Tecnológico de Monterrey

Deep Learning

Momento de Retroalimentación: Módulo 2 Implementación de un modelo de deep learning.

Profesor:
Obed Noé Sámano Abonce

Alumno:
Cristóbal Camarena Hernández

Grupo:
501

Fecha:
16 de noviembre de 2025

1. Introducción

La detección de acciones humanas en videos es un problema fundamental en visión por computadora con aplicaciones en seguridad, deportes, rehabilitación y entretenimiento. Tradicionalmente, este problema se aborda procesando directamente los frames RGB de los videos, lo cual requiere grandes recursos computacionales y puede verse afectado por iluminación, fondo y ropa.

Una alternativa prometedora es utilizar representaciones esqueletales, que capturan únicamente las posiciones de las articulaciones clave del cuerpo humano. Estas representaciones son más ligeras, invariantes a cambios visuales superficiales y capturan directamente la estructura biomecánica de las acciones.

1.1. Objetivos

- Implementar un modelo baseline (LSTM) para establecer una línea base de rendimiento
- Desarrollar un modelo avanzado (ST-GCN) optimizado para datos esqueletales
- Comparar el rendimiento de ambos modelos
- Analizar el impacto de diferentes técnicas de regularización y arquitecturas

2. Dataset

El dataset UCF101 contiene 13,320 videos anotados con 101 clases diferentes de acciones humanas. Cada video tiene anotaciones de esqueletos 2D en formato COCO, con 17 keypoints por persona.

2.1. Características del Dataset

- **Número de clases**: 101
- **Número total de videos**: 13,320
- **Keypoints por persona**: 17 (formato COCO)
- **Formato de keypoints**: (x, y, confidence_score)
- **Splits**: 3 splits de entrenamiento/test con distribuciones diferentes

2.2. Preprocesamiento

El preprocesamiento incluye las siguientes operaciones:

1. **Normalización**: Las coordenadas de los keypoints se normalizan dividiendo por las dimensiones de la imagen, resultando en valores en el rango [0, 1]
2. **Selección de persona**: Si hay múltiples personas detectadas, se selecciona la que tiene mayor confianza promedio

3. **Padding/Truncado**: Las secuencias se paddean o truncan a 300 frames para tener secuencias de longitud fija
4. **Concatenación de características**: Las coordenadas (x, y) se concatenan con los scores de confianza para formar vectores de características de 3 dimensiones por keypoint

3. Modelos Implementados

3.1. Modelo Baseline: LSTM Bidireccional

El modelo baseline utiliza una arquitectura LSTM bidireccional para procesar secuencias temporales de keypoints.

3.1.1. Arquitectura

- **Input**: Secuencias de keypoints aplanados de forma (T, V×C) donde T=300 frames, V=17 keypoints, C=3 características
- **LSTM**: 2 capas LSTM bidireccionales con 128 unidades ocultas cada una
- **Dropout**: 0.5 para regularización
- **Capas Fully Connected**:
  - FC1: 256 unidades con ReLU
  - Dropout: 0.5
  - FC2: 101 unidades (clases)
- **Output**: Probabilidades sobre las 101 clases

3.1.2. Justificación

El modelo LSTM es una elección natural para secuencias temporales y sirve como baseline efectivo. La bidireccionalidad permite capturar dependencias temporales en ambas direcciones, mejorando la comprensión del contexto de las acciones.

3.2. Modelo Principal: ST-GCN

ST-GCN (Spatial-Temporal Graph Convolutional Network) es un modelo diseñado específicamente para datos esqueletales que explota la estructura de grafo inherente a los esqueletos humanos.

3.2.1. Arquitectura

Estructura de Grafo Los 17 keypoints COCO se conectan formando un grafo no dirigido basado en las conexiones anatómicas del cuerpo humano:

- Conexiones de cabeza (nariz, ojos, orejas)
- Conexiones de torso (hombros, caderas)
- Conexiones de extremidades (brazos y piernas)

Bloks ST-GCN Cada bloque ST-GCN combina:

1. **Convolución Espacial (Graph Convolution)**: Aplica convolución sobre el grafo espacial de keypoints usando una matriz de adyacencia normalizada
2. **Convolución Temporal**: Aplica convolución 1D sobre la dimensión temporal
3. **Conexiones Residuales**: Facilita el entrenamiento de redes profundas

Arquitectura Completa

- **Input Layer**: Graph Convolution + BatchNorm + ReLU
- **4 Bloques ST-GCN** con canales [64, 64, 128, 256]
- **Global Average Pooling**: Agrega información temporal y espacial
- **Clasificación**: Fully Connected Layer con 101 salidas
- **Dropout**: 0.5 para regularización

3.2.2. Ventajas sobre LSTM

- **Explota estructura espacial**: El modelo comprende relaciones anatómicas entre keypoints
- **Convolución sobre grafo**: Más eficiente que procesar keypoints independientemente
- **Invariante a permutaciones**: El orden de los keypoints no afecta el resultado
- **Mejor para acciones complejas**: Captura patrones espaciales-temporales más sofisticados

4. Entrenamiento

4.1. Hiperparámetros

Hiperparámetro	Baseline LSTM	ST-GCN
Batch Size	32	32
Learning Rate	0.001	0.001
Weight Decay	1e-4	1e-4
Optimizer	Adam	Adam
Scheduler	ReduceLROnPlateau	ReduceLROnPlateau
Dropout	0.5	0.5
Epochs	50	100

Cuadro 1: Hiperparámetros de entrenamiento

4.2. Técnicas de Regularización

- **Dropout**: 0.5 en capas fully connected

- **Weight Decay**: 1e-4 (regularización L2)
- **Batch Normalization**: En bloques ST-GCN y después de convoluciones
- **Learning Rate Scheduling**: Reducción de LR cuando la pérdida de validación se estanca
- **Early Stopping**: Guardado del mejor modelo basado en accuracy de validación

4.3. Pipeline de Entrenamiento

1. Carga y preprocesamiento de datos
2. División en sets de entrenamiento y validación (split1)
3. Entrenamiento con monitoreo de pérdida y accuracy
4. Validación periódica para detectar overfitting
5. Guardado automático del mejor modelo
6. Registro de métricas en TensorBoard

5. Resultados

5.1. Métricas de Evaluación

Los modelos se evalúan usando:

- **Accuracy**: Porcentaje de predicciones correctas
- **Precision, Recall, F1-Score**: Por clase y promedios macro/weighted
- **Confusion Matrix**: Para análisis detallado de errores

5.2. Resultados Experimentales

Los modelos fueron entrenados y evaluados en el split 1 del dataset UCF101. Los resultados obtenidos se presentan en la Tabla 2.

Modelo	Accuracy (%)	F1-Score (Weighted)	Parámetros
Baseline LSTM	34.76	0.325	672,357
ST-GCN	14.06	0.108	935,973

Cuadro 2: Resultados de evaluación en el conjunto de test

5.2.1. Modelo Baseline (LSTM)

El modelo baseline fue entrenado durante 20 épocas debido a limitaciones de tiempo computacional. Los resultados obtenidos son:

- **Accuracy de validación**: 34.76 %
- **Accuracy de entrenamiento final**: 43.75 %
- **F1-Score (weighted)**: 0.325
- **Pérdida de validación**: 2.47
- **Número de parámetros**: 672,357

El modelo muestra un rendimiento moderado, con una diferencia significativa entre el accuracy de entrenamiento y validación, lo que sugiere cierto grado de overfitting. Sin embargo, el modelo logra aprender patrones básicos de las secuencias temporales de keypoints.

5.2.2. Modelo ST-GCN

El modelo ST-GCN fue entrenado durante únicamente 1 época debido a limitaciones computacionales severas. Los resultados obtenidos son:

- **Accuracy de validación**: 14.06 %
- **Accuracy de entrenamiento final**: 12.12 %
- **F1-Score (weighted)**: 0.108
- **Pérdida de validación**: 3.42
- **Número de parámetros**: 935,973

El modelo ST-GCN muestra un rendimiento muy bajo, lo cual es esperado dado que solo se entrenó durante 1 época. El modelo no tuvo suficiente tiempo para converger y aprender los patrones espaciales-temporales complejos que requiere esta arquitectura.

5.3. Análisis Comparativo

Los resultados experimentales muestran que el modelo baseline (LSTM) supera al modelo ST-GCN en este experimento. Sin embargo, esta comparación no es justa debido a las limitaciones computacionales que impidieron entrenar completamente el modelo ST-GCN.

Análisis del Baseline LSTM

- **Ventajas observadas**: Implementación más simple, entrenamiento más rápido, menor uso de memoria
- **Rendimiento**: 34.76 % de accuracy, lo cual es razonable considerando que solo se entrenó 20 épocas
- **Limitaciones**: El modelo muestra signos de overfitting (diferencia entre train y validation accuracy)
- **Observaciones**: El modelo logra aprender patrones temporales básicos pero tiene dificultades con la complejidad de 101 clases

Análisis del ST-GCN

- **Estado del entrenamiento**: Solo 1 época completada, claramente insuficiente para convergencia
- **Rendimiento actual**: 14.06 % de accuracy, cercano al rendimiento aleatorio (0.99 % para 101 clases)
- **Expectativas**: Con entrenamiento completo (50-100 épocas), se esperaría un rendimiento superior al baseline
- **Complejidad**: El modelo requiere más tiempo de entrenamiento debido a su arquitectura más compleja

Conclusión del Análisis La comparación directa de los resultados no es representativa del potencial real de cada modelo. El ST-GCN requiere significativamente más épocas de entrenamiento para alcanzar su rendimiento óptimo, mientras que el LSTM converge más rápidamente pero tiene un techo de rendimiento más bajo. En un entorno con recursos computacionales adecuados, se esperaría que el ST-GCN supere al baseline después de un entrenamiento completo.

6. Mejoras Implementadas

6.1. Optimizaciones de Arquitectura

1. **Normalización de Grafo**: La matriz de adyacencia se normaliza para estabilizar el entrenamiento
2. **Conexiones Residuales**: Permiten entrenar redes más profundas sin degradación
3. **Batch Normalization**: Acelera la convergencia y mejora la estabilidad
4. **Global Pooling**: Agrega información de forma invariante al número de frames

6.2. Técnicas de Entrenamiento

1. **Learning Rate Scheduling**: Reduce LR cuando el modelo deja de mejorar
2. **Early Stopping**: Previene overfitting guardando el mejor modelo
3. **Weight Decay**: Regularización L2 para prevenir sobreajuste
4. **Data Normalization**: Normalización de keypoints para estabilizar el entrenamiento

6.3. Preprocesamiento

1. **Selección de Persona**: Prioriza la persona con mayor confianza en videos con múltiples personas
2. **Padding Inteligente**: Padding con ceros mantiene la estructura temporal
3. **Normalización Relativa**: Normalización basada en dimensiones de imagen preserva proporciones

7. Limitaciones Computacionales

Este proyecto se enfrentó a limitaciones significativas de capacidad computacional que afectaron el alcance y los resultados del entrenamiento. Esta sección documenta estas limitaciones y su impacto.

7.1. Recursos Disponibles

El entrenamiento se realizó en un equipo con las siguientes características:

- **Procesador**: CPU (sin aceleración GPU)
- **Memoria**: Limitada para cargar batches grandes
- **Tiempo de cómputo**: Restringido por la disponibilidad del equipo

7.2. Impacto en el Entrenamiento

7.2.1. Modelo Baseline (LSTM)

El modelo baseline fue entrenado durante 20 épocas en lugar de las 50 planificadas:

- **Tiempo por época**: Aproximadamente 20-30 minutos
- **Tiempo total**: 7-10 horas para 20 épocas
- **Impacto**: El modelo no alcanzó su potencial máximo, pero logró converger parcialmente
- **Observación**: El accuracy de entrenamiento (43.75 %) sugiere que con más épocas podría mejorar significativamente

7.2.2. Modelo ST-GCN

El modelo ST-GCN fue severamente limitado por los recursos computacionales:

- **Tiempo por época**: Aproximadamente 2-3 horas (significativamente más lento que el baseline)
- **Épocas completadas**: Solo 1 época de las 100 planificadas
- **Impacto crítico**: El modelo no tuvo oportunidad de converger, resultando en un rendimiento cercano al aleatorio
- **Razón de la lentitud**:
  - Operaciones de convolución sobre grafo más complejas
  - Mayor número de parámetros (935,973 vs 672,357)
  - Operaciones matriciales más costosas en CPU

7.3. Adaptaciones Realizadas

Para mitigar las limitaciones, se implementaron las siguientes adaptaciones:

1. **Reducción de batch size**: Se utilizó batch size de 32, que es relativamente pequeño pero necesario para evitar problemas de memoria
2. **Limitación de épocas**: Se redujo el número de épocas planificadas para ambos modelos
3. **Optimización de código**: Se implementaron optimizaciones para reducir el tiempo de cómputo por iteración
4. **Guardado frecuente**: Se guardaron checkpoints cada 10 épocas para no perder progreso

7.4. Recomendaciones para Mejores Resultados

Para obtener resultados más representativos, se recomienda:

- **Entrenamiento extendido**: Entrenar el baseline durante al menos 50 épocas y el ST-GCN durante 100+ épocas
- **Batch size mayor**: Con más memoria, aumentar el batch size a 64 o 128 para estabilizar el entrenamiento
- **Subset de clases**: Para pruebas iniciales, trabajar con un subconjunto de 10-20 clases para validar la implementación más rápidamente

7.5. Lecciones Aprendidas

Las limitaciones computacionales destacaron varios aspectos importantes:

1. **Importancia de la planificación**: Es crucial estimar correctamente los recursos necesarios antes de comenzar el entrenamiento
2. **Arquitecturas eficientes**: El modelo LSTM, aunque más simple, demostró ser más práctico en entornos con recursos limitados
3. **Validación temprana**: Aunque el ST-GCN no convergió, la implementación fue validada y está lista para entrenamiento completo con mejores recursos
4. **Documentación de limitaciones**: Es importante documentar las limitaciones para interpretar correctamente los resultados

8. Conclusiones

Este proyecto implementó exitosamente dos modelos de deep learning para la clasificación de acciones humanas basada en esqueletos 2D. A pesar de las limitaciones computacionales, se logró:

- Implementar completamente ambos modelos (Baseline LSTM y ST-GCN)
- Entrenar el modelo baseline hasta un punto de convergencia parcial (20 épocas)
- Validar la implementación del ST-GCN (aunque sin convergencia debido a limitaciones)
- Establecer un pipeline completo de preprocesamiento, entrenamiento y evaluación
- Documentar las limitaciones y su impacto en los resultados

Resultados del Baseline El modelo baseline LSTM alcanzó un accuracy de 34.76 % después de 20 épocas, lo cual es un resultado razonable considerando:

- La complejidad del problema (101 clases)
- El número limitado de épocas de entrenamiento
- La ausencia de aceleración por GPU

Estado del ST-GCN El modelo ST-GCN, aunque no pudo ser entrenado completamente, demostró:

- La implementación es correcta y funcional
- La arquitectura está lista para entrenamiento completo
- Con recursos adecuados, se esperaría un rendimiento superior al baseline

Las técnicas de regularización implementadas (dropout, weight decay, batch normalization) están correctamente aplicadas y serían efectivas con entrenamiento completo. El preprocesamiento cuidadoso de los datos, incluyendo normalización y padding, es crucial para el rendimiento del modelo y está correctamente implementado.