

Données publiques

- Procédure d'entraînement de l'algorithme **Reptile** sur données publiques

Définition des modèles de base

```
""  
# 2 Modèle de base  
# =====  
def create_mlp(output_dim=None):  
    if output_dim is None:  
        output_dim = input_dim  
    return models.Sequential([  
        layers.Input(shape=(input_dim,)),  
        layers.Dense(64, activation='relu'),  
        layers.Dense(32, activation='relu'),  
        layers.Dense(output_dim, activation='linear')  
    ])
```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-1 : entraînement de l'algorithme **Reptile** sur données publiques*

- On entraîne l'algorithme **Reptile** sur données publiques

```
# 3 Reptile
# =====
def sample_task(X, k_support=10, k_query=20):
    idx = np.random.permutation(len(X))
    return X[idx[:k_support]], X[idx[k_support:k_support+k_query]]

def reptile_train(X_train, meta_epochs=5, inner_steps=3, k_support=10, k_query=20, meta_lr=0.1):
    meta_model = create_mlp()
    for epoch in range(meta_epochs):
        X_supp, X_q = sample_task(X_train, k_support, k_query)
        X_supp_tf, X_q_tf = tf.convert_to_tensor(X_supp, dtype=tf.float32), tf.convert_to_tensor(X_q, dtype=tf.float32)

        task_model = create_mlp()
        task_model.set_weights(meta_model.get_weights())
        inner_opt = optimizers.SGD(0.01)

        for _ in range(inner_steps):
            with tf.GradientTape() as tape:
                loss = tf.reduce_mean(tf.square(task_model(X_supp_tf) - X_supp_tf))
            grads = tape.gradient(loss, task_model.trainable_variables)
            inner_opt.apply_gradients(zip(grads, task_model.trainable_variables))

        meta_weights = meta_model.get_weights()
        task_weights = task_model.get_weights()
        new_weights = [mw + meta_lr*(tw - mw) for mw, tw in zip(meta_weights, task_weights)]
        meta_model.set_weights(new_weights)

        q_loss = tf.reduce_mean(tf.square(task_model(X_q_tf) - X_q_tf)).numpy()
        print(f"[Reptile Epoch {epoch+1}] Query Loss = {q_loss:.4f}")
    return meta_model
```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-2 : entraînement de l'algorithme **Reptile** sur données publiques*

```
# =====
# 4 Évaluation
# =====
def eval_model(model, X_train, X_test, y_test):
    mse_test = np.mean(np.square(X_test - model.predict(X_test)), axis=1)
    threshold = np.percentile(np.mean(np.square(X_train - model.predict(X_train)), axis=1), 95)
    y_pred = (mse_test > threshold).astype(int)
    return f1_score(y_test, y_pred, average='macro'), recall_score(y_test, y_pred, average='macro')

# =====
# 5 Exécution
# =====
print("=== Entraînement Reptile ===")
reptile_model = reptile_train(X_train)
print("=== Évaluation finale Reptile ===")
f1, rec = eval_model(reptile_model, X_train, X_test, y_test)
print(f"Reptile -> F1: {f1:.4f}, Recall: {rec:.4f}")
```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-3 : entraînement de l'algorithme **Reptile** sur données publiques*

- Voici le résultat

Reptile -> F1: 0.9744, Recall: 0.9744

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-4 : résultat de l'algorithme reptile sur données publiques

Données privées

- Même procédure pour les données privées

```
# -----  
# Autoencoder  
def build_autoencoder(input_dim, encoding_dim):  
    inputs = layers.Input(shape=(input_dim,))  
    e = layers.Dense(encoding_dim*2, activation='relu')(inputs)  
    e = layers.Dense(encoding_dim, activation='relu')(e)  
    bottleneck = layers.Dense(max(1, encoding_dim//2), activation='relu')(e)  
    d = layers.Dense(encoding_dim, activation='relu')(bottleneck)  
    d = layers.Dense(encoding_dim*2, activation='relu')(d)  
    outputs = layers.Dense(input_dim, activation='linear')(d)  
    return models.Model(inputs, outputs)
```

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-5 : entraînement de l'algorithme Reptile sur données privées

```

# -----
# Reptile class
class Reptile:
    def __init__(self, input_dim, encoding_dim, lr=1e-3):
        self.model = build_autoencoder(input_dim, encoding_dim)
        self.opt = tf.keras.optimizers.Adam(lr)

    def fit(self, X_train, support, epochs=5, inner_steps=1):
        X_train_tf = tf.convert_to_tensor(X_train, dtype=tf.float32)
        support_tf = tf.convert_to_tensor(support, dtype=tf.float32)
        for epoch in range(epochs):
            old_weights = self.model.get_weights()
            for _ in range(inner_steps):
                with tf.GradientTape() as tape:
                    recon = self.model(support_tf, training=True)
                    loss = tf.reduce_mean(tf.square(recon - support_tf))
                    grads = tape.gradient(loss, self.model.trainable_variables)
                    self.opt.apply_gradients(zip(grads, self.model.trainable_variables))
            # interpolation poids
            new_weights = self.model.get_weights()
            for i in range(len(new_weights)):
                new_weights[i] = old_weights[i] + 0.1*(new_weights[i]-old_weights[i])
            self.model.set_weights(new_weights)
            print(f"[Reptile] Epoch {epoch+1}/{epochs} done")

    def predict(self, X):
        return self.model.predict(X)

```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-6 : entraînement de l'algorithme **Reptile** sur données privées*

```

# Support set
support_idx = np.random.choice(X_test_with_anom.shape[0], size=20, replace=False)
X_support = X_test_with_anom[support_idx]

# -----
# Entraînement & évaluation
reptile = Reptile(input_dim, encoding_dim)
reptile.fit(X_train, support=X_support, epochs=8, inner_steps=2)

recon_reptile = reptile.predict(X_test_with_anom)
mse_test = np.mean(np.square(X_test_with_anom - recon_reptile), axis=1)
threshold = np.percentile(np.mean(np.square(X_train - reptile.predict(X_train))), axis=1, 95)
y_pred = (mse_test > threshold).astype(int)

print("=== Résultats Reptile ===")
print(f"F1: {f1_score(y_test, y_pred):.4f}, Recall: {recall_score(y_test, y_pred):.4f}")

```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-7 : suite entraînement de l'algorithme **Reptile** sur données privées*

Reptile': (0.9622641509433962, 1.0).

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-8 : résultat de l'algorithme reptile sur données privées