Sistemas Inmunes Artificiales y Modelos de Redes Neuronales Profundas para la Detección de Anomalías

Adán González Rodríguez
Sara Porto Álvarez

Dataset Utilizado

Hemos utilizado el Dataset: NAB, concretamente el **./archive/artificialWithAnomaly/artificialWithAnomaly/art_daily_flatmiddle.csv**

artificialWithAnomaly contiene datos generados artificialmente con diversos tipos de anomalías, en este caso se trata de un timeseries con una meseta. Lo usaremos para tratar de detectar anomalías.

Algoritmo de Selección Clonal (CSA) con ventana temporal

Código: Link al código de CSA

- **Imports** de las librerías necesarias
- Método **generate_single_timeseries_with_anomalies**
- Método **load_timeseries_from_csv**
- Método **plot_timeseries_with_windows**
- Clase **ClonalSelectionAIS**
  - Método **__init__**
  - Método **_init_population**
  - Método **_affinity**
  - Método **_evaluate_pop**
  - Método **_clone_and_mutate**
  - Método **fit**
  - Método **_compute_threshold**
  - Método **predict**
  - Método **_plot_iteration**
  - Método **main1**
  - Método **main2**

Algoritmo de Selección Negativa (NSA) con ventana temporal

Código: Link al código de NSA

- **Imports** de las librerías necesarias
- Método **generate_single_timeseries_with_anomalies**
- Método **load_timeseries_from_csv**
- Método **plot_timeseries_with_windows**
- Clase **NegativeSelectionVectors**
  - Método **__init__**
  - Método **_init_detectors**
  - Método **_filter_self_reactive**
  - Método **fit**
  - Método **predict**
  - Método **_plot_generation**
  - Método **main1**
  - Método **main2**

Red Neuronal Profunda (DNN) (o con Autoencoder en Keras)

Código: Link al código de DNN

- **Imports** de las librerías necesarias
- Método **generate_single_timeseries_with_anomalies**
- Método **load_timeseries_from_csv**
- Método **plot_timeseries_with_windows**
- Clase **DNNAnomalyDetector**
  - Método **__init__**
  - Método **forward**
  - Método **fit**
  - Método **evaluate_accuracy**
  - Método **predict**
  - Método **main1**
  - Método **main2**

Problema con los resultados con el dataset

Resultados → Demás pruebas con dataset sintético

DNN:

```
Confusion Matrix (Test):
[[55  5]
 [ 0  0]]

Classification Report (Test):
              precision    recall  f1-score   support

      Normal       1.00      0.92      0.96        60
     Anomaly       0.00      0.00      0.00         0

    accuracy                           0.92        60
   macro avg       0.50      0.46      0.48        60
weighted avg       1.00      0.92      0.96        60
```

CAS:

```
[[61]]

Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00        61
     Anomaly       0.00      0.00      0.00         0

    accuracy                           1.00        61
   macro avg       0.50      0.50      0.50        61
weighted avg       1.00      1.00      1.00        61
```

NSA:

```
[[61]]

Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00        61
     Anomaly       0.00      0.00      0.00         0

    accuracy                           1.00        61
   macro avg       0.50      0.50      0.50        61
weighted avg       1.00      1.00      1.00        61
```

Comparación entre algoritmos (con el dataset sintético)

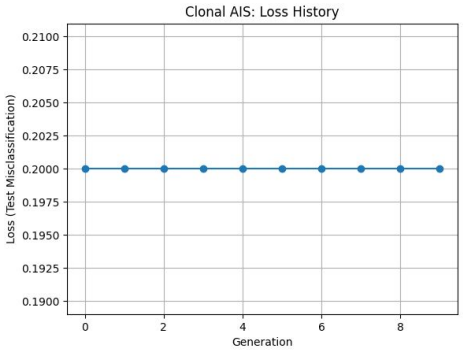
```python
def generate_single_timeseries_with_anomalies(
    n_points=400,
    anomaly_intervals=[(100, 120), (250, 270)],
    window_size=20,
    step=20,
    random_seed=42
):
    np.random.seed(random_seed)

    # 1) Build base normal wave
    t_axis = np.linspace(0, 4*np.pi, n_points)
    base_amp = 1.0
    wave = base_amp * np.sin(t_axis)
    noise = 0.1 * np.random.randn(n_points)
    T = wave + noise

    # 2) Insert anomalies
    for (start_idx, end_idx) in anomaly_intervals:
        # triple amplitude + bigger noise
        T[start_idx:end_idx] = 3.0 * base_amp * np.sin(t_axis[start_idx:end_idx])
        T[start_idx:end_idx] += 0.3 * np.random.randn(end_idx - start_idx)

    # 3) Slice into windows
    window_starts = range(0, n_points - window_size + 1, step)
    X, y = [], []
    for ws in window_starts:
        we = ws + window_size
        window_data = T[ws:we]
        # label=1 if overlaps any anomaly interval
        label = 0
        for (a_start, a_end) in anomaly_intervals:
            if not (we <= a_start or ws >= a_end):
                label = 1
                break
        X.append(window_data)
        y.append(label)

    X = np.array(X)
    y = np.array(y, dtype=int)
    return T, X, y, list(window_starts)
```

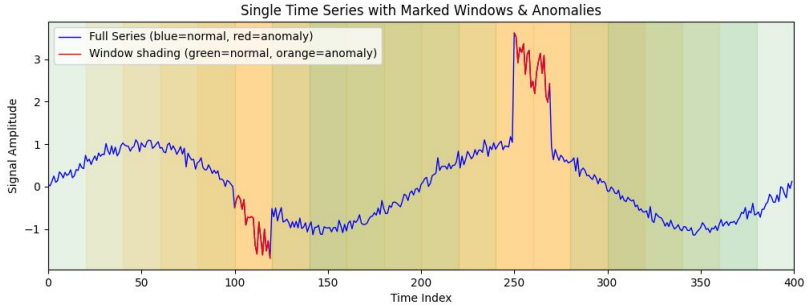
Métricas de evaluación utilizadas

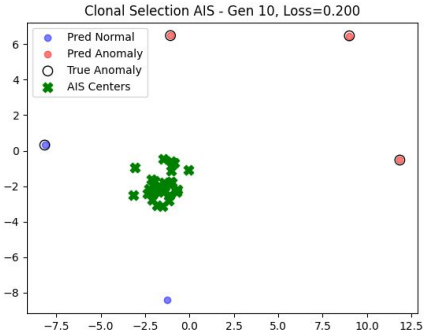
- **Matriz de confusión**
  - Cómo se comportó el modelo al clasificar las instancias
  - TP, FN, FP, TN

- **Reporte de clasificación**
  - ***Precision***: De todas las veces que el modelo predijo una clase (Normal o Anomaly), ¿cuántas fueron correctas?
  - ***Recall***: De todas las veces que realmente había una clase (Normal o Anomaly), ¿cuántas veces el modelo la detectó correctamente?
  - ***F1-Score***: Promedio equilibrado entre Precisión y Recall (útil cuando las clases están desbalanceadas)
  - ***Support***: Número de ejemplos reales en cada clase
  - ***Accuracy***: Proporción total de predicciones correctas sobre todas las muestras.

Evaluación CAS: Link a más pruebas con CAS

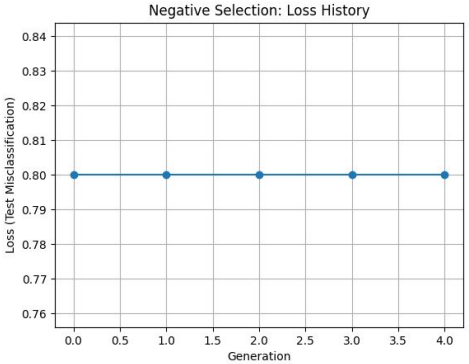


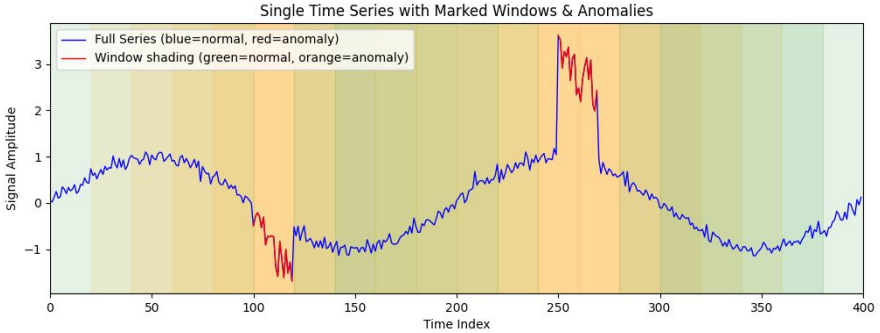


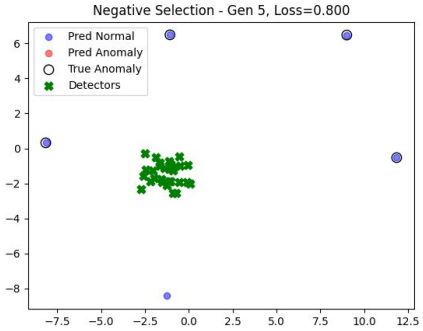



```
Confusion Matrix (Test):
[[1 0]
 [1 3]]

Classification Report:
              precision    recall  f1-score   support

      Normal       0.50      1.00      0.67         1
     Anomaly       1.00      0.75      0.86         4

    accuracy                           0.80         5
   macro avg       0.75      0.88      0.76         5
weighted avg       0.90      0.80      0.82         5
```

Evaluación NSA: Link a más pruebas con NSA

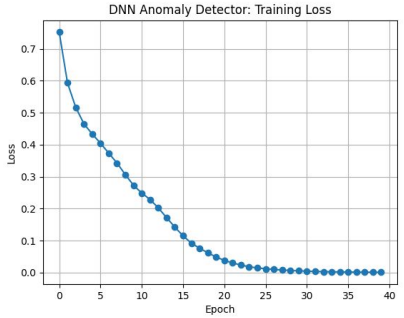


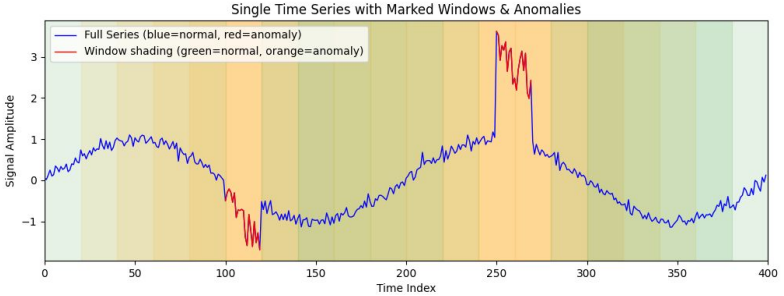


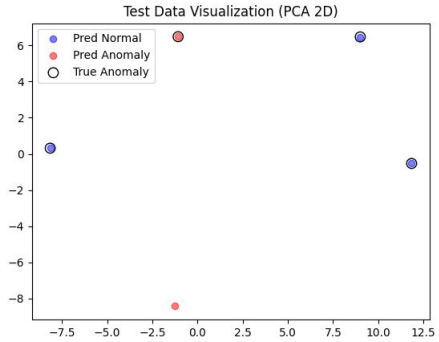



```
Confusion Matrix (Test):
[[1 0]
 [4 0]]

Classification Report:
              precision    recall  f1-score   support

      Normal       0.20      1.00      0.33         1
     Anomaly       0.00      0.00      0.00         4

    accuracy                           0.20         5
   macro avg       0.10      0.50      0.17         5
weighted avg       0.04      0.20      0.07         5
```

Evaluación DNN: Link a más pruebas con DNN








```
Epoch 40/40, Train Loss: 0.0012, Val Acc: 0.2000

Confusion Matrix (Test):
[[0 1]
 [3 1]]

Classification Report (Test):
              precision    recall  f1-score   support

      Normal       0.00      0.00      0.00         1
     Anomaly       0.50      0.25      0.33         4

    accuracy                           0.20         5
   macro avg       0.25      0.12      0.17         5
weighted avg       0.40      0.20      0.27         5
```

Comparativa final

Limitaciones, problemas de escalabilidad y ventajas

	Principal Limitación	**Principal Ventaja**	**Escalabilidad**
CSA	Sensible a la parametrización	Capacidad de adaptación y mejora con el tiempo	**Limitada** (alto costo computacional con más datos)
NSA	Alta tasa de falsos positivos si no se ajusta bien	No requiere datos etiquetados para entrenar	**Limitada** (crecimiento exponencial de detectores)
DNN	Necesita grandes volúmenes de datos y poder computacional	Captura patrones complejos y tiene alta precisión	**Alta** (escalable con paralelización)

Fin