch 14: early stopping
```

In [16]:

```python
model_3 = build_cnn(nb_conv=3)

checkpoint, early_stop, ckpt_dir = get_callbacks(3)

history_3 = model_3.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    epochs=20,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // BATCH_SIZE,
    callbacks=[checkpoint, early_stop],
    verbose=1
)
```

```
Epoch 1/20
438/438 [==============================] - ETA: 0s - loss: 1.1100 - accuracy: 0.5
604
Epoch 1: val_accuracy improved from -inf to 0.68750, saving model to checkpoints_
cnn_3\weights_epoch_01.h5
438/438 [==============================] - 173s 393ms/step - loss: 1.1100 - accur
acy: 0.5604 - val_loss: 0.8216 - val_accuracy: 0.6875
Epoch 2/20
438/438 [==============================] - ETA: 0s - loss: 0.8551 - accuracy: 0.6
808
Epoch 2: val_accuracy improved from 0.68750 to 0.72245, saving model to checkpoin
ts_cnn_3\weights_epoch_02.h5
438/438 [==============================] - 171s 389ms/step - loss: 0.8551 - accur
acy: 0.6808 - val_loss: 0.7551 - val_accuracy: 0.7224
Epoch 3/20
438/438 [==============================] - ETA: 0s - loss: 0.7460 - accuracy: 0.7
348
Epoch 3: val_accuracy improved from 0.72245 to 0.79368, saving model to checkpoin
ts_cnn_3\weights_epoch_03.h5
438/438 [==============================] - 175s 399ms/step - loss: 0.7460 - accur
acy: 0.7348 - val_loss: 0.5849 - val_accuracy: 0.7937
Epoch 4/20
438/438 [==============================] - ETA: 0s - loss: 0.6552 - accuracy: 0.7
700
Epoch 4: val_accuracy improved from 0.79368 to 0.80544, saving model to checkpoin
ts_cnn_3\weights_epoch_04.h5
438/438 [==============================] - 170s 389ms/step - loss: 0.6552 - accur
acy: 0.7700 - val_loss: 0.5199 - val_accuracy: 0.8054
Epoch 5/20
438/438 [==============================] - ETA: 0s - loss: 0.6082 - accuracy: 0.7
862
Epoch 5: val_accuracy improved from 0.80544 to 0.83367, saving model to checkpoin
ts_cnn_3\weights_epoch_05.h5
438/438 [==============================] - 170s 389ms/step - loss: 0.6082 - accur
acy: 0.7862 - val_loss: 0.4707 - val_accuracy: 0.8337
Epoch 6/20
438/438 [==============================] - ETA: 0s - loss: 0.5526 - accuracy: 0.8
070
Epoch 6: val_accuracy improved from 0.83367 to 0.84845, saving model to checkpoin
ts_cnn_3\weights_epoch_06.h5
438/438 [==============================] - 169s 386ms/step - loss: 0.5526 - accur
acy: 0.8070 - val_loss: 0.4491 - val_accuracy: 0.8485
Epoch 7/20
438/438 [==============================] - ETA: 0s - loss: 0.5237 - accuracy: 0.8
144
Epoch 7: val_accuracy did not improve from 0.84845
438/438 [==============================] - 169s 386ms/step - loss: 0.5237 - accur
acy: 0.8144 - val_loss: 0.5078 - val_accuracy: 0.8212
Epoch 8/20
438/438 [==============================] - ETA: 0s - loss: 0.4971 - accuracy: 0.8
238
Epoch 8: val_accuracy did not improve from 0.84845
438/438 [==============================] - 169s 386ms/step - loss: 0.4971 - accur
acy: 0.8238 - val_loss: 0.5472 - val_accuracy: 0.8243
Epoch 9/20
438/438 [==============================] - ETA: 0s - loss: 0.4860 - accuracy: 0.8
326
Epoch 9: val_accuracy did not improve from 0.84845
438/438 [==============================] - 169s 387ms/step - loss: 0.4860 - accur
acy: 0.8326 - val_loss: 0.4484 - val_accuracy: 0.8441
```

```
Epoch 10/20
438/438 [==============================] - ETA: 0s - loss: 0.4667 - accuracy: 0.8
398
Epoch 10: val_accuracy did not improve from 0.84845
438/438 [==============================] - 169s 385ms/step - loss: 0.4667 - accur
acy: 0.8398 - val_loss: 0.4371 - val_accuracy: 0.8458
Epoch 11/20
438/438 [==============================] - ETA: 0s - loss: 0.4522 - accuracy: 0.8
425
Epoch 11: val_accuracy did not improve from 0.84845
Restoring model weights from the end of the best epoch: 6.
438/438 [==============================] - 169s 386ms/step - loss: 0.4522 - accur
acy: 0.8425 - val_loss: 0.4533 - val_accuracy: 0.8451
Epoch 11: early stopping
```

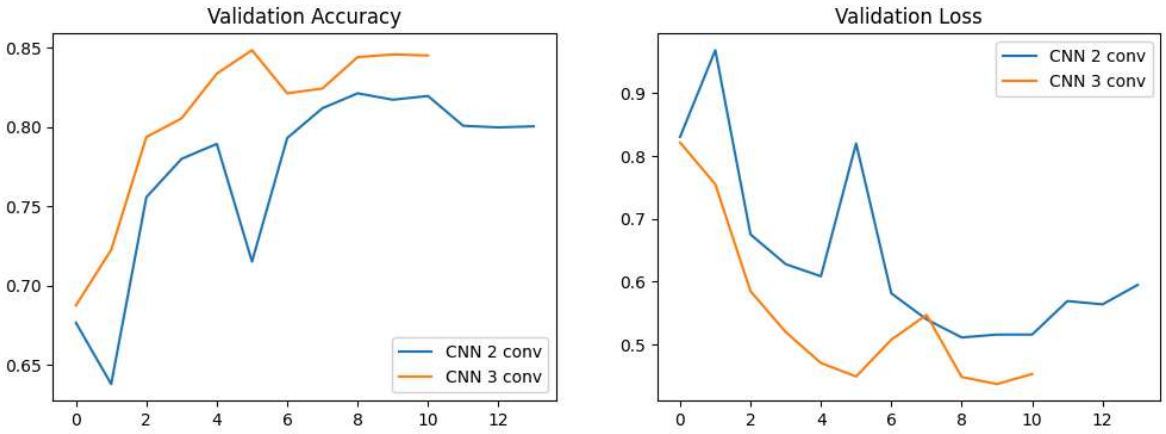
In [17]:
```python
import matplotlib.pyplot as plt

plt.figure(figsize=(12,4))

plt.subplot(1,2,1)
plt.plot(history_2.history["val_accuracy"], label="CNN 2 conv")
plt.plot(history_3.history["val_accuracy"], label="CNN 3 conv")
plt.title("Validation Accuracy")
plt.legend()

plt.subplot(1,2,2)
plt.plot(history_2.history["val_loss"], label="CNN 2 conv")
plt.plot(history_3.history["val_loss"], label="CNN 3 conv")
plt.title("Validation Loss")
plt.legend()

plt.show()
```



Comparaison CNN 2 convolutions vs CNN 3 convolutions

L'entraînement des deux architectures montre que l'ajout d'une troisième couche de convolution améliore significativement les performances sur le dataset Intel Image Classification.

- CNN 2 convolutions : atteint une précision maximale sur l'ensemble de validation d'environ 82,1 %, avec une perte de validation autour de 0,51.
- CNN 3 convolutions : atteint une précision maximale sur l'ensemble de validation d'environ 84,8 %, avec une perte de validation plus faible (~0,45).

Le graphique comparatif montre également que le modèle à 3 convolutions converge plus rapidement et généralise mieux, tout en maintenant un overfitting limité grâce à l'early stopping et au Dropout.

Donc avec de la troisième couche de convolution permet au CNN de capturer des caractéristiques plus complexes des images, ce qui améliore la précision globale et la stabilité du modèle. Pour ce dataset multiclasse, le CNN à 3 convolutions est donc préférable.