La investigación reportada en esta tesis es parte de los programas de investigación del CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada, B.C.).

La investigación fue financiada por el CONACYT (Consejo Nacional de Ciencia y Tecnología).

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de lo Estados Unidos Mexicanos (México). El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo o titular de los Derechos Autor.

CICESE © 2022, Todos los Derechos Reservados, CICESE

Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

---

Maestría en Ciencias
en Ciencias de la Computación

Traducción de Lenguaje de Señas Mexicano a texto mediante aprendizaje profundo

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Michelle Sainos Vizuett

Ensenada, Baja California, México

2022

Tesis defendida por

Michelle Sainos Vizuett

y aprobada por el siguiente Comité

Dr. Irvin Hussein López Nava

Director de tesis

Dr. Jesús Favela Vara

Dra. Carolina Álvarez Delgado

Dr. Edgar Leonel Chávez González

Dr. Pedro Gilberto López Mariscal

Coordinador del Posgrado en Ciencias de la Computación

Dr. Pedro Negrete Regagnon

Director de Estudios de Posgrado

Copyright © 2022, Todos los Derechos Reservados, CICESE
Prohibida su reproducción parcial o total sin la autorización por escrito del CICESE

Resumen de la tesis que presenta Michelle Sainos Vizuett como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Traducción de Lenguaje de Señas Mexicano a texto mediante aprendizaje profundo

Resumen aprobado por:

Dr. Irvin Hussein López Nava
Director de tesis

En México existen 4.2 millones de personas con discapacidad auditiva. Las personas con sordera severa o profunda son incapaces de escuchar a otros, por lo que sufren problemas de comunicación. Las Lenguas de Señas son usadas por las comunidades sordas del mundo para comunicarse entre sí. Este problema de comunicación es bidireccional, y pueden seguir dos direcciones: traducción de Lengua de Señas al lenguaje hablado o viceversa. El reconocimiento o clasificación de las señas es parte fundamental en la traducción. En el presente trabajo se busca desarrollar una herramienta capaz de clasificar señas dinámicas de la Lengua de Señas Mexicana (LSM) al español. Para lograrlo, se diseñó un diccionario de 100 señas que forman parte de la LSM con apoyo de expertos. Después, se diseñó un protocolo para la captura de los videos de las señas con 10 participantes no expertos, y 2 señistas expertos. Posteriormente, en una fase de procesamiento se transformaron y adecuaron los datos en tres pasos: (i) la obtención de las articulaciones del esqueleto humano mediante OpenPose y su selección por relevancia, (ii) la creación de 2 conjuntos de datos usando dos distintos esquemas de aumento de datos, y (iii) la normalización de estos conjuntos de datos. El aumento de datos se realizó usando un esquema de re-muestreo y un esquema basado en las modificaciones de las posiciones de las articulaciones. La última fase consistió en el entrenamiento y la evaluación de los modelos de aprendizaje de máquina usando los dos conjuntos de datos. Se propusieron 3 arquitecturas de aprendizaje profundo (LSTM, CNN-LSTM y ResNet) y un modelo de aprendizaje de máquina tradicional (KNN-DTW) que se tomó como modelo base. El modelo que con mejores resultados fue el modelo ResNet, con una exactitud de 0.81 sobre el conjunto de datos de prueba. A pesar de que el modelo con menor costo computacional fue el modelo de LSTM, el modelo ResNet tiene resultados comparables en tiempo de entrenamiento y tiempo de prueba lo que lo hace adecuado para ser mejorado en el futuro. El presente estudio no solo permitió la traducción de una Lengua de Señas al español en el contexto abordado, también dejó las bases para nuevos estudios en el área.

Palabras clave: Lengua de Señas Mexicana, Aprendizaje Profundo, articulaciones, OpenPose

Abstract of the thesis presented by Michelle Sainos Vizuett as a partial requirement to obtain the Master of Science degree in Master of Science.

Mexican Sign Language translation to text using deep learning

Abstract approved by:

Dr. Irvin Hussein López Nava
Thesis Director

In Mexico, there are 4.2 million people with hearing disabilities. Many people with hearing loss experience some issues because of their impaired ability to communicate with other people. Sign Languages are used by deaf communities around the world to communicate with each other effectively. This communication problem is bidirectional and follows two directions: translation from Sign Language to spoken language or vice versa. The recognition or classification of signs is a fundamental part of translation. The present work aims to develop a tool capable of classifying dynamic signs from the Mexican Sign Language (LSM) into Spanish. For this purpose, a corpus of 100 signs that are part of the LSM was defined with the support of experts. Afterward, a protocol to capture videos of the signs with 10 non-expert participants and 2 experts, was designed. Subsequently, in a processing phase, the data were transformed and adapted in three steps: (i) obtaining the joints of the human skeleton through OpenPose tool and their selection by relevance, (ii) the creation of 2 datasets using two different schemes of data augmentation, and (iii) the normalization of these datasets. Data augmentation was performed using a resampling scheme, and a scheme based on changes in joint positions. The last phase consisted of the training and evaluation of the machine learning models using the two datasets. Three deep learning architectures (LSTM, CNN-LSTM, and ResNet) were implemented. In addition, a traditional machine learning model (KNN-DTW) was used as the base model. The model with the best results was the ResNet, with an accuracy of 0.81 on the test dataset. Although the model with the lowest computational cost was the LSTM, the ResNet model has similar results in training- and testing time, which makes it a more suitable candidate for improvement in the future. The present study not only allowed the translation of a Sign Language into Spanish but also opened the foundations for new studies in the area.

Keywords: Mexican Sign Language, Deep Learning, OpenPose

Dedicatoria

A Rafa Sainos y Rosy Vizuett gracias por su amor sincero profundo e incomparable. Si no fuera por ustedes nunca me habría convertido en la persona que soy hoy.

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada Baja California por brindarme la oportunidad de realizar mi sueño.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de maestría.

A mi director de tesis Irvin Hussein López Nava por toda su atención y aportes a este trabajo.

A mi comité de tesis por sus valiosas observaciones.

A los participantes voluntarios de la adquisición de datos, Bere, Dann, Yulith, Hector, Sarid, Juan Ca, Esteban, Ricardo, Joan, Emilio y Luis. Este trabajo no pudo haber sido posible sin ustedes.

A la Asociación Regional de Sordos Ensenadenses y a Sonia Esther Rico Meneses por su ayuda con la propuesta del diccionario y los videos de intérprete.

A Bere que ha estado presente durante estos 2 años en las buenas y en las malas, una gran amiga y excelente persona. A mis hermanas Pam y Jaz que a pesar de la distancia las llevo en el corazón, gracias por toda la buena vibra. A Mariana, Abraham y Stef por todas las porras, las buenas partidas y los buenos ratos.

Tabla de contenido

Página

Resumen en español ... ii

Resumen en inglés ... iii

Dedicatoria ... iv

Agradecimientos ... v

Lista de figuras ... ix

Lista de tablas ... xii

Capítulo 1. Introducción

1.1. Motivación ... 1
1.2. Introducción al problema ... 2
1.2.1. Direcciones de traducción ... 3
1.3. Propuesta de investigación ... 4
1.3.1. Pregunta de investigación ... 4
1.3.2. Objetivos ... 4
1.3.3. Limitaciones ... 4
1.4. Organización de la tesis ... 5

Capítulo 2. Marco teórico

2.1. Aprendizaje de máquina ... 6
2.1.1. Tareas T ... 7
2.1.2. Métricas de rendimiento P ... 9
2.1.3. Ejemplos de experiencia E ... 10
2.1.4. Sobreajuste y subajuste ... 11
2.1.5. Construcción de un algoritmo de aprendizaje ... 13
2.2. Aprendizaje profundo ... 13
2.3. Redes neuronales artificiales ... 14
2.3.1. Modelos lineales ... 14
2.3.2. No linealidad ... 15
2.3.3. Retropropagación ... 17
2.3.4. Hiperparámetros ... 18
2.3.5. Regularización ... 21
2.3.6. Dropout ... 21
2.3.7. Normalización por lote (Batch-Normalization) ... 21
2.4. Costo computacional ... 22
2.4.1. Frameworks de aprendizaje profundo ... 22
2.5. Arquitecturas ... 23
2.5.1. Perceptrón multicapa ... 23
2.5.2. Redes neuronales convolucionales ... 23
2.5.2.1. Operación de convolución ... 24
2.5.2.2. Pooling ... 25
2.5.3. Redes neuronales recurrentes ... 25
2.6. OpenPose ... 25

2.7. Clasificación de series de tiempo . . . 28
2.7.1. Modelos de aprendizaje de máquina . . . 28
2.7.2. Series de tiempo . . . 29
2.7.3. Modelos de aprendizaje profundo . . . 30
2.7.3.1. Modelos discriminativos . . . 31
2.8. Lengua de Señas Mexicana . . . 31

Capítulo 3. Trabajo relacionado
3.1. Metodología de revisión . . . 34
3.2. Reconocimiento de señas . . . 36
3.3. Uso del cuerpo humano para la tarea de SLR . . . 36
3.4. Sistemas de reconocimiento de señas . . . 38
3.4.1. Adquisición de datos . . . 40
3.4.1.1. Adquisición de datos basado en Hardware . . . 40
3.4.1.2. Adquisición de datos basado en Software . . . 40
3.4.2. Pre-procesamiento . . . 41
3.4.2.1. Representación de las señas . . . 41
3.4.2.2. Normalización y aumento de datos . . . 42
3.4.2.3. Extracción de características . . . 42
3.4.2.4. Selección de características . . . 43
3.4.3. Métodos de clasificación de señas . . . 43
3.4.3.1. Modelos basados en aprendizaje profundo . . . 44
3.4.3.2. Enfoque de combinación de clasificadores . . . 46
3.4.4. Granularidad de los sistemas de traducción . . . 46
3.4.4.1. Modelos continuos . . . 47
3.4.4.2. Modelos aislados . . . 47
3.5. Conjuntos de datos de LS . . . 47
3.6. Estado del arte en torno al reconocimiento de la LSM . . . 49

Capítulo 4. Metodología
4.1. Diccionario de señas . . . 51
4.2. Captura de señas . . . 52
4.3. Pre-procesamiento . . . 54
4.3.1. Extracción de articulaciones . . . 55
4.3.2. Selección de articulaciones . . . 55
4.3.3. Aumentación de datos . . . 56
4.3.3.1. Muestreo de las series de tiempo (Sampling) . . . 56
4.3.3.2. Modificaciones de las articulaciones (Hand-Switch) . . . 57
4.3.4. Normalización . . . 58
4.4. Modelos de clasificación . . . 59
4.4.1. Modelo base: KNN (DTW) . . . 60
4.4.2. RNN con celdas LSTM . . . 61
4.4.3. CNN-LSTM . . . 62
4.4.4. ResNet . . . 63
4.5. Conjuntos de entrenamiento, validación y prueba . . . 64

Capítulo 5. Resultados
5.1. Modelo base KNN (DTW) . . . 66
5.2. RNN (LSTM) . . . 67
5.2.1. Métricas de rendimiento de los modelos para el conjunto Sampling . . . 67
5.2.2. Métricas de rendimiento de los modelos para el conjunto Hand-Switch . . . 69
5.3. CNN-LSTM . . . 70
5.3.1. Métricas de rendimiento de los modelos para el conjunto Sampling . . . 70
5.3.2. Métricas de rendimiento de los modelos para el conjunto Hand-Switch . . . 72
5.4. ResNet . . . 73
5.4.1. Métricas de rendimiento de los modelos para el conjunto Sampling . . . 73
5.4.2. Métricas de rendimiento de los modelos para el conjunto Hand-Switch . . . 74
5.5. Comparación entre modelos . . . 76
5.6. Discusión de resultados . . . 77

Capítulo 6. Conclusiones
6.1. Trabajo a futuro . . . 80

Literatura citada . . . 82

Anexo A . . . 92
.1. Descenso por gradiente . . . 92
.2. Descenso por gradiente estocástico (SGD) . . . 93

Anexo B . . . 94
.1. Lista de glosas . . . 94

Lista de figuras

Figura Página

1. Direcciones de traducción entre lenguas de señas y lenguajes hablados. 3
2. Ramas de la IA 6
3. Clasificación de algoritmos de aprendizaje de máquina. 7
4. Ejemplo de la tarea de clasificación de imágenes de señas. 8
5. Ejemplo de una matriz de confusión para una clasificación binaria. 9
6. División de un conjunto de datos para entrenar un algoritmo de ML. 10
7. Cuadro comparativo entre subajuste, ajuste adecuado y sobreajuste. 12
8. Problema de dimensionalidad en los datos. 13
9. Representación de modelos neuronales. 14
10. Función de activación ReLU. 16
11. Ejemplo de una red neuronal simple con una capa oculta. 16
12. Interpretación gráfica de la capa Softmax en un ejemplo multiclase. 19
13. Comparación del valor de la función de costo/pérdida usando diferentes algoritmos de optimización. 19
14. Problemas con el algoritmo de descenso por gradiente. 20
15. Red Neuronal con dropout en la última capa oculta del $40\,\%$. 21
16. Ilustración de un perceptrón multicapa. 23
17. Operaciones dentro de una capa convolucional. 24
18. Representación gráfica de una capa recurrente. 25
19. Flujo del funcionamiento de OpenPose. 26
20. Arquitectura de la CNN multi-fase. 27
21. Detección de puntos clave del cuerpo humano para tronco superior y manos. 28
22. Representación gráfica de un sistema de clasificación de series de tiempo que se basa en aprendizaje profundo. 30
23. Enfoques de DL para resolver el problema de clasificación de series de tiempo. 30
24. Ejemplos de Ideogramas y Dactilología. 32
25. Características de los ideogramas. 32
26. Diagrama de inclusión y exclusión de literatura. 35
27. Representación de la misma seña con dos diferentes señistas expertos. 37
28. Clasificación del trabajo previo sobre los sistemas de traducción de Lenguas de Señas. . . 38
29. Ejemplos de señas dinámicas y estáticas. 39

30. Artículos publicados del estado del arte en SLR con modelos de ML. 43
31. Diferencias entre modelos de reconocimiento aislados y modelos de traducción continuos. 46
32. Diagrama de la metodología a seguir en este trabajo. 51
33. Ejemplo de frase en señas traducida a glosas. 52
34. Set-up de grabación de videos de señas. 53
35. Muestra de participantes para las grabaciones de los videos de señas. 53
36. Diagrama de flujo del Pre-procesamiento de los videos de señas. 54
37. Representación gráfica de la extracción de articulaciones con OpenPose. 55
38. Selección de articulaciones. 55
39. Aumentación de secuencias de frames usando técnica de remuestreo. 56
40. Articulaciones de las manos detectadas por OpenPose. 58
41. Representación gráfica de la normalización de frames. 58
42. Modelo general de clasificación de señas para este trabajo. 59
43. Modelos propuestos para realizar la clasificación de señas. 60
44. Representación gráfica de un modelo KNN y la distancia DTW. 60
45. Representación gráfica de la arquitectura RNN con celdas LSTM. 61
46. Funcionamiento de un modelo CNN-LSTM. 62
47. Representación gráfica del modelo CNN-LSTM. 62
48. Representación gráfica del modelo ResNet. 63
49. Porcentajes de datos utilizados para el entrenamiento, validación y para prueba. 64
50. Matriz de confusión del modelo base KNN-DTW. 66
51. Curvas de aprendizaje del modelo RNN (LSTM) con el conjunto de Sampling. 67
52. Matriz de confusión del modelo RNN (LSTM) con el conjunto de Sampling. 68
53. Curvas de aprendizaje del modelo LSTM con el conjunto de Hand-Switch. 69
54. Matriz de confusión del modelo LSTM con el conjunto de Hand-Switch. 70
55. Curvas de aprendizaje del modelo CNN-LSTM con el conjunto de Sampling. 71
56. Matriz de confusión del modelo CNN-LSTM con el conjunto de Sampling. 71
57. Curvas de aprendizaje del modelo CNN-LSTM con el conjunto de Hand-Switch. 72
58. Matriz de confusión del modelo CNN-LSTM con el conjunto de Hand-Switch. 72
59. Curvas de aprendizaje del modelo ResNet con el conjunto de Sampling. 73
60. Matriz de confusión del modelo ResNet con el conjunto de Sampling. 74
61. Curvas de aprendizaje del modelo ResNet con el conjunto de Hand-Switch. 75

62. Matriz de confusión del modelo ResNet con el conjunto de Hand-Switch. 75
63. Comparación del F1-Score de todos los modelos. 77
64. Señas que comparten queirema y kinema. 78
65. Señas que comparten toponema y kinema. 78
66. Aproximación de la minimización de la función objetivo. 92

Lista de tablas

Tabla Página

1. Ejemplo del cálculo de los nodos ocultos en una red neuronal. 17
2. Principales conjuntos de *corpus* de Lenguas de Señas. 47
3. Trabajo previo en la traducción de LSM. 49
4. Tabla de resultados del modelo LSTM. 69
5. Tabla de resultados del modelo CNN-LSTM. 73
6. Tabla de resultados del modelo ResNet. 76
7. Tabla comparativa de modelos para el conjunto de entrenamiento Sampling. . . . 76
8. Tabla comparativa de modelos para el conjunto de entrenamiento Hand-Switch. . 77

Capítulo 1. Introducción

1.1. Motivación

De acuerdo con la Organización Mundial de la Salud (OMS), el 5 % de la población mundial sufre una discapacidad auditiva, este número parece ser pequeño pero equivale a 460 millones de personas en el mundo (Farooq et al., 2021). Además, se pronostica que en 2050 sean 900 millones de personas las que sufran de pérdida auditiva. En México existen 4.2 millones de personas con discapacidad auditiva. De esta cifra, 1.3 millones sufren pérdida auditiva severa (considerada como sordera) mientras que 2.9 millones sufren de pérdida auditiva moderada (Mejía-Peréz et al., 2022). La pérdida auditiva se clasifica como poca, moderada, severa y profunda basada en la intensidad de la sordera. Las personas con sordera severa o profunda son incapaces de escuchar a otros, por lo que sufren problemas de comunicación (Martinez-Seis et al., 2019).

Las Lenguas de Señas son usadas por las comunidades sordas del mundo para comunicarse entre sí. Estos lenguajes son basados en señas, en ellos la persona sorda gesticula usando las manos y expresiones faciales. Cada gesto o seña representa una palabra o frase en el lenguaje hablado (Bragg et al., 2019). La Lengua de Señas oficial en México es la Lengua de Señas Mexicana (**LSM**), reconocida como una Lengua oficial desde el 2005. De acuerdo con Lissi et al. (2012) las personas sordas en México son clasificadas según su manejo del lenguaje en cuatro grupos:

- **Monolingües**. Estos individuos solo se comunican en LSM.
- **Bilingües**. En este grupo se encuentran los individuos que se comunican en dos lenguas.
    - **LSM como primera lengua**. Estos individuos dominan el LSM como primera lengua y tienen conocimiento, hasta cierto grado del Español oral y/o escrito, de otra lengua de señas.
    - **Español como primera lengua**. Estos individuos dominan el español oral y/o escrito como primera lengua y tienen conocimiento del LSM o de otra lengua de señas como segunda lengua.
- **Multilingües**. Dominan más de una lengua. Usualmente dominan LSM, Español y otro lenguaje natural (como el Inglés) u otro de señas como la Lengua de Señas Americana (ASL).

- **Semilingües**. No son competentes en LSM ni en Español. Lo constituyen aquellos que no han recibido educación formal y por lo tanto desconocen el Español y la LSM, usando como medio de comunicación señas caseras o familiares.

Esta clasificación ayuda a comprender qué parte de la comunidad sorda en México enfrenta dificultades para entablar comunicación con una persona en el lenguaje hablado (español).

Para millones de personas, la comunicación en Lengua de Señas es el medio principal de interacción con el mundo. Sin embargo, la comunidad *oyente* generalmente no conoce los gestos pertenecientes a estas lenguas. En México el 60 % de las personas sordas son analfabetas, debido a que no tuvieron un acceso apropiado a la educación debido a su condición. Esto genera una barrera de comunicación importante entre la comunidad sorda y la comunidad oyente, lo que le resta oportunidades a miembros de la comunidad sorda en ámbitos educativos, laborales y sociales. Por estos motivos no es difícil imaginar las aplicaciones potenciales que involucra el reconocimiento y traducción de las Lenguas de Señas. En particular, las Redes Neuronales Profundas han emergido como una herramienta con mucho potencial para resolver esta tarea. De acuerdo con Cui et al. (2019), el impacto total de su aplicación al problema del Reconocimiento de Lengua de señas (SLR) se verá en el transcurso de esta década.

1.2. Introducción al problema

Existen más de 200 Lenguas de Señas en el mundo. Cada Lengua de Señas es distinta entre sí al igual que los lenguajes hablados. La LSM tiene su propia gramática[1] y léxico[2]. Además no tiene relación directa con el Español oral (Faurot et al., 1999). La LSM también es muy distinta a la Lengua de Señas Española (SSL) y a la Lengua de Señas Americana (ASL) aunque están relacionadas porque tanto la LSM como la SSL se derivan de la vieja Lengua de Señas Francesa (FSL) (Faurot et al., 1999). En el desarrollo de sistemas de reconocimiento de las Lenguas de Señas es importante tomar en cuenta este factor debido a que, en muchas ocasiones, no existen conjuntos de datos de las distintas Lenguas de Señas lo suficientemente robustos para este propósito.

La traducción de las Lenguas de Señas cae dentro del paraguas de los problemas de traducción de lenguajes naturales. Sin embargo, dado que las Lenguas de Señas tienen características particulares diferentes a los lenguajes hablados, existe una clara diferencia con las metodologías usadas en su traducción.

[1] Parte de la lingüística que estudia la estructura de las palabras y sus accidentes, así como la manera en que se combinan para formar oraciones.

[2] Conjunto de las palabras de una lengua.

Las Lenguas de Señas son meramente **visuales** y además, regularmente, no tienen una representación escrita (ágrafas). En el Capítulo 2 se apreciará que la traducción de una Lengua de Señas es una tarea multi-disciplinaria que usualmente combina técnicas de visión por computadora, seguimiento del movimiento humano, inteligencia artificial, métodos estadísticos y procesamiento del lenguaje natural.

1.2.1. Direcciones de traducción

El problema de comunicación entre los oyentes y la comunidad sorda es bidireccional. Los sistemas de traducción para Lenguas de Señas pueden seguir dos direcciones: traducción de Lengua de Señas al lenguaje hablado o viceversa, véase figura 1.

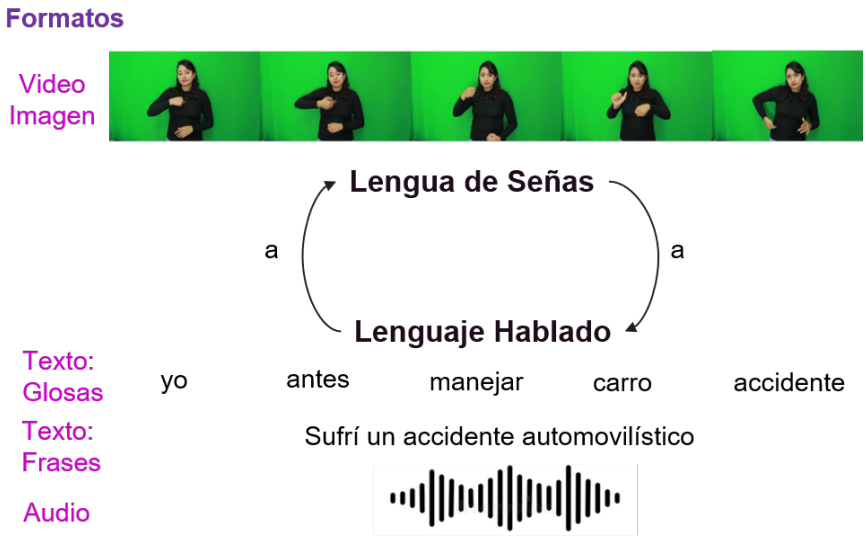

Figura 1. Direcciones de traducción entre lenguas de señas y lenguajes hablados, también se muestran los posibles formatos en los que se puede representar dicha traducción.

Ambas direcciones de traducción son igualmente importantes. El problema de traducción de Lengua de Señas a lenguaje hablado es importante para que la comunidad oyente (con pocos o nulos conocimientos de la Lengua de Señas) pueda comprender lo que comunica el sordo. Mientras que, el problema de traducción de lenguaje hablado a señas es útil para la comunidad sorda que no tiene conocimientos del lenguaje hablado.

La lengua de señas se caracteriza por ser ágrafa, es decir, que no tiene escritura y por esta razón resulta difícil identificar las señas cuando se pretende realizar una investigación gramatical (Burad V, 2019). Por ello, los científicos del área lingüística han buscado una forma material para plasmarlas e identificarlas en un soporte escrito mediante categorías llamadas **glosas**. La glosa es la transcripción de las señas a un medio escrito, también se puede definir como una forma de categorizar las señas por

medio de una o varias palabras escritas.

En la figura 1 se observa que para traducir de las Lenguas de Señas a lenguajes hablados o viceversa se requiere un formatos de codificación para representar dicha entrada/salida. Las Lenguas de Señas son representadas de forma meramente visual por lo que los formato de codificación suelen ser imágenes o secuencias de imágenes (videos). Para representar estas traducciones en lenguajes hablados se suelen usar formatos de texto o glosa y formatos de voz.

1.3. Propuesta de investigación

1.3.1. Pregunta de investigación

¿Con qué exactitud es posible desarrollar un modelo basado en aprendizaje profundo que reconozca señas que son parte de la Lengua de Señas Mexicana a partir videos?

1.3.2. Objetivos

En este trabajo se busca desarrollar modelos computacionales capaces de clasificar (en formato de glosa) señas estáticas y dinámicas de la LSM usando videos. Además se realizará una evaluación comparativa de dichos modelos. Para llevar a cabo el desarrollo de la herramienta descrita se deben cumplir los siguientes objetivos particulares:

- Proponer un diccionario o *corpus*[3] de señas con asesoría de expertos en LSM.
- Recolectar un conjunto de videos de tal forma que sea útil para entrenar un modelo de aprendizaje profundo.
- Proponer e implementar diversas arquitecturas de clasificación de secuencias espacio-temporales y entrenarlas con el conjunto de videos.
- Validar esta herramienta con videos de señistas expertos.

1.3.3. Limitaciones

A continuación se enlistan las principales limitaciones del estudio. Se trabajará con un diccionario limitado de 100 señas pertenecientes a la LSM. Estas señas tienen un contexto de situaciones de emergencia

[3] Conjunto de textos o de datos destinado a la investigación científica.

médica. La mayor parte de estas señas son dinámicas, i.e implican movimientos secuenciales del señista. En este trabajo se realizará un modelo de traducción aislado (véase Sección 3.4.4.2). El set-up experimental se adecuó para la captura de datos en un entorno controlado, i.e., fondo verde, ropa uniforme, una sola persona en la toma, y un protocolo de captura adecuado.

1.4. Organización de la tesis

El resto de la tesis está organizada de la siguiente forma. En el Capítulo 2 se introducen los conceptos básicos. En el Capítulo 3 se revisan los trabajos más importantes en torno a la traducción de Lenguas de Señas y en particular de la Lengua de Señas Mexicana. En el Capítulo 4 se describe la metodología para llevar a cabo los objetivos de este trabajo. En el Capítulo 5 se reportan los resultados de los modelos de reconocimiento de señas y se realiza un . Por último, en el Capítulo 6 se muestran las concusiones de este trabajo de tesis y el trabajo a futuro.

Capítulo 2. Marco teórico

Como se mencionó en el Capítulo 1, este trabajo se enfoca en el reconocimiento de señas de la LSM. Para entender cómo se resuelve esta tarea, es necesario introducir los conceptos de inteligencia artificial como herramientas. En este Capítulo se revisarán los conceptos necesarios para comprender los modelos propuestos en el Capítulo 4. Se iniciará con una introducción al aprendizaje de máquina, al aprendizaje profundo, las métricas de rendimiento de evaluación de los modelos, tipos de arquitectura profundas y transferencia de aprendizaje. Posteriormente, se explicará brevemente el funcionamiento del sistema OpenPose y finalmente se explicarán algunas definiciones y particularidades de la LSM.

2.1. Aprendizaje de máquina

En la figura 2 se observa un diagrama de la inteligencia artificial (**AI**) y como ella envuelve a los algoritmos de aprendizaje de máquina (**ML**) y los de aprendizaje profundo (**DL**).

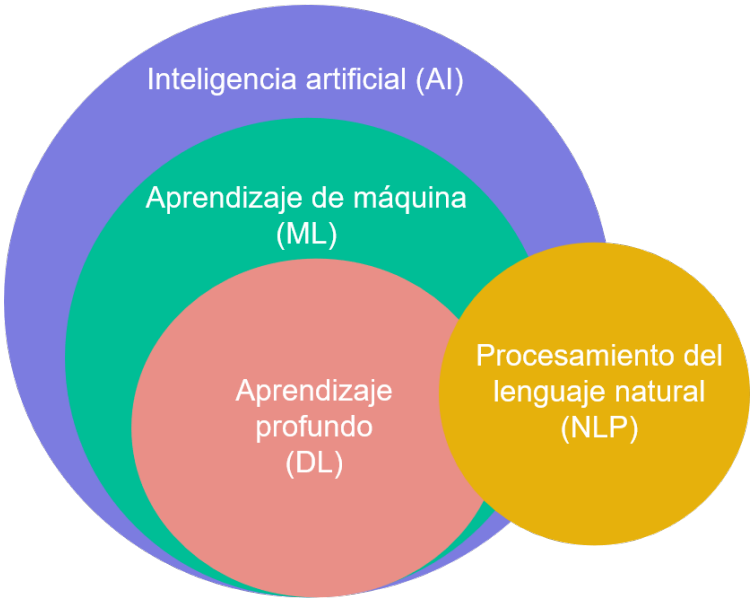

Figura 2. El aprendizaje profundo es un subconjunto del aprendizaje de máquina que a su vez representa un conjunto de algoritmos que forman parte de la inteligencia artificial. Por otro lado, el procesamiento del lenguaje natural son tareas que no necesariamente pertenecen a alguna de las ramas de la AI. Modificada de Decher Davina (2021).

En años recientes, han existido esfuerzos para desarrollar métodos automatizados para completar numerosas tareas de lingüística mediante algoritmos avanzados que puede "*aprender*" basados en experiencia pasada (Alpaydin, 2020). El concepto de aprendizaje de máquina o *machine learning* abarca un gran número de procesos estocásticos[1] que pueden ser usados para predecir el valor de cierto parámetro ba-

[1]Que está sometido al azar y que es objeto de análisis estadístico.

sado en **ejemplos similares** a los cuales ha sido expuesto el algoritmo.

Un algoritmo de aprendizaje de máquina es capaz de aprender de los datos. De acuerdo con Mitchell et al. (1997): "*Un programa de computadora aprende de la experiencia E con respecto a una tarea T y con una medida de rendimiento P si su rendimiento en la tarea T (medido con P) mejora con la experiencia E*". A continuación se explicarán descripciones intuitivas y ejemplos de diferentes tipos de tareas, medidas de rendimiento y experiencias que se pueden usar para construir algoritmos de ML.

2.1.1. Tareas T

Desde un punto de vista científico y filosófico, el aprendizaje de máquina es interesante debido a que su desarrollo implica desarrollar nuestro entendimiento sobre los principios sobre los que se basa la **inteligencia** (Goodfellow et al., 2016).

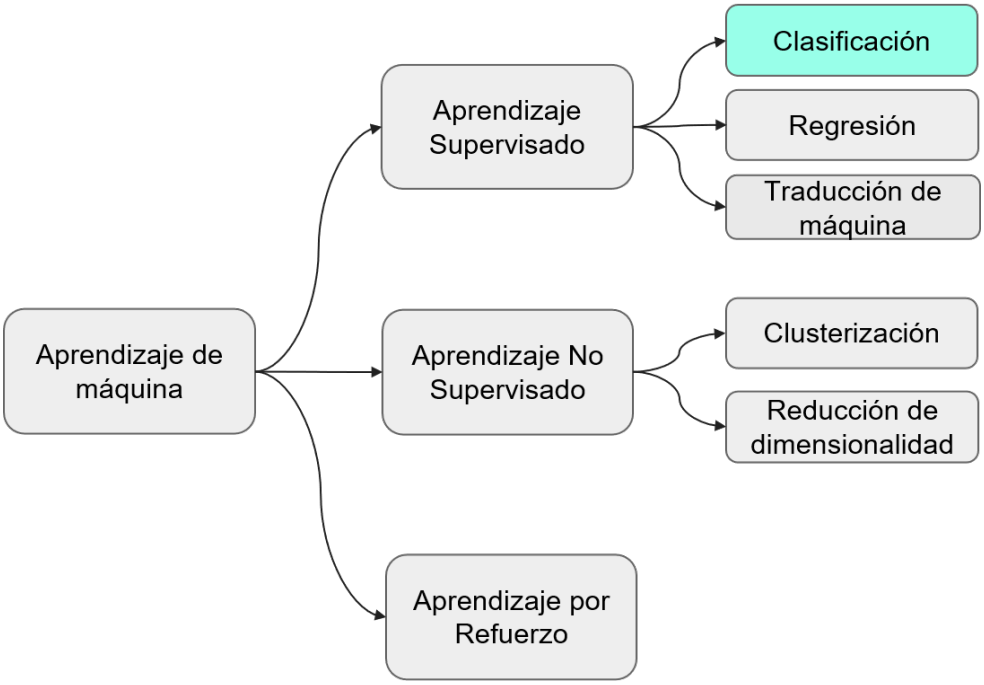

Figura 3. Clasificación de algoritmos de aprendizaje de máquina por el tipo de dato y las tareas que se desean alcanzar.

El proceso de aprendizaje por si mismo no es la tarea T. El aprendizaje es el medio para obtener la habilidad de realizar la tarea. Un ejemplo de esto sería si nosotros quisiéramos hacer que un robot camine, en este caso la tarea T sería la acción de caminar. Las tareas se realizan mediante el aprendizaje del algoritmo y dependen de ejemplos de cómo se deben realizar estas tareas. Usualmente los ejemplos (datos de entrada) se representan como un vector $x \in R^n$ donde cada entrada x_i del vector es una característica. En el caso de imágenes, x_i podría representar los valores numéricos de cada píxel. En la

figura 3 se muestra una clasificación de los algoritmos de aprendizaje de máquina de acuerdo con el tipo de datos de entradas.

Existen muchas tareas que pueden resolverse con aprendizaje de máquina. A continuación se abundará en la tarea de clasificación:

- **Clasificación.** En este tipo de tarea, se le pide al algoritmo especificar a cuál de las k categorías pertenece cierta entrada x_i. Para resolver esta tarea, el algoritmo de aprendizaje produce una función $f : R^n \rightarrow 1, ..., k$. Cuándo $y = f(x)$, el modelo asigna una entrada descrita por un vector x a una categoría identificada por un código numérico y. Un ejemplo de una tarea de clasificación es la de *reconocimiento de objetos en imágenes*. En este caso, el algoritmo aprende de unas serie de imágenes x (que se trataría de los píxeles de la imagen o características numéricas extraídas de cada imagen) con sus respectivas etiquetas codificadas z, el aprendizaje de dicho algoritmo está representado por ciertos pesos ω internos del modelo (parámetros de aprendizaje) que se calibrarán con los ejemplos con los que fue entrenado. Finalmente su salida sería una etiqueta y_n, correspondiente a la predicción del algoritmo a una entrada x_n. Actualmente la tarea de reconocimiento de objetos tiene los mejores resultados usando aprendizaje profundo (Krizhevsky et al., 2012; Ioffe y Szegedy, 2015). En la figura 4 se muestra la fase de **entrenamiento** y de **prueba** del problema de clasificación de imágenes de señas.

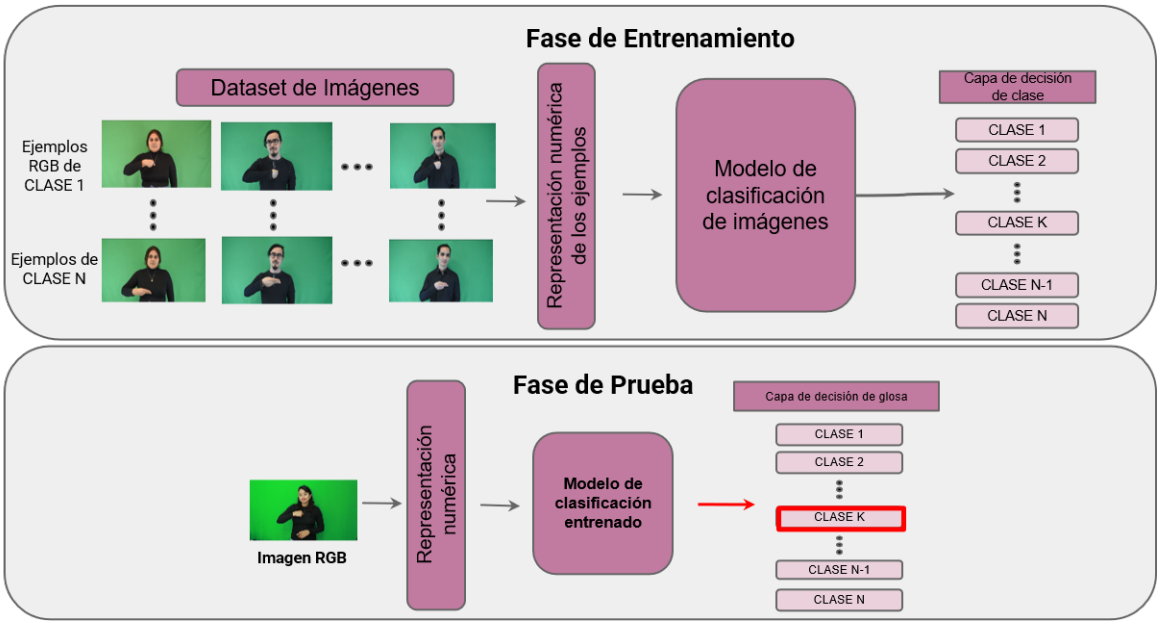

Figura 4. Ejemplo de la tarea de clasificación de imágenes de señas: *a)* Fase de entrenamiento, *b)* Fase de prueba.

2.1.2. Métricas de rendimiento P

Para evaluar las habilidades de un algoritmo de aprendizaje de máquina se debe seleccionar una medida cuantitativa de su rendimiento. Usualmente esta medida de rendimiento P es específica a la tarea T llevada a cabo por el sistema.

Para clasificación usualmente se mide la exactitud o *accuracy*. La *exactitud* es la proporción de ejemplos para los cuáles el modelo produce una predicción correcta. Formalmente, el *accuracy* se define de la siguiente forma:

$$Acc = \frac{\text{Número de predicciones corectas}}{\text{Número total de predicciones}} \tag{1}$$

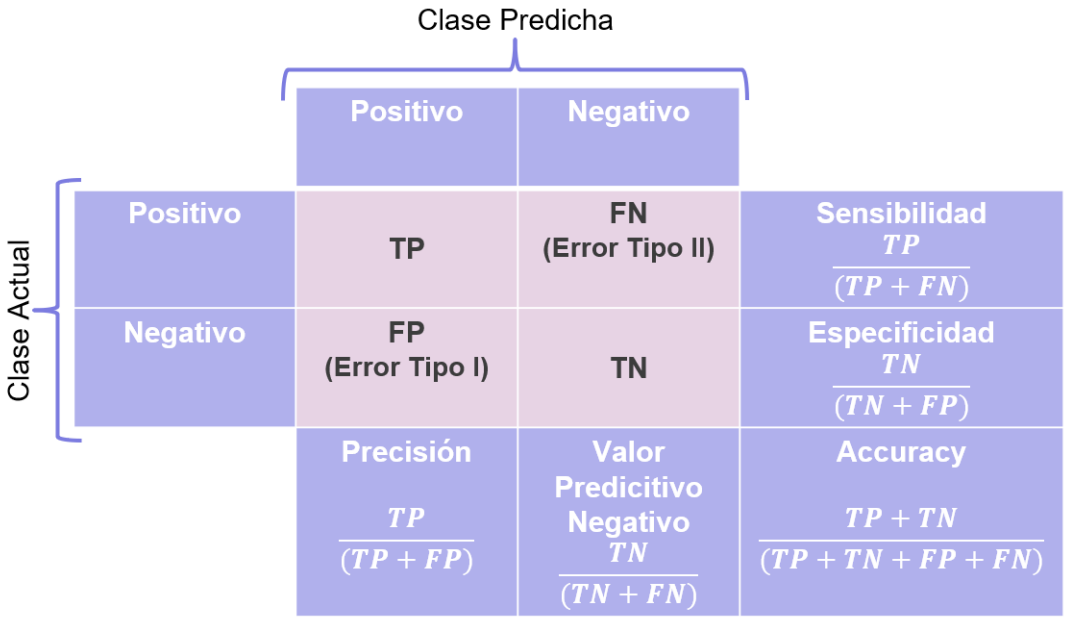

Figura 5. Ejemplo de una matriz de confusión para una clasificación binaria, este se puede extender para un problema multi-clase. Nótese que el accuracy no es la única métrica de rendimiento que se puede obtener, la sensibilidad, especificidad y precisión son otras métricas que dan una estimación de qué tanto se equivocó el modelo a la hora de predecir las categorías.

Dónde el número de predicccciones correctas se puede calcular como la suma de la tasa de *verdaderos positivos* (TP) y de *verdaderos negativos* (TN). Mientras que el número de predicciones totales se puede calcular como la suma de los TP, TN, *falsos positivos* (FP) y los *falsos negativos* (FN). Por lo que la ecuación 1 se puede reescribir de la siguiente forma:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

En la figura 5 se presentan las métricas más comunes de rendimiento para los problemas de clasificación;

en la figura se ejemplifica el caso de clasificación binaria (por ejemplo, dos señas distintas) pero este se puede extender a una clasificación multi-clase. Además, se puede obtener la **tasa de error**. La tasa de error es la proporción de ejemplos para los cuáles el modelo produce una salida incorrecta. Usualmente tanto la exactitud como la tasa de error están en un rango de $0-1$. Si la exactitud es cercana a 1, la tasa de error es cercana a 0 y esto implica qué, en términos generales, el modelo tiene un buen rendimiento ya que clasifica bien la mayor parte de los ejemplos. El caso contrario, en el que la tasa de error es cercana a 1 y el accuracy cercano a 0 implica que el modelo tiene un mal rendimiento en términos generales.

Usualmente nos interesa saber qué tan bien predice un modelo en datos que no ha visto antes (datos diferentes a los ejemplos con los que fue entrenado). Esto para tener una estimación de qué tan buen comportamiento tendrá el modelo cuando salga al mundo real. Para ello, usualmente se miden las métricas de rendimiento sobre un **conjunto de datos de prueba** que es distinto al **conjunto de datos de entrenamiento** sobre los cuáles el algoritmo fue aprendiendo, véase figura 6. La elección de la métrica de rendimiento para evaluar al modelo es difícil. En este trabajo se reportarán las métricas de *exactitud*, *precisión* y *sensibilidad* y F1-score.

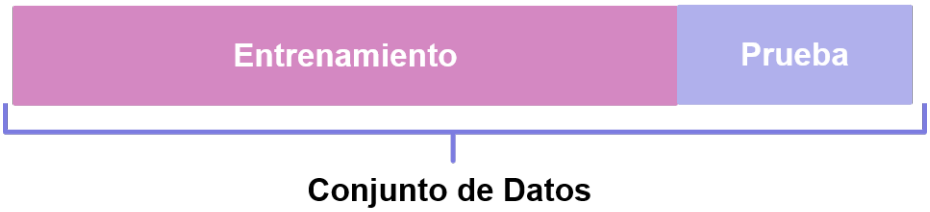

Figura 6. División de un conjunto de datos para entrenar un algoritmo de aprendizaje de máquina. Usualmente la mayor parte de los datos son apartados para el entrenamiento, es decir, para que el algoritmo aprenda a realizar la tarea requerida. Mientras que los datos restantes son apartados para prueba, sobre este conjunto se calculan las métricas de rendimiento para evaluar qué tan bien realiza la tarea el modelo de aprendizaje.

2.1.3. Ejemplos de experiencia E

Como se muestra en la figura 3 los algoritmos de aprendizaje de máquina se pueden dividir en **supervisados** y **no supervisados**. Esta división está basada en el tipo de experiencia (o conjunto de datos) disponible durante el proceso de aprendizaje.

Un conjunto de datos (*dataset*) es una colección de muchos ejemplos de la tarea que se desea realizar. Estos ejemplos pueden tener n dimensiones dependiendo del número de características disponibles en cada ejemplo. A continuación se explicarán a detalle los tipos de algoritmos de aprendizaje de máquina:

- **Algoritmos de aprendizaje no supervisado.** Son aquellos que adquieren experiencia de un con-

junto de datos (con m características o dimensiones cada ejemplo) y aprenden características útiles de la estructura de este conjunto de datos. En el contexto de aprendizaje profundo, usualmente se desea que el algoritmo aprenda la distribución de probabilidad que generó este dataset. Algunas otras tareas de algoritmos no supervisados realizan funciones de agrupamiento, esto consiste en dividir el conjunto de datos en cúmulos de ejemplos similares (Goodfellow et al., 2016).

- **Algoritmos de aprendizaje supervisado.** Son aquellos que adquieren experiencia de un conjunto de datos x (con m características) pero cada ejemplo está asociado a una **etiqueta** y. Un ejemplo de este tipo de algoritmos sería el mostrado en la figura 4, en el que cada subconjunto de imágenes RGB tiene una etiqueta (CLASE 1, CLASE 2, ..., ó CLASE N). En este caso el algoritmo aprende a clasificar las imágenes etiquetadas de entrenamiento. En la fase de prueba, se evalúa qué tan bien aprendió a etiquetar imágenes con las que no fue entrenado. En términos generales, los algoritmos supervisados involucran la observación de muchos ejemplos de un vector x asociados a un vector de etiquetas y y aprender a predecir y a partir de x usualmente estimando $p(y|x)$.
- **Algoritmos de aprendizaje por refuerzo.** Estos algoritmos interactúan con un ambiente donde existe un ciclo de retroalimentación entre el sistema de aprendizaje y sus experiencias (Goodfellow et al., 2016). Este tipo de algoritmos se encuentran fuera del alcance de este trabajo.

2.1.4. Sobreajuste y subajuste

El principal reto del aprendizaje de máquina es que el algoritmo tenga un buen rendimiento con *datos nuevos* o *ejemplos a los que no ha sido expuesto con anterioridad*. La habilidad de tener un buen rendimiento con entradas x no vistas antes se llama **generalización** (Goodfellow et al., 2016). Usualmente cuando se entrena un modelo de aprendizaje de máquina tenemos acceso a algún conjunto de entrenamiento; a partir de este entrenamiento se puede obtener una métrica de error. Lo interesante será obtener la métrica de error para el conjunto de datos de prueba como se mencionó en la sección anterior (**error de prueba**).

Cuando se hace uso de un algoritmo de aprendizaje de máquina no se fijan los parámetros antes del entrenamiento, estos parámetros de aprendizaje se van ajustando de acuerdo con los datos de entrenamiento. Primero se realiza el entrenamiento del algoritmo para obtener un **modelo entrenado** y se evalúa el **error de entrenamiento**. Posteriormente, se evalúa la tarea del modelo con los datos de prueba y se calcula su respectivo error de prueba. Se espera que el error de prueba sea mayor o igual al error de entrenamiento. Generalmente el error de entrenamiento debe ser pequeño. Sin embargo, un modelo con un buen rendimiento será el que *tenga una distancia pequeña entre el error de entrenamiento y el error de prueba*.

Estos dos factores determinarán si el modelo de aprendizaje presenta sobreajuste (*overfitting*) o subajuste (*underfitting*). El **subajuste** ocurre cuando el modelo no aprendió lo suficiente de los datos del conjunto de entrenamiento, en este caso no se espera que el comportamiento del modelo sea bueno con datos de prueba. El **sobreajuste** ocurre cuando el error obtenido con los datos de prueba es mucho mayor al error de entrenamiento, esto implica que el modelo no aprendió a generalizar para ejemplos de datos nunca vistos antes. Este último modelo no es adecuado para ser lanzado a un entorno donde tenga que actuar sobre datos reales y que será muy propenso a cometer errores.

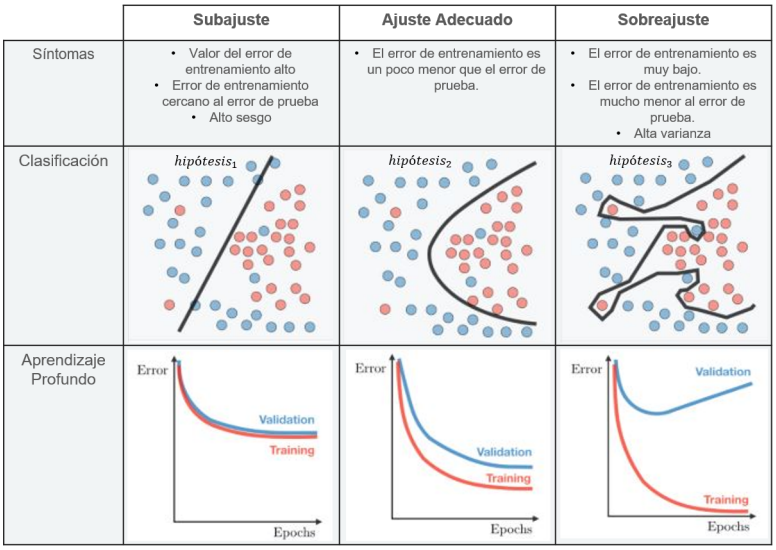

Figura 7. Cuadro comparativo entre subajuste, ajuste adecuado y sobreajuste. En el segundo renglón se observan los ejemplos de hipótesis o modelos de una clasificación binaria para los tres casos. Además se muestran los ejemplos de curvas de aprendizaje para el caso de aprendizaje profundo. Modificado de Shrivastava Abhishek (2020).

En la figura 7 se observan 3 diferentes modelos que resuelven el problema de clasificación binaria mostrado. En este caso, el modelo que generaliza mejor es $hipótesis_2$, aunque no clasifica correctamente todas las muestras, clasifica la mayoría usando un modelo con poca complejidad. El modelo $hipótesis_3$ clasifica correctamente todos los datos de entrenamiento, sin embargo no logrará clasificar bien datos diferentes a los de entrenamiento. Por otro lado, el modelo $hipótesis_1$ es demasiado sencillo para este problema de clasificación, no logra clasificar muchas de las muestras de entrenamiento por lo que se dice que no logró *aprender* a realizar esta tarea de clasificación. El modelo $hipótesis_2$, clasificará mejor datos nunca antes vistos. En cada problema existe un **espacio de hipótesis** o conjunto de posibles modelos para darle solución al problema en cuestión. Los algoritmos de aprendizaje de máquina usualmente tienen mejor comportamiento cuando su complejidad es adecuada respecto al problema. Los modelos con muy poca complejidad (como $hipótesis_1$) serán incapaces de resolver tareas complejas como la de clasificación de señas. Los modelos demasiados complejos (como $hipótesis_3$) tienden a sobreajustarse a los datos de

entrenamiento. Encontrar un modelo adecuado es un problema común en el aprendizaje de máquina.

2.1.5. Construcción de un algoritmo de aprendizaje

Los algoritmos de aprendizaje de máquina se pueden describir como instancias particulares de: la combinación de un dataset, una función de costo, un procedimiento de optimización y un algoritmo. Cada uno de estos componentes son independientes los unos de los otros, por lo que combinando distintos componentes es posible tener una amplia gama de algoritmos.

La función de costo o pérdida típicamente incluye al menos un término que provoca que el proceso de aprendizaje realice la estimación estadística. En el caso de clasificación multi-clase se suele usar la función de **pérdida de entropía cruzada categórica**. En el caso de clasificación de videos o imágenes la función de pérdida es:

$$L(\hat{y}, y) = -\sum_{k}^{K} y^{(k)} \log \hat{y}^{(k)} \tag{3}$$

dónde, y^k es 0 o 1, indicando si la etiqueta de la clase k es la clasificación correcta.

2.2. Aprendizaje profundo

Los algoritmos de aprendizaje de máquina tradicionales han resuelto una gran variedad de problemas importantes. Algunos de estos algoritmos tradicionales se basan en enfoques basados en instancias como K-Nearest Neighbors, enfoques basados en kernels como Support Vector Machines, enfoques basados en árboles de de cisión como Random Forest, entre otros. Sin embargo, han resuelto parcialmente problemas centrales de AI como la traducción de máquina y el reconocimiento de objetos. El desarrollo del aprendizaje profundo está motivado en parte por el hecho de que los algoritmos tradicionales no generalizan bien en ciertas tareas.

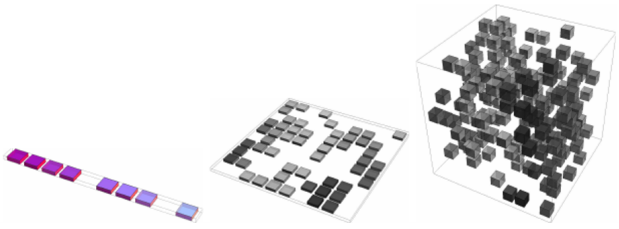

Figura 8. Mientras aumenta el número de dimensiones en los datos (de izquierda a derecha), el número de configuraciones de interés puede crecer exponencialmente. (*Izq*) En este ejemplo unidimensional tenemos una variable para la cuál solo queremos distinguir 10 regiones de interés. (*Centro*) Con 2 dimensiones es más complicado poder distinguir 10 diferentes valores de cada variable. (*Der*) Con 3 dimensiones el espacio aumenta $10^3 = 1,000$ regiones. Figura de Nicolas Chapados tomada de Goodfellow et al. (2016).

Los datos con muchas dimensiones son especialmente complicados por el hecho de que los algoritmos de aprendizaje de máquina son insuficientes para aprender funciones complicadas en espacios de alta dimensionalidad. Muchos problemas de aprendizaje de máquina se vuelven difíciles cuándo el número de dimensiones de los datos es alto, véase figura 8.

El proceso de optimización bajo el cuál funcionan las redes neuronales artificiales se explican en el Anexo A.

2.3. Redes neuronales artificiales

Una red neuronal es una técnica de aprendizaje de máquina que toma cierto número de entradas (*inputs*) y predice salidas (*outputs*). En muchas formas no son diferentes a otros algoritmos de aprendizaje de máquina, sin embargo tienen diferencias significativas que se mencionarán a continuación.

2.3.1. Modelos lineales

Una entrada x (una imagen, video, frase, etc) se representa por un conjunto de características $x_i(x)$. Cada característica se multiplica por un parámetro ω_i para obtener un resultado (y):

$$y(\omega, x) = \sum_j \omega_j x_j(x) \tag{4}$$

De forma gráfica, este modelo lineal se puede ilustrar como una serie de características o nodos de entrada conectados con flechas (que representan los pesos) a un nodo de salida, véase figura 9.

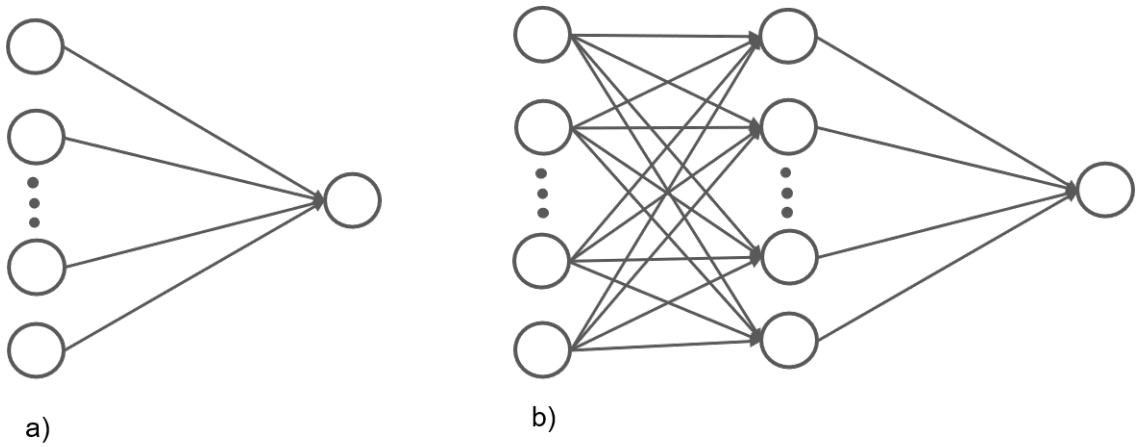

Figura 9. Modelos neuronales. *a)* representación gráfica de un modelo lineal; *b)* modelo con una capa oculta pero varias neuronas.

La salida es hasta ahora una combinación lineal de las entradas. Sin embargo, los modelos lineales no permiten definir relaciones más complejas entre las características.

La implementación de múltiples capas en las redes neuronales usando **capas ocultas**[2] es útil para agregar complejidad al modelo lineal. La red es procesada en l pasos: primero se realiza la combinación lineal de las entradas multiplicadas por los respectivos pesos y se produce un valor de un nodo oculto. Posteriormente, la combinación lineal de los valores de salida de los nodos ocultos recién calculados será la entrada para la siguiente capa. Este procedimiento se repite l veces, donde l es el número de capas ocultas. Suponiendo que $\overline{h} = (h_1, h_2, h_3, ..., h_m)^T$ son los nodos ocultos con m representando el número de nodos o neuronas en dicha capa y $W = \{\omega_{ij}\}$ es la matriz de pesos que conecta los nodos de entrada con los nodos de ocultos y $U = \{u_{ij}\}$. El cálculo en una red neuronal se puede representar como:

$$\begin{aligned} h_i &= \sum_i x_i \omega_{ji} \\ y_k &= \sum_j h_j u_{kj} \end{aligned} \tag{5}$$

2.3.2. No linealidad

Al añadir una capa oculta, la relación de entradas y salidas del modelo sigue siendo lineal. El uso de funciones de activación no lineales en las redes neuronales ayuda a resolver problemas con mayor complejidad que un problema lineal. Después de calcular la combinación lineal de características (x_i) con pesos (ω_{ij}) que se representa como

$$s_j = \sum_i x_i \omega_{ji}, \tag{6}$$

se aplica una función no lineal sobre esta respuesta $f(s_j)$ para obtener el valor del j-ésimo nodo $h_j = f(s_j)$. Una función de activación común es la ReLU (*Rectified Linear Unit*). Esta función no permite valores negativos y los lleva a 0, pero no altera los valores positivos, véase figura 10.

[2] Se les conoce así debido a que se observan sus entradas y salidas pero no los mecanismos que las conectan.

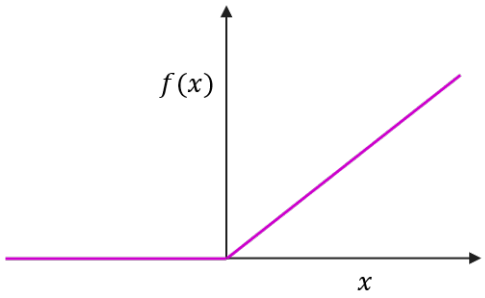

Figura 10. Función de activación ReLU. Se expresa con la siguiente ecuación $ReLU(x) = max(0, x)$ y sus salidas dan el rango de valores $(0, \infty)$

No necesariamente tiene que ser una sola capa oculta. El nombre de **aprendizaje profundo** para redes neuronales viene del hecho de que se pueden obtener mejores métricas de rendimiento apilando más capas ocultas.

Considerando el ejemplo de una red neuronal de la figura 11, los valores de entrada son $x = \{1, 0, 1\}$ mientras que los pesos ω_{ij} son los mostrados en las aristas. Las aristas punteadas representan pesos con valores iguales a 0. Los sesgos b se suman al producto de la combinación lineal para entrar como argumento de la función de activación f que en este ejemplo corresponde a una función ReLU. La tabla 1 muestra los resultados de los nodos ocultos y de la neurona de salida y_0 para este ejemplo. El cálculo de cada nodo se realiza de la siguiente forma:

$$h_i = ReLU(\sum_i x_j \omega_{ij} + b_i) \tag{7}$$

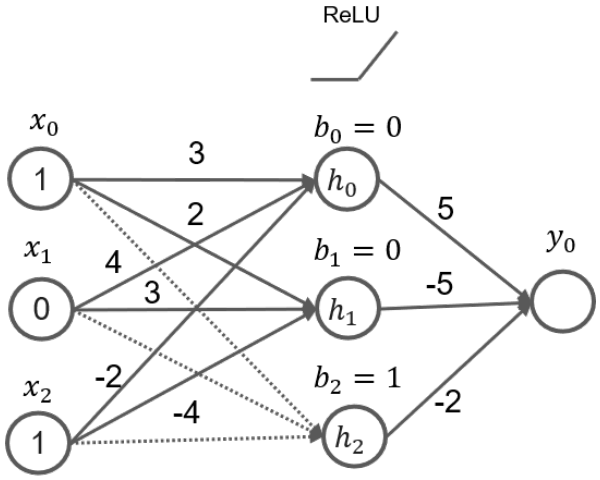

Figura 11. Ejemplo de una red neuronal simple con una capa oculta. Los valores b representan los **sesgos** que se suman al valor de la suma de la combinación lineal entre pesos y entradas para ser pasados por la función de activación, en este caso una ReLU. Modificada de Koehn (2017).

Tabla 1. Cálculos de los nodos ocultos y de salida de la red neuronal de la figura 11 para las entradas $x = \{1, 0, 1\}$.

Capa	Nodos h	Suma	Activación ReLU
Oculta	h_0	$1 \cdot 3 + 0 \cdot 4 + 1 \cdot (-2) + 0$	1
Oculta	h_1	$1 \cdot 2 + 0 \cdot 3 + 1 \cdot (-4) + 0$	0
Oculta	h_2	$1 \cdot 0 + 0 \cdot 0 + 1 \cdot (0) + 1$	1
Salida	y_0	$1 \cdot 5 + 0 \cdot (-5) + 1 \cdot (-2)$	0

2.3.3. Retropropagación

El entrenamiento de redes neuronales requiere de la optimización de los valores de los pesos de tal forma que la red obtenga la predicción de la salida correcta para un conjunto de entrenamiento. Esto se logra alimentando a la red repetidamente con los ejemplos de entrenamiento y actualizando los pesos (Koehn, 2017). Usualmente este proceso de actualización de pesos se hace durante varias iteraciones que en este contexto son conocidas como **épocas**.

La retropropagación es el método de entrenamiento más común. Primero actualiza los pesos de la capa de salida y propaga el error a las capas anteriores. El gradiente a un menor error (δ_i) es lo que actualiza los pesos de la red neuronal. Para los nodos de salida, el término de error δ_i se calcula a partir de la salida actual y_i del nodo y la salida objetivo que corresponde a la etiqueta actual del nodo (t_i):

$$\delta_i = (t_i - y_i) y_i' \tag{8}$$

dónde y_i' es la derivada de la función de activación. Para los nodos ocultos el término de error δ_j se calcula retropropagando el término de error δ_i para los nodos subsecuentes conectados por pesos ω_{ij}

$$\delta_j = \sum (\delta_i \omega_{ij}) \tag{9}$$

con h_j' como la derivada de la función de activación correspondiente a dicha capa oculta. Dados los términos de error, los pesos ω_{ij} de cada nodo h_j son actualizados usando un paso o tasa de aprendizaje μ.

$$\Delta\omega_{ij} = \mu \delta_i h_j \tag{10}$$

Una vez actualizados los pesos, se procesa el siguiente ejemplo de entrada hasta que se terminan por procesar todos los ejemplos de entrenamiento. Este proceso se realiza por varias épocas.

2.3.4. Hiperparámetros

Existen diversos problemas al entrenar redes neuronales ya que si no se seleccionan correctamente ciertos parámetros es posible que el descenso por gradiente no converja a un error lo suficientemente pequeño por lo que el modelo entrenado final puede no dar los resultados deseados. Esto puede resultar en dos casos: *subajuste* o *sobreajuste*. Un subajuste del modelo implica que los parámetros de aprendizaje (pesos y sesgos) no aprendieron suficiente de los datos de entrenamiento como se mencionó en la sección 2.1.4. En el caso del sobreajuste, la red neuronal puede no estar generalizando para ejemplos distintos a los que fue expuesta al entrenar, por lo que se debe recurrir a técnicas de regularización para minimizar este fenómeno. La selección de estos parámetros (mejor conocidos como **hiperparámetros**) es por lo tanto importante para obtener un modelo adecuado. A continuación se enlistan los hiperparámetros que se deben elegir:

- **Número de neuronas por capa.** Usualmente es el mismo en cada capa, este parámetro debe ser ajustado conforme a la complejidad del problema. Una tarea con mayor complejidad (por ejemplo clasificación de series de tiempo) necesita más neuronas para obtener predicciones con buenos resultados. Usualmente se eligen números de neuronas que son potencias de dos: 2^m.
- **Número de capas ocultas.** Este parámetro también depende de la complejidad del problema y del tipo de red neuronal. Por ejemplo existen redes pre-entrenadas del estado del arte que tienen más de 50 capas ocultas, estas son ideales para resolver problemas complejos como el de la clasificación de imágenes. Mientras que un problema sencillo como una regresión lineal simple puede ser resuelto solo con un par de capas ocultas.
- **Tipo de funciones de activación por capa.** Este es un parámetro por capa oculta. La capa de salida contiene el valor de salida, que corresponde a los posibles valores de clase, por ejemplo las señas. Las funciones de activación suelen ser las mismas entre capas exceptuando la penúltima capa. La función de activación de la penúltima capa suele ser una función sigmoide para el problema de clasificación bi-clase o una **softmax** para la clasificación multiclase ó en el caso de regresión no es necesaria. La función softmax extiende la idea de la regresión logística (sigmoide) y asigna probabilidades decimales a cada neurona de la última capa que corresponden a las clases, véase figura 12.

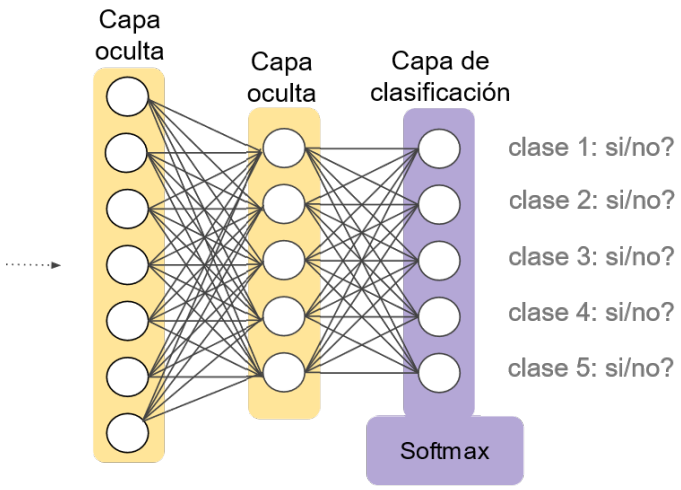

Figura 12. Interpretación gráfica de la capa Softmax en un ejemplo multiclase. Modificada de Google's Machine Learning Crash Course (2020).

- **Optimizador.** El optimizador es el algoritmo responsable de cambiar la tasa de aprendizaje y encontrar al menos algún mínimo local lo suficientemente bueno para obtener buenos resultados. Existen diversos algoritmos que son variaciones del descenso por graidente y del SGD. En la figura 13, se observa la comparación de estos algoritmos y su rendimiento en función del valor de su función de costo por épocas para el problema de clasificación de dígitos MNIST. El problema de MNIST consiste en clasificar una serie de fotografías de dígitos de 0 al 9 escritos a mano; véase Deng (2012). Nótese que para este problema, el algortimo Adam tiene mejores resultados. Usualmente se deben probar diversos algoritmos de optimización y quedarse con el que se obtengan los mejores resultados (método de prueba y error).

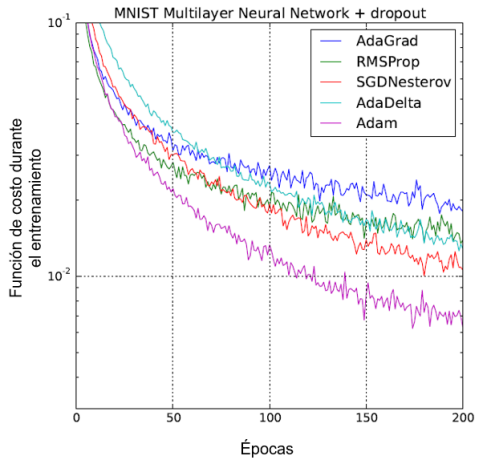

Figura 13. Comparación del valor de la función de costo/pérdida usando diferentes algoritmos de optimización para resolver el problema de MNIST usando una red neuronal profunda. Modificada de Kingma y Ba (2014).

- **Tasa de aprendizaje μ.** Este hiperparámetro controla el tamaño del paso del optimizador. En la

figura 14 se observa que este parámetro es importante para asegurar una buena convergencia del algoritmo de descenso por gradiente. Nótese que la tasa de aprendizaje debe ser lo suficientemente pequeña para poder encontrar un mínimo pero lo suficientemente grande para que encuentre una solución en el menor número de épocas posible.

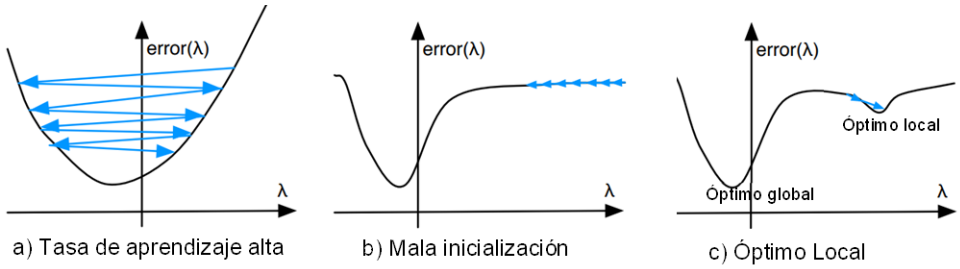

Figura 14. Problemas con el algoritmo de descenso por gradiente. *a)* una tasa de aprendizaje muy alta puede no converger en el óptimo. *b)* una mala inicialización de los pesos provoca que la convergencia sea lenta si la tasa de aprendizaje es pequeña. *c)* la existencia de óptimos locales pueden sesgar el resultado y consecuentemente el algoritmo de descenso por gradiente no encontrará el óptimo global. Tomada de Koehn (2017).

- **Función de pérdida.** Es usada para evaluar la solución candidata, i.e., conjunto de pesos. Esto implica que se busca una solución con la cuál se obtenga el mínimo posible de la función de pérdida/costo. La función de entropía cruzada es usada en problemas de clasificación.

- **Épocas.** El número de iteraciones que la red neuronal debe hacer sobre el conjunto de muestras de entrenamiento. Es necesario considerar suficientes épocas en la red neuronal para permitir que la solución candidata de pesos cambie. Pocas épocas de entrenamiento pueden provocar que el modelo subajuste.

- **Tamaño del lote.** Se podría pensar que la red neuronal actualiza los pesos y sesgos cada que 'entrena' con una muestra del conjunto de entrenamiento. Sin embargo, esto no sucede debido a que si cada ejemplo de entrenamiento es distinto entre sí, la red neuronal cambiará radicalmente los valores de los pesos en cada iteración. Este comportamiento no es el más adecuado. El tamaño de lote, también conocido como *batch size*, se refiere a la cardinalidad del sub-conjunto de lote de muestras de entrenamiento con las cuales se actualizarán los pesos. Un tamaño de lote grande ayuda a la generalización del modelo.

- **Inicialización de los pesos.** Este es un factor relevante al considerar el entrenamiento de la red neuronal. Como se mostró en la figura 14 cuando la inicialización de la solución posible (pesos y sesgos) se encuentra lejos de un mínimo local o el mínimo global es posible que la convergencia sea lenta. Los pesos suelen inicializarse los pesos en valores de 0.

2.3.5. Regularización

Existen algunas técnicas que disminuyen la complejidad de los modelos de redes neuronales durante el entrenamiento, i.e., previenen el posible sobreajuste. Algunas técnicas de regularización populares son el dropout y la normalización de capa. A continuación se explicarán estas herramientas con mayor detalle.

2.3.6. Dropout

El método de dropout es útil para evitar el estancamiento de la solución en el algoritmo de optimización de los mínimos locales. Durante el entrenamiento algunos nodos o neuronas de la red son ignorados, es decir que sus valores son fijados a 0 y sus parámetros asociados no son actualizados, véase figura 15. Estos nodos 'olvidados' se seleccionan aleatoriamente y pueden representar desde el $10\,\%$ o más neuronas de la capa oculta. El resultado final es un modelo más robusto (Koehn, 2017).

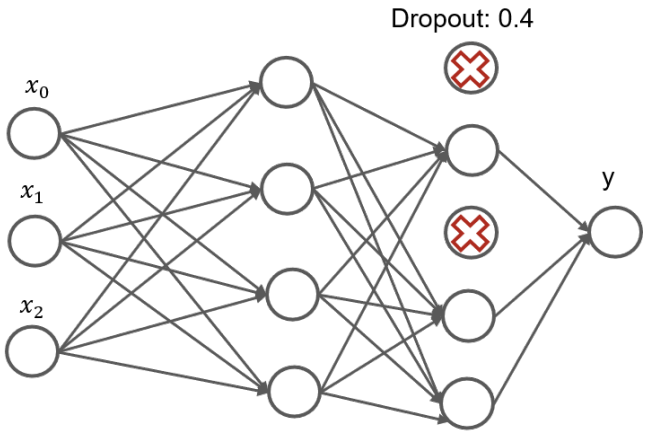

Figura 15. Red Neuronal con dropout en la última capa oculta del $40\,\%$.

2.3.7. Normalización por lote (Batch-Normalization)

El entrenamiento de una red neuronal es complejo cuando el número de capas ocultas aumenta. Esto debido a que el modelo es actualizado capa por capa de atrás hacia adelante (de la salida a la entrada) usando un error estimado que asume que los pesos en las capas anteriores son fijos. Dado que todas las capas cambian durante la actualización, el proceso de actualización está recurrentemente persiguiendo un objetivo móvil (Brownlee Jason, 2019).

La normalización por lote se propuso para ayudar a coordinar la actualización de múltiples capas en el modelo. Esto lo logra escalando la salida de la capa, específicamente estandarizando la activaciones

de cada variable de entrada por mini-lote así como las activaciones de un nodo de la capa anterior. Esto estabiliza y aumenta la velocidad del proceso de entrenamiento de las redes neuronales profundas.

2.4. Costo computacional

La ejecución de todas las operaciones necesarias para entrenar una red neuronal es computacionalmente costosa. Este tipo de operaciones matriciales son usadas también en el área de procesamiento de gráficos. Cuando las imágenes se renderizan en la pantalla las propiedades geométricas de objetos 3-D tienen que ser procesadas para generar los valores de color de la imagen 2-D en la pantalla (Koehn, 2017). Aprovechando la alta demanda de procesamiento de gráficos de alta velocidad, e.g., su uso para crear juegos de computadora hiper-realistas, el hardware de Unidades de Procesamiento Gráfico (GPU's) se ha abaratado con el paso de los últimos años.

El término general para escalar, vectores y matrices es el de tensores. Estos procesadores tienen un número masivo de núcleos. Los GPU's trabajan en paralelo, a comparación de un CPU y son excelentes opciones para operaciones tensoriales. Existen librerías como CUDA para C++ (y sus respectivos parches para Python) para programar en el GPU.

Existen otras herramientas en la nube que nos permiten tomar prestados recursos de GPU's más poderosas que las que se podrían tener en una computadora personal. Colaboratory, mejor conocido como **Colab**, es un producto de Google Research que permite a cualquier usuario con una cuenta de Google escribir y ejecutar código de Python (mediante Jupyter Notebooks) en el navegador. Los principales beneficios de usar Colab es que no requiere instalación de librerías u software adicional y brinda acceso a recursos computacionales como mayor RAM y GPU's.

2.4.1. Frameworks de aprendizaje profundo

Existen diversas herramientas para crear redes neuronales para problemas concretos. Al tiempo de escritura de este trabajo, los más prominentes son **pyTorch** (variante de Torch para Python) y **Keras-Tensorflow** (herramienta de Google Research). Estas herramientas proveen implementaciones eficientes de operaciones vectoriales y el cálculo de derivadas con soporte de GPU's. En particular **Keras-Tensorflow** es la API de alto nivel de Tensorflow para construir y entrenar modelos de aprendizaje profundo. Es especialmente amigable con el usuario y tiene la ventaja de ser modular y contiene arquitecturas del estado del arte, véase Abadi et al. (2015).

2.5. Arquitecturas

En la sección anterior se ha explicado cómo funciona una red neuronal con una capa oculta y además el funcionamiento de una red con más de una capa oculta. Las redes neuronales profundas suelen tener más de una capa oculta conectadas a la última capa de decisión (conocida como capa de clasificación). Existen arquitecturas del estado del arte que tienen más de 50 capas ocultas. En esta sección se explicarán a detalle algunos tipos de red neuronales y su funcionamiento.

2.5.1. Perceptrón multicapa

Un perceptrón multicapa (**MLP**) constituye la arquitectura más sencilla y tradicional de los modelos de aprendizaje profundo. También es conocida como fully-connected (FC) dado que las neuronas en la capa l_i están conectadas a cada neurona en la capa l_{i-1} con $i \in [1, L]$. Estas conexiones son modeladas por los pesos en una red neuronal. Este tipo de redes son la generalización de las redes explicadas en la sección 2.3.3, véase figura 16.

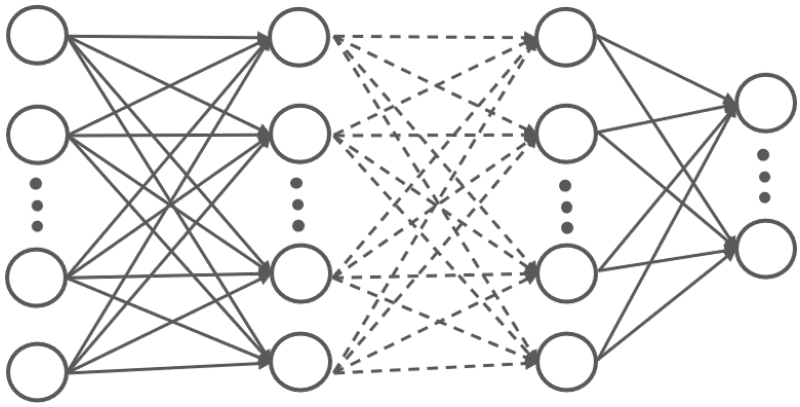

Figura 16. Ilustración de un perceptrón multicapa. Las líneas punteadas indican que puede tener más capas ocultas.

2.5.2. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN) son un tipo de redes neuronales especiales para datos de entrada tipo malla o red (LeCun et al., 1989). Este tipo de datos pueden ser series de tiempo, imágenes, video, etc. Las redes neuronales convolucionales han sido exitosas en aplicaciones prácticas como la clasificación de imágenes, video y series de tiempo. Como su nombre lo indica, estas redes realizan la operación de convolución en lugar de la multiplicación de matrices en al menos una de sus capas.

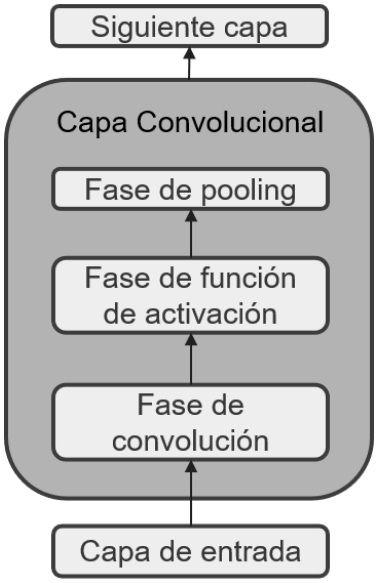

Figura 17. Operaciones dentro de una capa convolucional.Tomada de Goodfellow et al. (2016).

Una capa convolucional consiste de tres pasos, véase figura 17:

- En la primera fase, se realizan una serie de operaciones convolucionales en paralelo para producir un conjunto de activaciones lineales.
- En la segunda fase, se aplica una función no lineal a cada activación lineal.
- En la tercera fase se aplica una función de **pooling** para modificar la salida de la capa.

2.5.2.1. Operación de convolución

La convolución es una operación lineal que permite combinar dos conjuntos de información, en las CNN se aplica sobre los datos de entrada para filtrar la información y producir un mapa de características o **mapa de rasgos**. La convolución de dos conjuntos $x(t)$ y $\omega(t)$ se representa como:

$$s(t) = x(t) * \omega(t) = \sum_{\alpha=-\infty}^{\infty} x(\alpha)\omega(t-\alpha) \tag{11}$$

Donde $x(t)$ es el arreglo tensorial de entrada, $\omega(t)$ en este caso se conoce como kernel o filtro, la salida s es el mapa de rasgos. Si la entrada es una imagen o un arreglo con dos dimensiones, el kernel ω también debe tener 2 dimensiones:

$$s(i,j) = \sum_{m}\sum_{n} x(i-m, j-n)\omega(m,n) \tag{12}$$

La operación de convolución tiene ciertas características importantes que ayudan a mejorar un sistema de aprendizaje, una de ellas son las interacciones dispersas. Esto se realiza con un kernel mucho menor al tamaño de los arreglos de entrada. Usando un kernel pequeño se puede sintetizar la información de un arreglo de entrada grande (por ejemplo de una imagen se puede obtener información relevante como las esquinas y contornos).

2.5.2.2. Pooling

Esta operación reemplaza la salida de la capa de la red con un resumen de la salida (Goodfellow et al., 2016). La operación *max pooling* reporta la salida máxima de un vecindario rectangular de la salida. Por otro lado, la función *average pooling* obtiene el promedio de un vecindario rectangular. Las operaciones de pooling permiten que las representaciones de las salidas sean invariante a pequeñas variaciones de traslación en las entradas. Esto quiere decir que si la entrada sufre una traslación, los valores de las salidas de la función pooling no cambian.

2.5.3. Redes neuronales recurrentes

Las redes neuronales recurrentes (RNN) son una familia de redes neuronales usadas para procesar datos secuenciales $x^{(1)}, ..., x^{(t)}$. Al igual que las redes neuronales fully-connected y las redes convolucionales, las redes recurrentes utilizan los datos de entrada para entrenar. Estas redes usan su *memoria* ya que aprovechan la información de entradas de tiempos anteriores para predecir la salida actual. A diferencia de las redes neuronales fully-connected que asumen que las entradas y las salidas son independientes, las redes recurrentes dependen de elementos anteriores de la secuencia, véase figura 18.

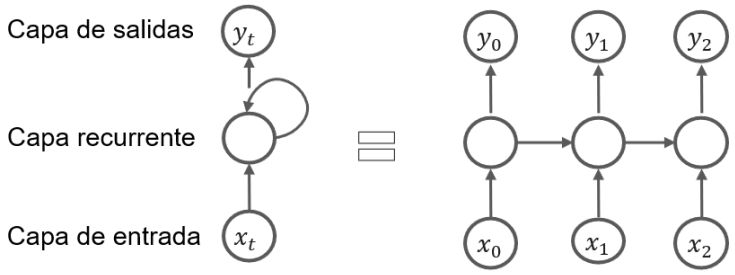

Figura 18. Representación gráfica de una capa recurrente.

2.6. OpenPose

OpenPose es el primer sistema multi-persona en tiempo real que detecta las articulaciones del cuerpo humano, el rostro, las manos y los pies en imágenes. El sistema puede detectar hasta 135 puntos clave

del cuerpo humano en total. La inferencia de múltiples poses (conjuntos de articulaciones del cuerpo humano) en imágenes presenta retos importantes, entre ellos el tiempo de ejecución aumenta en función del número de personas que deben ser detectadas en la imagen.

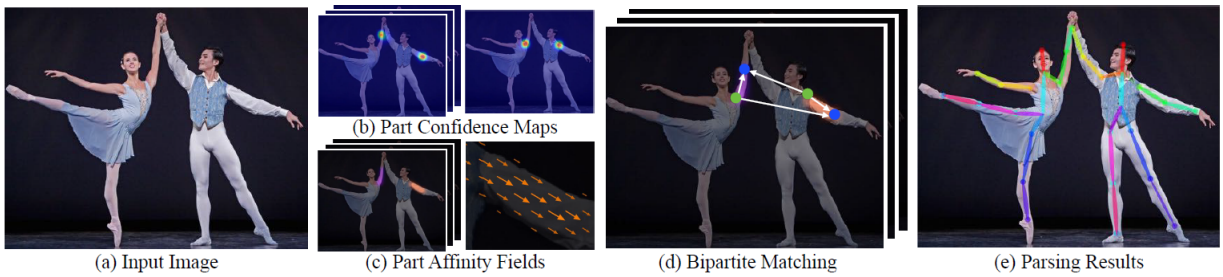

Figura 19. Flujo del funcionamiento de OpenPose. *a*) OpenPose toma la imagen completa como entrada para la CNN para ser predicha. *b*) Mapas de confianza para la detección de partes del cuerpo. *c*) PAFs para la asociación de partes del cuerpo. *d*) Paso de análisis entre conjuntos de coincidencias bipartitas para asociar partes del cuerpo candidatas. Tomada de Cao et al. (2019).

Cao et al. (2019) presentaron un método eficiente para estimar la pose del esqueleto humano para imágenes con más de una persona con un costo computacional competitivo en comparación con otros métodos. OpenPose usa un esquema de anotaciones de asociación mediante Part Affinity Fields (PAFs)[3], véase figura 19.

OpenPose es una librería de código abierto disponible para la detección de poses. Además, comparan su rendimiento con otros sistemas como Mask R-CNN y AlphaPose demostrando que su enfoque tiene una ventaja computacional ante ellos.

En la figura 19 se observa el flujo del funcionamiento del método que usa la librería OpenPose. El sistema toma una entrada a color, una imagen de $w \times h$ y produce las ubicaciones 2-D de los puntos clave para cada persona en la imagen. Primero, una red neuronal predice el conjunto de mapas de confianza 2-D (S) de las ubicaciones de las partes del cuerpo y un conjunto de campos vectoriales 2-D (L) de PAFs que codifican el grado de asociación entre partes. El conjunto $S = (S_1, S_2, ..., S_j)$ tiene j mapas de confianza donde $S_j \in \mathrm{Re}^{\omega x h}$, $j \in 1, ..., J$. El conjunto $L = (L_1, L_2, ..., L_C)$ tiene C campos vectoriales, uno por extremidad[4]. Cada ubicación de la imagen en L_C codifica un vector 2-D. Finalmente, los mapas de confianza y los PAFs son analizados para dar como salida puntos clave 2-D para todas las personas en la imagen.

[3] Los PAFs son un conjunto 2D de campos vectoriales que codifican la ubicación y orientación de extremidades sobre el dominio de la imagen

[4] con extremidad los autores Cao et al. (2017) se refieren a pares de partes incluso cuando algunos pares no son extremidades humanas (por ejemplo el rostro)

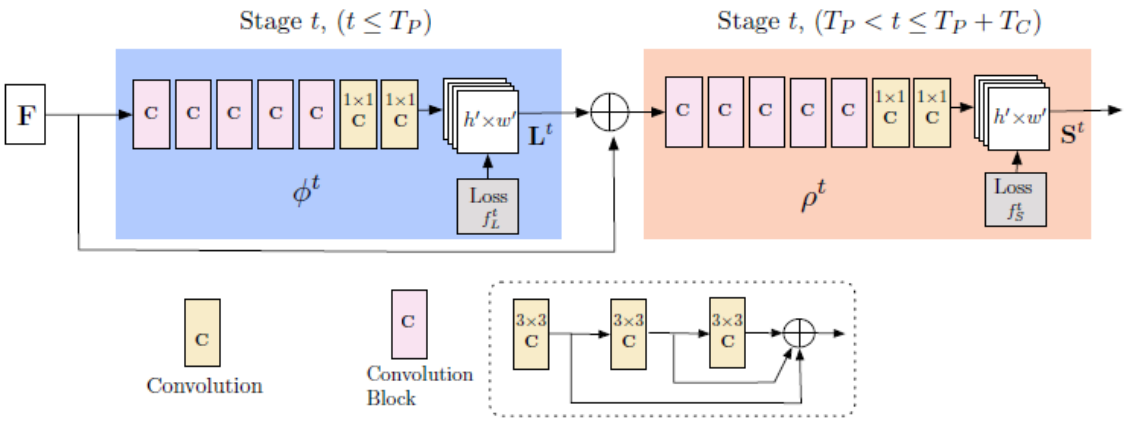

Figura 20. Arquitectura de la CNN multi-fase. La primer fase predice PAFs L^t, mientras que el último conjunto predice mapas de confianza S^t. Las predicciones en cada fase y sus correspondientes características de imágenes son concatenadas para la siguiente etapa. Se realizan convoluciones con un kernel de tamaño 3. Tomada de Cao et al. (2019).

La arquitectura de la red neuronal encargada de realizar este reconocimiento de puntos clave es la mostrada en la figura 20. Esta arquitectura predice iterativamente campos de afinidad que codifican asociaciones parte por parte (mostrada en color azul) y detecta los mapas de confianza (mostrado en color beige).

El sistema OpenPose fue creado para ayudar a la comunidad con proyectos de investigación del movimiento del cuerpo humano. Este sistema puede correr en distintas plataformas incluyendo Ubuntu, Windows, Mac OSX y sistemas embedidos (por ejemplo Nvidia Tegra TX2). Además provee soporte para diversos tipos de hardware como GPUs de CUDA. El usuario puede elegir una imagen de entrada o elegir la captura de las imágenes vía cámara. Además, el usuario puede seleccionar cada detector (cuerpo, pies, rostro y manos) y controlar el número de GPUs a usar entre otras cosas.

OpenPose detecta 3 diferentes bloques: cuerpo + pies, manos y rostro. El bloque núcleo es el detector de cuerpo + pies. De forma alternativa, se puede seleccionar si usar los modelos de cuerpo originales entrenados con los datasets de COCO y MPII. En total el sistema puede detectar 21 puntos claves para cada mano (representando los falanges), 21 del cuerpo completo (incluyendo ojos, nariz y pies) y 67 puntos clave del rostro.

En la figura 21 se observa la detección del cuerpo humano para tronco superior y manos. Como se mencionó anteriormente, OpenPose recibe como entrada una imagen y da como salida las coordenadas 2D de los puntos clave del cuerpo humano. En el caso de la figura 21, se observa un cuadro (frame) de dos personas realizando una seña usando el torso superior.

a) Seña: Doctor a) Seña: Mareado

Figura 21. Detección de puntos clave del cuerpo humano para tronco superior y manos. *a)* Frame de la seña "doctor", en este caso existe oclusión de las manos detectadas. *b)* Frame de la seña "mareado", no se observa oclusión de los puntos clave.

2.7. Clasificación de series de tiempo

La clasificación de series de tiempo (**CST**) es uno de los problemas más importantes y que conlleva más retos en la minería de datos (Yang y Wu, 2006). Con el aumento de los datos temporales y su disponibilidad, se han propuesto cientos de algoritmos para resolver el problema de CST (Bagnall et al., 2017).

2.7.1. Modelos de aprendizaje de máquina

Uno de los enfoques más populares para llevar a cabo el problema de CST es es el uso del algoritmo de clasificación de vecinos cercanos (**KNN**) acoplado con una función de distancia (Lines y Bagnall, 2015). En particular, la distancia Dynamic Time Warping (**DTW**) usada en el algoritmo clasificador de KNN ha demostrado obtener resultados comparables a otros algoritmos del estado del arte (Bagnall et al., 2017). Lines y Bagnall (2015) compararon varias medidas de distancia y demostraron que ninguna otra métrica de distancia mejora los resultados obtenidos con DTW. Además demostraron que ensamblando los clasificadores KNN individuales se obtienen mejores resultados. Estos enfoques usan ya sea ensambles o árboles de decisión (random forest).

Bagnall et al. (2015) desarrollaron un ensamble de 35 clasificadores llamado COTE (*Collective Of Transformation-based Ensembles*). Este ensamble no permite tener varios clasificadores sobre la misma transformación, ensambla diferentes clasificadores sobre diferentes representaciones de series de tiempo. Lines et al. (2016) extendieron COTE con un sistema de voto jerárquico (*Hierarchical Vote*), esta modificación es conocida como HIVE-COTE y ha demostrado alcanzar mejoras significativas sobre COTE.

HIVE-COTE es actualmente considerado el algoritmo del estado del arte en clasificación de series de tiempo (Bagnall et al., 2017). Para que HIVE-COTE logre alcanzar su alta exactitud se vuelve computacionalmente muy costoso, a tal grado que es impráctico para ser usado en problemas reales (Bagnall et al., 2017). Este enfoque requiere entrenar 37 clasificadores y además realizar una validación cruzada de cada hiperparámetro de estos algoritmos, esto hace que este enfoque sea impráctico para entrenar con datos de alta dimensión. Uno de los 37 clasificadores es la Transformada Shapelet cuya complejidad es $O(n^2 \cdot l^4)$ con n siendo el número de series de tiempo en el conjunto de datos y l la longitud de cada serie de tiempo. Además de la complejidad del tiempo de entrenamiento, también está la alta complejidad del tiempo de clasificación, el clasificador de vecinos cercanos necesita revisar en conjunto de entrenamiento antes de tomar una decisión en el momento de obtener una clasificación de prueba. Dado que KNN es un componente esencial de HIVE-COTE su implementación para evaluar en tiempo real es impráctico.

2.7.2. Series de tiempo

A continuación se presentan algunas definiciones para entender el problema de CST.

- Una serie de tiempo univariada $X = [x_1, x_2, ..., x_l]$ es un conjunto ordenado de valores reales. La longitud de X es igual al número de valores l.
- Una serie de tiempo multi-dimensional $X = [X^1, X^2, ..., X^M]$ consiste de M series de tiempo univariadas con $X^i \in \mathrm{Re}^l$
- Un conjunto de datos $D = \{(X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)\}$ es una colección de pares (X_i, Y_i) donde X_i puede ser una serie de tiempo univariada o multi-variada con Y_i como su etiqueta en codificación one-hot[5]. Para un conjunto de datos que contiene K clases, el vector one-hot Y_i es un vector de longitud K donde cada elemento $j \in [1, K]$ es igual a 1 si la clase de X_i es j y 0 en otro caso.

La tarea de CST consiste en entrenar un clasificador con un conjunto de datos D de tal forma que se pueda mapear del espacio de posibles entradas a una distribución de probabilidad sobre los valores de clase (etiquetas).

[5]La codificación One-Hot es un método para etiquetar a qué clase pertenecen los datos y la idea es asignar 0 a toda la dimensión, excepto 1 para la clase a la que pertenecen los datos.

2.7.3. Modelos de aprendizaje profundo

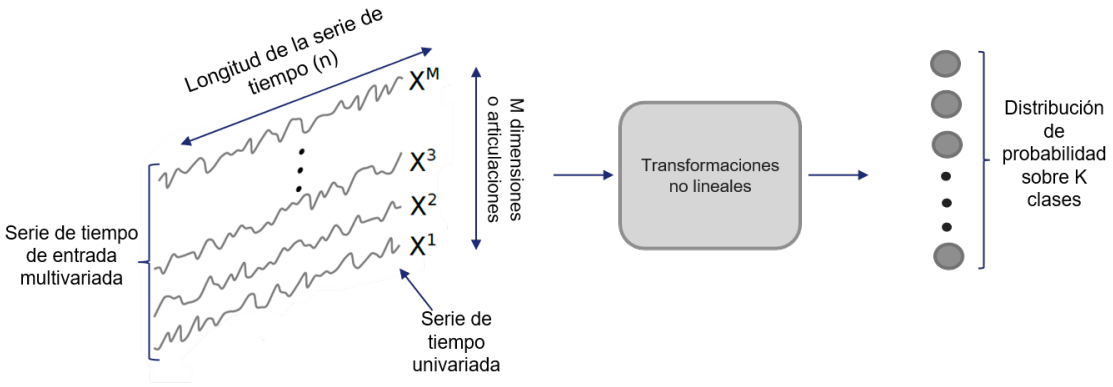

Figura 22. Representación gráfica de un sistema de clasificación de series de tiempo que se basa en aprendizaje profundo.

En la subsección 2.7.1 se estableció el estado del arte actual de clasificadores de series de tiempo que no usan aprendizaje profundo. El éxito del aprendizaje profundo en múltiples tareas de clasificación han motivado a usar este tipo de herramientas para resolver problemas de CST. En la figura 22 se observa un sistema de clasificación de series de tiempo.

Los enfoques de aprendizaje profundo para CST pueden separarse en dos categorías principales: modelos *generativos* y modelos *discriminativos*, véase figura 23. En este trabajo nos concentraremos en los modelos discriminativos ya que de acuerdo con Bagnall et al. (2017) el consenso informal en la literatura es que los modelos generativos usualmente son menos precisos y más complicados que los modelos discriminativos.

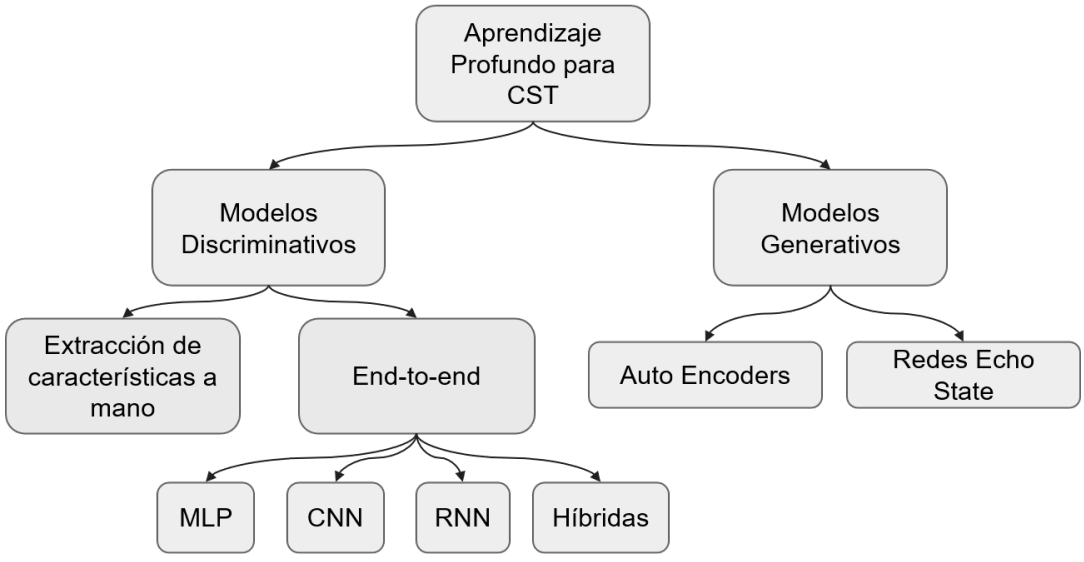

Figura 23. Resumen de los diferentes enfoques de aprendizaje profundo para resolver el problema de clasificación de series de tiempo. Modificada de Ismail Fawaz et al. (2019).

2.7.3.1. Modelos discriminativos

Un modelo discriminativo en aprendizaje profundo es un clasificador que aprende directamente el mapeo entre las entradas crudas, o con muy poco pre-procesamiento, de una serie de tiempo y entrega como salida una distribución de probabilidad sobre las clases del conjunto de datos. Se han propuesto muchas arquitecturas discriminativas en la tarea de CST, este modelo puede ser subdividido en dos grupos: *modelos de aprendizaje con características extraídas a mano* y *modelos end-to-end*. Para los fines de este trabajo, nos concentraremos en los modelos end-to-end.

Los modelos *end-to-end* incorporan el proceso de aprendizaje de características y el proceso de clasificación (Nweke et al., 2018). Wang et al. (2017) diseñaron una red neuronal multi-capa (MLP), el problema de este modelo es que la información temporal se pierde y las características aprendidas no son invariantes en el tiempo. Usualmente los modelos CNN son mejores ya que estos aprenden filtros invariantes en el espacio (características) de las series de tiempo crudas (Wang et al., 2017).

Ismail Fawaz et al. (2019) concluyeron que las CNN son las redes más usadas para el problema de CST debido a su robustez y relativamente bajo tiempo de entrenamiento comparadas con arquitecturas RNN o MLP. Se han propuesto muchas variantes de CNNs tales como las Redes Residuales (ResNets) que tienen la particularidad de añadir atajos lineales en forma de conexión para las capas convolucionales potencialemnte mejorando la exactitud de los modelos (He et al., 2016).

2.8. Lengua de Señas Mexicana

La Lengua de Señas Mexicana está compuesta por la dactilología y los ideogramas. Se conoce como **dactilología** al equivalente al deletreo en la lengua oral, y está representada con el abecedario (De Fleischmann y Pérez, 2011). Cada palabra se puede representar con la articulación de mano correspondiente de cada letra que la conforma. De acuerdo con esto, la palabra mamá puede representarse con cada una de sus letras en señas (M-A-M-A). Sin embargo esto no es óptimo para el señista, es por ello que existen los ideogramas. Los **ideogramas** representan una palabra con una o varias configuraciones de mano. En este caso entra casi todo el resto de las palabras existentes. En la figura 24 se muestra un ejemplo de un ideograma (de la seña AMIGO) y del abecedario en LSM que forma la dactilología.

Figura 24. Ejemplos de Ideogramas y Dactilología. *a)* Ejemplo de ideograma como secuencia de imágenes. *b)* Ejemplo del abecedario de señas con el que se forma la datilología, tomado de Morales et al. (2019).

Los ideogramas tienen las siguientes características:

- **Queirema.** Esta característica se refiere a la la forma de la mano al realizar la seña.
- **Toponema.** Se refiere a la posición de la mano en el marco de referencia del cuerpo donde se ejecutará la seña.
- **Kinema.** Se refiere al movimiento y la trayectoria de desplazamiento de la mano para interpretar la seña.

En la figura 25, se observan dos ideogramas correspondientes a las señas de DÍA y DIABETES. Ambas señas comparte queirema, es decir la misma colocación de a mano (en forma de *D*). Sin embargo en ambas señas tienen una orientación, posición respecto al cuerpo y trayectoria de movimiento distintas.

Figura 25. Características de los ideogramas. *a)* Queirema de la mano de la letra **D**. *b)* Ejemplo de ideograma (seña: DÍA). *c)* Ejemplo de ideograma (seña: DIABETES).

Otra característica de la Lengua de Señas Mexicana es que no guarda relación con el Español habado ni escrito. La principal diferencias entre la LSM y la lengua oral es en su estructura. Esto debido a que la LSM es ágrafa, no se produce (emisión vocal) ni se percibe (atención auditiva) como las lenguas orales (Pichardo-Lagunas, 2016). La razón detrás de esta diferencia estructural es el hecho de que las Lenguas de Señas basan su funcionamiento en la percepción visual, dado que involucran formas y conceptos mientras que los lenguajes escritos involucran reglas gramaticales.

Capítulo 3. Trabajo relacionado

Existe trabajo previo en ambas direcciones de traducción para distintas Lenguas de Señas, ver figura 1. Por un lado, la traducción de un lenguaje hablado a una Lengua de Señas involucra convertir palabras/letras o frases a gestos usando tecnología de avatar u otras representaciones gráficas (Kapoor et al., 2021) ; (Stoll et al., 2018) ; (Stoll et al., 2020) ; (Saunders et al., 2021). Por otro lado, para la traducción de Lengua de Señas a lenguaje hablado, existen enfoques de reconocimiento de señas propuestos en la literatura (Abraham et al., 2019) ; (Camgoz et al., 2018) ; (Orbay y Akarun, 2020) ; (Harini et al., 2020) ; (Guo et al., 2018) ; (Yin, 2020) ; (Camgoz et al., 2020) ; (Antony et al., 2020). Muchos de estos utilizan hardware especializado (Ko et al., 2018) ; (Hisham y Hamouda, 2021) ; (Ko et al., 2019) . Otros enfoques se concentran en el desarrollo de software y mejora en técnicas de visión por computadora (Koller et al., 2018) ; (De Coster et al., 2020) ; (Lakhotiya et al., 2021) ; (Bantupalli y Xie, 2018) ; (Vedak et al., 2019) ; (Sonare et al., 2021) ; (Idoko, 2020) ; (Fernandes et al., 2020). Para facilitar la normalización de las Lenguas de Señas, al igual que para evaluar diferentes algoritmos de traducción, algunos investigadores han creado repositorios que incluyen gestos en diccionarios de palabras (Koller et al., 2018) ; (Pichardo-Lagunas, 2015) ; (De Coster et al., 2019). Estos diccionarios están conformados con distintos formatos dependiendo del uso previsto.

En este Capítulo se presenta un estudio comprensivo de aspectos relacionados al problema de traducción de Lenguas de Señas. Se presentarán los enfoques de reconocimiento de señas del estado del arte y se revisarán los repositorios disponibles de *corpora* de distintas Lenguas de Señas. Por último, se discute sobre el trabajo relacionado en torno a la LSM.

3.1. Metodología de revisión

Para seleccionar los documentos más relevantes como trabajo previo al propuesto se realizó un proceso de búsqueda, descarte y lectura de documentación. El primer paso consistió en la recolección de referencias bibliográficas usando tres motores de búsqueda: SpringerLink, IEEE Xplore y Google Scholar. Se recolectaron 270 referencias perteneciendo principalmente a las categorías de artículos de revista, artículos de conferencia y reseñas. Las tesis de maestría se omitieron por considerarse literatura gris, de igual forma se trató de evitar usar textos en español (a excepción de 2 textos en específico que abordaban LSM).

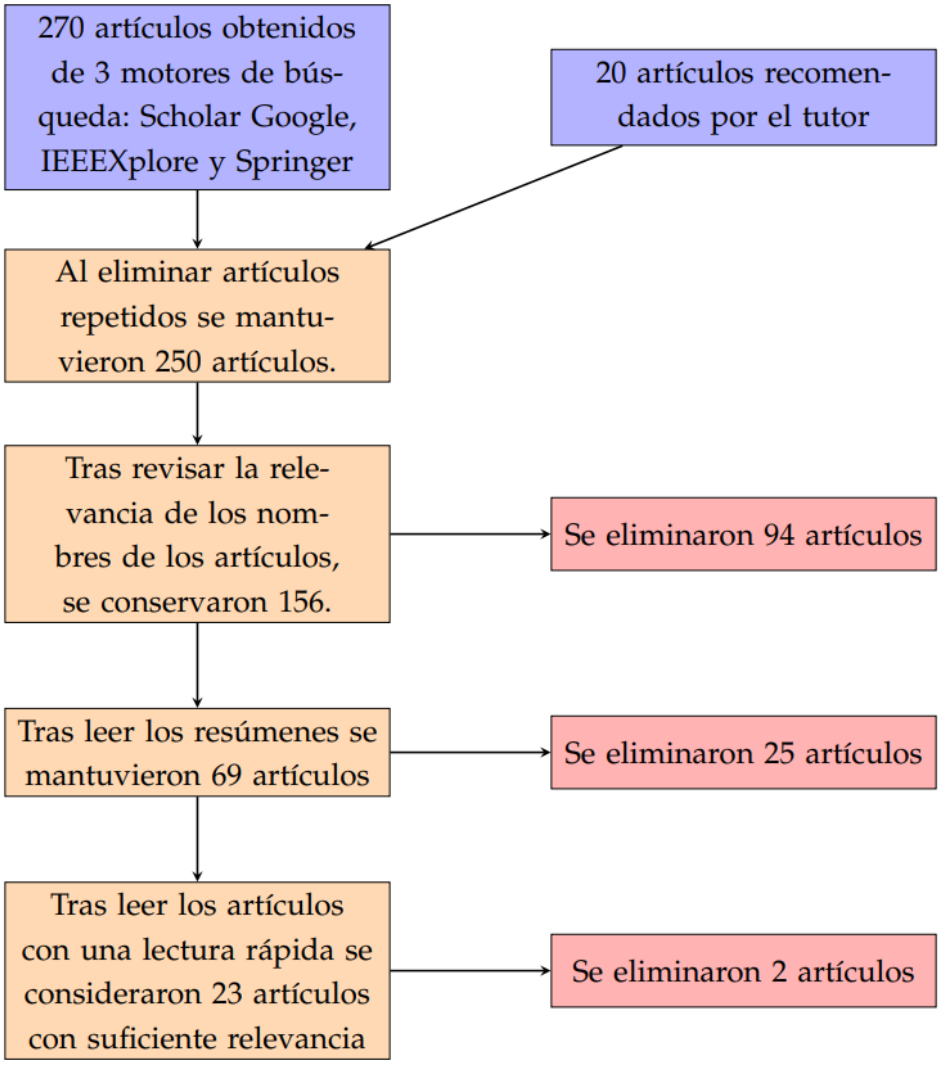

Figura 26. Diagrama de inclusión y exclusión de literatura.

La cadena usada en los 3 motores de búsqueda fue la siguiente:

(Finger Spelling OR Mexican Sign Language OR Sign Language)
AND (Recognition OR Translation)
AND (Deep Learning OR Computer Vision)

Con esta cadena se engloban tres aspectos importantes: acción (traducción), objeto de acción (LSM o Lengua de señas) y método (aprendizaje profundo o visión por computadora). Los trabajos con métodos de ML fueron excluídos de esta búsqueda. Además de esta búsqueda, se incluyeron 20 artículos relacionados con el tema que fueron recomendados por el respectivo tutor del proyecto de tesis. Estos 20 artículos en su mayoría eran específicamente trabajo relacionado con el LSM. El conjunto de referencias bibliográficas sumó 250 artículos, de este conjunto se eliminaron los duplicados por lo que se obtuvieron

270 artículos para ser analizados mediante *screening*. Cabe mencionar que las referencias recopiladas a partir de los motores de búsqueda se limitaron en un rango de fechas de 2010 al 2021 exceptuando una de 1999.

El proceso de *screening* consistió en eliminar las referencias bibliográficas que no concordaban o no tenían relevancia suficiente para el tema de investigación. El primer paso dentro del *screening* consistió en eliminar por título las referencias bibliográficas alejadas del tema por abordar. Durante este paso se conservaron 156. Se incluyeron los artículos que presentaban nombres ambiguos o genéricos para su posterior análisis. Una vez descartados los títulos poco consistentes con el tema de investigación, se leyeron los resúmenes de los 156 artículos restantes. Durante este proceso se descartaron 94 artículos que no mantenían una relación fuerte con la Lengua de Señas o la metodología especificada. Finalmente, se analizaron 69 artículos para lectura rápida. De estos artículos se observó que 22 de ellos no fueron lo suficientemente sustanciosos por lo que se conservaron los 20 restantes para realizar el análisis de la literatura. En la figura 26 se resume el proceso anteriormente descrito. La mayor parte de estos artículos se concentran en la metodología de traducción y reconocimiento de gestos para diversas Lenguas de Señas principalmente Lengua de Señas Americana (ASL), Lengua de Señas Británica (BSL), Lengua de Señas Alemana (DGS), Lengua de Señas China (CSL), Lengua de Señas India (ISL) y Lengua de Señas Arábe (ArSL). Una cuarta parte de los artículos seleccionados para lectura rápida se relacionan directamente con la traducción de LSM.

3.2. Reconocimiento de señas

El **SLR** es el campo dedicado a la interpretación automática de gestos con las manos y otro tipo de seña usados en las Lenguas de Señas. De acuerdo con Al-Qurishi et al. (2021), aún no es posible diseñar una herramienta de **SLR** con un 100 % de eficacia frente a un vocabulario amplio. Esto debido a los problemas inherentes a esta tarea. Antes de que cualquier aplicación práctica pueda ser considerada, es imperativo perfeccionar la interpretación de los algoritmos al punto que los falsos positivos sean inusuales (Cheok et al., 2019). Por ello es muy importante continuar desarrollando nuevas metodologías y evaluar sus propios méritos, de esta forma se podrían alcanzar soluciones más confiables.

3.3. Uso del cuerpo humano para la tarea de SLR

Como se ha mencionado anteriormente, las lenguas de señas son lenguajes meramente visuales. La interpretación de señas ha sido sujeto a investigación científica por varias décadas. Algunas de las publicaciones más recientes al respecto hablan de las oportunidades y los retos que tiene el área.

Figura 27. Representación de la misma seña con dos diferentes señistas expertos. A pesar de que la comparación cuadro a cuadro es muy distinta entre cada intérprete, se conserva la forma y trayectoria de las manos.

La mayor parte de los caracteres y palabras de las Lenguas de Señas son expresados con señas sencillas que involucran movimiento de las manos, esto hace que el reconocimiento de las señas sea relativamente fácil dependiendo de la posición, forma y movimiento de la manos. Sin embargo, el proceso de reconocer la mano involucra muchos problemas debido a la diversidad del tamaño de las manos entre señistas, color de piel e incluso estilos. Como se observa en la figura 27, la misma seña (glosa: TEMBLOR) representada por dos diferentes señistas es distinta ya que existen cambios físicos y de estilo de realizar la seña de ambos señistas. Estos problemas se pueden resolver diseñando sistemas de reconocimiento que sean independientes de estas variables. Una de las ventajas de las técnicas de aprendizaje profundo es que tienen la capacidad de identificar conexiones latentes entre diversas variables.

Otro problema que surge en el SLR es que dependiendo de las variaciones regionales de las Lenguas de Señas, los gestos pueden ser realizados con una mano o con ambas manos para expresar ciertas palabras. Usualmente las señas en las que solamente se utiliza una mano tienen significados básicos tales como letras y números. El análisis de señas debe reconocer contenido lingüístico a partir de imágenes y de video u otros formatos.

La configuración del cuerpo humano juega un rol esencial en el SLR. La idea básica es determinar la pose exacta del cuerpo completo basado en la posición de ciertos puntos fijos del cuerpo. Esta tarea puede ser realizada de diversas maneras, de acuerdo con Moryossef et al. (2020) y los algoritmos de aprendizaje profundo son sumamente eficientes en esta tarea siempre que se tenga un conjunto de entrenamiento con muestras correctamente seleccionadas.

3.4. Sistemas de reconocimiento de señas

Actualmente, las soluciones a la tarea de SLR se pueden dividir en cuatro grupos: de acuerdo con el tipo de seña, la granularidad de traducción, el método de adquisición de los datos y el método de clasificación usado. En la figura 28 se ilustra esta clasificación de acuerdo con la literatura revisada. En esta clasificación solo se incluyen trabajos con guantes y acelerómetros que usan métodos de IA.

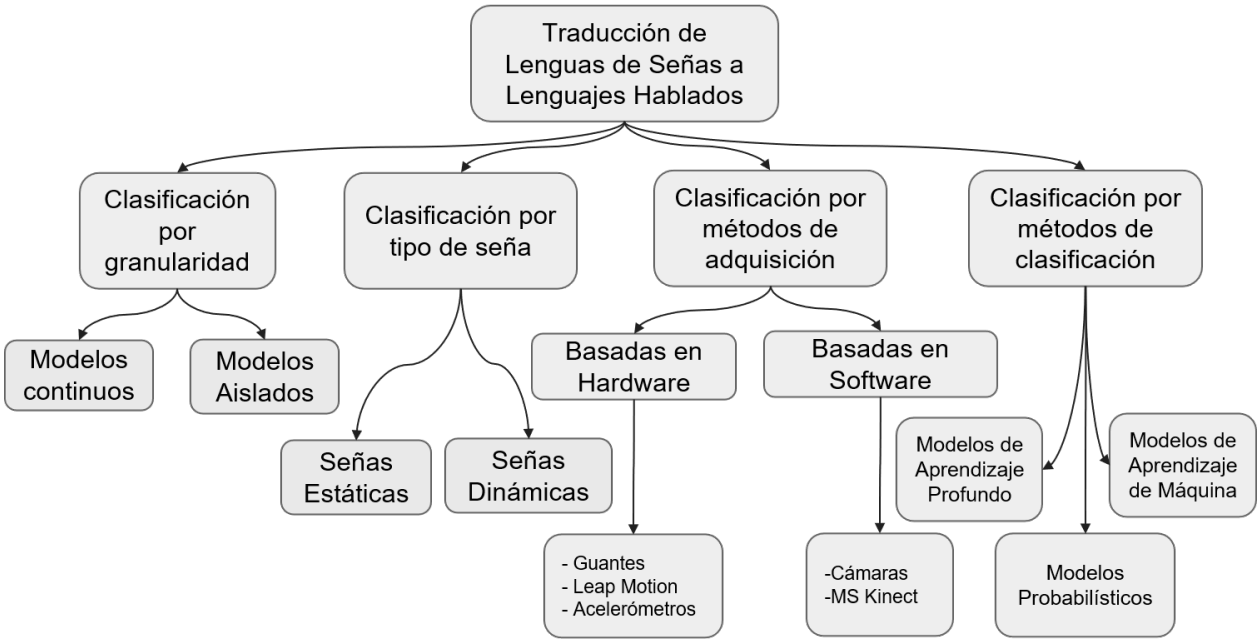

Figura 28. Clasificación del trabajo previo sobre los sistemas de traducción de Lenguas de Señas.

Mientras que algunos investigadores concuerdan que los modelos con aprendizaje profundo representan el enfoque con mejores resultados, existen otros enfoques que han tenido un éxito relativo como por ejemplo enfoques de aprendizaje de máquina. Además, estos sistemas de traducción se pueden dividir esencialmente en dos grupos dependiendo del tipo de método de recolección de datos. Un grupo de métodos se enfoca en sensores externos para obtener información de las acciones de los intérpretes de señas. Sin embargo, existen consideraciones prácticas en cuanto a las técnicas basadas en sensores. Es por ello que la mayoría de investigaciones de los últimos años se ha dirigido hacia métodos basados en **CV**. Estos métodos se basan en imágenes, videos o datos de profundidad de señistas para determinar la *semántica* de las señas.

Existen dos tipos de señas de acuerdo con su movimiento (Sharma et al., 2021): las estáticas y las dinámicas. Las señas estáticas suelen ser números y letras mientras que las señas dinámicas conforman la mayor parte del universo de palabras en las Lenguas de Señas. Como se puede apreciar en la figura

29, las señas dinámicas se representan con una serie de frames del movimiento del cuerpo del intérprete mientras que las señas estáticas se pueden representar con un solo frame. Esto tiene grandes implicaciones en los alcances de los modelos ya que la rápida evolución de componentes de software provoca que el desarrollo de sistemas para reconocer señas en tiempo real sean cada vez más factibles. Sin embargo, esto solo aplica para señas estáticas debido a que solo se necesita un frame.

a) Seña Dinámica: EXPLOSIÓN

a) Seña Estática: YO

Figura 29. Ejemplos de señas dinámicas y estáticas. *a)* Las señas dinámicas se representan con una serie de cuadros (frames) con el movimiento del torso superior del cuerpo del intérprete. *b)* Las señas estáticas se pueden representar con un solo frame.

Otro problema que se debe valorar al crear un sistema de traducción es la granularidad que tendrá el mismo. Es decir, se debe decidir si el sistema de traducción se limitará a reconocer señas aisladas de la Lengua de Señas o si será capaz de darle sentido a un conjunto de señas para traducir oraciones completas.

La mayor parte de la investigación en torno al reconocimiento de las Lenguas de Señas involucra problemas al interpretar movimientos del cuerpo, manos e incluso rostro, asociados con señas que representan un significado. El primer paso para realizar el SLR de señas es obtener información de entrada en cierto formato sobre estos movimientos corporales. Esta información debe ser adaptada o pre-procesada de tal forma que un sistema inteligente sea capaz de decodificar esta información, reconocer el gesto realizado y darle un significado al conjunto de información.

Además de esto, existen otros problemas sobre el tipo de información de entrada con el que se representan las señas y que sirve como entrada del sistema. Se debe considerar cuánta información sobre el cuerpo de los señistas es suficiente para representar cada seña. En las siguientes subsecciones se hablará en detalle el trabajo relacionado en cuanto a estos temas.

3.4.1. Adquisición de datos

Con los avances en las herramientas de reconocimiento automático, el SLR es una ventana de oportunidad para la creación de conjuntos de datos adaptados para esta tarea. Usualmente estas técnicas de Inteligencia Artificial requieren grandes conjuntos de datos para poder ser usadas. En esta Subsección se elaborará en los 2 tipos de enfoques para adquirir estos datos de señas.

3.4.1.1. Adquisición de datos basado en Hardware

Este enfoque fue creado para evitar enfoques basados en visión por computadora. En la mayoría de los casos, los enfoques basados en hardware usan sensores vestibles. Uno de los más populares son los guantes con sensores. El trabajo pionero de Starner et al. (1998) reporta un ejemplo de un sistema con sensores vestibles (guantes).

En esta misma categoría se encuentran los enfoques basados en sistemas de captura de movimiento usando sensores vestibles en las manos o el cuerpo. De acuerdo con Huang et al. (2015) los enfoques basados en sensores no son *naturales* debido a los instrumentos que deben usarse. En los años recientes, ha aumentado el uso de este tipo de enfoque para el reconocimiento de acciones humanas (Adeyanju et al., 2021). Existen otros dispositivos que entran en esta categoría como Leap Motion Controller (LMC) o Google Tango. El dispositivo Leap Motion es un excelente producto para realizar el seguimiento de las manos (Wang et al., 2016). Sin embargo el dispositivo es costoso económicamente lo que ha frenado su uso entre las comunidades sordas.

La mayor parte de los dispositivos de hardware para adquirir información de señas se basa en seguimiento de coordenadas o aceleración. El problema más evidente de estos dispositivos es su costo económico y su accesibilidad al público practicante de las Lenguas de Señas. Además de que el uso de dispositivos vestibles suele restringir el movimiento de los señistas.

3.4.1.2. Adquisición de datos basado en Software

En años recientes, la investigación en torno al reconocimiento de señas se ha enfocado en sistemas basados en visión debido a que tienen menos limitaciones para los usuarios. Este enfoque se basa en el uso de dispositivos de visión y de registro de profundidad para adquirir datos. Muchos de los trabajos de SLR se basan en este tipo de enfoque (Adeyanju et al., 2021). De acuerdo con Rioux-Maldague y Giguere (2014) el uso de datos de profundidad ha aumentado debido a la variedad de sensores 3D disponibles en

el mercado.

Además de los datos de profundidad, los datos de estimación de la pose han facilitado las técnicas basadas en visón por computadora. Rioux-Maldague y Giguere (2014) usaron una combinación de imágenes RGB e imágenes de profundidad para señas de un alfabeto manual (dactilología). El uso de datos de seguimiento de la pose da una estimación de coordenadas de las articulaciones o las extremidades del cuerpo humano. Koller et al. (2016) describe que en el SLR es determinante conocer la configuración de las manos, la posición y los ángulos formados entre los dedos para realizar un buen trabajo de reconocimiento.

Algunos retos que se presentan con los dispositivos de visión como cámaras convencionales es el hecho de que no son muy precisas. Sin embargo, la estimación de la pose disminuye este problema aunque también crea problemas como el de oclusión de extremidades. En este trabajo se utiliza este enfoque para la adquisición de datos.

3.4.2. Pre-procesamiento

El pre-procesamiento de las señas adquiridas depende del tipo de datos de entrada y el tipo de modelo usado para realizar el reconocimiento de señas. Esta fase involucra la representación de la seña, normalización, aumento de datos, extracción y selección de características de tal forma que esta información modificada sea la entrada para el modelo de reconocimiento.

3.4.2.1. Representación de las señas

Como se mencionó anteriormente, las Lenguas de Señas utilizan gestos manuales y no manuales para expresar estructuras gramáticas. Estas gesticulaciones varían desde la orientación, posición, forma y trayectoria de la mano. La representación de la seña involucra el tipo de codificación que se usará para representarla. En los enfoques basados en visión, usualmente las representaciones se basan en imágenes RGB o imágenes RGB-D (dispositivos de captura con profundidad).

Además de esto, es importante determinar si se usan únicamente características pertenecientes a las manos o si se incluyen otro tipo de características. Wei et al. (2016) observó que las características del cuerpo complementan bastante bien las características de las manos y de hecho se obtienen mejores resultados en SLR que usando únicamente características de las manos. En resumen, comprobaron que

usar características del cuerpo mejora 2.27 % los resultados de SLR. Esto puede ser atribuido al hecho de que las articulaciones del cuerpo son más robustas que las articulaciones de las manos.

3.4.2.2. Normalización y aumento de datos

En aprendizaje de máquina, la normalización se refiere a la tarea de estandarizar las entradas basadas en un conjunto de reglas, con el objetivo de mejorar el rendimiento del algoritmo de aprendizaje. El tipo de normalización depende del tipo de entrada (imagen, video, texto, etc). En el trabajo de Meng y Li (2021) grabaron videos de señas de varios usuarios. Los autores seleccionaron a un señista y estandarizaron los puntos clave de las posiciones de las articulaciones de todos los registros con respecto a este señista.

Muchos estudios en este campo incluyen la **generación de datos** (*data augmentation*), esto cumple el propósito de mejorar el rendimiento del modelo de reconocimiento debido a que se compensa la escasa cantidad que usualmente es recolectada. El muestreo aleatorio, o el descarte de algunos cuadros de los videos, es una de las técnicas más efectivas encontrada en la literatura (Al-Qurishi et al., 2021). Usualmente esta técnica es combinada con transformaciones de la imagen como el brillo, saturación, posición, rotación, etc. La herramienta PoseLTSM también emplea algunas operaciones para aumentar el número de imágenes a partir de las originales.

3.4.2.3. Extracción de características

La extracción de características es un paso crucial en todos los modelos de SLR debido al impacto que tiene sobre los modelos dado que ciertas características de los datos pueden facilitar la categorización de las señas. En todos los casos, la extracción de características se realiza sobre los datos originales y hacen referencia a posiciones del cuerpo humano (puntos clave del rostro, torso y manos) necesarias para la comunicación de las lenguas de señas. Estas características se basan en operaciones estadísticas y a cada una se le asigna un nivel de confianza proporcional a que tan certero sea el cálculo.

Una extracción de características útil es la de estimación de la pose mediante coordenadas espaciales 2D. Autores como Wu y Shao (2014) y Tang et al. (2015) realizaron procesos de extracción de características para realizar experimentos de reconocimiento de Lenguajes de Señas. Otros autores como Cihan Camgoz et al. (2017) usaron Redes Neuronales Convolucionales para realizar la extracción de características a partir de imágenes. Aunque este proceso parece que omite el paso de extracción automática de características, en realidad realiza esta extracción en el mismo paso que se realiza el reconocimiento. En otro experimento, Konstantinidis et al. (2018a) extrajo una mezcla de características

de video y del esqueleto humano a partir de secuencias de video. Las características de video fueron las imágenes mientras que las características del esqueleto humano fueron coordenadas del cuerpo, la cara y las manos.

3.4.2.4. Selección de características

Este paso es crucial en el diseño de sistemas de reconocimiento de señas basados en aprendizaje de máquina. Básicamente involucra una reducción de los datos a un número limitado de parámetros estadísticamente relevantes que son alimentados como entrada al modelo de reconocimiento. La idea es incluir solo aquellos que contribuyen significativamente a que el algoritmo distinga de forma efectiva entre las clases (señas) limitando el costo computacional. Por ejemplo, en el trabajo experimental de Konstantinidis et al. (2018b) se seleccionaron solo 12 de los 18 puntos clave del cuerpo humano (articulaciones). Esta selección fue basada en el hecho de que los señistas del conjunto de datos seleccionado se encontraban sentados y sus extremidades inferiores no se mueven.

3.4.3. Métodos de clasificación de señas

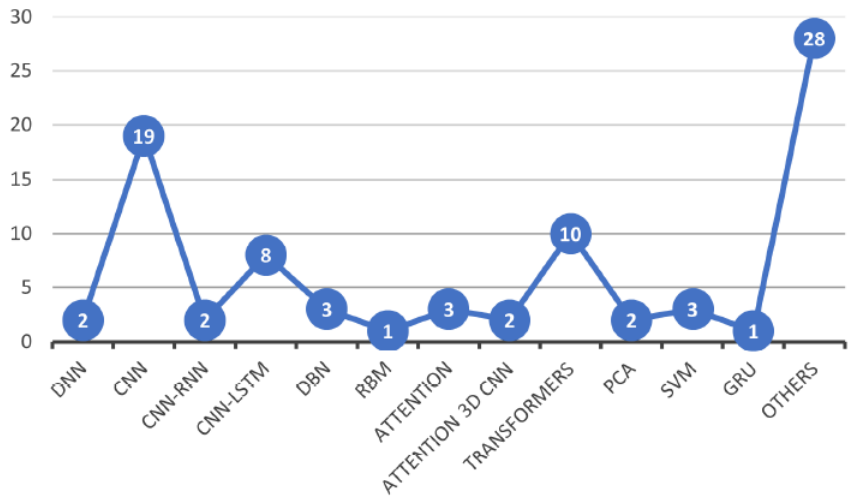

Figura 30. Artículos publicados del estado del arte en SLR con modelos de Aprendizaje de Máquina. Tomada de Al-Qurishi et al. (2021).

Los modelos de SLR se enfocan en identificar desde un punto de vista semántico una seña. El reconocimiento de señas puede involucrar el reconocimiento de patrones y visión por computadora. El objetivo de los modelos de clasificación de señas es reconocer señas (que ya fueron realizadas) e inferir su significado. Las técnicas de reconocimiento de señas incluyen modelos de aprendizaje de máquina, modelos probabilísticos y modelos de aprendizaje profundo. Como se detalló en la Sección 2.1, el aprendizaje de

máquina se refiere a aquellos algoritmos que aprenden a realizar una tarea (en este caso reconocimiento de señas) a partir de datos de entrada sin ser específicamente programados para ello.

En la figura 30 se observan el número de publicaciones, revisadas por Al-Qurishi et al. (2021), para distintos algoritmos de aprendizaje de máquina y aprendizaje profundo. Como se puede observar, existe trabajo relacionado que hace uso específicamente de algoritmos como Máquinas de Soporte Vectorial (SVM) y Análisis de Componentes Principales (PCA). Sin embargo, la mayor parte del estado del arte se basa en modelos de aprendizaje profundo.

3.4.3.1. Modelos basados en aprendizaje profundo

El aprendizaje profundo, cómo se explicó en la Sección 2.2, es un sub-campo del aprendizaje de máquina que se enfoca en aprender representaciones de los datos. Los avances en este campo han provocado grandes avances también en el campo del reconocimiento de Lenguas de Señas (Wu y Shao, 2014). Las técnicas de aprendizaje profundo aplicadas en los trabajos recientes de SLR usan redes neuronales convolucionales, recurrentes, residuales, transferencia de aprendizaje e incluso arquitecturas híbridas.

En muchos casos, como se muestra en la figura 30, las redes neuronales convolucionales (**CNN**) son una excelente opción debido a que son fácilmente entrenables cuando se cuenta con una gran cantidad de datos etiquetados. En el trabajo de Koller et al. (2016) se unió una red neuronal convolucional con un algoritmo EM (expectation-maximization), esto permitió que el modelo convolucional entrenara con muchas imágenes de ejemplo. Esta red alcanzó una exactitud del 67 % en más de 300 imágenes de manos realizando señas que fueron etiquetadas manualmente. En el trabajo de Ameen y Vadera (2017) se propuso una red neuronal convolucional enfocada en clasificar imágenes del abecedario de Lengua de Señas Americana usando mezclas de imágenes y de datos de profundidad. Este modelo alcanzó una precisión de 82 %. Yang y Zhu (2017) integraron una red neuronal convolucional a un método de reconocimiento de video. En este estudio el modelo fue entrenado para el reconocimiento de un diccionario limitado de Lengua de Señas China y alcanzó una exactitud de 99 %. En el trabajo de Hossen et al. (2018) se desarrolló una red que consistió de una capa convolucional con activación tipo ReLU, una capa max-pooling, una capa totalmente conectada (fully-connected), una capa de dropout y una capa softmax. Este modelo alcanzó una exactitud de 84.68 %, la cual es notablemente alta tomando en cuenta que se usó un conjunto de datos muy pequeño para entrenar y probar el modelo. Existen muchos trabajos enfocados al SLR de la Lengua de Señas India. Uno de los trabajos más notables es el de Rao et al. (2018) que alcanzó una tasa de reconocimiento del 92.56 % usando una arquitectura convolucional.

Otra arquitectura comúnmente usada es la red neuronal recurrente (**RNN**) dado que facilita el procesamiento de datos secuenciales. Fang et al. (2017) usaron una RNN bidireccional y celdas Long Short-Term Memory (**LSTM**) en su experimento para facilitar la traducción de lenguas de señas a nivel palabra y frase. La salida del experimento indicó que el modelo RNN pudo capturar exitosamente características importantes de las glosas de la Lengua de Señas Americana. Rakun et al. (2018) usaron una RNN con celdas LSTM para reconocer Lengua de Señas Indonesa. Se utilizó el enfoque LSTM debido a la necesidad de que el modelo pudiera usar secuencias completas de entrada y no dependiera de datos pre-procesados por cada cuadro de video. El resultado del experimento indicó que el modelo LSTM de dos capas alcanzó una exactitud del 95 % en el reconocimiento de palabras. Sin embargo, el modelo alcanzó 77 % cuando se usaron palabras conjugadas como palabras que involucran género o tiempo. Algunos estudios han usado LSTM para reconocer Lengua de Señas Indonesa. En Adimas et al. (2019) se usó una red LSTM de 2 capas para identificar Lengua de Señas Indonesa, la red neuronal alcanzó muy alta exactitud (91.74 %).

En ocasiones el uso de un solo tipo de red neuronal es complicado ya que esta red puede llegar a ser muy compleja en términos del número de capas. Como resultado, algunos experimentos combinan técnicas de aprendizaje profundo. Cui et al. (2017) introdujeron una red neuronal convolucional híbrida para mapear segmentos de video a glosas. Los autores usaron una red neuronal convolucional recurrrente (RCNN) para extraer las características y facilitar el aprendizaje de secuencias. Con esta arquitectura propuesta pudieron igualar los resultados con el estado del arte sin introducirle a la red información adicional. Los autores Ariesta et al. (2018) usaron una combinación de una CNN y una RNN bidireccional (Bi-RNN). Usaron la CNN para obtener características de cada cuadro de los videos de entrenamiento y la Bi-RNN para generar características únicas del comportamiento secuencial presentado en los videos. En promedio el enfoque híbrido alcanzó resultados similares a los del estado del arte.

En los últimos años se han desarrollado arquitecturas neuronales basadas en atención con arquitecturas de **transformer**. En este tipo de modelos se utilizan bloques encoder-decoder para entrenar modelos y realizar la clasificación de muestras de Lenguas de Señas. Este enfoque ha demostrado ser exitoso y tener ventajas sobre otros. La única limitación de este tipo de modelos es la falta de información posicional para las secuencias de entrada por lo que la extracción de características es otro paso necesario en estos modelos neuronales. A pesar de que estos métodos basados en transformers suelen tener mejores resultados que arquitecturas de otro tipo, aún no se encuentran ni cerca del nivel necesario para ser usado en

aplicaciones del día a día (Adeyanju et al., 2021). Esto debido a que los diccionarios disponibles para LS aún están basados en temas muy específicos (pronóstico del tiempo, emergencias, saludos, etc).

3.4.3.2. Enfoque de combinación de clasificadores

En su mayoría, el reconocimiento de acciones humanas en 3D se resuelve mediante el entrenamiento de clasificadores de aprendizaje profundo. Sin embargo, no es computacionalmente factible entrenar un clasificador para cada variante posible de datos de entrenamiento para seleccionar la combinación de mejor rendimiento de técnicas de pre-procesamiento para un conjunto de datos determinado. Sedmidubsky y Zezula (2021) proponen un procedimiento que determina la mejor combinación de rendimiento de técnicas de pre-procesamiento de forma muy eficaz. Sedmidubsky y Zezula (2021) entrenaron un clasificador independiente para cada técnica de preprocesamiento disponible y estimaron la precisión de una combinación específica mediante la fusión eficiente de los resultados de la clasificación correspondiente en base a una regla de voto mayoritario. Esto permite para decidir si es mejor entrenar un solo modelo o es mejor entrenar un conjunto de clasificadores independientes cuyos resultados son fusionados dentro de la fase de clasificación.

3.4.4. Granularidad de los sistemas de traducción

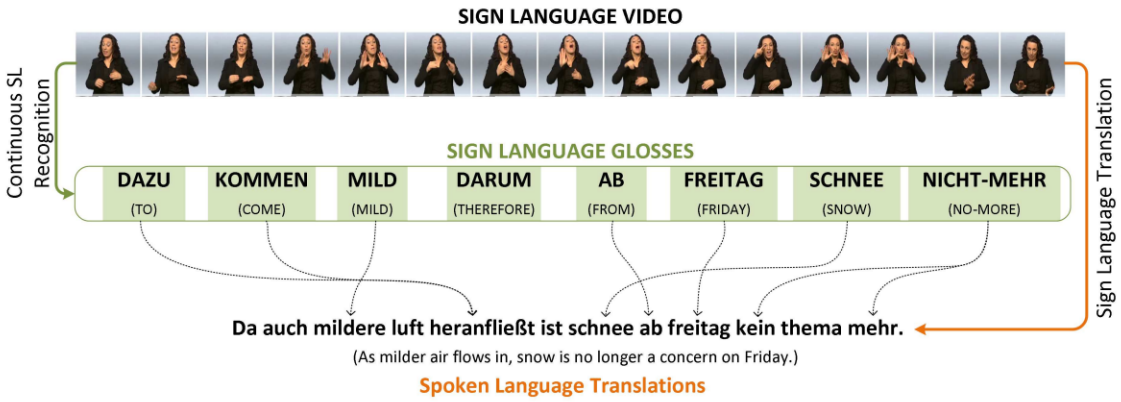

Figura 31. Diferencias entre modelos de reconocimiento aislados y modelos de traducción continuos. A partir de un video de señas de una frase se puede realizar el reconocimiento aislado de cada glosa o realizar directamente la traducción al lenguaje habado. Tomada de Camgoz et al. (2018).

Existen dos tipos de sistemas relacionados al nivel de traducción de Lenguas de Señas (su granularidad): el modelo aislado y el modelo continuo. En la figura 31 se observa la diferencia entre el reconocimiento de señas aislada y la traducción de una frase en señas al lenguaje hablado.

3.4.4.1. Modelos continuos

Estos modelos realizan la traducción de muestras de señas compuestas en oraciones. Por ejemplo, los autores Wu y Shao (2014) propusieron una red dinámica para el reconocimiento continuo de señas. Este modelo usa las posiciones 3D de las articulaciones. Koller et al. (2016) propusieron un algoritmos basado en EM para el reconocimiento continuo de lengua de señas. Cihan Camgoz et al. (2017) desarrollaron un sistema end-to-end diseñado para el reconocimiento continuo, este modelo está basado en modelado por sub-unidad. De forma similar Wang et al. (2018) sugirieron un método de fusión temporal para poder traducir lenguajes visuales continuos en formato de video a frases textuales.

3.4.4.2. Modelos aislados

Hasta ahora, la mayor parte del reconocimiento de Lenguas de Señas se realiza de forma aislada a partir de muestras de señas. Estos modelos examinan una secuencia de imágenes o movimientos del cuerpo obtenidos por algún sensor. Koller et al. (2016) usaron un conjunto de datos de señas aisladas de las Lenguas de Señas Danesa y Neo-Zelandeza. Konstantinidis et al. (2018a) también propusieron un sistema de SLR diseñado para extraer aspectos discriminativos de videos, donde cada video de señas corresponde a una palabra. En el trabajo de Kumar et al. (2018) los autores usaron un sistema de reconocimiento de glosas aisladas para facilitar la traducción. Este sistema incluyó pre-procesamiento de videos y un módulo de red neuronal para series de tiempo.

3.5. Conjuntos de datos de LS

Tabla 2. Principales conjuntos de *corpus* de Lenguas de Señas.

Autores	Nombre	Lengua de Señas
Martínez et al. (2002)	Lengua de Señas Americana	The Purdue RVL-SLL Database
Athitsos et al. (2008)		ASLLVD
Zahedi et al. (2006)		RWTH BOSTON-104
Bungeroth et al. (2008)		RWTH BOSTON-400
Barczak et al. (2011)		Massey Dataset
Assaleh et al. (2012)	Lengua de Señas Árabe	Arabic Sign Language Database
Shohieb et al. (2015)		Signs World Atlas
Porfirio et al. (2013)	Lengua de Señas Brasileña	LIBRAS-HCRGBDS
Schembri et al. (2013)	Lengua de Señas Británica	British Sign Language Corpus
Von Agris y Kraiss (2007)	Lengua de Señas Alemana	SIGNUM
Forster et al. (2014)		RWTH-PHOENIX-Weather 2014

Entre mayores son los diccionarios disponibles se pueden realizar sistemas más sofisticados. Para considerar un conjunto de datos robusto se deben tomar en cuenta dos aspectos importantes: el primero es la cantidad de señas etiquetadas y el segundo es la cantidad de repeticiones (preferentemente con distintos señistas) para cada una de estas señas. En esta Sección se presentarán algunos de los conjuntos de datos disponibles que contienen señas que han sido usados para evaluar herramientas de SLR.

Dependiendo de la Lengua de Señas seleccionada, actualmente existen dccionarios digitales de alta calidad. Para la Lengua de Señas Americana existen varios conjuntos de datos disponibles incluyendo el RWTH-Boston-1, RWTH-Boston-50 y RWTH-Boston-400 que varían entre el número de señas disponibles que contienen (10 a 400 señas). El American Sign Language Lexicon Video Dataset (ASLLVD) es el *corpus* más importante para el estudio de Lengua de Señas Americana ya que contiene 30,000 señas etiquetadas realizadas por 6 señistas. En cada uno de sus videos marcan el inicio y el final de cada seña. Para la Lengua de Señas Alemana existen conjuntos de datos robustos como SIGNUM y RWTH-PHOENIX-Weather. Estos conjuntos de datos contienen entre 35 y 1225 señas y además tienen bastantes frases formadas por 9 señistas distintos. Cada uno de estos videos se encuentran etiquetados y contienen información de las manos y del rostro de los señistas. El *corpus* IITA-ROBITA ISL es una diccionario digital de la Lengua de Señas India y fue desarrollado colaborativamente entre 2010 y 2017 por diversos equipos de investigación. El problema con este conjunto de datos es el hecho de que todo el conjunto de señas solo se realizó con un señista. Es decir, no se tienen más repeticiones para cada seña lo que limita su uso para herramientas de aprendizaje profundo.

Uno de los factores que limita el desarrollo de herramientas de SLR es el acceso a este tipo de conjunto de datos especializados. El hecho de que se requiera un conjunto de datos distinto para cada Lengua de Señas en el mundo y cada variante regional aumenta la complejidad de la tarea de tener un corpus adecuado para la tarea de reconocimiento. En algunos casos, como el caso de Ko et al. (2019), los investigadores han tenido que desarrollar conjuntos de datos a partir del registro de señas mediante video. Como se mencionó anteriormente, un conjunto de datos típico debe contener múltiples repeticiones de la misma seña preferentemente realizada por distintos señistas. Esto con el objetivo de generar modelos de reconocimiento de señas independientes del señista. Algunos conjuntos de datos son mayores que otros y esto debe tomarse en cuenta para entender la calidad de los resultados. En la tabla 2 se muestran algunas referencias de conjuntos de datos de lenguas de señas que tienen mejor contenido lingüístico en el mundo.

3.6. Estado del arte en torno al reconocimiento de la LSM

En la tabla 3 se resume el trabajo más relevante de la traducción y reconocimiento de LSM de la última década. Nótese que se especifica la granularidad de traducción, el tipo de seña o gesto (dinámica o estática), el tipo de enfoque de adquisición de datos y el tipo de modelo de reconocimiento.

Tabla 3. Trabajo previo en la traducción de LSM, NN indica redes neuronales y LM indica Leap Motion.

Autores	Granularidad	Gesto	Enfoque
Luis-Pérez et al. (2011)	Letras/Palabras	Estáticos	Segmentación de imágenes
Trujillo-Romero et al. (2013)	Letras/Palabras	Estáticos	Análisis de datos 3D
Caballero-Morales et al. (2013)	Letras/Palabras	Dinámicos	Uso de MS Kinect
Solís-V et al. (2014)	Letras	Estáticos	Momentos normalizados y NN
Galicia et al. (2015)	Letras/Palabras	Dinámicos	Uso de MS Kinect
Solís et al. (2015)	Letras/Palabras	Dinámicos	Momentos Jacobi-Fourier
López et al. (2015)	Letras/Palabras	Dinámico	Uso de MS Kinect
Pichardo-Lagunas (2015)	Letras/Palabras	Estáticos	Creación de conjunto de datos
Garcia-Bautista et al. (2016)	Frases	Dinámico	Traductor Señas a Diálogo
Solís et al. (2016)	Letras/Palabras	Estáticos	Momentos normalizados y NN
Pichardo-Lagunas (2016)	Letras/Palabras	Dinámicos	Generación de video
Sosa-Jiménez et al. (2017)	Letras/Palabras	Dinámicos	Uso de MS Kinect
García-Bautista et al. (2017)	Letras/Palabras	Dinámicos	Uso de MS Kinect y DTW
Jimenez et al. (2017)	Letras/Números	Estático	Uso de MS Kinect y AdaBoost
Martinez-Seis et al. (2019)	Letras/Palabras	Dinámicos	Imágenes de smartphones
Martínez-Gutiérrez et al. (2019)	Letras/Números	Estático	Uso de MS Kinect y AdaBoost
Martinez-Seis et al. (2019)	Letras/Palabras	Estáticos	IntraRealSense f200 NN
Trujillo-Romero et al. (2021)	Letras/Palabras	Ambos	Uso de MS Kinect y NN
Carmona-Arroyo et al. (2021)	Letras	Estático	Uso de MS Kinect/LM y ML

Pichardo-Lagunas (2016) señala que el LSM no cuenta con un vocabulario establecido oficialmente ni con reglas gramaticales normadas, ya que en ocasiones es desarrollada de manera intuitiva por sectores de la población con discapacidad auditiva. Sin embargo existen diccionarios de LSM que se han construido con la finalidad de normalizar y estandarizar el lenguaje (Martinez-Seis et al., 2019). Los diccionarios digitales de DIESELME (Cruz-Aldrete, 2014) y Manos con Voz (De Fleischmann y Pérez, 2011) son glosarios digitales adecuados para el aprendizaje de la LSM. Pichardo-Lagunas (2015) realizó una recolección de imágenes y videos de señas pertenecientes al LSM y su respectivo significado en Español. Además de las palabras y sus señas, el corpora incluye colocaciones, sinónimos y funciones sintácticas. El corpora se presenta en una base de datos multimedia. Tiene formato SQL[1]con diez tablas con referencias a videos e imágenes. Este corpus fue generado con el soporte de la comunidad de la Casa de Cultura de la Delegación Cuauhtemoc de la CDMX, México. Este contiene 1,505 palabras en Español relacionadas con

[1] `http://cienciadedatosupiita.com/content/bd_creation.sql`

1,019 videos de señas. Cada seña de estos 3 diccionarios digitales fueron hechas por un solo intérprete y tienen formatos de codificación distintos. Esto los hace inadecuados para su uso en un modelo de reconocimiento de aprendizaje de máquina cuyo objetivo final sea la generalización.

Capítulo 4. Metodología

En este Capítulo se explicará el flujo de trabajo propuesto en este trabajo para reconocer señas a partir de videos. La primera parte consiste en diseñar un diccionario de señas de la LSM de tal forma que sea escalable para reforzarse en un trabajo futuro, véase Sección 4.1. Una vez propuesto el diccionario se procede al diseño de la captura de los videos de señas, esta parte también involucra diseñar un set-up de captura de videos y realizar una prueba piloto previa a la captura, véase Sección 4.2. Posteriormente se planteó una fase de pre-procesamiento de los datos adquiridos. Esta fase consiste de 3 pasos necesarios para transformar y adecuar los datos de tal forma que faciliten el proceso de clasificación, esto se describe en la Sección 4.3. En esta sección se describen dos enfoques de aumentación de datos que se usarán en este trabajo para crear dosn conjuntos de datos de señas. La tercera fase consiste en la definición y entrenamiento de los modelos de aprendizaje de máquina. Los modelos propuestos se detallan en la Sección 4.4 Finalmente, el último paso consiste en la validación de los modelos de clasificación con datos de expertos. La estrategia de división del conjunto de datos para prueba y entrenamiento se describe en la Sección 4.5. En la figura 32 se observa el diagrama de esta metodología.

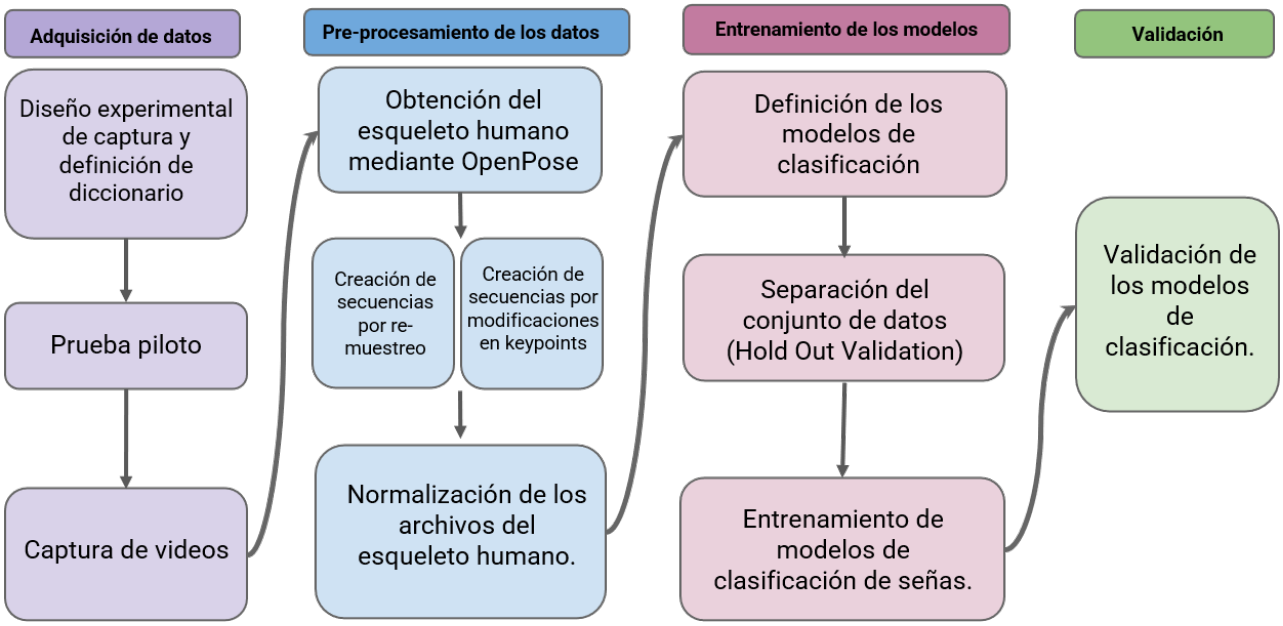

Figura 32. Diagrama de la metodología a seguir en este trabajo.

4.1. Diccionario de señas

En esta fase, se decidió diseñar un *corpus* de señas de la LSM debido a la falta de diccionarios digitales de LSM adecuados para los modelos de aprendizaje de máquina.

El diseño del siguiente diccionario se realizó con el propósito de extenderlo, de ser necesario para trabajos futuros. Para la creación de este *corpus* se propusieron 110 frases en español, todas ellas en contexto médico, de emergencias y frases cordiales. Además, este diccionario se realizó consultando cada frase con expertos en Lengua de Señas Mexicana de la Asociación Regional de Sordos Ensenadenses (ARSE) y una experta externa en LSM. Esto para identificar la existencia de frases o señas que no aplicaran en el contexto de una persona sorda y descartar repeticiones. En total se descartaron 10 frases, un ejemplo de una repetición es el siguiente par de frases:

Tengo escurrimiento nasal

Tengo síntomas de gripe

Ambas frases tienen distinto significado en la lengua hablada pero en señas tienen la misma interpretación.

Se le pidió a 2 expertos que reprodujeran la traducción a LSM de estas frases como se muestra en la figura 33. Se descartaron las señas que fueron reproducidas de forma distinta entre los expertos. La razón por la cuál los intérpretes expertos realizaron algunas señas de forma distinta fue debido a los regionalismos que existen en la LSM. Como se mencionó en el Capítulo 2, la LSM no es una lengua del todo unificada y existen variantes de señas dependiendo de la región de la República Mexicana. Finalmente se redujo el diccionario a 100 *glosas* en LSM debido a los descartes de frases y señas. En el Anexo B se muestra la lista de 100 glosas.

Figura 33. Ejemplo de frase en señas traducida a glosas.

4.2. Captura de señas

La captura de señas se realizó con 10 participantes no expertos. Esto debido a la dificultad que implicaba conseguir participantes sordos expertos en LSM. Primero se grabaron los videos de señas instructivos con uno de los expertos, de tal forma que un participante no experto en LSM fuese capaz de reproducir cada seña. Estos videos se le mostraron a cada uno de los participantes para que los reprodujeran.

Cada uno de estos videos fue capturado en un *set-up* de grabación especial para ayudar al sistema de reconocimiento de articulaciones (OpenPose). Se utilizó un fondo verde, una cámara RGB e iluminación constante como se observa en la figura 34.

Figura 34. Set-up de grabación de videos de señas. *a)* Se muestra el set de grabación que consiste en una cámara RGB, un conjunto de iluminación y una pantalla verde. *b)* Captura de videos de señas con participantes.

Las características del grupo de participantes son las siguientes: 8 de ellos fueron hombres y 2 mujeres, 4 participantes usaron lentes durante las grabaciones, además todos los participantes son *diestros*, es decir, que su mano dominante para realizar las señas fue la derecha y su mano de apoyo la izquierda. Cada participante es físicamente distinto el uno del otro, véase figura 35, lo importante en el diseño de esta adquisición de datos es que se realice correctamente cada seña. Por el hecho de que los modelos neuronales no trabajarán directamente con la imagen sino que recibirán como entrada las posiciones de las articulaciones de los participantes, las características físicas de los individuos no es relevante. Lo único relevante para el sistema de detección de OpenPose es la **trayectoria** de los movimientos de las articulaciones. Además los participantes firmaron una carta de consentimiento informado donde autorizaron utilizar su rostro para fines de investigación y divulgación.

Figura 35. Muestra de participantes para las grabaciones de los videos de señas.

Una vez organizado el set de grabación y propuesto el diccionario de señas se realizó una prueba piloto

de la captura de señas. En esta prueba piloto se le mostró al participante los videos instructivos de cada seña y se le pidió reproducirlas a una velocidad baja. Una vez finalizada la prueba piloto, se realizaron las grabaciones con cada participante. La captura de señas consistió en los siguientes pasos:

1. Se le explicó al participante la realización de las señas con la mano dominante y el uso de la mano de apoyo.
2. Se le pidió observar el video instructivo de la seña.
3. Se le pidió al participante imitar la seña realizada en el video instructivo y practicarla dos veces.
4. Se realizó la grabación de video a 30 cuadros por segundo (fps) de la seña a una velocidad lenta. Cada grabación de video cuenta con una duración de 2-4 segundos.
5. Una vez finalizada la grabación de la seña se procedió a repetir los 3 pasos anteriores con la siguiente seña.
6. En la seña 50 se realizó un descanso de 5 minutos.

Además se le solicitó a los participantes portar una camiseta oscura de manga larga durante las grabaciones de los videos. Esto para que las extremidades del participante pudieran contrastar con el fondo verde del set-up de grabación y así ayudar al sistema de reconocimiento de articulaciones. El registro de señas se realizó en un tiempo de 2 horas para todo el diccionario propuesto por participante.

4.3. Pre-procesamiento

En esta fase se describe el trabajo realizado sobre los datos de video registrados para cada seña previo a la clasificación. En la fase de captura de datos se obtuvieron 1,000 videos de señas, cada uno de estos videos con 120 frames en promedio. El preprocesamiento de cada uno de estos videos de señas consiste en la extracción de puntos clave, selección de características (articulaciones), generación de secuencias para aumentar el conjunto de datos y la normalización de cada frame del video, véase figura 36. A continuación se explicarán estos pasos en las siguiente sub-secciones.

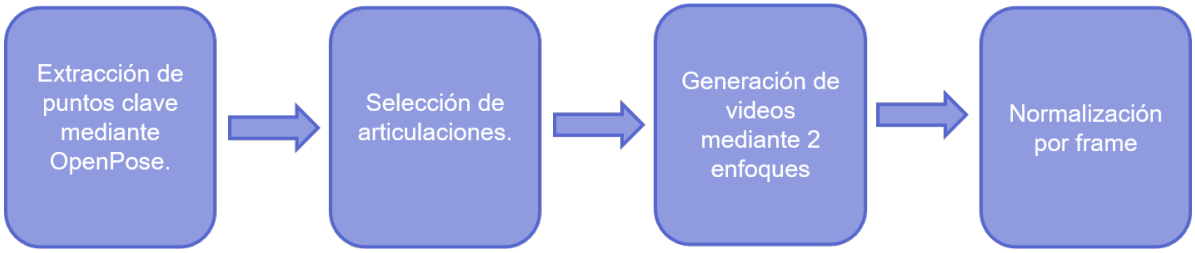

Figura 36. Diagrama de flujo del Pre-procesamiento de los videos de señas.

4.3.1. Extracción de articulaciones

Consiste en la extracción de puntos clave (*keypoints*) del cuerpo en cada frame para cada video de señas. Mediante el sistema de OpenPose se estiman las coordenadas de las articulaciones del participante sobre cada imagen que constituye un video. Este proceso se repite para cada frame de video y para cada seña. Finalemente se almacenan estas coordenadas en un formato de salida JSON para cada frame. OpenPose obtiene 137 keypoints, 70 correspondientes al rostro, 21 para cada una de las manos y 25 para el resto del cuerpo. En la figura 37 se observan los keypoints para determinados frames de los videos capturados de dos diferentes señas.

Figura 37. Representación gráfica de la extracción de articulaciones con OpenPose. Cada articulación contiene coordenadas (x, y) y un nivel de confianza c que va del 0-1 donde 0 significa un nivel de confianza nulo y 1 un nivel de confianza máximo.

4.3.2. Selección de articulaciones

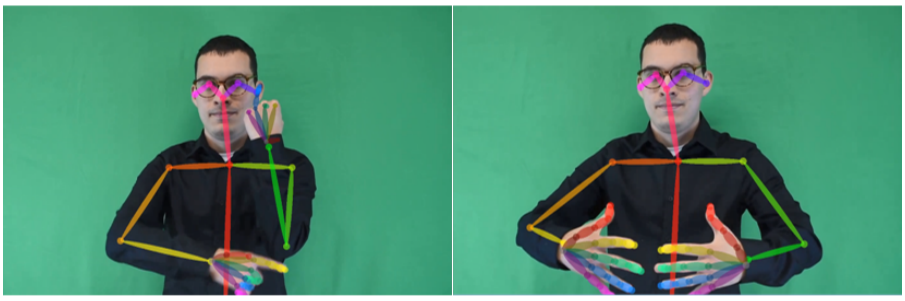

Figura 38. Selección de articulaciones.

Una vez estimadas las coordenadas de las articulaciones usando OpenPose, se deben seleccionar las más importantes de tal forma que ayuden al reconocimiento de la seña y disminuyan el costo computacional del algoritmo de aprendizaje profundo. En este caso, dado que la mayor parte de los participantes son *no expertos* en LSM los gestos faciales de algunas señas no fueron realizados. En el diccionario se eligieron señas en las que los gestos faciales no son indispensables para identificar a la seña. Se eligieron las

articulaciones de los codos, hombros pecho, nariz, ojos, orejas y además las 21 articulaciones de cada una de las manos. En total se seleccionaron 41 articulaciones de las 137 disponibles como se observa en la figura 38.

4.3.3. Aumentación de datos

Dado que el diccionario disponible contiene 100 señas y estas señas fueron realizadas por 10 participantes inexpertos, se tienen únicamente 10 muestras por cada una de las 100 clases. Esta cifra no es suficiente para entrenar un modelo de aprendizaje profundo. Existen muchas técnicas de *data augmentation* en la literatura, una de las más populares es la transformación numérica a nivel imagen (rotaciones, traslaciones, etc). Sin embargo, resulta costoso computacionalmente realizar estas transformaciones a nivel imagen para generar nuevos *frames* de cada uno de los videos.

En este trabajo se usan dos enfoques distintos para compensar esta falta de muestras de series de tiempo. El primer enfoque consiste en aprovechar la serie de tiempo de los keypoints y obtener submuestras de la misma, este enfoque estrictamente no es *data augmentation*. El segundo enfoque consiste en realizar transformaciones o alteraciones sobre la serie de tiempo (keypoints). Es importante recalcar que ninguna de estas transformaciones afectan la forma de realizar la seña (variantes regionales), solo se modifican las series de tiempo. A continuación se abundará más en cada uno de ellos.

4.3.3.1. Muestreo de las series de tiempo (Sampling)

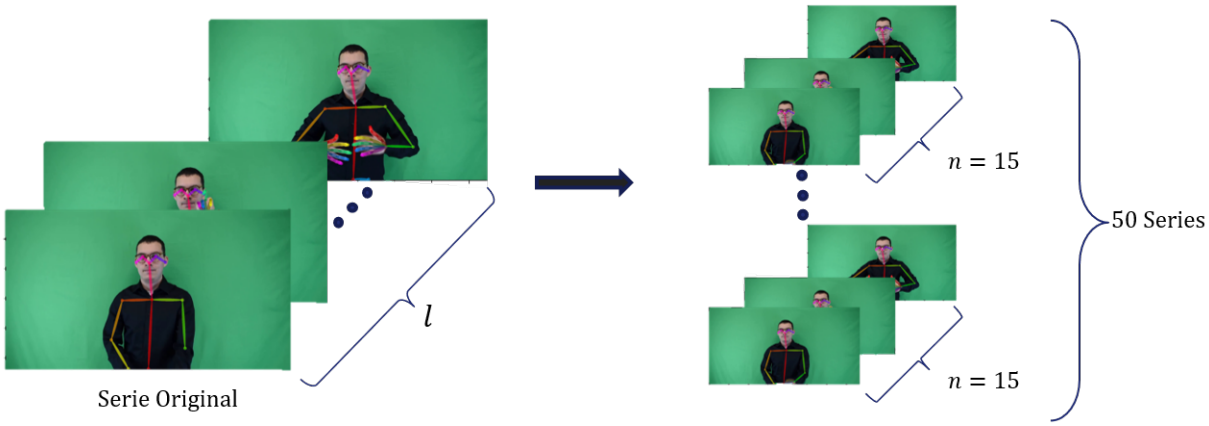

Figura 39. Aumentación de secuencias de frames usando técnica de remuestreo sugerida por Ko et al. (2019).

Ko et al. (2019) propusieron solucionar el problema de falta de datos mediante la extracción de secuencias de frames usando la técnica de **re-muestreo**. Esta técnica consiste en utilizar los frames disponibles

de los videos y crear combinaciones diferentes con una extensión limitada. En la figura 39 se muestra gráficamente esta técnica.

Dada una secuencia de video

$$S = (f_1, f_2, ..., f_l) \tag{13}$$

con l frames y seleccionando aleatoriamente un número fijo de cuadros, en este caso $n = 15$. La longitud promedio de espacios entre frames se calcula de la siguiente forma:

$$z = \lfloor \frac{l}{n-1} \rfloor \tag{14}$$

A continuación se extrae un conjunto de frames con los índices de la siguiente secuencia:

$$Y = (y, y + z, y + 2z, ..., y + (n-1)z) \tag{15}$$

donde $y = \lfloor \frac{l - z(n-1)}{s} \rfloor$ a Y se le conoce como **secuencia base**. Posteriormente se calcula una secuencia de números aleatorios

$$R = (r_1, r_2, ..., r_n) \tag{16}$$

con valores que van de $[1, z]$. Una secuencia nueva Y_{new} se genera sumando la secuencia base con la secuencia aleatoria:

$$Y_{new} = (Y_1 + R_1, Y_2 + R_2, ..., Y_n + R_n) \tag{17}$$

En este trabajo se generaron 50 secuencias nuevas de $n = 15$ frames c/u a partir de cada secuencia de video original. Por lo tanto, el conjunto de datos contiene 500 muestras por cada una de las clases (señas) dentro del diccionario propuesto.

4.3.3.2. Modificaciones de las articulaciones (Hand-Switch)

Park y Sohn (2020) propusieron realizar la aumentación de datos a nivel *keypoint* o articulación. En particular, encontraron buenos resultados aumentando el tamaño de las falanges de los dedos y removiendo articulaciones de manera aleatoria.

El método de modificación de las falanges de los dedos, consiste en aumentar o disminuir de forma aleatoria la longitud de los dedos en cada frame de la secuencia temporal. La longitud de los dedos varía de persona en persona. Por lo tanto, el reducir o aumentar la longitud del dedo de forma aleatoria en un

solo "video"simula que se grabaron videos (Park y Sohn, 2020). En la figura 40 se muestran en círculos rojos las articulaciones que fueron aleatoriamente modificadas. Las modificaciones aleatorias se realizaron entre ± 14 mm como máxima modificación, esto representa el 10 % de cambio en las coordenadas que entrega OpenPose como salida.

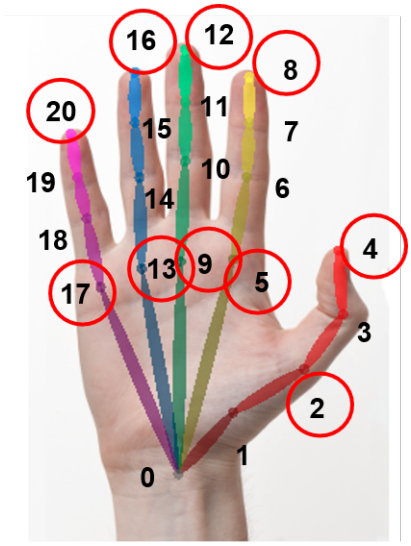

Figura 40. Articulaciones de las manos detectadas por OpenPose. En círculos rojos se resaltan las falanges que sufren modificaciones aleatorias para el proceso de data augmentation. Modificada de Eidos Team (2019).

Además de modificar la longitud de los dedos, se dejaron de utilizar algunas articulaciones de forma aleatoria. De acuerdo con Park y Sohn (2020), la justificación de remover articulaciones aleatorias es por el hecho de que OpenPose puede no reconocer de forma precisa los keypoints. Remover keypoints de forma aleatoria disminuye la dependencia en OpenPose para el aprendizaje.

En este enfoque, se generaron 50 secuencias nuevas de $n = 15$ frames c/u. Por lo que el conjunto de datos contiene 500 muestras por cada seña del diccionario.

4.3.4. Normalización

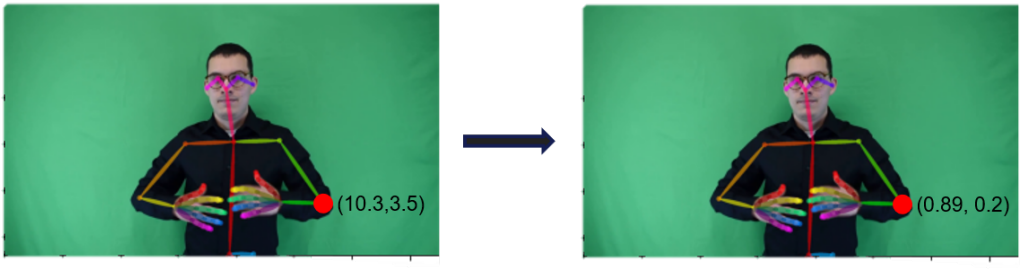

Figura 41. Representación gráfica de la normalización de frames.

El último paso del pre-procesamiento consiste en *normalizar* las secuencias de datos, véase figura 41. Esta normalización se aplicó en cada frame usando la media (μ) y la desviación estándar (σ) del vector de puntos clave compuesto por cada una de las coordenadas espaciales (x, y). La normalización de cada frame es necesaria para reducir la varianza de los datos y así facilitar el proceso de clasificación. Sean los vectores de coordenadas espaciales:

$$V_x = (v_1^x, v_2^x, v_k^x) \qquad \text{y} \qquad V_y = (v_1^y, v_2^y, v_k^y) \tag{18}$$

Estos vectores contienen las coordenadas de cada una de las k articulaciones para un solo frame. Cada elemento se normaliza respecto al valor de μ y σ de dicha coordenada (x o y) en ese frame:

$$\begin{aligned} \dot{V_x} &= \frac{V_x - \mu(V_x)}{\sigma(V_x)} \\ \dot{V_y} &= \frac{V_y - \mu(V_y)}{\sigma(V_y)} \end{aligned} \tag{19}$$

4.4. Modelos de clasificación

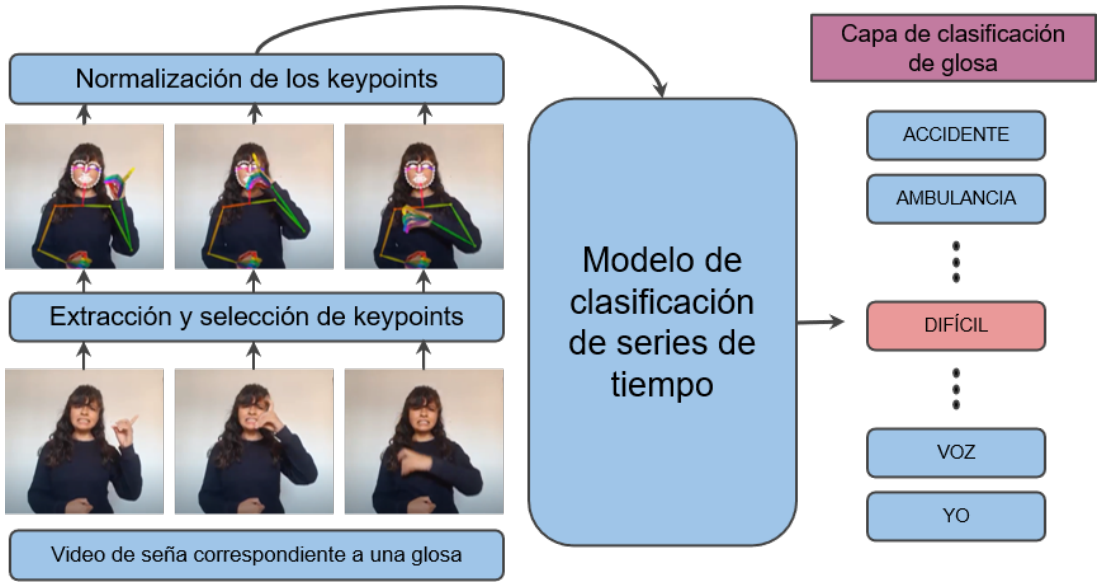

Figura 42. Modelo general de clasificación de señas usando la ubicación de los puntos clave del cuerpo humano.

En esta Sección se ahondará en los modelos seleccionados para clasificar las series de tiempo producto del pre-procesamiento de los videos detallado en la Sección anterior. Los modelos de clasificación propuestos realizan lo mostrado en la figura 42. En esta figura se observa que el modelo de clasificación de series

de tiempo recibe como entrada secuencias espacio-temporales etiquetadas con las que entrenará para lograr la clasificación adecuada de cada una de las 100 señas.

Como se explicó en la Sección 2.7, los modelos de aprendizaje profundo más populares para resolver la tarea de CST son los *modelos discriminativos* como las arquitecturas CNN, RNN e híbridas. Recordemos que nuestro conjunto de datos contiene 100 señas distintas, i.e., 100 **clases**. Como se muestra en la figura 42 todos los modelos propuestos tendrán una última capa tipo *dense*[1] de decisión con 100 neuronas. Las secuencias de entrada para estos modelos discriminativos serán secuencias *espaciales* (debido a las distintas ubicaciones de los keypoints) y *temporales* (debido a que se trata de secuencias que varían su posición respecto al tiempo). En la figura 43 se muestra el resumen de los modelos propuestos para resolver este problema de clasificación. Estos modelos fueron propuestos con base en el trabajo relacionado en torno al problema de SLR, véase Sección 3.

Figura 43. Modelos propuestos para realizar la clasificación de señas.

En las siguientes sub-secciones se detallarán cada uno de estos modelos.

4.4.1. Modelo base: KNN (DTW)

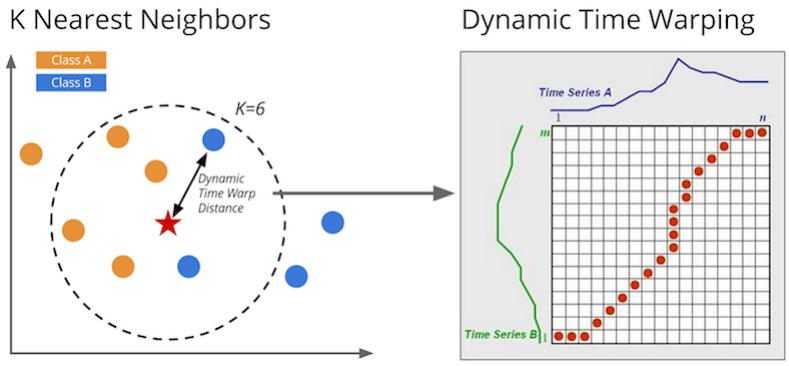

Figura 44. Representación gráfica de un modelo KNN y la distancia DTW. Tomada de Regan Mark (2016).

[1]Este término se usa para referirse a una capa que está totalmente conectada con la capa anterior.

Este modelo es el descrito en la sub-sección 2.7.1. Consiste del algoritmo de Vecinos más cercanos usando la distancia DTW, véase figura 44. Se seleccionó este modelo como base (baseline) debido a que es el más sencillo de todos los propuestos y suele tener resultados similares a los del *estado del arte* en muchas tareas de CST. El algoritmo de KNN se fijó con $k = 5$ vecinos, este parámetro fue seleccionado experimentalmente. Además se especificó que todos los puntos en el vecindario tuvieran el mismo peso.

4.4.2. RNN con celdas LSTM

Como se describió en la sub-sección 2.5.3, los modelos LSTM son un tipo de Redes Neuronales Recurrentes que tienen la capacidad de aprender sobre secuencias de entrada largas. Este modelo puede admitir secuencias $m - dimensionales$, estas m dimensiones corresponden al número de articulaciones de entrada para el modelo. La longitud de la serie de tiempo se representa como n, recordando que todas las secuencias tienen una $n = 15$ frames. Además se tiene una tercera dimensión $l = 2$ que corresponde a las variables espaciales (x, y) en cada frame.

Este modelo en particular aprende a extraer características de las secuencias de entrenamiento y aprende a mapear estas características a las diferentes clases (señas). Una ventaja de usar modelos RNN con celdas LSTM es que aprenden directamente de las secuencias.

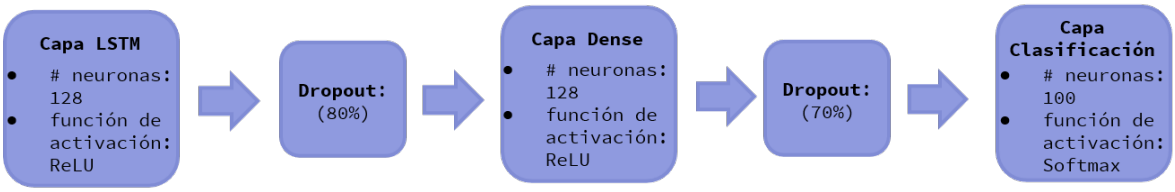

Figura 45. Representación gráfica de la arquitectura RNN con celdas LSTM.

La arquitectura RNN (LSTM) propuesta para este trabajo es la que se muestra en la figura 45. Se realizaron distintas variantes de esta configuración respetando la arquitectura. Sin embargo, se decidió mostrar esta configuración debido a que con ella se obtienen mejores resultados de clasificación. La arquitectura consiste de una capa RNN con 128 celdas LSTM y con función de activación ReLU conectada con una capa *dense* con 128 neuronas que finalmente se encuentra conectada con una capa dense con 100 neuronas y una función de activación Softmax. Entre estas capas se aplica una regularización *Dropout*[2] del 80 % y el 70 % respectivamente.

[2]Recordando que el *dropout* nos ayuda a disminuir el posible sobreajuste.

4.4.3. CNN-LSTM

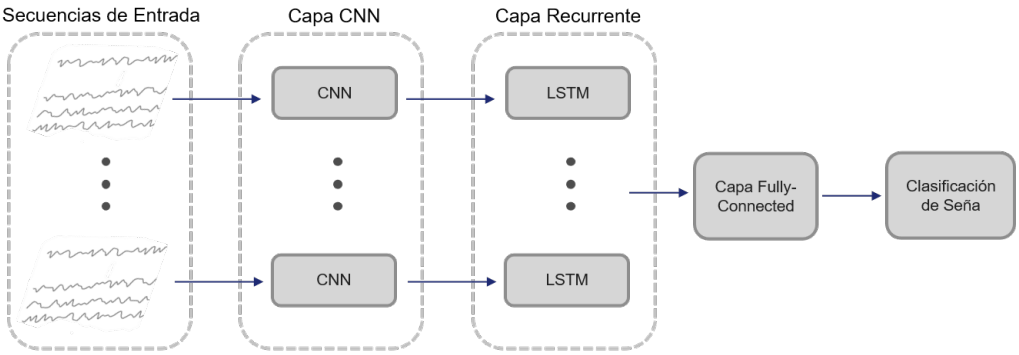

Figura 46. Funcionamiento de un modelo CNN-LSTM.

La arquitectura CNN-LSTM usa capas convolucionales para realizar la extracción de características de los datos de entrada y las combina con LSTM para realizar la predicción de la secuencia, véase figura 46. Esta arquitectura fue nombrada Long-term recurrent convolutional network (LRCN). Fue originalmente desarrollada para la tarea de generación de secuencias de texto describiendo imágenes. Para ese propósito, se usó una CNN pre-entrenada con un problema de clasificación de imágenes como extractor de características. Esta arquitectura es ideal para problemas con estructuras espaciales (2-D o 3-D) y temporales.

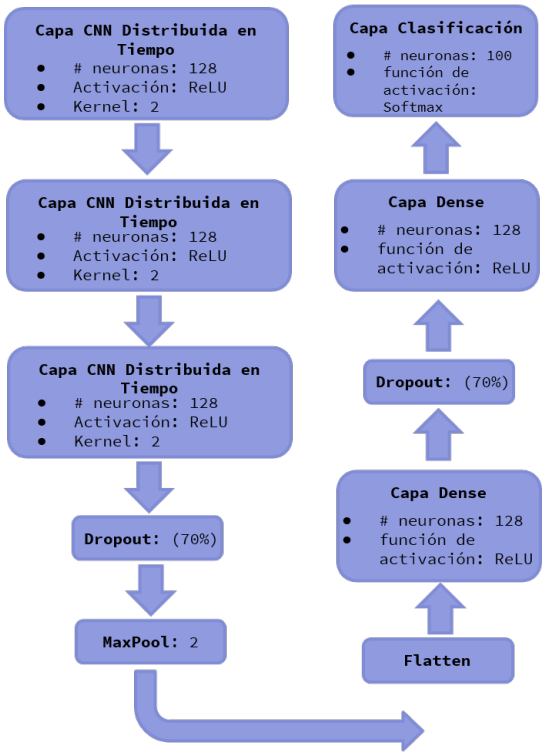

Figura 47. Representación gráfica del modelo CNN-LSTM.

En la figura 47 se observa la arquitectura CNN-LSTM propuesta en este trabajo. Esta arquitectura consiste en 3 capas convolucionales distribuidas en tiempo, i.e. con celdas LSTM, cada una con 128 neuronas una función de activación ReLU y Kernel de tamaño 2. Posteriormente la salida de estas capas sufre un MaxPooling y un aplanamiento para ser conectada con un clasificador. El clasificador[3] consiste de 3 capas dense, las primeras dos con 128 neuronas y la última con 100 neuronas correspondientes a cada clase.

4.4.4. ResNet

Esta arquitectura fue propuesta por Wang et al. (2017) y es una red neuronal residual medianamente profunda. Originalmente esta red se compone de 11 capas de las cuáles las primeras 9 son convolucionales. La característica principal de las capas ResNet son sus *atajos* o conexiones residuales entre capas convolucionales consecutivas. Estas conexiones residuales permiten el flujo del gradiente, es decir, el entrenamiento de la red neuronal profunda es mucho más sencillo dado que se reduce el efecto de desvanecimiento del gradiente.

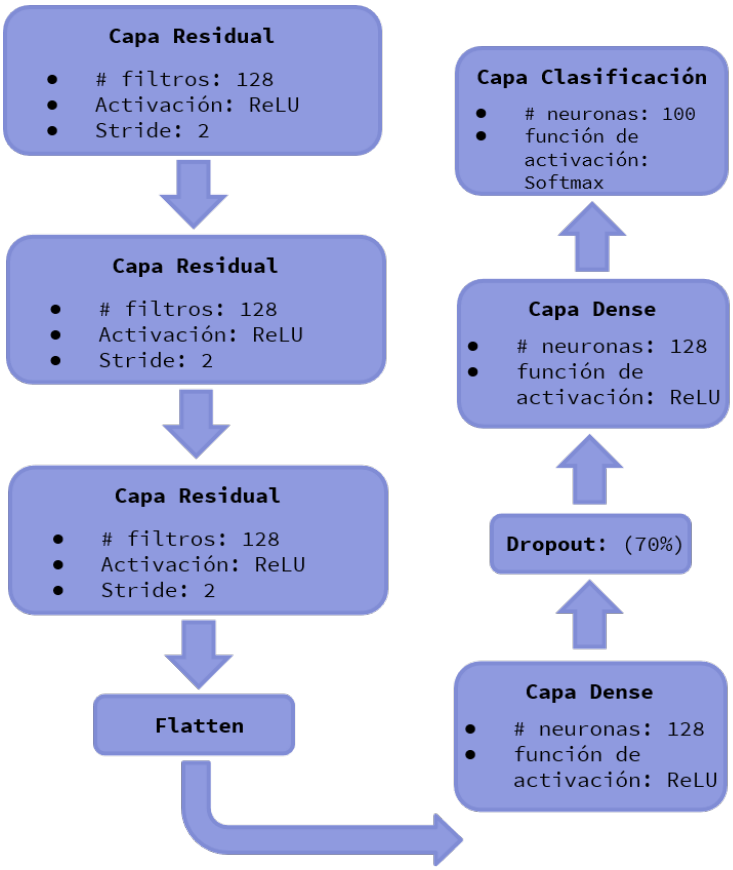

Figura 48. Representación gráfica del modelo ResNet.

[3]Se refiere a la fase de la red neuronal encargada de realizar la clasificación.

En la figura 48 se observa el modelo ResNet propuesto para este problema. El modelo cuenta con 3 capas residuales con 128 filtros. Posteriormente la salida de estas capas residuales se aplana para entrar a una fase de clasificador. Este clasificador consiste de tres capas *dense* con la última capa teniendo 100 neuronas.

4.5. Conjuntos de entrenamiento, validación y prueba

Como se mencionó en la sección 4.2, se grabaron a 10 participantes inexpertos. Además, se registraron y pre-procesaron videos de 2 expertos en LSM para cada seña. En total se cuentan **1200** videos originales para las 100 señas. A cada video de los participantes inexpertos se le realizó *data augmentation* usando los dos enfoques descritos en la Subsección 4.3.3. Es decir, se generaron **2 subconjuntos de datos**, el primero correspondiente al enfoque de data augmentation de re-muestreo (*Sampling*) y el segundo subconjunto correspondiente al enfoque de modificaciones a nivel keypoint (*Hand-Switch*). En cada uno de estos subconjuntos de datos se generaron 50 secuencias "sintéticas" a partir de cada video original, cada secuencia tiene un tamaño fijo de 15 frames. Por lo tanto, cada subconjunto de datos cuenta con **500** series de tiempo para cada clase (50,000 series de tiempo en total).

El esquema de separación del conjunto de datos que se usó en este trabajo es el de *Hold-Out Validation*. Este enfoque consiste en dividir el conjunto de datos en 3 subconjuntos: *entrenamiento, validación y prueba*. Usualmente se escogen proporciones de 70 %, 15 % y 15 % para cada subconjunto del total de datos.

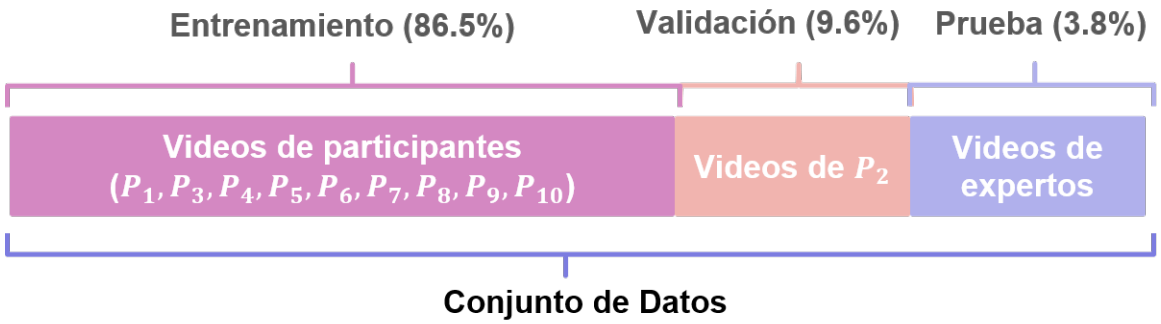

Figura 49. Porcentajes de datos utilizados para el entrenamiento, validación y para prueba en el esquema Hold-Out Validation.

En este caso, se decidió utilizar los videos de inexpertos para los conjuntos de entrenamiento y validación. A los datos de los expertos en LSM solo se les realizó el esquema de sampling y se obtuvieron 10 secuencias a partir de cada video original. Estos datos fueron reservados para probar el modelo. El subconjunto de

validación es útil para monitorear el aprendizaje de los modelos neuronales profundos. En este trabajo, se usaron los videos del participante inexperto 2 (**P2**) para la validación. En la figura 49 se muestra esquemáticamente el porcentaje de datos asignados para cada subconjunto. En resumen, se utilizarán **45,000** secuencias de señas para el entrenamiento, **5,000** secuencias para la validación de los modelos y **2,000** secuencias para evaluar el rendimiento de cada modelo ante un escenario parecido al real.

Capítulo 5. Resultados

En este capítulo se muestran los resultados obtenidos para los modelos de clasificación de señas propuestos utilizando los dos enfoques de *data augmentation*. Primero se describirán los resultados para el modelo base seleccionado debido a que los modelos propuestos se compararán con sus resultados. Posteriormente se mostrarán los resultados de los 3 modelos propuestos (LSTM, CNN-LSTM y ResNet) para cada uno de los conjuntos de datos propuestos: Sampling y Hand-Switch. Se detallarán las métricas de rendimiento que describen que tan bien realizan la clasificación de las señas. Una vez descritos estos resultados, se compararán estos modelos entre sí con base en su rendimiento al realizar la tarea y sus tiempo de ejecución. Finalmente, se realizará una discusión de resultados donde se analizará el motivo de estos resultados.

5.1. Modelo base KNN (DTW)

Como se mencionó en las Secciones 2.7 y 4.4, el modelo KNN con distancia DTW alcanza resultados comparables al estado del arte en diversos problemas. Por ello se probó este modelo de clasificación de aprendizaje de máquina sobre el conjunto de datos de Sampling y se tomaron sus resultados como base (*baseline*) para comparar los modelos de aprendizaje profundo propuestos. En la figura 50 se observa la matriz de confusión generada sobre los datos de prueba de dicho modelo. Nótese que se tienen 20 series de tiempo etiquetadas para cada clase en este conjunto de prueba.

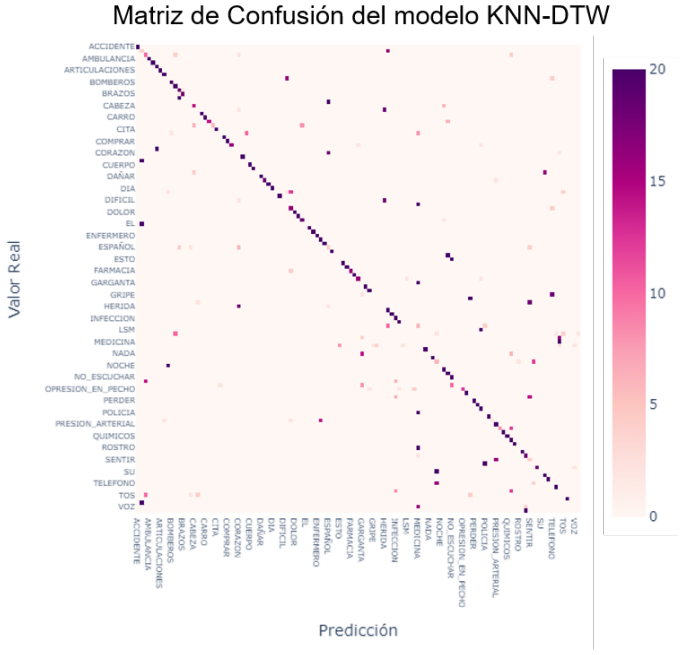

Figura 50. Matriz de confusión del modelo base KNN-DTW.

En la matriz de confusión se observa claramente la tendencia sobre la diagonal principal. Esto significa que, en un porcentaje visible, el modelo predice que las secuencias pertenecen a la clase que en realidad corresponden. A pesar de que en la matriz de confusión se observan relativamente pocas *confusiones*, este modelo alcanzó una exactitud o *accuracy* de **0.59**. En este trabajo podemos tomar la métrica de exactitud como referencia debido a que los conjuntos de datos son *balanceados*, i.e. todas las clases tienen el mismo número de muestras etiquetadas. La precisión de este modelo alcanzó **0.54** mientras que el F1-Score (métrica que combina la precisión y el recall) alcanzó **0.55**. El modelo tardó 14 minutos para entrenar y se tomó 33 minutos para clasificar el conjunto de prueba.

5.2. RNN (LSTM)

A continuación se describirán los resultados de los modelos de aprendizaje profundo para el conjunto de entrenamiento creado utilizando el enfoque de re-muestreo (Sampling) y el enfoque Hand-Switch para el modelo LSTM.

5.2.1. Métricas de rendimiento de los modelos para el conjunto Sampling

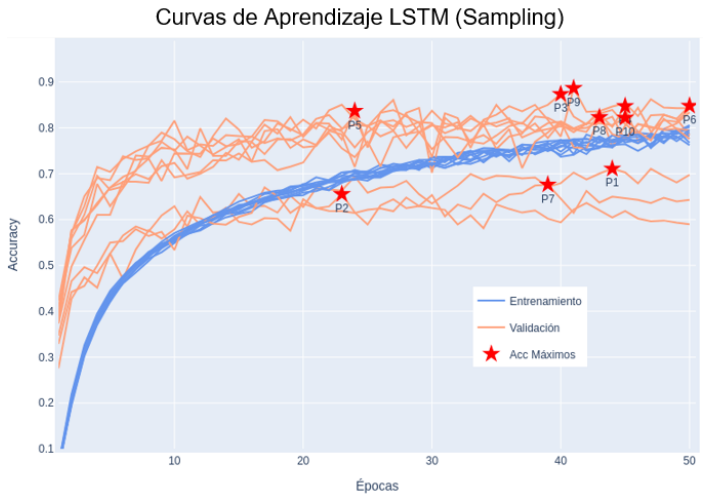

Figura 51. Curvas de aprendizaje del modelo RNN (LSTM) con el conjunto de Sampling.

Como se describió en la Sección 4.4, el modelo con RNN con celdas LSTM es el más sencillo de los propuestos en este trabajo. La capa recurrente de este modelo ayuda a extraer características temporales (a nivel de frame) de las secuencias. En las figuras 51 y 52 se observan los resultados obtenidos del entrenamiento y su rendimiento con los respectivos datos de prueba.

En la figura 51, se observan las curvas de aprendizaje del modelo sobre distintas subdivisiones del

conjunto de datos. Aprovechando la sencillez del modelo, se realizó un enfoque Leave One Out para determinar cuál será el subconjunto de validación para los siguientes modelos. En estas curvas de aprendizaje se monitorea el entrenamiento del modelo por época (iteración). En este caso podemos observar que existe muy poco sobre-ajuste debido a que las curvas de entrenamiento son similares y se encuentran a poca distancia de las curvas de validación. Para el conjunto de validación se eligieron los datos del participante inexperto 2 (P2) debido a que su curva de validación presenta menor similitud a las de entrenamiento y esto ayudará a obtener una evaluación más justa de los entrenamientos. Para este modelo, el entrenamiento fue exitoso, ya que se alcanzó un accuracy de **0.9** para entrenamiento y **0.65** para validación.

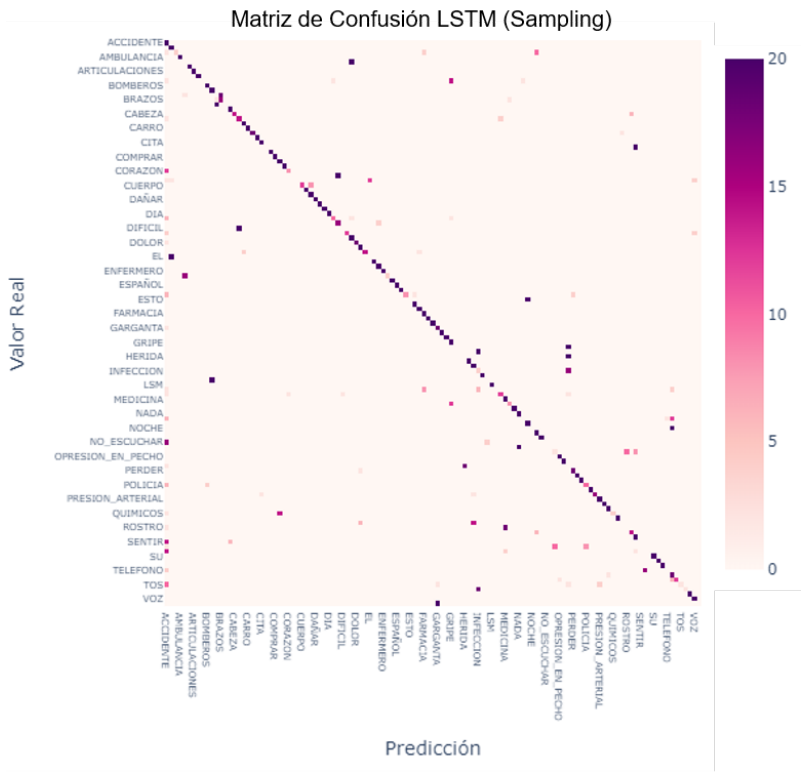

Figura 52. Matriz de confusión del modelo RNN (LSTM) con el conjunto de Sampling.

Una vez entrenado el modelo LSTM, en la figura 52 se muestran los resultados de la matriz de confusión sobre el conjunto de prueba. Nótese que, en comparación con la matriz de confusión del modelo 50, el modelo LSTM clasifica mejor. Esto debido a que se observan más muestras sobre la diagonal principal. El score de exactitud obtenido sobre el conjunto de prueba fue de **0.67**. El F1-Score y la precisión alcanzados para el conjunto de prueba fueron **0.64** y **0.62** respectivamente. Este modelo tardó **7 minutos** en realizar el entrenamiento y **0.7 segundos** en clasificar los datos de prueba.

5.2.2. Métricas de rendimiento de los modelos para el conjunto Hand-Switch

Los resultados obtenidos a partir del modelo LSTM usando el conjunto de entrenamiento de enfoque Hand-Switch se muestran en las figuras 53 y 54. En la figura 53 se muestran las curvas de aprendizaje durante la fase de entrenamiento. Se observa que existe un claro sobre-ajuste en el modelo, ya que la curva de validación está muy por debajo de la curva de entrenamiento. Esto debido a que el modelo no esta aprendiendo a generalizar apropiadamente. En el entrenamiento se obtuvo un *accuracy* de **0.97** mientras que en la validación se obtuvo **0.57**.

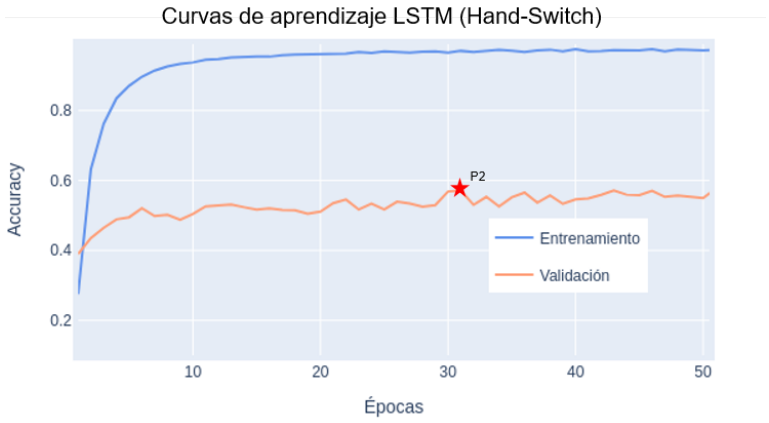

Figura 53. Curvas de aprendizaje del modelo LSTM con el conjunto de Hand-Switch.

En la figura 54 se muestra su respectiva matriz de confusión. Los elementos de la misma no se encuentran tan concentrados en la diagonal principal si la comparamos con los resultados de ResNet (Sampling) o o el modelo baseline. El accuracy de prueba alcanzado fue de **0.59**, el F1-Score de **0.52** y la precisión de **0.54**. Este modelo tardó aproximadamente lo mismo en entrenamiento y prueba que el LSTM (Sampling).

En la tabla 4 se muestran los resultados del modelos LSTM con los dos conjuntos de datos.

Tabla 4. Tabla de resultados del modelo LSTM.

	Train Acc	Val Acc	Test Acc	Test F1-Score	Test Prec
Sampling	0.9	0.65	0.67	0.64	0.62
Hand-Switch	0.97	0.57	0.59	0.52	0.54

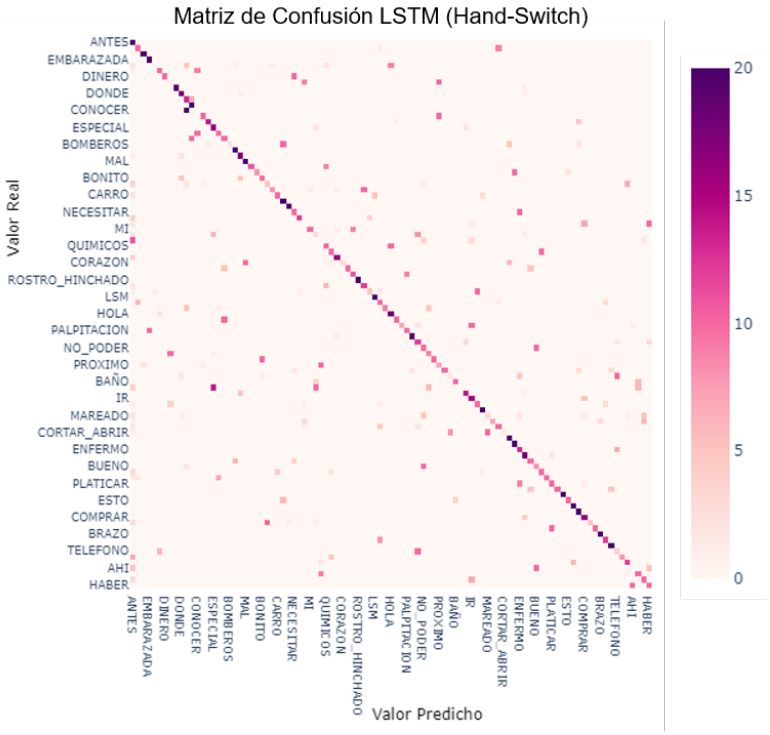

Figura 54. Matriz de confusión del modelo LSTM con el conjunto de Hand-Switch.

5.3. CNN-LSTM

A continuación se describirán los resultados de los modelos de aprendizaje profundo para el conjunto de entrenamiento creado utilizando el enfoque de re-muestreo (Sampling) y el enfoque Hand-Switch para el modelo CNN-LSTM.

5.3.1. Métricas de rendimiento de los modelos para el conjunto Sampling

Este modelo híbrido aprovecha la estructura espacial y temporal de la secuencia de datos de entrada, véase Sección 4.4. En las figuras 55 y 56 se muestran los resultados generados por este modelo para el conjunto de Sampling.

En la figura 55 se observan las curvas de aprendizaje con el conjunto de entrenamiento de Sampling y el conjunto de validación P2. Durante el entrenamiento se alcanzó un accuracy de **0.98** mientras que la validación alcanzó un accuracy máximo de **0.77**. Estos resultados muestran un sobre-ajuste del modelo con los datos de entrenamiento, i.e. el modelo no está generalizando lo suficiente.

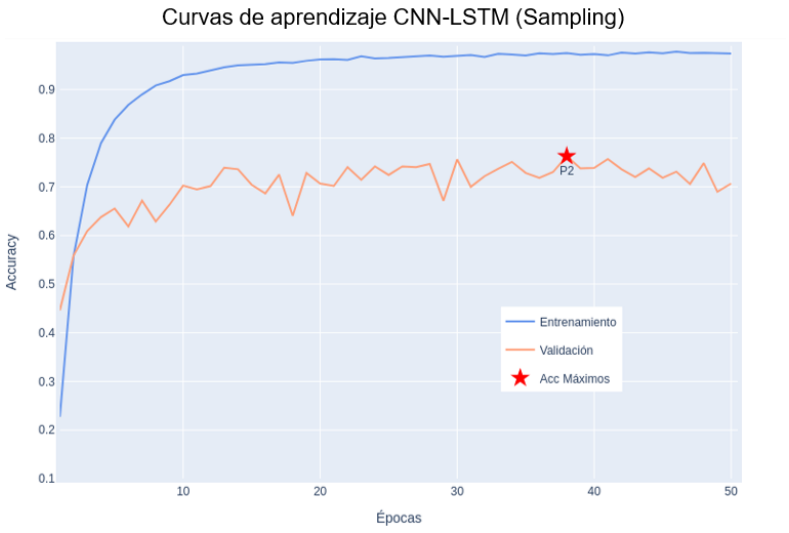

Figura 55. Curvas de aprendizaje del modelo CNN-LSTM con el conjunto de Sampling.

En la figura 56 se muestran los resultados de la clasificación con el conjunto de prueba a través de la matriz de confusión. Se observa que tiene más valores sobre la diagonal principal que la del modelo base. Este modelo híbrido alcanzó un *accuracy* sobre el conjunto de prueba de **0.76**, en cuánto a F1-score y precisión, alcanzó **0.77** y **0.72** respectivamente. Además modelo tardó **12 minutos** en entrenar y **0.96 segundos** en obtener las predicciones de las clases del conjunto de prueba.

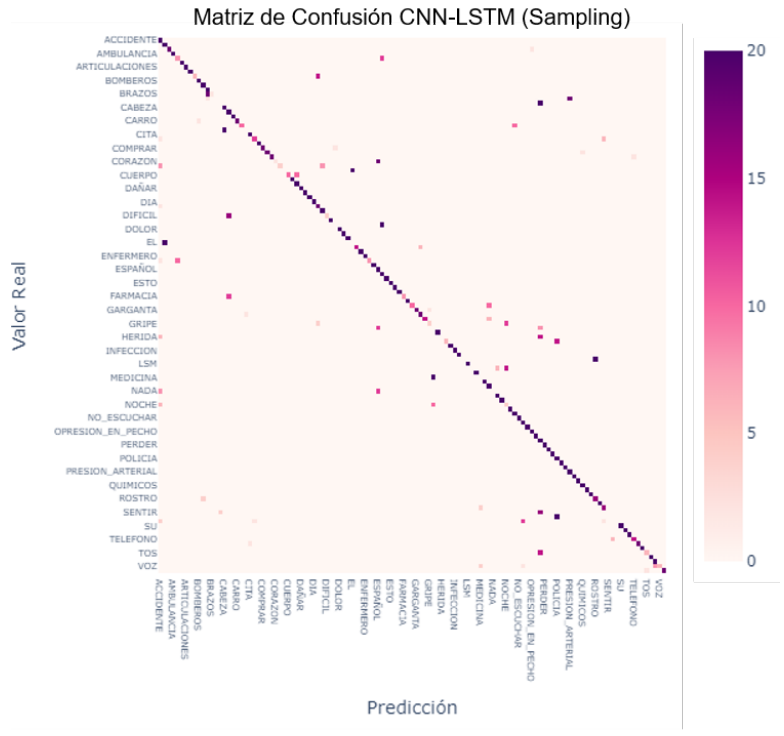

Figura 56. Matriz de confusión del modelo CNN-LSTM con el conjunto de Sampling.

5.3.2. Métricas de rendimiento de los modelos para el conjunto Hand-Switch

Los resultados obtenidos a partir del modelo híbrido CNN-LSTM usando el conjunto de entrenamiento de enfoque Hand-Switch se muestran en las figuras 57 y 58.

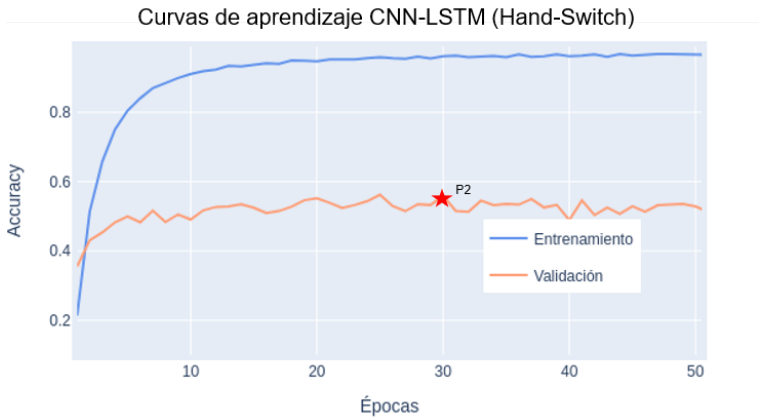

Figura 57. Curvas de aprendizaje del modelo CNN-LSTM con el conjunto de Hand-Switch.

En la figura 53 se observan las curvas de aprendizaje durante el entrenamiento del modelo. Las curvas de aprendizaje denotan sobre-ajuste en el modelo. En el entrenamiento se obtuvo un accuracy de **0.99** mientras que en la validación se obtuvo **0.59**.

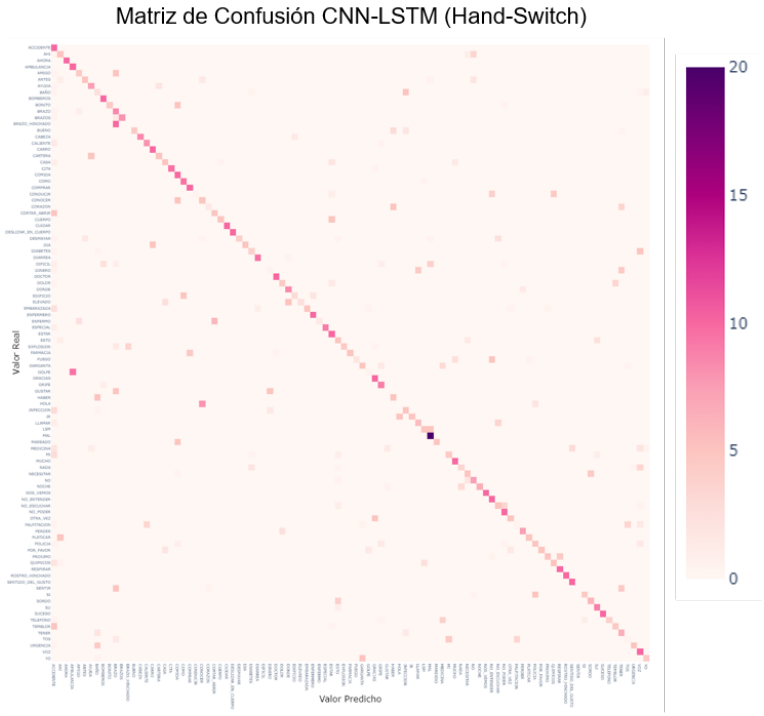

Figura 58. Matriz de confusión del modelo CNN-LSTM con el conjunto de Hand-Switch.

En la figura 54 se observa la matriz de confusión. Los elementos de la matriz se concentran en su mayoría en la diagonal principal pero ésta no se observa tan resaltada como en modelos anteriores. El accuracy de prueba alcanzado fue de **0.55**, el F1-Score de **0.5** y la precisión de **0.5**. Este modelo tardó aproximadamente lo mismo en entrenamiento y prueba que el modelo CNN-LSTM (Sampling).

En la tabla 5 se muestran los resultados del modelos CNN-LSTM con los dos conjuntos de datos.

Tabla 5. Tabla de resultados del modelo CNN-LSTM.

	Train Acc	Val Acc	Test Acc	Test F1-Score	Test Prec
Sampling	0.98	0.77	0.76	0.77	0.72
Hand-Switch	0.99	0.59	0.55	0.5	0.5

5.4. ResNet

A continuación se describirán los resultados de los modelos de aprendizaje profundo para el conjunto de entrenamiento creado utilizando el enfoque de re-muestreo (Sampling) y el enfoque Hand-Switch para el modelo ResNet.

5.4.1. Métricas de rendimiento de los modelos para el conjunto Sampling

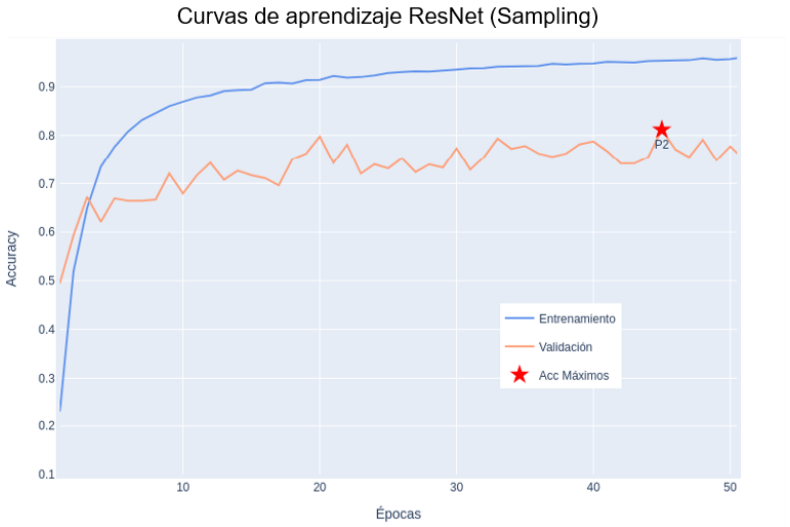

Figura 59. Curvas de aprendizaje del modelo ResNet con el conjunto de Sampling.

Como se mencionó en la Sección 4.4, la arquitectura ResNet representa el estado del arte en muchos problemas de clasificación. En este caso no fue la excepción, ya que es el modelo que mejor score de

exactitud alcanzó entrenando con el conjunto de Sampling. En las figuras 59 y 60 se muestran los resultados obtenidos de este modelo.

En la figura 59 se observan las curvas de aprendizaje obtenidas durante el entrenamiento del modelo. El score de accuracy de entrenamiento obtenido fue de **0.98** mientras que el accuracy de validación máximo fue de **0.825**. Esto implica poco sobre-ajuste del modelo entrenado.

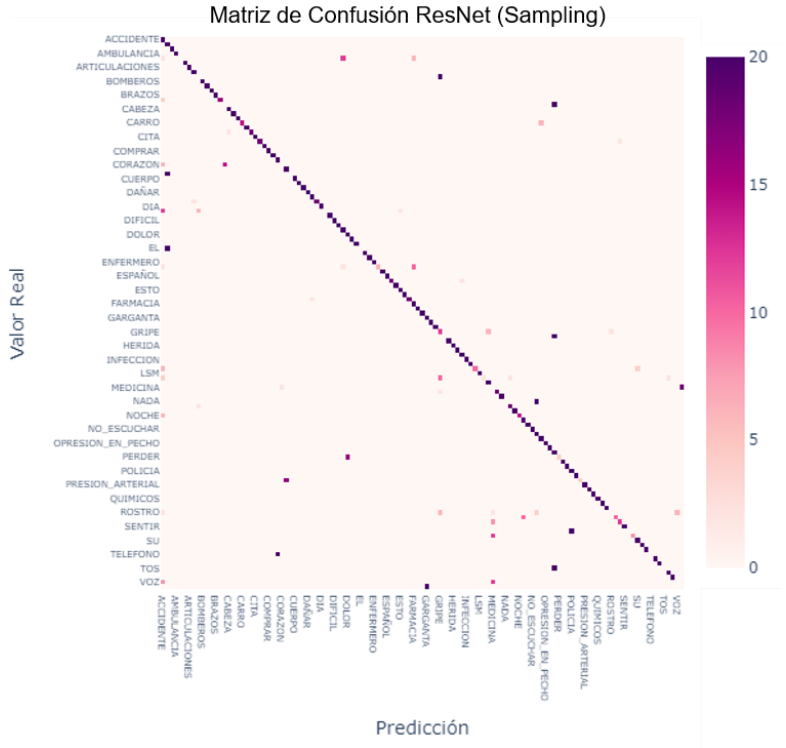

Figura 60. Matriz de confusión del modelo ResNet con el conjunto de Sampling.

En la figura 60 se observa su respectiva matriz de confusión sobre el conjunto de prueba. A comparación del modelo *baseline* y los modelos anteriores, se observan muchos más valores sobre la diagonal principal. El accuracy de prueba obtenido sobre el conjunto de datos fue **0.81**, mientras que el F1-Score y la precisión alcanzados fueron de **0.78** y **0.77** respectivamente. El modelo tardó **8 minutos** en entrenar y obtuvo las predicciones de clases sobre el conjunto de prueba en **0.77 segundos**.

5.4.2. Métricas de rendimiento de los modelos para el conjunto Hand-Switch

En las figuras 61 y 62, se muestran los resultados obtenidos a partir del modelo ResNet usando el conjunto de entrenamiento de enfoque Hand-Switch.

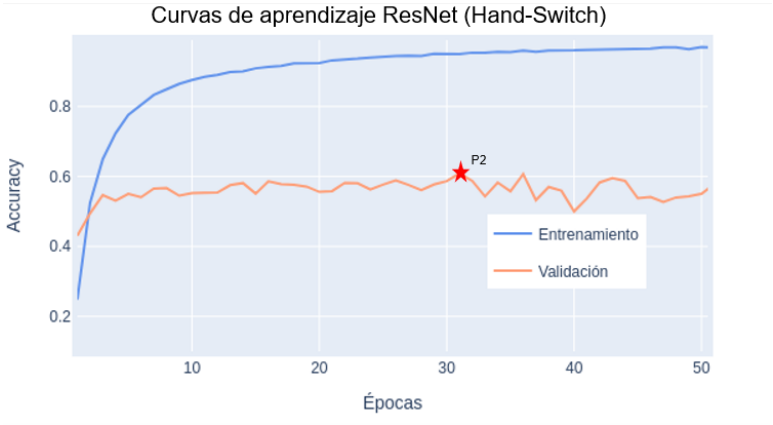

Figura 61. Curvas de aprendizaje del modelo ResNet con el conjunto de Hand-Switch.

En la figura 61 se observan las curvas de aprendizaje obtenidas durante el entrenamiento del modelo con el conjunto de datos Hand-Switch. El score de accuracy de entrenamiento obtenido fue de **0.99** mientras que el accuracy de validación máximo fue de **0.61**. Este modelo tiene mejores métricas de entrenamiento que los modelos anteriores que usan este conjunto de datos.

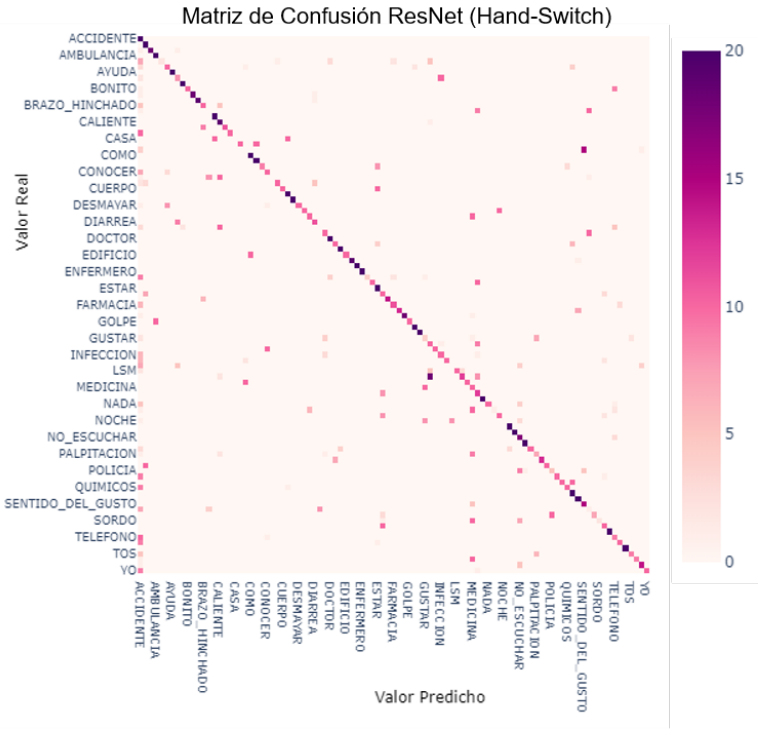

Figura 62. Matriz de confusión del modelo ResNet con el conjunto de Hand-Switch.

En la figura 62 se observa su respectiva matriz de confusión sobre el conjunto de prueba. A comparación del modelo *baseline* y los modelos de Sampling anteriores, se observan más dispersa pero aún conservando una tendencia sobre la diagonal principal. El accuracy de prueba obtenido sobre el conjunto de

datos fue **0.63**, mientras que el F1-Score y la precisión alcanzados fueron de **0.59** y **0.6** respectivamente. El modelo hizo tiempos promedio similares a los del modelo ResNet Sampling en entrenamiento y prueba.

En la tabla 6 se muestran los resultados del modelos ResNet con los dos conjuntos de datos.

Tabla 6. Tabla de resultados del modelo ResNet.

	Train Acc	Val Acc	Test Acc	Test F1-Score	Test Prec
Sampling	0.98	0.825	0.81	0.78	0.77
Hand-Switch	0.99	0.61	0.63	0.59	0.6

5.5. Comparación entre modelos

En esta sub-sección se compararán los modelos propuestos y los respectivos enfoques de data augmentation. En las tablas 7 y 8 se muestra el resumen de los resultados de los modelos para el conjunto de Sampling y Hand-Switch respectivamente. En la tabla 7, correspondiente al conjunto de Sampling, se resalta que el modelo con mejores métricas de rendimiento (exactitud, F1-Score, precisión y recall) sobre los datos de prueba, i.e. datos de los intérpretes expertos de LSM. El modelo con mejores métricas de rendimiento es el ResNet. Con respecto a los tiempo de ejecución, el modelo LSTM fue el que tomó menor costo computacional en su entrenamiento y en obtener las predicciones de clase para el conjunto de prueba. En la tabla 8 se resalta que también el modelo Resnet es aquel que alcanza mejores

Tabla 7. Tabla comparativa de modelos para el conjunto de entrenamiento Sampling.

	KNN (DTW)	LSTM	CNN-LSTM	ResNet
Test Accuracy	0.59	0.67	0.76	**0.81**
Test F1-Score	0.55	0.62	0.72	**0.77**
Test Precisión	0.54	0.64	0.77	**0.78**
Test Recall	0.59	0.67	0.76	**0.81**
Train Time	14 min	**7 min**	12 min	8 min
Test Time	33 min	**0.5 seg**	0.96 seg	0.7 seg

métricas de rendimiento para el conjunto de prueba. Las métricas de evaluación son significativamente menores usando el conjunto de datos con modificaciones en los keypoints (Hand-Switch). De hecho, los modelos LSTM y CNN-LSTM no generaron métricas superiores al modelo *baseline*. En la figura 63 se muestran los diagramas de caja y bigote con los *scores* de F1-Score (para las 100 clases) de cada uno de los modelos. La línea roja entre los diagramas de caja de cada modelo representan la media de F1-Score. Los modelos con la caja más compacta es el ResNet (para el conjunto de Sampling) y el LSTM (para el conjunto Hand-Switch). Sin embargo, el modelo LSTM (Hand-Switch) tiene un F1-score promedio de

Tabla 8. Tabla comparativa de modelos para el conjunto de entrenamiento Hand-Switch.

	KNN (DTW)	LSTM	CNN-LSTM	ResNet
Test Accuracy	0.59	0.59	0.55	**0.63**
Test F1-Score	0.55	0.52	0.5	**0.59**
Test Precisión	0.54	0.54	0.5	**0.6**
Test Recall	0.59	0.59	0.55	**0.63**
Train Time	14 min	**7 min**	12 min	8 min
Test Time	33 min	**0.5 seg**	0.96 seg	0.7 seg

0.55. El modelo con el F1-Score promedio más alto es también el ResNet (Sampling). Todos los modelos presentan una notable dispersión, es decir, los valores de F1-Score son muy variables por clase.

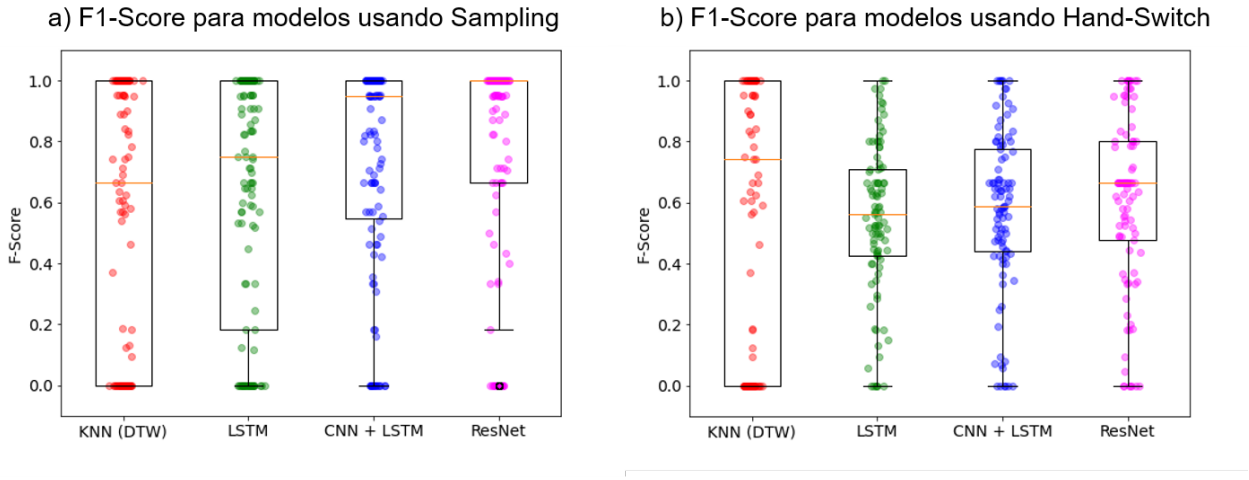

Figura 63. Comparación del F1-Score de todos los modelos.

5.6. Discusión de resultados

Las señas seleccionadas para esta tarea de clasificación son señas que son diferenciables entre sí. Sin embargo, algunas señas como las mostradas en la figura 64 son confundidas entre sí por los modelos de aprendizaje profundo. En la figura 64 se muestran dos señas que son confundidas por los modelos KNN, LSTM y ResNet entre sí. Ambas señas comparten tanto la forma de las manos (queirema) como la trayectoria de desplazamiento de las mismas (kinema), la diferencia entre ellas es la posición de las manos con respecto al cuerpo (toponema). Este tipo de confusiones son entendibles, ya que existen señas muy similares entre si.

a) Señal: CABEZA

b) Señal: CASA

Figura 64. Confusiones de los modelos. Nótese que ambas señas comparte queirema y kinema.

Otra confusión común entre los modelos son aquellas señas que comparten el mismo kinema y toponema. En la figura 65 se observa que las señas COMIDA y SENTIDO DEL GUSTO comparten la trayectoria de la mano y la posición de la mano respecto al cuerpo.

a) Señal: COMIDA

b) Señal: SENTIDO DEL GUSTO

Figura 65. Confusiones de los modelos. Nótese que ambas señas comparte toponema y kinema.

En las tablas 8 y 7 se observa que el modelo con mejores resultados de *accuracy*, F1-score, precisión y recall es el modelo ResNet para los dos conjuntos de datos (Sampling y Hand-Switch). Este modelo alcanza una exactitud de 0.81 en promedio para todas las clases. El modelo ResNet es el modelo más complejo en términos del número de parámetros de entrenamiento, i.e. pesos y sesgos, ya que cuenta con 272,225 parámetros. La tarea de clasificación de series espacio-temporales es una tarea compleja que no puede ser resuelta con un modelo lineal simple, se requiere un modelo robusto para darle solución a este problema. Los modelos de transferencia de aprendizaje para la clasificación de series de tiempo sin

duda son una opción que se puede considerar para aumentar la exactitud de los modelos de clasificación para un trabajo a futuro.

El conjunto de datos que obtuvo mejores resultados fue el conjunto Sampling sobre el conjunto Hand-Switch. Esto posiblemente debido al hecho de que las secuencias espacio-temporales son muy similares entre sí debido a que la variación del aumento o disminución de los dedos de la mano fue de menos del 10 %. Esto disminuye la variabilidad entre las muestras por cada clase. Este problema es posible que sea resuelto aumentando el porcentaje de aumento del tamaño de los dedos. Otras variantes a explorar sería el de rotar las secuencias y aplicar *zoom in* y *zoom out* de los archivos JSON. En la literatura consultada, se encontraron pocos resultados de LS para realizar esquemas de *data augmentation* sobre las secuencias de archivos JSON. Sin embargo, el enfoque de remuestreo de frames presenta buenos resultados.

Capítulo 6. Conclusiones

En este trabajo de tesis se implementaron 3 modelos de aprendizaje profundo para reconocer ideogramas que forman parte de la Lengua de Señas Mexicana. Se propuso un diccionario con 100 gestos con expertos en LSM y se llevaron a cabo grabaciones del diccionario propuesto de señas con 10 participantes inexpertos en LSM. Además se implementaron dos formas de *data augmentation* para incrementar el tamaño de este conjunto de ideogramas.

El modelo que mejores resultados obtuvo fue el modelo **ResNet**, este modelo obtuvo una exactitud de **0.81** en el conjunto de datos de prueba. A pesar de que el modelo con mejor costo computacional fue el modelo de LSTM, el modelo ResNet tiene resultados comparables en tiempo de entrenamiento y tiempo de prueba lo que lo hace adecuado para ser mejorado en el futuro.

Se realizaron dos conjuntos de datos de acuerdo a los dos esquemas de *data augmentation* usados. Con el esquema de Sampling se obtuvieron mejores resultados en todos los modelos. Con el conjunto Hand-Switch no se obtuvieron buenos resultados de forma general esto debido a que las secuencias espacio-temporales probablemente eran muy similares entre sí a pesar de cambiar la longitud de los dedos. Otros enfoques de modificación de los archivos JSON es aplicar rotación y operaciones de *zoom in* ó *zoom out*.

En este trabajo se concluyó que el modelo ResNet (recordando que es el modelo con mayor complejidad computacional) es el que mejor clasifica los conjuntos de señas. A pesar de que los otros modelos presentan resultados con menor exactitud promedio, estos modelos comparten muchas de las "confusiones" que el modelo ResNet. Estas confusiones son debidas a que varias señas comparten toponemas, queiremas y/o kinemas entre sí.

6.1. Trabajo a futuro

A continuación se enlistarán algunas observaciones sobre el trabajo que se puede realizar para enriquecer el trabajo presentado:

- Aumentar el diccionario de ideogramas de LSM. De esta forma se podrá realizar la clasificación de más señas pertenecientes a la LSM con el mismo contexto de emergencias u otro distinto.

- Aumentar las repeticiones de la seña para aumentar el número de ejemplos por clase. Enriquecer el registro de videos existente con videos de participantes nuevos ayudará a robustecer los modelos de aprendizaje profundo debido a que aumentará la variabilidad intra-clase del conjunto de datos.

- Probar diversas arquitecturas con transferencia de aprendizaje para obtener exactitudes más altas en la clasificación. Los modelos pre-entrenados para la clasificación de series de tiempo han sido poco estudiados en los problemas de reconocimiento de las Lenguas de Señas. Sin embargo, sería de sumo interés explorar los resultados que se pudieran generar aplicando este enfoque.

- Ampliar el trabajo de tesis usando modelos de traducción continuos de frases en LSM a español usando el diccionario propuesto. Los modelos aislados son útiles para reconocer seña a seña las glosas correspondientes a la LSM. Sin embargo, el estado del arte en cuanto a la traducción de Lenguas de Señas justifica el uso de modelos de aprendizaje profundo enfocados en *machine translation*.

- Realizar la traducción de texto en español a Lengua de Señas Mexicana. El siguiente paso en los flujos de traducción sería el de realizar la traducción a Lengua de Señas mediante *avatars* o generación de *keypoints*. Esta dirección de traducción implica distintos retos a los enfrentados en el trabajo presentado.

Literatura citada

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., y Zheng, X.. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Recuperado el 20 de junio del 2022 de `https://www.tensorflow.org/`.

Abraham, E., Nayak, A., y Iqbal, A.. 2019. Real-time translation of indian sign language using LSTM. En: 2019 Global Conference for Advancement in Technology (GCAT)..IEEE, pp. 1–5.

Adeyanju, I., Bello, O., y Adegboye, M.. 2021. Machine learning methods for sign language recognition: A critical review and analysis. Intelligent Systems with Applications, Vol. 12, pp. 200056.

Adimas, D. M., Rakun, E., y Hardianto, D.. 2019. Recognizing indonesian sign language gestures using features generated by elliptical model tracking and angular projection. En: 2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS)..IEEE, pp. 25–31.

Ahmed, A. M., Alez, R. A., Tharwat, G., Taha, M., Ghribi, W., Badawy, A. S., Changalasetty, S. B., y Bose, J. S. C.. 2017. Towards the design of automatic translation system from arabic sign language to arabic text. En: 2017 International Conference on Inventive Computing and Informatics (ICICI)..IEEE, pp. 325–330.

Al-Qurishi, M., Khalid, T., y Souissi, R.. 2021. Deep learning for sign language recognition: Current techniques, benchmarks, and open issues. IEEE Access, pp. 126917–126951.

Alpaydin, E.. 2020. Introduction to machine learning. MIT press, Cambridge.

Ameen, S. y Vadera, S.. 2017. A convolutional neural network to classify american sign language fingerspelling from depth and colour images. Expert Systems, Vol. 34(3), pp. e12197.

Antony, R., Paul, S., Alex, S., et al.. 2020. Sign language translation system. International Journal of Scientific Research & Engineering Trends, pp. 1486–1493.

Ariesta, M. C., Wiryana, F., Zahra, A., et al.. 2018. Sentence level indonesian sign language recognition using 3d convolutional neural network and bidirectional recurrent neural network. En: 2018 Indonesian Association for Pattern Recognition International Conference (INAPR)..IEEE, pp. 16–22.

Assaleh, K., Shanableh, T., y Zourob, M.. 2012. Low complexity classification system for glove-based arabic sign language recognition. En: International Conference on Neural Information Processing..Springer, pp. 262–268.

Athitsos, V., Neidle, C., Sclaroff, S., Nash, J., Stefan, A., Yuan, Q., y Thangali, A.. 2008. The american sign language lexicon video dataset. En: 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops..IEEE, pp. 1–8.

Bagnall, A., Lines, J., Hills, J., y Bostrom, A.. 2015. Time-series classification with cote: the collective of transformation-based ensembles. IEEE Transactions on Knowledge and Data Engineering, Vol. 27(9), pp. 2522–2535.

Bagnall, A., Lines, J., Bostrom, A., Large, J., y Keogh, E.. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data mining and knowledge discovery, Vol. 31(3), pp. 606–660.

Bahdanau, D., Serdyuk, D., Brakel, P., Ke, N. R., Chorowski, J., Courville, A., y Bengio, Y.. 2015. Task loss estimation for sequence prediction. arXiv preprint eprint:1511.06456. Recuperado de https://arxiv.org/pdf/1511.06456.

Bantupalli, K. y Xie, Y.. 2018. American sign language recognition using deep learning and computer vision. En: 2018 IEEE International Conference on Big Data (Big Data)..IEEE, pp. 4896–4899.

Barczak, A., Reyes, N., Abastillas, M., Piccio, A., y Susnjak, T.. 2011. A new 2d static hand gesture colour image dataset for asl gestures. Massey University Research Letters, pp. 12—-20.

Bengio, Y., Ducharme, R., y Vincent, P.. 2000. A neural probabilistic language model. Advances in neural information processing systems, Vol. 13.

Bragg, D., Koller, O., Bellard, M., Berke, L., Boudreault, P., Braffort, A., Caselli, N., Huenerfauth, M., Kacorri, H., Verhoef, T., et al.. 2019. Sign language recognition, generation, and translation: An interdisciplinary perspective. En: The 21st international ACM SIGACCESS conference on computers and accessibility..pp. 16–31.

Brownlee Jason. 2019. A gentle introduction to batch normalization for deep neural networks. Recuperado el 12 de junio del 2022 de https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/.

Bungeroth, J., Stein, D., Dreuw, P., Ney, H., Morrissey, S., Way, A., y van Zijl, L.. 2008. The ATIS sign language corpus. En: Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco..European Language Resources Association.

Burad V. 2019. La glosa: Un sistema de notación para la lengua de señas. Recuperado el 12 de junio del 2022 de https://cultura-sorda.org/la-glosa-un-sistema-de-notacion-para-la-lengua-de-senas/#:~:text=Caracter%C3%ADsticas%20de%20la%20glosa,caracter%C3%ADsticas%20sem%C3%A1nticas%2C%20sint%C3%A1cticas%20y%20morfol%C3%B3gicas.

Caballero-Morales, S.-O. y Trujillo-Romero, F.. 2013. 3d modeling of the mexican sign language for a speech-to-sign language system. Computación y Sistemas, Vol. 17(4), pp. 593–608.

Camgoz, N. C., Hadfield, S., Koller, O., Ney, H., y Bowden, R.. 2018. Neural sign language translation. En: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition..pp. 7784–7793.

Camgoz, N. C., Koller, O., Hadfield, S., y Bowden, R.. 2020. Sign language transformers: Joint end-to-end sign language recognition and translation. En: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition..pp. 10023–10033.

Cao, Z., Simon, T., Wei, S.-E., y Sheikh, Y.. 2017. Realtime multi-person 2d pose estimation using part affinity fields. En: Proceedings of the IEEE conference on computer vision and pattern recognition..pp. 7291–7299.

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., y Sheikh, Y.. 2019. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. arXiv preprint eprint:1812.08008. . Recuperado de http://openaccess.thecvf.com/content_cvpr_2017/papers/Cao_Realtime_Multi-Person_2D_CVPR_2017_paper.pdf.

Carmona-Arroyo, G., Ríos-Figueroa, H., y Avendaño, M.. 2021. Mexican sign language static alphabet recognition using 3d affine invariants. Machine Vision Inspection Systems, Vol. 2, pp. 171–192.

Cheok, M. J., Omar, Z., y Jaward, M. H.. 2019. A review of hand gesture and sign language recognition techniques. International Journal of Machine Learning and Cybernetics, Vol. 10(1), pp. 131–153.

Cihan Camgoz, N., Hadfield, S., Koller, O., y Bowden, R.. 2017. Subunets: End-to-end hand shape and continuous sign language recognition. En: Proceedings of the IEEE international conference on computer vision..pp. 3056–3065.

Cruz-Aldrete, M.. 2014. Hacia la construcción de un diccionario de lengua de señas mexicana. Revista de Investigación, Vol. 38(83), pp. 57–80.

Cui, R., Liu, H., y Zhang, C.. 2017. Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. En: Proceedings of the IEEE conference on computer vision and pattern recognition..pp. 7361–7369.

Cui, R., Liu, H., y Zhang, C.. 2019. A deep neural framework for continuous sign language recognition by iterative training. IEEE Transactions on Multimedia, Vol. 21(7), pp. 1880–1891.

Dabwan, B. A.. 2020. Convolutional neural network-based sign language translation system. International Journal of Engineering, Science and Mathematics, Vol. 9(6), pp. 47–57.

De Coster, M., Van Herreweghe, M., y Dambre, J.. 2019. Towards automatic sign language corpus annotation using deep learning. En: 6th workshop on sign language translation and avatar technology.

De Coster, M., Van Herreweghe, M., y Dambre, J.. 2020. Sign language recognition with transformer networks. En: 12th International Conference on Language Resources and Evaluation..European Language Resources Association (ELRA), pp. 6018–6024.

De Fleischmann, M. E. S. y Pérez, R. G.. 2011. Manos con voz: diccionario de lengua de señas mexicana. Consejo Nacional para Prevenir la Discriminación, Ciudad de México.

Decher Davina. 2021. Artificial intelligence, machine learning, deep learning—buzzwords explained. Recuperado el 23 de agosto del 2022 de `https://www.lengoo.com/blog/ai-buzzwords/`.

Deng, L.. 2012. The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, Vol. 29(6), pp. 141–142.

Eidos Team. 2019. Training a hand detector like the openpose one in tensorflow. Recuperado el 20 de agosto del 2022 de `https://eidos-ai.medium.com/training-a-hand-detector-like-the-openpose-one-in-tensorflow-45c5177d6679`.

Fang, B., Co, J., y Zhang, M.. 2017. Deepasl: Enabling ubiquitous and non-intrusive word and sentence-level sign language translation. En: Proceedings of the 15th ACM conference on embedded network sensor systems..pp. 1–13.

Farooq, U., Rahim, M. S. M., Sabir, N., Hussain, A., y Abid, A.. 2021. Advances in machine translation for sign language: approaches, limitations, and challenges. Neural Computing and Applications, pp. 1–43.

Faurot, K., Dellinger, D., Eatough, A., y Parkhurst, S.. 1999. The identity of mexican sign as a language. Summer Institute of Linguistics.

Fawaz Ismail, H., Forestier, G., Weber, J., Idoumghar, L., y Muller, P.-A.. 2018. Transfer learning for time series classification. En: 2018 IEEE International Conference on Big Data (Big Data)..Vol. 1, pp. 1367–1376.

Fernandes, L., Dalvi, P., Junnarkar, A., y Bansode, M.. 2020. Convolutional neural network based bidirectional sign language translation system. En: 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)..IEEE, pp. 769–775.

Forster, J., Schmidt, C., Koller, O., Bellgardt, M., y Ney, H.. 2014. Extensions of the sign language recognition and translation corpus rwth-phoenix-weather. En: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)..pp. 1911–1916.

Galicia, R., Carranza, O., Jiménez, E., y Rivera, G.. 2015. Mexican sign language recognition using movement sensor. En: 2015 IEEE 24th International Symposium on Industrial Electronics (ISIE)..IEEE, pp. 573–578.

Garcia-Bautista, G., Trujillo-Romero, F., y Diaz-Gonzalez, G.. 2016. Advances to the development of a basic mexican sign-to-speech and text language translator. En: Applications of Digital Image Processing XXXIX..International Society for Optics and Photonics, Vol. 9971, p. 99713E.

García-Bautista, G., Trujillo-Romero, F., y Caballero-Morales, S. O.. 2017. Mexican sign language recognition using kinect and data time warping algorithm. En: 2017 International Conference on Electronics, Communications and Computers (CONIELECOMP)..IEEE, pp. 1–5.

Goodfellow, I., Bengio, Y., y Courville, A.. 2016. Deep Learning. MIT Press. Recuperado el 08 de Mayo del 2022 de: http://www.deeplearningbook.org.

Google's Machine Learning Crash Course. 2020. Machine learning crash cours, multi-class neural networks: Softmax. Recuperado el 07 de junio del 2022 de https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax?hl=es_419.

Grandini, M., Bagli, E., y Visani, G.. 2020. Metrics for multi-class classification: an overview. arXiv preprint eprint:2008.05756. Recuperado de https://arxiv.org/pdf/2008.05756.

Guo, D., Zhou, W., Li, H., y Wang, M.. 2018. Hierarchical LSTM for sign language translation. En: Proceedings of the AAAI Conference on Artificial Intelligence..Vol. 32.

Harini, R., Janani, R., Keerthana, S., Madhubala, S., y Venkatasubramanian, S.. 2020. Sign language translation. En: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)..IEEE, pp. 883–886.

He, K., Zhang, X., Ren, S., y Sun, J.. 2016. Deep residual learning for image recognition. En: Proceedings of the IEEE conference on computer vision and pattern recognition..pp. 770–778.

He, S.. 2019. Research of a sign language translation system based on deep learning. En: 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)..IEEE, pp. 392–396.

Hisham, B. y Hamouda, A.. 2021. Arabic sign language recognition using ada-boosting based on a leap motion controller. International Journal of Information Technology, Vol. 13(3), pp. 1221–1234.

Hossen, M., Govindaiah, A., Sultana, S., y Bhuiyan, A.. 2018. Bengali sign language recognition using deep convolutional neural network. En: 2018 joint 7th international conference on informatics, electronics & vision (iciev) and 2018 2nd international conference on imaging, vision & pattern recognition (icIVPR)..IEEE, pp. 369–373.

Huang, J., Zhou, W., Li, H., y Li, W.. 2015. Sign language recognition using real-sense. En: 2015 IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP)..IEEE, pp. 166–170.

Huang, J., Zhou, W., Zhang, Q., Li, H., y Li, W.. 2018. Video-based sign language recognition without temporal segmentation. En: Thirty-Second AAAI Conference on Artificial Intelligence.

Idoko, J. B.. 2020. Deep Learning-Based on Sign Language Translation System. Tesis de doctorado, Near East University. pp. 93.

Ioffe, S. y Szegedy, C.. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. En: International conference on machine learning..PMLR, pp. 448–456.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., y Muller, P.-A.. 2019. Deep learning for time series classification: a review. Data mining and knowledge discovery, Vol. 33(4), pp. 917–963.

Jimenez, J., Martin, A., Uc, V., y Espinosa, A.. 2017. Mexican sign language alphanumerical gestures recognition using 3d haar-like features. IEEE Latin America Transactions, Vol. 15(10), pp. 2000–2005.

Kamal, S. M., Chen, Y., Li, S., Shi, X., y Zheng, J.. 2019. Technical approaches to chinese sign language processing: a review. IEEE Access, Vol. 7, pp. 96926–96935.

Kapoor, P., Mukhopadhyay, R., Hegde, S. B., Namboodiri, V., y Jawahar, C.. 2021. Towards automatic speech to sign language generation. arXiv preprint eprint:2106.12790. Recuperado de `https://arxiv.org/pdf/2106.12790`.

Kim, T., Keane, J., Wang, W., Tang, H., Riggle, J., Shakhnarovich, G., Brentari, D., y Livescu, K.. 2017. Lexicon-free fingerspelling recognition from video: Data, models, and signer adaptation. Computer Speech & Language, Vol. 46, pp. 209–232.

Kingma, D. P. y Ba, J.. 2014. Adam: A method for stochastic optimization. arXiv preprint eprint:1412.6980. Recuperado de `https://arxiv.org/pdf/1412.6980?source=post_page`.

Ko, S.-K., Son, J. G., y Jung, H.. 2018. Sign language recognition with recurrent neural network using human keypoint detection. En: Proceedings of the 2018 conference on research in adaptive and convergent systems..pp. 326–328.

Ko, S.-K., Kim, C. J., Jung, H., y Cho, C.. 2019. Neural sign language translation based on human keypoint estimation. Multidisciplinary Digital Publishing Institute: Applied Sciences, Vol. 9(13), pp. 2683.

Koehn, P.. 2017. Neural machine translation. arXiv preprint eprint:1709.07809. Recuperado de `https://arxiv.org/pdf/1709.07809.pdf%2C2ndpublicdraft`.

Koller, O., Ney, H., y Bowden, R.. 2016. Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. En: Proceedings of the IEEE conference on computer vision and pattern recognition..pp. 3793–3802.

Koller, O., Zargaran, S., Ney, H., y Bowden, R.. 2018. Deep sign: Enabling robust statistical continuous sign language recognition via hybrid cnn-hmms. International Journal of Computer Vision, Vol. 126(12), pp. 1311–1325.

Konstantinidis, D., Dimitropoulos, K., y Daras, P.. 2018a. A deep learning approach for analyzing video and skeletal features in sign language recognition. En: 2018 IEEE international conference on imaging systems and techniques (IST)..IEEE, pp. 1–6.

Konstantinidis, D., Dimitropoulos, K., y Daras, P.. 2018b. Sign language recognition based on hand and body skeletal data. En: 2018-3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)..IEEE, pp. 1–4.

Krizhevsky, A., Sutskever, I., y Hinton, G. E.. 2012. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, Vol. 25, pp. 84–90.

Kumar, G. y Bhatia, P. K.. 2014. A detailed review of feature extraction in image processing systems. En: 2014 Fourth international conference on advanced computing & communication technologies..IEEE, pp. 5–12.

Kumar, S. S., Wangyal, T., Saboo, V., y Srinath, R.. 2018. Time series neural networks for real time sign language translation. En: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)..IEEE, pp. 243–248.

Lakhotiya, H., Pandita, H. S., y Shankarmani, R.. 2021. Real time sign language recognition using image classification. En: 2021 2nd International Conference for Emerging Technology (INCET)..IEEE, pp. 1–4.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., y Jackel, L. D.. 1989. Backpropagation applied to handwritten zip code recognition. Neural computation, Vol. 1(4), pp. 541–551.

Lemaréchal, C.. 2012. Cauchy and the gradient method. Doc Math Extra, Vol. 251(254), pp. 10.

Lines, J. y Bagnall, A.. 2015. Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery, Vol. 29(3), pp. 565–592.

Lines, J., Taylor, S., y Bagnall, A.. 2016. Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. En: 2016 IEEE 16th international conference on data mining (ICDM)..IEEE, pp. 1041–1046.

Lissi, M., Svartholm, K., y González, M.. 2012. El enfoque bilingüe en la educación de sordos: Sus implicaciones para la enseñanza y aprendizaje de la lengua escrita. Recuperado el 01 de julio del 2022 de https://cultura-sorda.org/el-enfoque-bilingue-en-la-educacion-de-sordos-sus-implicancias-para-la-ensenanza-y-aprendizaje-de-la-lengua-escrita/.

López, E., Velásquez, J., Eleuterio, R., y Gil, L.. 2015. Interfaz de reconocimiento de movimientos para el lenguaje de señas mexicano implementando el kinect. Revista Aristas: Investigación Básica y Aplicada, Vol. 4(7), pp. 130–133.

Luis-Pérez, F. E., Trujillo-Romero, F., y Martínez-Velazco, W.. 2011. Control of a service robot using the mexican sign language. En: Mexican International Conference on Artificial Intelligence..Springer, pp. 419–430.

Martínez, A. M., Wilbur, R. B., Shay, R., y Kak, A. C.. 2002. Purdue rvl-slll asl database for automatic recognition of american sign language. En: Proceedings. Fourth IEEE International Conference on Multimodal Interfaces..IEEE, pp. 167–172.

Martinez-Seis, B., Pichardo-Lagunas, O., Rodriguez-Aguilar, E., y Saucedo-Diaz, E.-R.. 2019. Identification of static and dynamic signs of the mexican sign language alphabet for smartphones using deep learning and image processing. Research in Computing Science, Vol. 148(11), pp. 199–211.

Martínez-Gutiérrez, M., Rojano-Cáceres, J., Benitez-Guerrero, E., y Sánchez-Barrera, H.. 2019. Data acquisition software for sign language recognition. Research in Computing Science, Vol. 148, pp. 205–211.

Mejía-Peréz, K., Córdova-Esparza, D.-M., Terven, J., Herrera-Navarro, A.-M., García-Ramírez, T., y Ramírez-Pedraza, A.. 2022. Automatic recognition of mexican sign language using a depth camera and recurrent neural networks. Multidisciplinary Digital Publishing Institute: Applied Sciences, Vol. 12(11), pp. 5523.

Meng, L. y Li, R.. 2021. An attention-enhanced multi-scale and dual sign language recognition network based on a graph convolution network. Multidisciplinary Digital Publishing Institute: Sensors, Vol. 21(4), pp. 1120.

Mitchell, T. M. et al.. 1997. Machine learning. McGraw-hill, New York.

Morales, E. M., Aparicio, O. V., Arguijo, P., Armenta, R. Á. M., y López, A. H. V.. 2019. Traducción del lenguaje de señas usando visión por computadora. Res. Comput. Sci., Vol. 148(8), pp. 79–89.

Moryossef, A., Tsochantaridis, I., Aharoni, R., Ebling, S., y Narayanan, S.. 2020. Real-time sign language detection using human pose estimation. En: European Conference on Computer Vision..Springer, pp. 237–248.

Mubashira, N. y James, A.. 2020. Transformer network for video to text translation. En: 2020 International Conference on Power, Instrumentation, Control and Computing (PICC)..IEEE, pp. 1–6.

Nunnari, F., España-Bonet, C., y Avramidis, E.. 2021. A data augmentation approach for sign-language-to-text translation in-the-wild. En: 3rd Conference on Language, Data and Knowledge (LDK 2021)..Schloss Dagstuhl-Leibniz-Zentrum für Informatik, p. 8.

Nweke, H. F., Teh, Y. W., Al-Garadi, M. A., y Alo, U. R.. 2018. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. Expert Systems with Applications, Vol. 105, pp. 233–261.

Orbay, A. y Akarun, L.. 2020. Neural sign language translation by learning tokenization. En: 2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)..IEEE, pp. 222–228.

Park, C.-I. y Sohn, C.-B.. 2020. Data augmentation for human keypoint estimation deep learning based sign language translation. Multidisciplinary Digital Publishing Institute: Electronics, Vol. 9, pp. 1257.

Pichardo-Lagunas, O.. 2015. Resource creation for automatic translation system from texts in spanish into mexican sign language. Research in Computing Science, Vol. 100, pp. 129–137.

Pichardo-Lagunas, O.. 2016. Sistema de traducción directa de español a lsm con reglas marcadas. Research in Computing Science, Vol. 115, pp. 29–41.

Porfirio, A. J., Wiggers, K. L., Oliveira, L. E., y Weingaertner, D.. 2013. Libras sign language hand configuration recognition based on 3d meshes. En: 2013 IEEE International Conference on Systems, Man, and Cybernetics..IEEE, pp. 1588–1593.

Potter, L. E., Araullo, J., y Carter, L.. 2013. The leap motion controller: a view on sign language. En: Proceedings of the 25th Australian computer-human interaction conference: augmentation, application, innovation, collaboration..pp. 175–178.

Rakun, E., Arymurthy, A. M., Stefanus, L. Y., Wicaksono, A. F., y Wisesa, I. W. W.. 2018. Recognition of sign language system for indonesian language using long short-term memory neural networks. Advanced Science Letters, Vol. 24(2), pp. 999–1004.

Rao, G. A., Syamala, K., Kishore, P., y Sastry, A.. 2018. Deep convolutional neural networks for sign language recognition. En: 2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)..IEEE, pp. 194–197.

Regan Mark. 2016. K nearest neighbors dynamic time warping. Recuperado el 22 de junio del 2022 de https://github.com/markdregan/K-Nearest-Neighbors-with-Dynamic-Time-Warping.

Rioux-Maldague, L. y Giguere, P.. 2014. Sign language fingerspelling classification from depth and color images using a deep belief network. En: 2014 Canadian Conference on Computer and Robot Vision..IEEE, pp. 92–97.

Rodriguez, J., Chacón, J., Rangel, E., Guayacan, L., Hernández, C., Hernandez, L., y Martinez, F.. 2020. Understanding motion in sign language: A new structured translation dataset. En: Proceedings of the Asian Conference on Computer Vision.

Rumelhart, D. E., Hinton, G. E., y Williams, R. J.. 1986. Learning representations by back-propagating errors. Nature, Vol. 323(6088), pp. 533–536.

Saunders, B., Camgoz, N. C., y Bowden, R.. 2021. Continuous 3d multi-channel sign language production via progressive transformers and mixture density networks. International Journal of Computer Vision, pp. 1–23.

Schembri, A., Fenlon, J., Rentelis, R., Reynolds, S., y Cormier, K.. 2013. Building the british sign language corpus. Language Documentation & Conservation, Vol. 7, pp. 136–154.

Sedmidubsky, J. y Zezula, P.. 2021. Efficient combination of classifiers for 3d action recognition. Multimedia Systems, Vol. 27(5), pp. 941–952.

Sharma, A., Sharma, N., Saxena, Y., Singh, A., y Sadhya, D.. 2021. Benchmarking deep neural network approaches for indian sign language recognition. Neural Computing and Applications, Vol. 33(12), pp. 6685–6696.

Shohieb, S. M., Elminir, H. K., y Riad, A. M.. 2015. Signsworld atlas; a benchmark arabic sign language database. Journal of King Saud University-Computer and Information Sciences, Vol. 27(1), pp. 68–76.

Shrivastava Abhishek. 2020. Underfitting vs just right vs overfitting in machine learning. Recuperado el 01 de junio del 2022 de https://www.kaggle.com/getting-started/166897.

Solís, F., Toxqui, C., Martínez, D., et al.. 2015. Mexican sign language recognition using jacobi-fourier moments. Scientific Research Journal: Engineering, Vol. 7(10), pp. 700.

Solís, F., Martínez, D., Espinoza, O., et al.. 2016. Automatic mexican sign language recognition using normalized moments and artificial neural networks. Scientific Research Journal: Engineering, Vol. 8(10), pp. 733.

Solís-V, J.-F., Toxqui-Quitl, C., Martínez-Martínez, D., y Margarita, H.-G.. 2014. Mexican sign language recognition using normalized moments and artificial neural networks. En: Optics and Photonics for Information Processing VIII..International Society for Optics and Photonics, Vol. 9216, p. 92161A.

Sonare, B., Padgal, A., Gaikwad, Y., y Patil, A.. 2021. Video-based sign language translation system using machine learning. En: 2021 2nd International Conference for Emerging Technology (INCET)..IEEE, pp. 1–4.

Sosa-Jiménez, C. O., Ríos-Figueroa, H. V., Rechy-Ramírez, E. J., Marin-Hernandez, A., y González-Cosío, A. L. S.. 2017. Real-time mexican sign language recognition. En: 2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)..IEEE, pp. 1–6.

Starner, T., Weaver, J., y Pentland, A.. 1998. Real-time american sign language recognition using desk and wearable computer based video. IEEE Transactions on pattern analysis and machine intelligence, Vol. 20(12), pp. 1371–1375.

Stoll, S., Camgöz, N. C., Hadfield, S., y Bowden, R.. 2018. Sign language production using neural machine translation and generative adversarial networks. En: Proceedings of the 29th British Machine Vision Conference (BMVC 2018)..University of Surrey.

Stoll, S., Camgoz, N. C., Hadfield, S., y Bowden, R.. 2020. Text2sign: Towards sign language production using neural machine translation and generative adversarial networks. International Journal of Computer Vision, Vol. 128(4), pp. 891–908.

Tang, A., Lu, K., Wang, Y., Huang, J., y Li, H.. 2015. A real-time hand posture recognition system using deep neural networks. ACM Transactions on Intelligent Systems and Technology (TIST), Vol. 6(2), pp. 1–23.

Trujillo-Romero, F. y Caballero-Morales, S.-O.. 2013. 3d data sensing for hand pose recognition. En: CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing..IEEE, pp. 109–113.

Trujillo-Romero, F. y García-Bautista, G.. 2021. Reconocimiento de palabras de la lengua de señas mexicana utilizando información rgb-d. Revista Electrónica de Computación, Informática, Biomédica y Electrónica., Vol. 10, pp. 1–23.

Vedak, O., Zavre, P., Todkar, A., y Patil, M.. 2019. Sign language interpreter using image processing and machine learning. International Research Journal of Engineering and Technology (IRJET).

Von Agris, U. y Kraiss, K.-F.. 2007. Towards a video corpus for signer-independent continuous sign language recognition. Gesture in Human-Computer Interaction and Simulation, Lisbon, Portugal, May, Vol. 11, pp. 2.

Wang, N., Ma, Z., Tang, Y., Liu, Y., Li, Y., y Niu, J.. 2016. An optimized scheme of mel frequency cepstral coefficient for multi-sensor sign language recognition. En: International Conference on Smart Computing and Communication..Springer, pp. 224–235.

Wang, S., Guo, D., Zhou, W.-g., Zha, Z.-J., y Wang, M.. 2018. Connectionist temporal fusion for sign language translation. En: Proceedings of the 26th ACM international conference on Multimedia..pp. 1483–1491.

Wang, Z., Yan, W., y Oates, T.. 2017. Time series classification from scratch with deep neural networks: A strong baseline. En: 2017 International joint conference on neural networks (IJCNN)..IEEE, pp. 1578–1585.

Wei, S., Chen, X., Yang, X., Cao, S., y Zhang, X.. 2016. A component-based vocabulary-extensible sign language gesture recognition framework. Multidisciplinary Digital Publishing Institute: Sensors, Vol. 16(4), pp. 556.

Wu, D. y Shao, L.. 2014. Multimodal dynamic networks for gesture recognition. En: Proceedings of the 22nd ACM international conference on Multimedia..pp. 945–948.

Yang, Q. y Wu, X.. 2006. 10 challenging problems in data mining research. International Journal of Information Technology & Decision Making, Vol. 5(04), pp. 597–604.

Yang, S. y Zhu, Q.. 2017. Video-based chinese sign language recognition using convolutional neural network. En: 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)..IEEE, pp. 929–934.

Yin, K.. 2020. Sign language translation with transformers. arXiv preprint eprint:2004.00588. Recuperado de https://www.academia.edu/download/62815371/transformer_slt_320200404-119107-10ildcf.pdf.

Yuan, T., Sah, S., Ananthanarayana, T., Zhang, C., Bhat, A., Gandhi, S., y Ptucha, R.. 2019. Large scale sign language interpretation. En: 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)..IEEE, pp. 1–5.

Zahedi, M., Dreuw, P., Rybach, D., Deselaers, T., y Ney, H.. 2006. Continuous sign language recognition-approaches from speech recognition and available data resources. En: Second workshop on the representation and processing of sign languages: lexicographic matters and didactic scenarios..pp. 21–24.

Anexo A

.1. Descenso por gradiente

En el aprendizaje profundo la mayor parte de los algoritmos involucran optimización. La optimización se refiere a la tarea de minimizar o maximizar[1] una función $f(x)$ alterando el valor x. La función que se busca minimizar es conocida como **función objetivo**, véase figura 66. Cuando la función objetivo es minimizada es conocida como **función de costo**. El valor que minimiza la función con el superíndice *, por ejemplo:

$$x^* = \text{argmin} f(x) \tag{20}$$

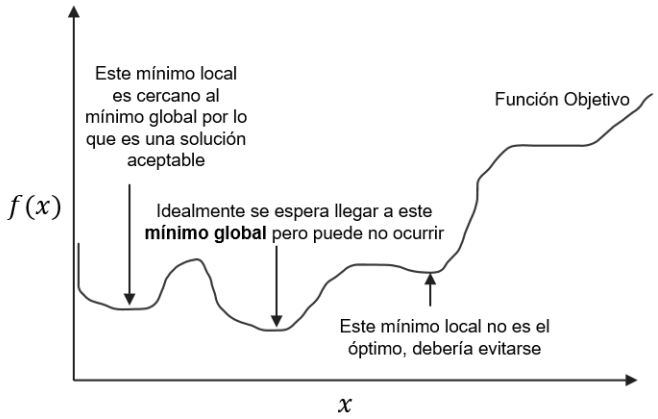

Figura 66. Aproximación de la minimización de la función objetivo. Los algoritmos de optimización no siempre encuentran el mínimo global cuando existen múltiples mínimos locales. En el contexto de aprendizaje profundo generalmente se aceptan soluciones locales mientras tengan muy bajos valores en la función de costo. Modificada de Goodfellow et al. (2016).

Suponiendo que existe una función $y = f(x)$, dónde x e y son números reales. La derivada de esta función se denota como $f'(x)$. La derivada $f'(x)$ nos da la pendiente de $f(x)$ en el punto x. En otras palabras, especifica cómo escalar un pequeño cambio en la entrada para obtener el cambio correspondiente en la salida $f(x + \epsilon) \approx f(x) + \epsilon f'(x)$.

La derivada es útil para minimizar la función debido a que nos dice cómo cambiar el valor x para tener un menor valor de y. Es posible minimizar $f(x)$ moviendo x en pequeños pasos con el signo opuesto de la derivada. Esta técnica es el algoritmo de **descenso por gradiente** (Lemaréchal, 2012).

[1] Usualmente en aprendizaje profundo se busca la minimización de una función.

.2. Descenso por gradiente estocástico (SGD)

La mayor parte del aprendizaje profundo se basa en este algoritmo de optimización. El algoritmo de Descenso por Gradiente Estocástico (SGD) es una extensión del algoritmo por descenso por gradiente explicado en la sub-sección anterior. Un problema en aprendizaje de máquina son los enormes conjuntos de datos de entrenamiento para asegurar una buena generalización pero esto implica mayor costo computacional.

La funciones de costo usadas por los algoritmos de ML suelen descomponerse en una suma sobre los ejemplos de entrenamiento. De acuerdo con Goodfellow et al. (2016), el condicional negativo de verosimilitud logarítmica de los datos de entrenamiento pueden ser escritos como:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta) \tag{21}$$

donde m es el número de ejemplos de entrenamiento, L es la función de pérdida o de costo $L(x, y, \theta) = -\log p(y|x;\theta)$.

Para esta función de costo aditiva, el descenso por gradiente es:

$$\nabla_\theta J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta) \tag{22}$$

El costo computacional de esta operación es $O(m)$. En cuanto el tamaño del conjunto de entrenamiento se vuelve más grande, el tiempo de cálculo del paso del gradiente se vuelve muy largo. El SGD realiza el cálculo de la expectativa usando un conjunto de muestras. Específicamente en cada paso del algoritmo, se muestrea un **minibatch** o mini-lote de ejemplos $B = x^{(1)}, ..., x^{(m')}$ uniforme del conjunto de entrenamiento. El tamaño del minilote m' es fijo y usualmente pequeño. Se puede ajustar un conjunto de entrenamiento con billones de ejemplos usando actualizaciones calculadas con solo cientos de ejemplos. El gradiente estimado es:

$$g = \frac{1}{m'}\nabla_\theta \sum_{i=1}^{m'} L(x^{(i)}, y^{(i)}, \theta) \tag{23}$$

usando ejemplos del minibatch B. A pesar de que los algoritmos de optimización no garantizan ni siquiera aterrizar en un mínimo local en un tiempo de ejecución razonable, normalmente encuentran un valor lo suficientemente bajo en la función de costo que es útil.

Anexo B

.1. Lista de glosas

1. ACCIDENTE
2. AHÍ
3. AHORA
4. AMBULANCIA
5. AMIGO
6. ANTES
7. AYUDA
8. BAÑO
9. BOMBEROS
10. BONITO
11. BRAZO
12. BRAZO_HINCHADO
13. BRAZOS
14. BUENO
15. CABEZA
16. CALIENTE
17. CARRO
18. CARTERA
19. CASA
20. CITA
21. COMIDA
22. COMO
23. COMPRAR
24. CONDUCIR
25. CONOCER
26. CORAZÓN
27. CUERPO
28. CUIDAR
29. DESMAYAR
30. DÍA
31. DIABETES
32. DIARREA
33. DIFÍCIL
34. DINERO
35. DOCTOR
36. DOLOR
37. DONDE
38. EDIFICIO
39. ELEVADO
40. EMBARAZADA
41. ENFERMERO
42. ENFERMO
43. ESPECIAL
44. ESTAR
45. ESTO
46. EXPLOSIÓN
47. FARMACIA
48. FUEGO
49. GARGANTA
50. GOLPE
51. GRACIAS
52. GRIPE
53. GUSTAR
54. HABER
55. HERIDA_ABIERTA
56. HOLA
57. HOSPITAL
58. INFECCIÓN
59. IR
60. LLAMAR
61. LSM
62. MAL
63. MAREADO

64. MEDICINA
65. MI
66. MUCHO
67. NADA
68. NECESITAR
69. NO
70. NO_ENTENDER
71. NO_ESCUCHAR
72. NO_NADA
73. NO_PODER
74. NOCHE
75. NOS_VEMOS
76. OTRA_VEZ
77. PALPITACIÓN
78. PERDER
79. PLATICAR
80. POLICÍA
81. POR_FAVOR
82. PRÓXIMO
83. QUEMADURA_EN_CUERPO
84. QUÍMICOS
85. RESPIRAR
86. ROSTRO
87. ROSTRO_HINCHADO
88. SENTIDO_DEL_GUSTO
89. SENTIR
90. SI
91. SORDO
92. SU
93. SUCESO
94. TELÉFONO
95. TEMBLOR
96. TENER
97. TOS
98. URGENCIA
99. VOZ
100. YO