

Model1_cnn

November 17, 2025

```
[85]: import sys
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
```

```
[86]: import tensorflow as tf
print(tf.__version__)
```

2.20.0

```
[87]: import pickle
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from keras import layers, models
path_to_data = 'ift-3395-6390-kaggle-2-competition-fall-2025/train_data.pkl'

# --- Load data ---
with open(path_to_data, "rb") as f:
    train_data = pickle.load(f)

X = train_data["images"]
y = train_data["labels"].reshape(-1)

# --- Preprocessing ---
X = X.astype("float32") / 255.0

# --- Split ---
X_train, X_val, y_train, y_val = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

class_counts = np.bincount(y_train)
total = len(y_train)
```

```

class_weight = {i: total/(5*class_counts[i]) for i in range(5)}

data_augmentation = models.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.05),
    layers.RandomZoom(0.1),
    layers.RandomTranslation(0.1, 0.1),
    layers.RandomContrast(0.1),
    layers.RandomBrightness(0.1)
])

# --- CNN definition ---
model = models.Sequential([
    data_augmentation,
    layers.Conv2D(32, (3,3), activation="relu", padding="same"),
    layers.BatchNormalization(),
    layers.Conv2D(32, (3,3), activation="relu"),
    layers.MaxPooling2D(),

    layers.Conv2D(64, (3,3), activation="relu", padding="same"),
    layers.BatchNormalization(),
    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D(),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(0.3),
    layers.Dense(5, activation="softmax")
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
from keras.callbacks import EarlyStopping

early_stop = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)
# --- Train ---
history = model.fit(
    X_train, y_train,
    class_weight=class_weight,

```

```

    epochs=100,
    batch_size=32,
    validation_data=(X_val, y_val),
    callbacks=[early_stop],
    shuffle=True
)

# --- Save CNN ---
model.save("cnn_model.keras")

```

```

Epoch 1/100
27/27          1s 17ms/step -
accuracy: 0.1782 - loss: 2.0157 - val_accuracy: 0.0602 - val_loss: 1.6147
Epoch 2/100
27/27          0s 16ms/step -
accuracy: 0.1713 - loss: 1.6608 - val_accuracy: 0.1204 - val_loss: 1.6092
Epoch 3/100
27/27          0s 15ms/step -
accuracy: 0.1921 - loss: 1.6330 - val_accuracy: 0.0602 - val_loss: 1.6203
Epoch 4/100
27/27          0s 14ms/step -
accuracy: 0.2037 - loss: 1.6227 - val_accuracy: 0.2407 - val_loss: 1.6019
Epoch 5/100
27/27          0s 13ms/step -
accuracy: 0.3287 - loss: 1.6160 - val_accuracy: 0.4491 - val_loss: 1.5929
Epoch 6/100
27/27          0s 13ms/step -
accuracy: 0.2153 - loss: 1.6158 - val_accuracy: 0.1898 - val_loss: 1.6032
Epoch 7/100
27/27          0s 13ms/step -
accuracy: 0.1343 - loss: 1.6186 - val_accuracy: 0.1806 - val_loss: 1.6024
Epoch 8/100
27/27          0s 13ms/step -
accuracy: 0.1481 - loss: 1.6156 - val_accuracy: 0.1806 - val_loss: 1.6106
Epoch 9/100
27/27          0s 13ms/step -
accuracy: 0.2500 - loss: 1.6112 - val_accuracy: 0.0602 - val_loss: 1.6125
Epoch 10/100
27/27          0s 15ms/step -
accuracy: 0.0961 - loss: 1.6103 - val_accuracy: 0.1806 - val_loss: 1.6127

```

```
[88]: import matplotlib.pyplot as plt
```

```

# Accuracy
plt.figure(figsize=(8,5))
plt.plot(history.history["accuracy"], label="Train accuracy")
plt.plot(history.history["val_accuracy"], label="Validation accuracy")

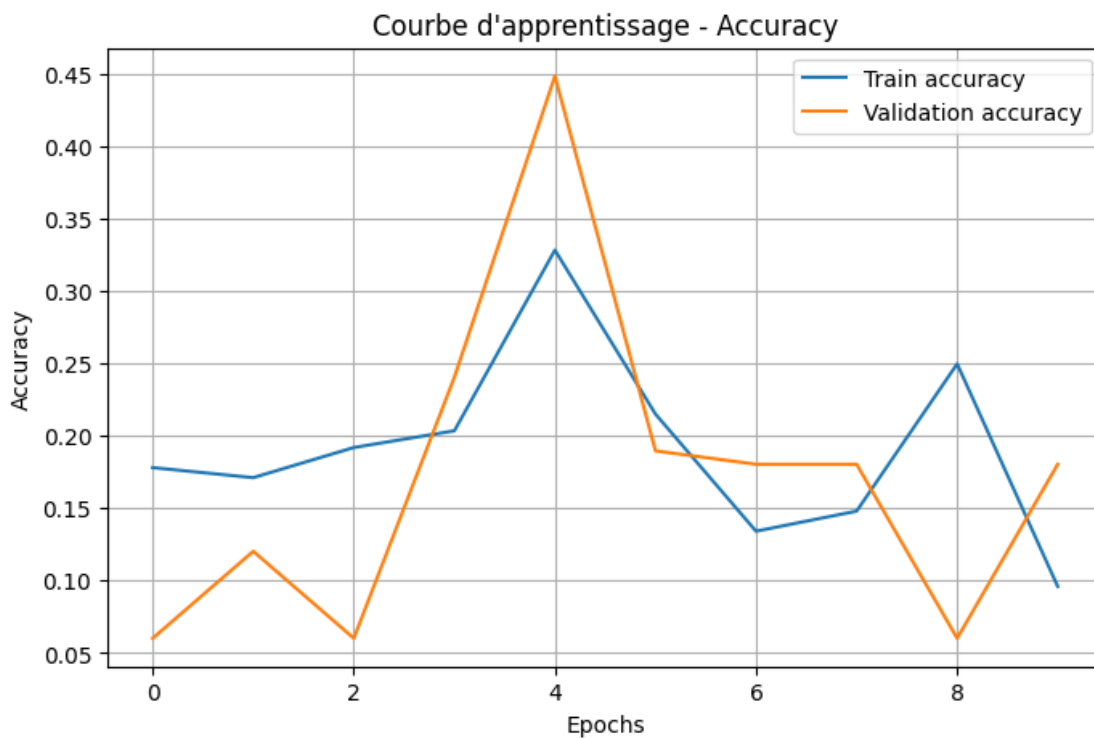
```

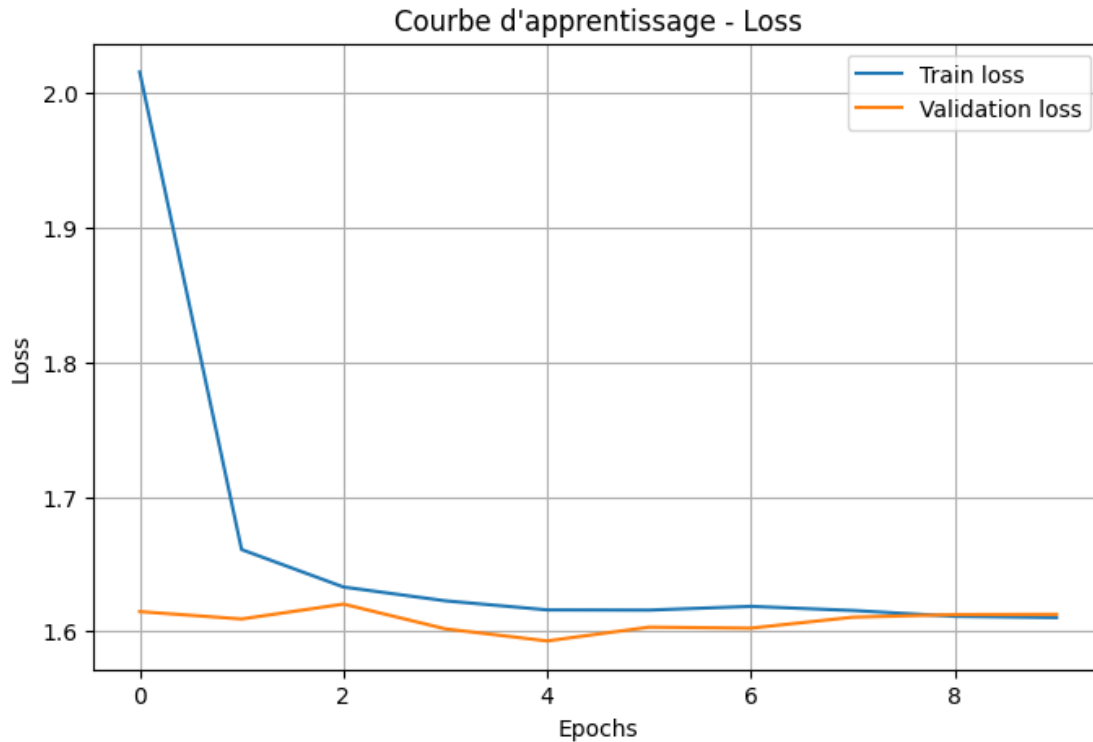
```

plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Courbe d'apprentissage - Accuracy")
plt.legend()
plt.grid(True)
plt.show()

# Loss
plt.figure(figsize=(8,5))
plt.plot(history.history["loss"], label="Train loss")
plt.plot(history.history["val_loss"], label="Validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Courbe d'apprentissage - Loss")
plt.legend()
plt.grid(True)
plt.show()

```





```
[89]: y_val_pred = model.predict(X_val)
y_val_pred_classes = y_val_pred.argmax(axis=1)
from sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns

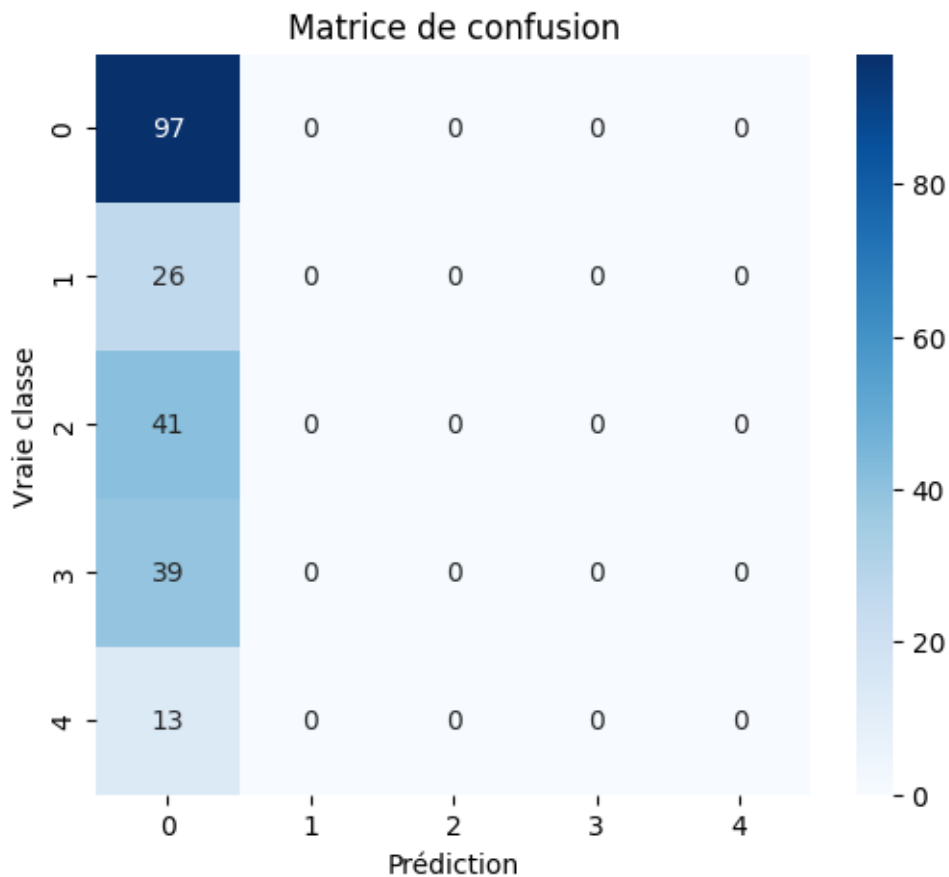
cm = confusion_matrix(y_val, y_val_pred_classes)

plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
plt.title("Matrice de confusion")
plt.xlabel("Prédiction")
plt.ylabel("Vraie classe")
plt.show()
from sklearn.metrics import recall_score

recall_per_class = recall_score(y_val, y_val_pred_classes, average=None)
recall_macro = recall_score(y_val, y_val_pred_classes, average="macro")
acc = accuracy_score(y_val, y_val_pred_classes)
print("Recall par classe :", recall_per_class)
print('acc:', acc)
print("Recall macro :", recall_macro)
```

7/7

0s 10ms/step



Recall par classe : [1. 0. 0. 0. 0.]
 acc: 0.44907407407407407
 Recall macro : 0.2

```
[90]: import pickle
import numpy as np
import pandas as pd
"""
# -----
# 1. Charger le modèle entraîné
# -----
model, scaler = pickle.load(open("model_softmax.pkl", "rb"))

# -----
# 2. Charger le test_data.pkl
# -----
with open("ift-3395-6390-kaggle-2-competition-fall-2025/test_data.pkl", "rb") as f:
    test_data = pickle.load(f)
```

```

X_test_imgs = test_data["images"]

# Apply to test set
X_test_feats = np.array([extract_features(img) for img in X_test_imgs],
    dtype=np.float32)

# -----
# 4. Normaliser avec les stats du train
# -----
X_test_norm = scaler.transform(X_test_feats)

# -----
# 5. Prédire
# -----
y_pred = model.compute_output(X_test_norm).astype(int)

# -----
# 6. Générer le CSV Kaggle
# -----
df = pd.DataFrame({
    "ID": np.arange(1, len(y_pred)+1),
    "Label": y_pred
})

df.to_csv("ift3395_YamirPoldoSilvaV7.csv", index=False)

print("Fichier 'submission.csv' généré !")
"""

```

```

[90]: '\n# ----- \n# 1. Charger le modèle entraîné\n#
----- \nmodel, scaler =
pickle.load(open("model_softmax.pkl", "rb"))\n\n# ----- \n#
2. Charger le test_data.pkl\n# ----- \nwith
open("ift-3395-6390-kaggle-2-competition-fall-2025/test_data.pkl", "rb") as f:\n
test_data = pickle.load(f)\n\nX_test_imgs = test_data["images"]\n\n\n# Apply to
test set\nX_test_feats = np.array([extract_features(img) for img in
X_test_imgs], dtype=np.float32)\n\n# ----- \n# 4. Normaliser
avec les stats du train\n# ----- \nX_test_norm =
scaler.transform(X_test_feats)\n\n# ----- \n# 5. Prédire\n#
----- \ny_pred =
model.compute_output(X_test_norm).astype(int)\n\n# ----- \n#
6. Générer le CSV Kaggle\n# ----- \nndf = pd.DataFrame({\n
"ID": np.arange(1, len(y_pred)+1),\n    "Label":
y_pred\n})\n\nndf.to_csv("ift3395_YamirPoldoSilvaV7.csv",
index=False)\n\nprint("Fichier \'submission.csv\' généré !")\n'

```