# perceptron_to_debug

December 9, 2025

[ ]:

```
import sys
import numpy as np
import pickle
np.random.seed(0)
from scripts import (
    #train_test_split,
    StandardScaler,
    accuracy,
    confusion_matrix,
    recall_per_class,
    balanced_accuracy,
    KernelPerceptron,
    SVM,
    rbf_kernel,
)
import pandas as pd
```

[49]:

```
from sklearn.model_selection import train_test_split
# --- Load training data ---
path_to_data = 'ift-3395-6390-kaggle-2-competition-fall-2025/train_data.pkl'
with open(path_to_data, "rb") as f:
    train_data = pickle.load(f)

X_imgs = train_data["images"].astype(np.float32)
y = train_data["labels"].reshape(-1)

X_train_imgs, X_val_imgs, y_train, y_val = train_test_split(
    X_imgs, y, test_size=0.2, random_state=0, stratify=y
)

train_min = X_train_imgs.min()
train_max = X_train_imgs.max()

train_mean = X_train_imgs.mean()

train_std = X_train_imgs.std()
X_train_imgs = (X_train_imgs - train_min) / (train_max - train_min + 1e-6)
X_train_imgs = (X_train_imgs - train_mean) / (train_std + 1e-6)
```

```
X_val_imgs = (X_val_imgs - train_min) / (train_max - train_min + 1e-6)
X_val_imgs = (X_val_imgs - train_mean) / (train_std + 1e-6)
```

[50]:
```
def extract_simple_stats(img):
    gray = img.mean(axis=2)

    return np.array([gray.mean(), gray.std(), gray.min(), gray.max()], dtype=np.
 ↪float32)
```

[51]:
```
def radial_profile(img):
    """Calcule le profil radial moyen d'une image."""
    h, w = img.shape
    y, x = np.ogrid[:h, :w]
    r = np.sqrt((x - w//2)**2 + (y - h//2)**2).astype(int)
    profile = np.bincount(r.ravel(), img.ravel()) / np.bincount(r.ravel())
    return np.log(profile + 1e-6)
```

[52]:
```
def fft_features(images):
    """Extrait les caractéristiques FFT d'images."""
    gray = images.mean(axis=3)
    F = np.fft.fft2(gray, axes=(1, 2))
    return np.abs(np.fft.fftshift(F, axes=(1, 2)))
```

[53]:
```
def extract_channel_stats(img):
    # img.shape est (H, W, 3)
    features = []
    for c in range(img.shape[2]): # Itere sur les 3 canaux
        channel = img[:, :, c]
    features.extend([channel.mean(), channel.std(), channel.min(), channel.
 ↪max()])

    return np.array(features, dtype=np.float32)
```

[54]:
```
def simple_augment(images, labels):
    flips = images[:, :, ::-1, :]
    noise = images + 0.01*np.random.randn(*images.shape)
    aug_imgs = np.concatenate([images, flips, noise], axis=0)
    aug_labels = np.concatenate([labels, labels, labels])
    return aug_imgs, aug_labels
```

[55]:
```
X_train_imgs, y_train = simple_augment(X_train_imgs, y_train)

fft_mag_train = fft_features(X_train_imgs)

X_fft_train = np.array([radial_profile(img) for img in fft_mag_train], dtype=np.
 ↪float32)
```

```
X_stats_train = np.array([extract_simple_stats(img) for img in X_train_imgs],␣
 ↪dtype=np.float32)
X_color_train = np.array([extract_channel_stats(img) for img in␣
 ↪X_train_imgs],dtype=np.float32)

X_train = np.hstack([X_fft_train,X_stats_train, X_color_train])

fft_mag_val = fft_features(X_val_imgs)

X_fft_val = np.array([radial_profile(img) for img in fft_mag_val], dtype=np.
 ↪float32)
X_stats_val = np.array([extract_simple_stats(img) for img in␣
 ↪X_val_imgs],dtype=np.float32)
X_color_val = np.array([extract_channel_stats(img) for img in X_val_imgs],␣
 ↪dtype=np.float32)

X_val = np.hstack([X_fft_val,X_stats_val, X_color_val])

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
```

[56]:

```
class_counts = np.bincount(y_val)
class_weights = (1.0 / class_counts)
class_weights /= class_weights.sum()
sample_weights = class_weights[y_train]
```

[57]:

```
model = KernelPerceptron(kernel_fn=rbf_kernel, n_classes=5, sigma=1,␣
 ↪learning_rate=1.0,  sample_weights=None, lam=0.0)
model.fit(X_train, y_train, max_epochs=50)
```

[58]:

```
y_pred_val = model.predict(X_val)
acc = (y_pred_val == y_val).mean()
print("Test accuracy =", acc)
cm = confusion_matrix(y_val, y_pred_val)
bal_acc = balanced_accuracy(y_val, y_pred_val)
rec = recall_per_class(cm)
print("Balanced acc :", bal_acc)
print("Recall par classe :", rec)
print("Recall moyen :", rec.mean())
print(cm)
```

```
Test accuracy = 0.44907407407407407
Balanced acc : 0.3190054350979672
Recall par classe : [0.70103093 0.19230769 0.31707317 0.23076923 0.15384615]
Recall moyen : 0.3190054350979672
[[68  8 11  6  4]
```

```
 [10  5  5  6  0]
 [11  5 13  9  3]
 [14  6  7  9  3]
 [ 6  0  3  2  2]]
```

[59]:

```python
model_pkg = {
'model': model,
'scaler': scaler,
'train_min': train_min,
'train_max':train_max,
'train_mean': train_mean,
'train_std': train_std
}

pickle.dump(model_pkg, open("model_perceptron.pkl","wb"))
```

[60]:

```python
model_pkg_pred = pickle.load(open("model_perceptron.pkl", "rb"))

model= model_pkg_pred['model']
scaler = model_pkg_pred['scaler']
train_min = model_pkg_pred['train_min']
train_max = model_pkg_pred['train_max']
train_mean = model_pkg_pred['train_mean']
train_std = model_pkg_pred['train_std']

# --------------------------
# 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"].astype(np.float32)
X_test_imgs = (X_test_imgs - train_min) / (train_max - train_min + 1e-6)
X_test_imgs = (X_test_imgs - train_mean) / (train_std + 1e-6)


fft_mag_test = fft_features(X_test_imgs)
X_fft_test = np.array([radial_profile(img) for img in fft_mag_test], dtype=np.
 ↪float32)
X_stats_test = np.array([extract_simple_stats(img) for img in␣
 ↪X_test_imgs],dtype=np.float32)
X_color_test = np.array([extract_channel_stats(img) for img in␣
 ↪X_test_imgs],dtype=np.float32)
X_test = np.hstack([X_fft_test, X_stats_test, X_color_test])
X_test = scaler.transform(X_test)
```

```python
y_pred = model.predict(X_test).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_YAPS_MCS_V101.csv", index=False)

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

#print(df.head())

df1 = pd.read_csv("ift3395_YAPS_MCS_V101.csv")
df2 = pd.read_csv("IFT3395_YAPS_MCS_V14_ref.csv")

comparison = df1.compare(df2)
print(comparison)
print("Nombre de différences :", len(comparison))
```

```
Fichier 'submission.csv' généré !
    Label
     self other
11    4.0   0.0
14    1.0   0.0
17    2.0   3.0
29    3.0   4.0
31    3.0   2.0
..    …     …
379   3.0   2.0
384   1.0   0.0
390   2.0   0.0
391   0.0   3.0
394   1.0   0.0

[90 rows x 2 columns]
Nombre de différences : 90
```