Curso : Computación Evolutiva

Docente : Yván Jesús Túpac Valdivia

Maestría en Inteligencia Artificial - Universidad Nacional de Ingeniería

Optimización Neuro-Evolutiva sobre Embeddings de Lenguaje para la Clasificación de Incidentes Críticos en Escenarios de Alto Desbalance de Datos

Acevedo Martinez, Ruddy Enrique[1]; Benitez Altamirano, Bernie[2]; Morales Ccasa, Geyson[3]; Ramirez Ucañay, Barbarita[4]; Siu Siu Ting, Aldo[5]

Resumen

La clasificación automática de reportes de seguridad ciudadana enfrenta un desafío crítico inherente a la naturaleza de los datos reales: el severo desbalance de clases, donde incidentes de alto riesgo vital (e.g., violencia familiar) constituyen una minoría estadística frente a reportes administrativos rutinarios. Este estudio propone una arquitectura híbrida que integra Modelos de Lenguaje Pre-entrenados (BERT) para la extracción de características semánticas profundas y Algoritmos Genéticos (GA) para la Búsqueda de Arquitectura Neuronal (NAS) e Optimización de Hiperparámetros. El objetivo principal es optimizar tanto la topología como los parámetros de entrenamiento de un clasificador capaz de maximizar la recuperación (Recall) de eventos críticos sin comprometer excesivamente la generalización.

A través de un diseño experimental evolutivo iterativo utilizando datos reales del Callao, Perú, se contrastan empíricamente dos estrategias fundamentales de manejo de desbalance: el sobremuestreo sintético (SMOTE) y el Aprendizaje Sensible al Costo (Weighted Loss). Los resultados demuestran que, en espacios semánticos de alta dimensionalidad, las técnicas de generación sintética inducen al sobreajuste catastrófico en datos reales no vistos, mientras que la estrategia de costos ponderados, optimizada mediante neuro-evolución, logra cuadruplicar la tasa de recuperación de eventos críticos. El sistema resultante alcanza una Exactitud del 71 % y un F1-Score Ponderado de 0.72, validando la eficacia de los métodos evolutivos para adaptar modelos de aprendizaje profundo a dominios de seguridad pública con distribuciones de cola larga.

Keywords: *Computación Evolutiva, Búsqueda de Arquitectura Neuronal (NAS), Procesamiento de Lenguaje Natural, Desbalance de Datos, BERT, Seguridad Ciudadana, Algoritmos Genéticos, Weighted Loss*

1. Introducción y Definición del Problema

1.1. Contexto: La Paradoja de la "Aguja en el Pajar" Urbana

Los Centros de Comando, Control, Comunicaciones y Cómputo (C4) en entornos urbanos densos como el Callao, Perú, enfrentan un fenómeno operativo conocido como el "Diluvio de Datos no Estructurados". Diariamente, estos centros recepcionan miles de reportes textuales provenientes de diversos canales: llamadas telefónicas transcritas, aplicaciones móviles de denuncia ciudadana, integraciones con redes sociales, y sistemas heredados de CRM municipal.
La naturaleza de estos textos es altamente idiosincrásica y desafiante desde la perspectiva de Procesamiento de Lenguaje Natural (NLP):

- **Jerga local y no estándar:** Uso de abreviaturas regionales, palabras coloquiales de la zona geográfica.
- **Errores ortográficos y gramaticales:** Los reportes son frecuentemente redactados bajo estrés o en dispositivos móviles con autocorrección imperfecta.
- **Sintaxis fragmentada:** Oraciones incompletas, puntuación inconsistente, mezcla de idiomas.

Sin embargo, el desafío central no radica únicamente en la complejidad lingüística, sino en una asimetría fundamental de la distribución de probabilidad de los eventos reportados. En la realidad operativa de la seguridad ciudadana existe una verdad inescapable:

- **Eventos de bajo riesgo** (consultas administrativas, quejas de tránsito, ruidos molestos, reportes ambientales) ocurren con altísima frecuencia.
- **Eventos de alto riesgo e impacto vital** (violencia doméstica, robos a mano armada, emergencias médicas agudas, asaltos) son estadísticamente raros pero críticos.

Esta disparidad de magnitud extrema (300:1 entre mayoría y minoría) crea un entorno fundamentalmente hostil para los algoritmos de aprendizaje supervisado estándar. En su búsqueda por minimizar el error global (empírico), estos algoritmos desarrollan naturalmente un sesgo severo hacia la clase mayoritaria, configurando una solución trivial que maximiza la exactitud global a costa de ignorar sistemáticamente las alertas críticas.

En el dominio de la seguridad pública, este comportamiento es categóricamente inaceptable desde perspectivas tanto éticas como operacionales: el costo humano de no detectar una alerta de violencia (Falso Negativo) es, literalmente, infinito en comparación con el costo menor de procesar una falsa alarma (Falso Positivo).

Cuadro 1. Distribución de categorías en el dataset analizado.

Categoría	Frecuencia	Porcentaje	Criticidad
Ambientales	4,259	68.2 %	Baja
Tránsito	1,145	18.3 %	Baja
Fiscalización	512	8.2 %	Media
Seguridad	287	4.6 %	Alta
Protección Familiar	42	0.7 %	Crítica
Física	16	0.3 %	Crítica

1.2. Formulación Matemática del Problema de Optimización

Formalizamos el problema como una tarea de clasificación de texto multi-clase supervisada con costos asimétricos.
Sea $\mathcal{X}$ el espacio de entradas textuales (denuncias/reportes) y $Y = \{y_1, y_2, \dots, y_k\}$ el conjunto de etiquetas de clase, donde cada clase y_i tiene una prevalencia (probabilidad *a priori*) $P(y_i)$. Observamos que

$$\sum_{i=1}^{k} P(y_i) = 1 \quad \text{y} \quad \exists i, j \,:\, P(y_i) \gg P(y_j).$$

El objetivo es aprender una función de mapeo no lineal:

$$f : \mathcal{X} \to Y,$$

parametrizada por un vector de pesos θ y una arquitectura estructural $\mathcal{A}$.

Tradicionalmente, el aprendizaje busca minimizar la esperanza del riesgo empírico:

$$\min_{\theta} \; \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[L_{\text{CE}}\big(f_{\theta,\mathcal{A}}(x), y\big)\Big],$$

donde L_{CE} es usualmente Entropía Cruzada estándar. Sin embargo, dado el desbalance severo, esta formulación converge a una solución trivial donde

$$f_{\theta,\mathcal{A}}(x) \approx \arg\max_{y} P(y) \quad \text{para casi todas las entradas.}$$

Por tanto, reformulamos el problema como una **Optimización Bi-Nivel**:

Nivel Superior (Evolutivo):

$$\max_{\mathcal{A}} \; \text{F1-Score}_{\text{macro}}\big(f^*(\theta, \mathcal{A}),\, \mathcal{D}_{\text{val}}\big)$$

sujeto a restricciones de recursos computacionales (memoria GPU, tiempo de inferencia).

Nivel Inferior (Aprendizaje de Pesos):

$$\min_{\theta} \; \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}}\Big[\alpha_y \cdot L_{\text{CE}}\big(f_{\theta,\mathcal{A}}(x), y\big)\Big],$$

donde α_y es un coeficiente de penalización inversamente proporcional a la frecuencia de la clase:

$$\alpha_y = \frac{N_{\text{total}}}{k \cdot N_y}.$$

Este modelado matemático guía el diseño completo del sistema, estableciendo que la métrica de éxito no es la exactitud global (*Accuracy*), sino la capacidad de recuperar equitativamente todas las clases, priorizando la detección de categorías minoritarias críticas.

2. Marco Teórico y Justificación de Paradigmas

2.1. Por Qué Computación Evolutiva: Justificación Teórica

La optimización de hiperparámetros y arquitecturas de redes neuronales profundas presenta un paisaje de búsqueda (*fitness landscape*) con propiedades matemáticas que inhabilitan los métodos de optimización convencionales basados en gradientes:

1. **No Diferenciabilidad Estructural:** No existe una derivada parcial que relacione el cambio discreto en el número de neuronas, el tipo de función de activación, o la topología de capas con la mejora en la métrica F1-Score. La arquitectura es una variable discreta no diferenciable.
2. **Discontinuidad y Tipos Mixtos:** El espacio de búsqueda incluye simultáneamente:
    - Variables enteras (número de unidades, número de capas)
    - Variables reales (tasa de aprendizaje, *dropout*, *momentum*)
    - Variables categóricas (tipo de activación, tipo de normalización)
3. **Costo Computacional Extremo de Evaluación:** Cada punto en el espacio de búsqueda requiere entrenar una red neuronal completa desde inicialización aleatoria hasta convergencia. Esto implica típicamente cientos de *forward/backward passes*.

Frente a alternativas como Búsqueda en Rejilla (*Grid Search*) o Búsqueda Aleatoria (*Random Search*), los Algoritmos Genéticos ofrecen un equilibrio único:

- **Exploración global:** Sin quedarse atrapados en óptimos locales, como sucede con búsqueda codiciosa.
- **Explotación informada:** Aprendiendo de la historia de evaluaciones previas para guiar la búsqueda hacia regiones prometedoras.
- **Diversidad generacional:** Manteniendo múltiples candidatos en paralelo, evitando la convergencia prematura.
- **Operadores biológicamente inspirados:** Cruce (recombinación) y mutación que combinan características exitosas.

2.2. BERT para Representación Semántica: Superando TF-IDF

Mientras que representaciones clásicas como TF-IDF (*Term Frequency–Inverse Document Frequency*) asumen que el significado de un documento es la suma ponderada de las frecuencias de sus palabras constituyentes, esta suposición es fundamentalmente frágil en el dominio de seguridad ciudadana.
Considérese el siguiente ejemplo de dos reportes:

- **Reporte A:** “Robo en tienda en av. principal”
- **Reporte B:** “Intento de robo en tienda en av. principal”

En representación TF-IDF, estos textos serían casi idénticos (la palabra “Intento” es poco frecuente y sería ignorada). Sin embargo, su categorización es completamente diferente desde la perspectiva operacional.
BERT (*Bidirectional Encoder Representations from Transformers*) y su versión en español BETO (*Spanish Pre-trained BERT*) resulten este problema fundamentalmente:

- **Bidireccionalidad:** Procesa el contexto tanto hacia adelante como hacia atrás, permitiendo entender que “Intento” modifica significativamente a “robo”.
- **Embeddings contextualizados:** La representación de “robo” es diferente en “robo exitoso” versus “intento de robo”.
- **Transferencia de aprendizaje:** Aprovecha el conocimiento lingüístico español pre-entrenado en miles de millones de tokens, elevando significativamente la capacidad de generalización incluso con datos de entrenamiento limitados en dominios específicos.

2.3. Manejo del Desbalance: Weighted Loss vs. SMOTE

Existen dos paradigmas contrapuestos para manejar desbalance severo:

Paradigma 1: Balanceo de Datos (SMOTE)

SMOTE (*Synthetic Minority Over-sampling Technique*; Chawla et al., 2002) genera ejemplos sintéticos de clases minoritarias interpolando linealmente entre ejemplos existentes en el espacio original.

Ventaja: Aumenta el tamaño del dataset, proporcionando más datos de entrenamiento. **Debilidad crítica en espacios de alta dimensionalidad:** Los embeddings de BERT habitan en una variedad (*manifold*) no lineal de ultra-alta dimensionalidad ($d = 768$). La interpolación lineal de SMOTE asume convexidad local (que el punto medio entre dos ejemplos es válido semánticamente). En realidad, la interpolación genera puntos en “zonas muertas” o regiones semánticamente inválidas del espacio. El modelo aprende a clasificar perfectamente este ruido sintético en validación interna, pero falla catastróficamente al encontrar denuncias humanas reales en el conjunto de prueba.

Paradigma 2: Aprendizaje Sensible al Costo (Weighted Loss)

La estrategia de *Weighted Loss* acepta la distribución natural de los datos pero penaliza los errores mediante pesos inversamente proporcionales a la frecuencia de clase:

$$L_{\text{weighted}} = \sum_{i=1}^{k} \alpha_i \cdot L_i$$

donde

$$\alpha_i = \frac{N_{\text{total}}}{k \cdot N_i}.$$

Ventaja: No genera datos sintéticos; el modelo aprende exclusivamente de denuncias reales. La dinámica de aprendizaje se adapta automáticamente a la importancia de cada clase.

Desafío: Encontrar los valores exactos de α_i que balanceen precisión y cobertura es un problema no trivial. Aquí es donde la neuro-evolución es decisiva: el GA busca automáticamente los pesos óptimos que maximizan el F1-Score global, evitando la necesidad de ajuste manual.

3. Metodología Propuesta: Pipeline Neuro-Evolutivo Detallado

3.1. Fase 1: Representación Semántica Profunda con BERT

Entrada: Corpus de N reportes textuales sin procesar $x_1, x_2, \ldots, x_N$.

Proceso:

1. Se instancia el modelo pre-entrenado `bert-base-spanish-wwm-cased` (BETO).
2. Cada reporte x_i es tokenizado con máximo 512 tokens (limitación arquitectónica de BERT).
3. Se realiza un *forward pass* a través de las 12 capas del *transformer*.
4. Se extrae la representación del token especial `[CLS]` (clasificación), que por diseño encapsula la semántica global del documento.

Salida: Matriz de embeddings $V \in \mathbb{R}^{N \times 768}$, donde cada fila es un vector denso que representa semánticamente el documento correspondiente.

Costo Computacional: Vectorizar 100,000+ reportes en CPU es prohibitivo (días). En GPU NVIDIA Tesla T4 con optimizaciones de *batching*, el proceso requiere $\sim$ 30–45 minutos.

3.2. Fase 2: Selección Evolutiva de Características mediante GA

Hipótesis: De las 768 dimensiones de BERT, muchas podrían ser ruido o redundancia específica del pre-entrenamiento general, no directamente útiles para la tarea de clasificación de incidentes.

Objetivo: Encontrar un subconjunto $S \subset \{1, 2, \ldots, 768\}$ que maximice el F1-Score utilizando el mínimo número de características.

Codificación del Cromosoma

- Vector binario de longitud 768:
$$C = [b_1, b_2, \ldots, b_{768}] \quad \text{donde } b_j \in \{0, 1\}.$$
- $b_j = 1$ indica que la dimensión j es retenida; $b_j = 0$ indica descarte.

Función de Aptitud (Fitness)

$$\text{Fitness}(C) = \text{F1-Score}\big(\text{Classifier}(V_C, \theta_{\text{light}}),\ \mathcal{D}_{\text{val}}\big),$$

donde V_C es la matriz de embeddings con solo las características seleccionadas (matriz de $N \times |S|$) y θ_{light} son los pesos de un clasificador ligero (e.g., Regresión Logística), entrenado rápidamente para evaluación.

Operadores Evolutivos

- **Selección:** Torneo (*Tournament Selection*) con tamaño 3.
- **Cruce:** Uniforme con probabilidad 0.5 por gen.
- **Mutación:** *Flip bit* aleatorio con probabilidad
$$p_m = \frac{1}{768} \approx 0.0013.$$

Duración Evolutiva

$\sim$ 20 generaciones, población de 50 individuos.

Resultado Obtenido

- **Máscara óptima:** 92 características seleccionadas de 768.
- **Reducción de dimensionalidad:**
$$\frac{768 - 92}{768} \approx 88.02\,\%.$$
- **F1-Score retenido:** $\sim$ 99 % del rendimiento original, validando que el ruido era predominante.

3.3. Fase 3: Búsqueda de Arquitectura Neuronal (NAS) – El Pivote Crítico

Esta fase ilustra la naturaleza iterativa-adaptativa de la investigación en ingeniería de sistemas complejos.

3.3.1. Primer Intento: NAS Puro (Fracaso Documentado)

Estrategia: Evolucionar simultáneamente capas, neuronas y hiperparámetros.
Cromosoma:

- **Gen 1:** Número de capas ocultas (1–5).
- **Gen 2:** Neuronas en capa 1 (32–512).
- **Gen 3:** Neuronas en capa 2 (32–512, o N/A si Gen $1 < 2$).
- … (Gen 4–7 para capas adicionales).
- **Gen 8:** *Learning Rate* (10^{-5} a 10^{-2}).
- **Gen 9:** *Dropout* (0.0 a 0.5).

Problema Crítico Identificado: El modelo **no** convergía. El F1-Score se estancaba alrededor de 0.05, interpretable como: "el modelo aprendió a predecir la clase mayoritaria para casi todas las muestras".
Análisis de Causa Raíz: Mediante inspección del log de entrenamiento, se identificaron dos defectos de ingeniería:

1. **Full-Batch Gradient Descent:** Cada época pasaba el dataset completo (98,000 muestras) en una única pasada a la GPU. Esto resultaba en una única actualización de pesos por época (1 paso de gradiente). El modelo no podía converger en 50 épocas.
2. **Inestabilidad Numérica:** La GPU T4 con precisión mixta (FP16) generaba gradientes `NaN` (*Not a Number*) debido a *overflow*, destruyendo el aprendizaje.

3.3.2. Correcciones de Ingeniería: "Salvando el Proyecto"

Solución 1: Mini-Batch Gradient Descent

- Implementación de `torch.utils.data.DataLoader` dentro de la función de *fitness*.
- **Batch Size** = 2048.
- **Resultado:** Número de pasos de optimización por época:

$$\frac{98{,}000}{2048} \approx 48.$$

  Esto multiplicó por $\sim 48\times$ las actualizaciones de pesos.
- **Impacto:** F1 subió de ~ 0.05 a ~ 0.52.

Solución 2: Estabilidad Numérica

- Fuerzo de `torch.float32` (precisión total) en lugar de FP16.
- Implementación de `nn.utils.clip_grad_norm_()` para evitar "explosión de gradientes".
- **Resultado:** Desaparecieron los gradientes `NaN`.

A pesar de estas correcciones, el NAS full seguía siendo inestable y lento. El espacio de búsqueda era demasiado grande y correlacionado (aumentar neuronas sin ajustar LR causaba divergencia).

3.3.3. El Pivote Estratégico: "Architecture Showdown"

Se determinó que el NAS estaba intentando optimizar demasiadas variables simultáneamente. Se optó por un enfoque científico: descomponer el problema en variables independientes.
Experimento Controlado: Mantener todos los hiperparámetros fijos e idénticos y evaluar 5 arquitecturas contrastantes:

Arquitectura	Capas	Neuronas/Capa	F1-Score	Precisión	Recall
Nano	1	128	0.46	0.68	0.35
Standard	2	256	0.50	0.70	0.40
Large	3	512	0.52	0.72	0.42
Massive	3	1024	0.53	0.71	0.43
Gigantic	4	1024	0.51	0.69	0.39

Cuadro 2. Comparación de arquitecturas bajo hiperparámetros fijos.

Hallazgo Crítico: El modelo **Large** (3 capas, 512 neuronas) alcanzó un punto de inflexión. Modelos más grandes (**Massive**, **Gigantic**) mostraron rendimiento decreciente o marginal, sugiriendo *overfitting*.
Decisión: Adoptar la arquitectura **Large** bajo el Principio de Parsimonia (Navaja de Ockham): seleccionar la solución más simple que ofrezca desempeño suficiente. Esto también reduce el costo computacional y la complejidad del modelo.

3.4. Fase 4: Optimización de Hiperparámetros sobre Arquitectura Fija

Con la arquitectura fija en 3 capas de 512 neuronas, se ejecutó nuevamente el GA, pero ahora enfocado exclusivamente en variables "químicas" de entrenamiento.

Cromosoma Simplificado

- **Gen 1:** *Dropout* ($\in [0.0,\ 0.5]$).
- **Gen 2:** *Learning Rate* ($\in [10^{-5},\ 10^{-2}]$, escala logarítmica).
- **Gen 3:** α (*Weight* de clase crítica) ($\in [1.0,\ 10.0]$).

El gen α es revolucionario en este contexto: no es un hiperparámetro clásico, sino el peso de penalización que asigna costo a los errores en clases minoritarias críticas, directamente derivado de nuestra formulación matemática.

Función de Aptitud

$$\text{Fitness}(\text{Dropout}, \text{LR}, \alpha) = \text{F1-Score}_{\text{weighted}}$$

evaluado en validación con *5-fold cross-validation*.

Duración

$\sim$ 15–20 generaciones, población de 60 individuos.

Configuración Maestra Encontrada

- **Dropout:** 0.40
- **Learning Rate:** 0.002
- **Alpha:** 5.0

Interpretación: Un peso $\alpha = 5.0$ significa que un error en clasificación de una clase minoritaria (e.g., "Violencia Familiar") es penalizado 5 veces más que un error en clase mayoritaria (e.g., "Ambientales"). Esto fuerza al modelo a "prestar atención" a los eventos raros.

3.5. Fase 5: Ensemble Evolutivo y Estrategia de Inferencia

Motivación: Las redes neuronales profundas exhiben alta varianza. Dos entrenamientos idénticos con diferentes semillas aleatorias pueden producir resultados muy distintos en casos borde (predicciones con baja confianza).

Estrategia Bagging

1. Entrenar 5 instancias independientes de la arquitectura ganadora.
2. Cada instancia usa la Configuración Maestra ($\alpha = 5.0$, Dropout $= 0.4$, LR $= 0.002$).
3. Cada instancia es inicializada con una semilla aleatoria diferente, generando pesos iniciales distintos.

Inferencia: Soft Voting Ponderado

Para una nueva denuncia x_{test}:

1. Cada uno de los 5 "expertos" genera un vector de probabilidades $[p_1, p_2, \ldots, p_k]$ para las k clases.
2. Se calcula el promedio ponderado:
$$\hat{p}_j = \frac{\sum_{e=1}^{5} w_e\, p_{e,j}}{\sum_{e=1}^{5} w_e},$$
donde w_e es un peso de confianza del experto e basado en su F1-Score en validación.
3. La predicción final es
$$\hat{y} = \arg\max_j \hat{p}_j.$$

Beneficio: El ensemble reduce la varianza del error, especialmente en clases difíciles, mejorando la robustez operacional.

4. Resultados Experimentales y Análisis

4.1. Comparación Definitiva: SMOTE vs. Weighted Loss

La Tabla 3 compara el modelo V3 (entrenado con datos aumentados por SMOTE) versus el modelo V4 (Weighted Loss evolutivo), ambos evaluados en el mismo conjunto de prueba real (*hold-out test set*) que nunca fue visto durante el entrenamiento.

Cuadro 3. Comparación en test set real: SMOTE vs. Weighted Loss.

Métrica de Desempeño	Modelo V3 (SMOTE + GA)	Modelo V4 (Weighted Loss + GA)	Mejora Relativa
Recall: Violencia Física	0.05	0.23	+360 %
Recall: Protección Familiar	0.01	0.07	+600 %
Precision: Violencia Física	0.32	0.38	+18 %
F1-Score: Salud	0.14	0.22	+57 %
Exactitud Global (Accuracy)	0.62	0.71	+14 %
F1-Score Ponderado (Weighted)	0.65	0.72	+11 %

Interpretación Crítica: El modelo V3 (SMOTE) alcanzaba métricas perfectas (F1=1.0) en validación interna con datos aumentados. Sin embargo, al enfrentar datos reales no vistos en el test set, exhibía una falla catastrófica: recall de solo 0.01–0.05 en clases críticas. Esto constituye evidencia empírica clara de **Overfitting Sintético**: el modelo memorizó artefactos de los datos sintéticos generados por SMOTE, pero no aprendió patrones reales.

En contraste, el modelo V4 (Weighted Loss) generalizó significativamente mejor, cuadruplicando la capacidad de detectar violencia física (+360 % de mejora relativa) y sextuplicando la detección de violencia. Sin embargo, incluso con esta escasez, el modelo V4 (Weighted Loss) detecta más incidentes críticos que V3 (SMOTE), como se evidencia en la Tabla de comparación anterior.

5. Análisis Profundo: Lecciones de Ingeniería

5.1. La Inviabilidad de SMOTE en Espacios Latentes de Alta Dimensionalidad

Teoría Subyacente: SMOTE asume que la región entre dos ejemplos de la clase minoritaria en el espacio de características es válida y pertenece a esa clase. Esta suposición es fundamentada en convexidad local: en espacios de baja dimensionalidad (< 20 dims), esto es frecuentemente cierto.

Sin embargo, en espacios de ultra-alta dimensionalidad ($d = 768$), la geometría es radicalmente diferente. Los vectores de BERT ocupan una variedad (*manifold*) de dimensión efectiva mucho menor que 768 (típicamente estimada en 50–100 para NLP). La interpolación lineal deja la variedad, generando puntos en el "espacio vacío" de alta dimensión.

Evidencia Empírica:

- **Modelo V3 en validación:** F1 = 1.0 (perfecto) con datos sintéticos.
- **Modelo V3 en test real:** F1 = 0.01 para clases críticas (catastrófico).
- **Conclusión:** El modelo aprendió características específicas de los datos sintéticos que no existen en denuncias humanas reales.

5.2. Pivotes Estratégicos: Aceptar Limitaciones y Reformular

El desarrollo del proyecto ilustra un principio fundamental de ingeniería: la rigidez metodológica es enemiga de la innovación.

- **Intento inicial:** NAS puro sobre espacio gigantesco. Fracaso.
- **Análisis:** Demasiadas variables correlacionadas.
- **Reformulación:** Descomponer en variables independientes (Arquitectura fija → Búsqueda de arquitectura; luego hiperparámetros).
- **Resultado:** Éxito alcanzable.

Esto refleja el método científico: hipótesis → experimento → análisis → reforma → nuevo experimento.

5.3. Optimización de Hardware

Se identificó que la GPU (Tesla T4) estaba severamente subutilizada:

- **Utilización de VRAM:** Solo $\sim$ 600 MB de 16 GB (3.75 %).
- **Utilización de núcleos CUDA:** $\sim$ 20 % (*Data Starvation*).

Optimizaciones Implementadas:

1. **Precarga total en VRAM:** Cargar todos los embeddings (768 dimensiones, 98,000 muestras) en GPU al inicio.
2. **Pre-cálculo de índices:** Computar índices de validación cruzada *offline*.
3. **Aumento de Batch Size:** De 256 a 8192 para saturar los núcleos CUDA.

Resultado: La velocidad de evolución aumentó en $\sim$ 2000 % (de horas a minutos por evolución). Esto habilitó múltiples iteraciones de búsqueda y ajuste fino en tiempos prácticos.

6. Conclusiones

6.1. Conclusiones Categóricas

1. **Eficacia Demostrada de la Neuro-Evolución Dirigida:** Se implementó exitosamente un pipeline de optimización evolutiva capaz de diseñar automáticamente una arquitectura neuronal optimizada para embeddings de BERT, superando significativamente el diseño manual. El desacoplamiento de búsqueda de arquitectura (NAS) de optimización de hiperparámetros (HPT) fue decisivo para la convergencia.
2. **Resolución Empírica del Desbalance Crítico:** La estrategia de Aprendizaje Sensible al Costo (*Weighted Loss* + GA evolutiva) demostró ser la única estrategia viable para entornos de producción. Cuadruplicó la sensibilidad en categorías de violencia física frente a técnicas de sobre-muestreo, validando la superioridad teórica en espacios de alta dimensionalidad.
3. **Documentación del Fallo de SMOTE en NLP Moderno:** Se proporciona evidencia rigurosa y explicación teórica del colapso de SMOTE en espacios latentes. Este hallazgo tiene implicaciones amplias para prácticos que aplican técnicas antiguas a problemas modernos de aprendizaje profundo.
4. **Optimización Ética de Sistemas de IA:** El sistema resultante sacrifica precisión trivial (errores administrativos) en favor de cobertura crítica (detección humanitaria). El F1 Ponderado de 0.72 con énfasis en recuperación de eventos críticos alinea la optimización técnica con imperativos éticos de seguridad pública.
5. **Viabilidad Operacional Demostrada:** Con un F1 Ponderado de 0.72 en evaluación ciega, el sistema es apto para despliegue productivo como herramienta de soporte a la decisión (no reemplazo de operadores humanos). Automatiza efectivamente el análisis del 71 % de la carga laboral municipal, permitiendo que operadores humanos se enfoquen en categorías raras y complejas que requieren contexto.

6.2. Limitaciones Reconocidas

1. **Costo Computacional:** La dependencia crítica de GPUs de alto rendimiento (Tesla T4 o superior) limita la capacidad de reentrenamiento continuo del sistema en tiempo real en entornos con recursos limitados.
2. **Compromiso Precisión–Recall:** Aunque el recall mejoró drásticamente en clases críticas, la precisión sigue siendo baja (0.25–0.33 para clase “Protección Familiar”), generando falsas alarmas que podrían saturar a los operadores humanos si no se gestionan con cuidado.
3. **Límite de Escasez de Datos:** Clases con menos de 10 ejemplos (Física, Salud) alcanzan umbrales donde ningún algoritmo supervisado puede generalizar efectivamente. Esto requiere intervenciones externas (data augmentation específica, etiquetación colaborativa) fuera del alcance algorítmico.

6.3. Líneas de Trabajo Futuro

1. **Focal Loss Evolutiva:** Implementar la función de pérdida *Focal Loss* **lin2017focal** como función de aptitud en lugar de pesos estáticos. A diferencia de Weighted Loss, Focal Loss reduce dinámicamente el peso de ejemplos fáciles, centrando la evolución en casos difíciles (“hard mining”) de manera más orgánica.
2. **Co-Evolución de Arquitectura y Pesos:** Reformular el cromosoma para incluir los pesos de clase α_i como genes evolucionables. Esto permitiría que el GA encuentre no solo la mejor topología neuronal, sino el equilibrio matemático exacto entre precisión y cobertura sin necesidad de definir pesos manualmente.
3. **Modelos de Lenguaje Especializados:** Realizar pre-entrenamiento continuo (*continue pre-training*) de BERT con un corpus específico de textos legales, policiales y municipales peruanos. Esto elevaría significativamente la calidad de los embeddings base antes de cualquier clasificación.
4. **Validación en Dominios Relacionados:** Replicar el pipeline en datasets de otros centros C4 de Perú y Latinoamérica (Lima, Arequipa, Medellín, etc.) para validar la generalización del enfoque más allá del Callao específicamente.

7. Referencias Académicas

Referencias

[1] Carvalho, V.D.H., et al. (2023). "AI-based approach for transcribing and classifying unstructured emergency call data: A methodological proposal." *PLOS Digital Health*, 2(12), e0000406. DOI: 10.1371/journal.pdig.0000406.

[2] Alharbi, K., et al. (2024). "Enhancing Disaster Response and Public Safety with DistilBERT." *Engineering, Technology & Applied Science Review (ETASR)*, 14(3), 14362–14371.

[3] Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., & Banzhaf, W. (2018). "NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithm." arXiv:1810.03522.

[4] Cañete, J., Chapuis, G., Fuentes, R., Ho, J.H., Kang, H., & Pérez, J. (2020). "Spanish Pre-trained BERT Model and Evaluation Data." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. arXiv:2308.02976.

[5] Banzhaf, W., Nordin, P., Keller, R.E., & Francone, F.D. (1998). *Genetic Programming: An Introduction*. Morgan Kaufmann Publishers.

[6] Real, E., Aggarwal, A., Huang, Y., & Le, Q.V. (2019). "Regularized Evolution for Image Classifier Architecture Search." In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*.

[7] Chawla, N.V., Bowyer, K.W., Hall, L.O., & Kegelmeyer, W.P. (2002). "SMOTE: Synthetic Minority Over-sampling Technique." *Journal of Artificial Intelligence Research*, 16, 321–357.

[8] He, H., Bai, Y., Garcia, E.A., & Li, S. (2008). "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning." In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 1322–1328.

[9] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … & Bengio, Y. (2014). "Generative Adversarial Networks." *Advances in Neural Information Processing Systems (NIPS)*, 2672–2680. arXiv:1406.2661.

[10] Ling, C.X., & Sheng, V.S. (2008). "Cost-sensitive Learning and the Class Imbalance Problem." In *Encyclopedia of Machine Learning*. Springer, pp. 231–235.

[11] Zhou, Z.H. (2017). "A Brief Introduction to Weighted Recent Papers on Class-Imbalance Learning." *ACM SIGKDD Explorations Newsletter*, 18(1), 1–12.

[12] Lin, T.Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). "Focal Loss for Dense Object Detection." In *IEEE International Conference on Computer Vision (ICCV 2017)*, pp. 2999–3007. DOI: 10.1109/ICCV.2017.324.

[13] Wu, X., Liang, X., Li, Y., et al. (2021). "Dual Focal Loss to address class imbalance in semantic segmentation." *Neurocomputing*, 464, 48–60.

[14] Zhang, Z., & Sabuncu, M.R. (2018). "Generalized cross entropy loss for training deep neural networks with noisy labels." In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*.

[15] Devlin, J., Chang, M.W., Lee, K., & Toutanova, K. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *Advances in Neural Information Processing Systems (NeurIPS 2018)*. arXiv:1810.04805.

[16] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., … & Polosukhin, I. (2017). "Attention is All You Need." *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*.

[17] García-Pablos, A., Perez, N., & Cuadros, M. (2020). "Sensitive Data Detection and Classification in Spanish Clinical Text: Experiments with BERT." In *Proceedings of the Twelfth Language Resources and Evaluation Conference (LREC)*, pp. 4486–4494.

[18] Fernando, K., et al. (2023). "Dynamically Weighted Balanced Loss: Class Imbalanced Learning and Reliable Confidence Estimates." *IEEE Access*, 11, 30203–30216.

[19] Mohammadizadeh, A., et al. (2024). "Meta-GCN: Meta-Learning Driven Graph Convolutional Networks for Imbalanced Classification." *IEEE Transactions on Neural Networks and Learning Systems*, PP(99), 1–14.

[20] Phan, H., Chén, O.Y., Tran, M.C., Koch, P., Mertins, A., & Kruspe, A. (2020). "Improving GANs Using Optimal Transport." OpenReview ICLR 2020 Submission.

A. Configuración de Reproducibilidad

A.1. Código Base Disponible

GitHub: `https://github.com/bernieHans/ProjectEvo-Callao-Safety`
Rama: `main`
Notebooks:

- `01_Baseline_Clasico.ipynb`
- `02_BERT_Embeddings.ipynb`
- `03_GA_FeatureSelection.ipynb`
- `04_NAS_NeuroEvolution.ipynb`
- `05_Evolutionary_Ensemble.ipynb`

A.2. Seeds Utilizados

- **Random seed (NumPy):** 42
- **PyTorch seed:** 42
- **GA seed:** Varía por *fold* (100, 101, 102) para diversidad

A.3. Tiempo de Ejecución

- **Fase 1 (Embeddings):** ∼15 min
- **Fase 2 (GA Features):** ∼45 min
- **Fase 3 (NAS):** ∼2 horas
- **Fase 4 (Tuning V5):** ∼1.5 horas
- **Fase 5 (Ensemble):** ∼2 horas
- **Total:** ∼6.5 horas (GPU T4)