



UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

Reconocimiento de Lenguaje de Señas

Por:

Juan E. Cifuentes, Leonardo Da Costa, Rodolfo L. Sánchez

Realizado con la asesoría de:

Ivette C. Martínez, Carlos D. Castillo

PROYECTO DE GRADO

Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero en Computación

Sartenejas, Diciembre 2010



UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

ACTA FINAL PROYECTO DE GRADO

Reconocimiento de Lenguaje de Señas

Presentado por:

Juan E. Cifuentes, Leonardo Da Costa, Rodolfo L. Sánchez

Este Proyecto de Grado ha sido aprobado por el siguiente jurado examinador:

Carolina Chang

Gabrigla Montoya

Ivette C. Martínez (Tutor)

Sartenejas, Diciembre 2010



MENCIÓN DE HONOR

Quienes suscribimos, profesores CAROLINA CHANG, GABRIELA MONTOYA e IVETTE CAROLINA MARTÍNEZ, miembros del jurado designado por la Coordinación de la Carrera de Ingeniería de Computación de la Universidad Simón Bolívar para considerar y evaluar el trabajo de grado titulado **RECONOCIMIENTO DE LENGUAJE DE SEÑAS**, presentado por JUAN E. CIFUENTES, LEONARDO DA COSTA y RODOLFO SÁNCHEZ para optar al título de INGENIERO DE COMPUTACIÓN, emitimos el veredicto **APROBADO CON MENCIÓN DE HONOR** por unanimidad. La mención de honor está sustentada en que este Trabajo de Grado, adicionalmente a cumplir con los requisitos necesarios para otorgar la Mención Especial establecidos en el documento "Criterios para otorgar una Mención Especial en Proyectos de Grado Pasantías Largas e Intermedias" del Decanato de Estudios Profesionales, las siguientes razones:

1. La propuesta de un método base para el reconocimiento de señas en Lenguaje de Señas Venezolano (LSV). En base a este método se implementó un prototipo funcional, robusto, extensible y que funciona con datos verdaderos.
2. La calidad del trabajo teórico hecho en la descripción del método base que incluye la integración de descriptores geométricos como la transformación de distancia con clasificadores tanto lineales como no lineales para el reconocimiento de las señas.
3. Las mejoras propuestas e implementadas sobre el método base hacen factible la realización del reconocimiento del LSV a tiempo real. Una herramienta que permita este reconocimiento tendría un alto impacto para la facilitación de la comunicación entre la población con discapacidades auditivas y la población venezolana en general.
4. La calidad del trabajo experimental realizado, que incluye comparación con métodos del estado del arte y la realización de pruebas estadísticas que demuestran que las mejoras propuestas al método base incrementan significativamente la velocidad y precisión del reconocimiento.

En fe de todo lo cuál levantamos y firmamos la presente acta en el Valle de Sartenejas, Baruta, Edo. Miranda, a los 6 días del mes de enero de dos mil once.

CAROLINA CHANG

Jurado

Gabriela J. Montoya C.
GABRIELA MONTOYA

Jurado

IVETTE C. MARTÍNEZ

Tutor

RECONOCIMIENTO DE LENGUAJE DE SEÑAS
Por
Juan E. Cifuentes, Leonardo Da Costa y Rodolfo L. Sánchez

RESUMEN

El lenguaje de señas es una lengua natural que consiste en un conjunto de señas gesto-espaciales percibidas a través de la vista, que permiten comunicarse a personas que presentan deficiencias auditivas o del habla. Con la finalidad de facilitar el proceso de comunicación entre las personas con deficiencia y la sociedad en general, nace el interés de desarrollar mecanismos capaces de facilitar su interpretación. En Venezuela es utilizado por la comunidad de sordos o mudos el Lenguaje de Señas Venezolano, LSV por sus siglas.

En este trabajo se desarrolla un Traductor LSV-Castellano, cuyo objetivo es traducir un video del deletreo de una palabra en LSV al Castellano. Para lograrlo, se utilizan técnicas de manipulación de imágenes y los siguientes métodos de reconocimiento de patrones: Clasificador de Formas a través de Distancias Internas, Comparación de Plantillas con Transformación de Distancias, Perceptrón, Red Neural, SVM-Pegasos por Lotes Pequeños y SVM-Pegasos con Kernel. Para evaluar la calidad de clasificación de frames individuales se realizaron un conjunto de experimentos sobre una serie de videos de deletreo de palabras. Al evaluar los tiempos de ejecución y la precisión de cada método, se determinó que la Red Neural posee el mejor desempeño entre los métodos estudiados.

Para la clasificación de videos se propuso un método basado en Filtros Lineales y se crearon dos estructuras para mejorar su desempeño: Árbol de Decisión de Clusters y Diccionario Trie. Se realizaron experimentos para la evaluación del desempeño del proceso de traducción aplicando cada una de estas estructuras. A partir de estos experimentos se obtuvo que la estructura Árbol de Decisión de Clusters y el Diccionario Trie produjeron traducciones más precisas y mayor cantidad de frames procesados por segundo. Se realizó un experimento adicional con la integración de ambas estructuras, sobre el método basado en Filtros Lineales, generando un mejor desempeño en el proceso de traducción de videos.

Índice general

Índice de tablas	IX
Índice de figuras	XI
Lista de Abreviaturas	XV
1 Introducción	1
2 Marco Teórico	5
2.1 Lenguaje de Señas	5
2.2 Manipulación de Imágenes	6
2.2.1 Modelo de Color RGB	6
2.2.2 Croma	6
2.2.3 Convolución	7
2.2.3.1 Convolución Gaussiana	7
2.2.4 Segmentación	7
2.2.4.1 Segmentación Binaria	8
2.2.5 Filtro de Extracción de Bordes Canny	8
2.2.6 Transformación de Distancias	9
2.3 Clasificación de Imágenes	9
2.3.1 Algoritmos de Clasificación por Plantillas	10
2.3.1.1 Clasificación de Formas a través de Distancias Internas . . .	10
2.3.2 Algoritmos por Aprendizaje Supervisado	11
2.3.2.1 Normalización por Varianza y Media	11
2.3.2.2 Normalización por Mínimos y Máximos	11
2.3.2.3 Perceptrón	11
2.3.2.4 Red Neural	13
2.3.2.5 SVM-Pegasos	13
2.3.2.5.1 SVM-Pegasos por Lotes Pequeños	14

2.3.2.5.2	SVM-Pegasos con Kernel	14
2.4	Estructuras Complementarias	15
2.4.1	Filtros Lineales	15
2.4.2	Ventana Deslizante	15
2.4.3	Cluster	16
2.4.4	Árbol de Decisión	16
2.4.5	Trie	16
3	Etapas de Pre-Procesamiento (Manipulación de Frames)	17
3.1	Extracción de Áreas de Interés	17
3.2	Eliminación de Ruido	18
3.3	Recorte y Obtención de Líneas	20
4	Etapas de Procesamiento (Traducción de Videos)	21
4.1	Etiquetado de Frames	22
4.1.1	Algoritmos de Clasificación por Plantillas	23
4.1.1.1	Comparación de Plantillas con Transformación de Distancias	24
4.1.1.2	Clasificación de Formas a través de Distancias Internas . . .	25
4.1.2	Algoritmos de Clasificación por Aprendizaje	26
4.1.2.1	Perceptrón	27
4.1.2.2	Red Neural	27
4.1.2.3	SVM-Pegasos por Lotes Pequeños	28
4.1.2.4	SVM-Pegasos con Kernel	29
4.2	Traducción de Video	29
4.2.1	Proceso de Traducción	30
5	Mejoras al Proceso de Traducción de Videos	35
5.1	Empleo de Árbol de Decisión de Clusters	36
5.2	Empleo de Diccionario Trie	40
5.3	Empleo de Integración de Árboles de Decisión de Clusters y Diccionario Trie	44
6	Evaluación Experimental	48

6.1	Experimento #1: Determinar el Mejor Método de Clasificación	49
6.1.1	Condiciones y Parámetros	49
6.1.2	Resultados y Análisis	50
6.2	Experimento #2: Determinar el Mejor Proceso de Traducción de Video . . .	52
6.2.1	Condiciones y Parámetros	54
6.2.2	Resultados y Análisis	54
7	Conclusiones y Recomendaciones	58
	Bibliografía	61
A	Pseudo-Códigos Implementados	63
A.1	Perceptrón	63
A.2	Red Neural	64
A.3	SVM-Pegasos	65
A.4	SVM-Pegasos por Lotes Pequeños	66
A.5	SVM-Pegasos con Kernel	67
B	Lista de Tablas	68
C	Experimentos de Clasificación de Señas Estáticas	71
C.1	Experimento #1: Clasificador por Comparación de Plantillas con Transformación de Distancias	71
C.2	Experimento #2: Clasificador de Formas a través de Distancias Internas . .	73
C.3	Experimento #3: Clasificador Perceptrón	75
C.4	Experimento #4: Clasificador Red Neural	78
C.5	Experimento #5: Clasificador SVM-Pegasos por Lotes Pequeños	82
C.6	Experimento #6: Clasificador SVM-Pegasos con Kernel	84
D	Entonación de las Redes Neurales	90
D.1	Mejora del Método Elegido	90
E	Configuración del Árbol de Decisión de Clusters	92
E.1	Nivel #1: Descripción de las plantillas utilizadas	92
E.2	Nivel #2: Descripción de las plantillas utilizadas	93

Índice de tablas

6.1	Estadísticas de los Métodos de Clasificación	51
6.2	Tipos de traducción y la cantidad de ocurrencias en los resultados obtenidos bajo cada uno de los procesos de traducción empleados	54
6.3	Estadísticas de fps a partir de los resultados obtenidos bajo cada uno de los procesos de traducción empleados	56
B.1	Abecedario en Lenguaje de Señas Venezolano bajo su representación en RGB, Canny y Etiquetado.	68
B.1	Abecedario en Lenguaje de Señas Venezolano bajo su representación en RGB, Canny y Etiquetado.	69
B.1	Abecedario en Lenguaje de Señas Venezolano bajo su representación en RGB, Canny y etiquetado.	70
C.1	Tiempos de Ejecución por Configuración	73
C.2	Tiempos de Ejecución por Configuración del método SCUID	74
C.3	Porcentajes e Iteraciones Correpondientes a los Mejores Resultados por Con- figuración del Perceptrón	78
C.4	Porcentajes e Iteraciones Correpondientes a los Mejores Resultados por Con- figuración de la Red neural	81
C.5	Tabla resumen de las configuraciones de σ del SVM-Pegasos con Kernel . . .	88
E.1	Representación de las plantillas de contorno que definen los tres valores posi- bles del atributo Forma.	93
E.2	Porcentajes de Clasificación de la Red Neural Lateral para el Cluster de Forma 1 Modificando la Configuración de sus Capas Ocultas	94
E.3	Porcentajes de Clasificación de la Red Neural Lateral para el Cluster de Forma 3 Modificando la Configuración de sus Capas Ocultas	94

E.4	Porcentajes e Iteraciones Correspondientes a los Mejores Resultados por Configuración para Red Neural del Cluster Forma 1	99
E.5	Porcentajes e Iteraciones Correspondientes a los Mejores Resultados por Configuración para Red Neural del Cluster Forma 3	100
E.6	Configuración final de las redes neurales encargadas de definir el atributo Lateral sobre cada uno de los clusters correspondientes a Forma 1 y Forma 3 .	101
F.1	Resultados del Proceso Base de Traducción sobre 37 videos	102
F.2	Resultados del Proceso Base de Traducción empleando Árbol de Decisión de Clusters sobre 37 videos	103
F.3	Resultados del Proceso Base de Traducción empleando Diccionario Trie sobre 37 videos	104
F.4	Resultados del Proceso Base de Traducción empleando Árbol de Decisión de Clusters y Diccionario Trie sobre 37 videos	105

Índice de figuras

1.1	Representación gráfica de las dos etapas principales del mecanismo de traducción implementado.	4
2.1	Ejemplo del uso de Croma en un Noticiero de Televisión.	7
2.2	Ejemplo de la Aplicación de Convolución Gaussiana con una Máscara de 3x3.	8
2.3	Ejemplo de la Aplicación de Segmentación Binaria.	8
2.4	Ejemplo de la Aplicación del Filtro de Extracción de Bordes Canny.	9
2.5	Ejemplo de la Aplicación de Transformación de Distancias.	10
2.6	Ejemplo de la Representación Visual de un Trie	16
3.1	Ejemplo del Proceso de la sub-etapa Extracción de Áreas de Interés (EAI).	18
3.2	Ejemplo del Proceso de la sub-etapa Eliminación de Ruido (ER).	19
3.3	Ejemplo del Proceso de la sub-etapa Recorte y Obtención de Líneas (ROL).	20
4.1	Ejemplo de etiquetado de un frame de video.	23
4.2	Ejemplo del Pre-Procesamiento adicional del método SCUID.	26
4.3	Ejemplo de Desplazamiento de Ventanas de 5 y 10 frames.	31
4.4	Ejemplos de las diferentes situaciones en el análisis de una Ventana.	33
5.1	Ejemplo de confusión entre las señas U y R, donde la seña a clasificar es una R y es etiquetada como U debido al parecido entre éstas.	35
5.2	Representación de la sub-etapa Recorte y Obtención de Líneas (ROL) de la Etapa de Pre-Procesamiento con proceso extra.	37
5.3	Representación del Árbol de decisión de Clusters con el atributo Forma.	37
5.4	Representación del Árbol de Decisión de Clusters con los atributos Forma y Lateral.	38
5.5	Ejemplo de la representación visual del Diccionario Trie implementado	41

6.1	Gráfica de comparación entre los métodos de clasificación estudiados, a través de la precisión y la cantidad de frames procesados por segundo (fps)	50
C.1	Número de Plantillas Vs Porcentaje de Clasificación empleando Comparación de Plantillas con DT	72
C.2	Número de Plantillas Vs Porcentaje de Clasificación empleando SCUID	74
C.3	Gráficas de Porcentajes de Clasificación de Perceptrones, bajo una Normalización por Mínimos y Máximos variando la tasa de aprendizaje y Tasas de Aprendizaje 0,01 y 0,03	75
C.4	Gráficas de Porcentajes de Clasificación de Perceptrones, bajo una Normalización por Mínimos y Máximos variando la tasa de aprendizaje y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2	76
C.5	Gráficas de Porcentajes de Clasificación de Perceptrones, bajo una Normalización por Varianza y Media variando la tasa de aprendizaje y Tasas de Aprendizaje 0,01 y 0,03	76
C.6	Gráficas de Porcentajes de Clasificación de Perceptrones, bajo una Normalización por Varianza y Media variando la tasa de aprendizaje y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2	77
C.7	Porcentajes de Clasificación de una Red Neural Modificando la Configuración de sus Capas Ocultas	79
C.8	Gráfica de Porcentajes de Clasificación de Red Neural bajo una Normalización por Mínimos y Máximos variando la tasa de aprendizaje y Tasas de Aprendizaje 0,01 y 0,03	79
C.9	Gráfica de Porcentajes de Clasificación de Red Neural bajo una Normalización por Mínimos y Máximos variando la tasa de aprendizaje y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2	80
C.10	Gráfica de Porcentajes de Clasificación de Red Neural bajo una Normalización por Varianza y Media variando la tasa de aprendizaje y Tasas de Aprendizaje 0,01 y 0,03	80

C.11	Gráfica de Porcentajes de Clasificación de Red Neural bajo una Normalización por Varianza y Media variando la tasa de aprendizaje y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2	81
C.12	Porcentajes de Clasificación bajo valores λ de SVM-Pegasos por Lotes Pequeños con una normalización por Mínimos y Máximos	83
C.13	Gráfica de barras de las configuraciones λ del SVM-Pegasos por Lotes Pequeños con una normalización de Varianza y Media	84
C.14	Porcentajes de clasificación de un SVM-Pegasos con Kernel con Normalización por Mínimos y Máximos, bajo los valores de λ evaluados y σ : 2^{-15} 2^{-13} 2^{-11} 2^{-9}	85
C.15	Porcentajes de clasificación de un SVM-Pegasos con Kernel con Normalización por Mínimos y Máximos, bajo los valores de λ evaluados y σ : 2^{-7} 2^{-5} 2^{-3} 2^{-1} 2^1 2^3	86
C.16	Porcentajes de clasificación de un SVM-Pegasos con Kernel con Normalización por Varianza y Media, bajo los valores de λ evaluados y σ : 2^{-15} 2^{-13} 2^{-11} 2^{-9} 2^{-7} 2^{-5}	87
C.17	Porcentajes de clasificación de un SVM-Pegasos con Kernel con Normalización por Varianza y Media, bajo los valores de λ evaluados y σ : 2^{-3} 2^{-1} 2^1 2^3 . . .	88
D.1	Porcentajes de Clasificación de la Red Neural Variando las Iteraciones de Entrenamiento	90
E.1	Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 1, bajo una Normalización por Mínimos y Máximos y Tasas de Aprendizaje 0,01 y 0,03	95
E.2	Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 1, bajo una Normalización por Mínimos y Máximos y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2	96
E.3	Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 1, bajo una Normalización por Varianza y Media y Tasas de Aprendizaje 0,01 y 0,03	96

E.4	Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 1, bajo una Normalización por Varianza y Media y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2	97
E.5	Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 3, bajo una Normalización por Mínimos y Máximos y Tasas de Aprendizaje 0,01 y 0,03	97
E.6	Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 3, bajo una Normalización por Mínimos y Máximos y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2	98
E.7	Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 3, bajo una Normalización por Varianza y Media y Tasas de Aprendizaje 0,01 y 0,03	98
E.8	Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 3, bajo una Normalización por Varianza y Media y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2	99

Lista de Abreviaturas

CRT	<i>Cathode Ray Tube.</i> Turbo de Rayos Catódicos.
CPTD	Comparación de Plantillas con Transformación de Distancias.
DE	Desviación Estándar.
DT	<i>Distance Transformation.</i> Transformación de Distancias.
EAI	Extracción de Áreas de Interés.
ER	Eliminación de Ruido.
FPS	<i>Frames Per Second.</i> Frames Por Segundo.
GB	<i>Gigabyte.</i>
GHz	<i>Gigahertz</i>
IBM	International Business Machines.
LSV	Lenguaje de Señas Venezolano.
MAX	Máximo.
MB	<i>Megabyte.</i>
MED	Media.
MIN	Mínimo.
MIT	<i>Massachusetts Institute of Technology.</i>
Pegasos	<i>Primal Estimated sub-GrAdient SOLver for SVM.</i> Solucionador Fundamental de Sub-Gradientes Estimados para SVM.
PX	<i>Pixel.</i> Píxel.
RAM	<i>Random Access Memory.</i> Memoria de Acceso Aleatorio.
RBF	<i>Radial Basic Function.</i>
RGB	Red, Green and Blue. Rojo, Verde y Azul-
ROL	Recorte y Obtención de Líneas.
SCUID	<i>Shape Classification Using the Inner-Distance.</i> Clasificación de Formas a través de Distancias Internas
SEG	Segundo.
SiSi	Say It, Sign It.
SVM	<i>Support Vector Machine.</i> Máquina de Vector de Soporte.
UCAB	Universidad Católica Andrés Bello.

Capítulo 1

Introducción

El lenguaje de señas es una lengua natural de expresión que consiste en un conjunto de señas gesto-espaciales definidas, las cuales son percibidas a través de la vista. Este tipo de lenguaje es comúnmente utilizado por individuos que presentan deficiencias auditivas o del habla. El lenguaje de señas puede variar de una región a otra debido a los contextos socio-culturales de cada población. En Venezuela es utilizado, por la comunidad sorda, el Lenguaje de Señas Venezolano, LSV por su siglas [16].

La Organización Mundial de la Salud plantea que en el 2005, aproximadamente 278 millones de personas poseían una discapacidad auditiva profunda alrededor del mundo, donde el 80 % de ellos vivían en países sub-desarrollados [14]. Gracias al análisis estadístico aportado por M. Lewis [15] se estima que la población de sordos en Venezuela es de 1.246.674 sobre un total de 26.726.000 personas aproximadamente.

Globalmente la producción actual de mecanismos que ayudan a la gente con esta deficiencia, i.e. prótesis auditivas, es menor al 10 % de la necesidad total. En países en vías de desarrollo, sólo 1 de cada 40 personas con esta necesidad poseen algún mecanismo de ayuda a esta discapacidad. Viendo la baja disponibilidad de soporte existente, para facilitar el proceso de comunicación entre las personas con deficiencia y la sociedad en general, nace el interés de desarrollar mecanismos capaces de facilitar el proceso de interpretación y enseñanza del lenguajes de señas.

En estos últimos años se han desarrollado algunos mecanismos para ayudar a la interpretación y enseñanza del lenguaje de señas. La compañía IBM desarrolló un sistema llamado “Say It, Sign It” [7] (SiSi por sus siglas), capaz de traducir el lenguaje hablado a lenguaje de señas. SiSi captura la voz de una persona a través de un micrófono, mientras que en la pantalla de un computador se muestra un avatar que ejecuta su traducción en lenguaje de

señas.

En Venezuela también se han realizado trabajos relacionados con la enseñanza e interpretación de lenguaje de señas. En el año 2008, el ingeniero L. Terán presentó como trabajo de tesis, para optar por el título de Ingeniero en Informática de la UCAB, un software educativo para el uso del alfabeto manual LSV como herramienta de enseñanza del español en niños sordos [19]. Este software está conformado por un conjunto de juegos, donde los niños pueden aprender el sistema de deletreo manual y a la vez las letras del abecedario español. Sin embargo este sistema no posee ningún mecanismo capaz de interpretar el lenguaje de señas ejecutado por el individuo que lo utiliza.

Con la finalidad de desarrollar un mecanismo para interpretar el Lenguaje de Señas Venezolano, en este trabajo se presenta un reconocedor de lenguaje de señas, cuyo objetivo es traducir un video al Castellano en el cual es ejecutado el deletreo de una palabra en LSV utilizando su alfabeto manual. La idea base para el desarrollo del mecanismo de traducción es extraer el área de interés del video, la mano, como una forma que será analizada bajo una representación tridimensional referente al volumen en espacio-tiempo. Esta forma de representación fue propuesta en el año 2007 por M. Irani et al. [2], donde se propone la representación de acciones humanas, como caminar, correr y bailar, en tres dimensiones que conforman volúmenes a partir de siluetas humanas a través del tiempo.

En este trabajo, para lograr la correcta extracción de la mano, es utilizado un guante verde a partir del cual se desarrolló un mecanismo de extracción de áreas de color utilizando métodos de visión por computadora como: Croma, Segmentaciones, Convoluciones y Canny. Una vez extraído el guante, se implementaron diversos algoritmos de clasificación con la finalidad de etiquetar a cada frame con un valor correspondiente a la letra que representa la seña. Los algoritmos implementados se dividieron en dos grupos, donde el primero abarca a los algoritmos de clasificación por plantillas, mientras que el segundo contiene algoritmos basados en aprendizaje. Los algoritmos de clasificación por plantillas utilizados se basan en el trabajo publicado por H. Ling y D. Jacobs [9], que consiste en la clasificación de formas usando distancias internas, y el uso de Transformación de Distancias propuesto por G. Borgefors [3]. Los algoritmos de aprendizaje estudiados fueron: Perceptrón[11], Redes Neuronales[11] y

las simplificaciones de SVM propuestas por Yoram Singer et al. [18] , Pegasos por Lotes y Pegasos con Kernel.

Además de la clasificación individual de frames, se presenta un proceso de traducción de videos basado en el análisis de conjuntos de frames a través de filtros lineales, para poder refinar la clasificación de frames individuales convirtiéndolos en una palabra. Seguido a este proceso se presentan dos estructuras: Árbol de Decisión de Clusters y Diccionario Trie, las cuales son utilizadas junto al planteamiento de filtros lineales, para conseguir mejoras a nivel de precisión y tiempos en el proceso de traducción de videos.

El trabajo está estructurado en siete capítulos, incluida la introducción. El segundo capítulo, Marco Teórico, presenta los conceptos básicos necesarios para el análisis y desarrollo del mecanismo de Reconocimiento de Lenguaje de Señas. En este capítulo se pueden encontrar conceptos como: lenguaje de señas, RGB, Croma, Convolución, Filtro de Extracción de Bordes Canny, Segmentación, Normalizaciones, Algoritmos de Clasificación por Plantillas, Algoritmos de Clasificación por Aprendizaje Supervisado, Filtro Lineal, Ventana Deslizante, Cluster, Árbol de Decisión y Trie.

Los siguientes tres capítulos engloban lo referente al desarrollo del mecanismo encargado de la traducción de videos en Lenguaje de Señas Venezolano (por sus siglas LSV) al Castellano. El Capítulo 3, Etapa de Pre-Procesamiento (Manipulación de Frames), describe el proceso utilizado para la extracción del guante empleando técnicas de filtrado de imágenes sobre los frames del video de entrada. El Capítulo 4, Etapa de Procesamiento (Traducción de Videos), presenta las diferentes técnicas de clasificación de frames propuestas y un mecanismo base de traducción de videos de palabras deletreadas en LSV. El Capítulo 5, Mejoras al Proceso de Traducción de Videos, presenta las estructura creadas para mejorar el proceso de traducción base con respecto a precisión y tiempo de ejecución. A modo de ilustración en la Figura 1.1 se representa el mecanismo de traducción de videos compuesto por los Capítulos 3, 4 y 5.

El Capítulo 6, llamado Evaluación Experimental, presenta los experimentos realizados durante el desarrollo del proyecto y los resultados obtenidos. Dentro de este Capítulo se presentan dos experimentos principales, donde en el primero se contrastan todos los métodos

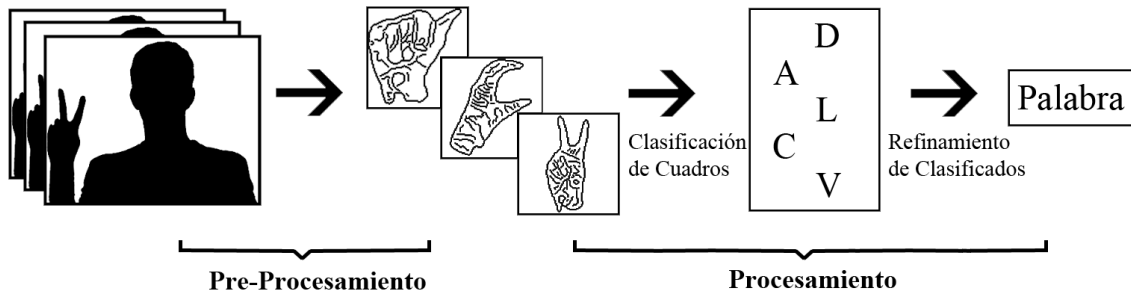


Figura 1.1: Representación gráfica de las dos etapas principales del mecanismo de traducción implementado.

de clasificación de frames individuales, presentados con la finalidad de elegir el más apto para este problema en particular. Seguido a éste, se presenta un segundo experimento donde se contrasta el proceso base de traducción con las estructuras propuestas para la obtención de mejoras en el proceso de traducción a través de la aplicación de éstos a un conjunto de videos, con la finalidad de buscar el mejor mecanismo para este problema. Finalmente, como cierre a este trabajo, el Capítulo 7 presenta las conclusiones obtenidas y propuestas para trabajos futuros relacionados al mecanismo descrito.

Capítulo 2

Marco Teórico

En este Capítulo se plantean los fundamentos teóricos necesarios para la comprensión del proyecto. Se hace una introducción al lenguaje de señas y las técnicas necesarias para la traducción automática de videos con LSV descritas en tres secciones principales: Manipulación de Imágenes, Clasificación de Imágenes y Estructuras Complementarias, con los elementos teóricos correspondientes a cada uno.

2.1. Lenguaje de Señas

El lenguaje de señas es utilizado generalmente por personas con deficiencias auditivas y/o del habla, logrando a través de éste establecer una comunicación con otras personas. El lenguaje de señas es una lengua natural de expresión que consiste en un conjunto de gestos y señas que permiten la comunicación entre el individuo que lo utiliza y aquellas personas capaces de interpretarlas. Las señas generalmente son ejecutadas por medio de las manos las cuales son acompañadas, en algunas ocasiones, de gestos corporales y faciales. No existe un lenguaje de señas universal, al igual que existen diversos lenguajes en el mundo dependiendo de la región y comunidad de sordos que la conforman, por ejemplo en Reino Unido es utilizado el *British Sign Language* (BSL), en Estados Unidos el *American Sign Language* (ASL) y en Venezuela el Lenguaje de Señas Venezolano (LSV).

Los lenguajes de señas poseen un alfabeto manual mediante el cual las personas con deficiencias auditivas y/o del habla, que saben leer y escribir, representan las letras del alfabeto con el que se escribe el lenguaje hablado. Gracias al alfabeto manual un individuo puede ejecutar el deletreo de palabras del lenguaje hablado a través del lenguaje de señas.

2.2. Manipulación de Imágenes

Un video esta compuesto por una secuencia de imágenes llamadas frames, las cuales pueden ser representadas en un computador a través de estructuras matriciales de 3 dimensiones denominadas imágenes digitales. Estas representaciones pueden usar un modelo de color RGB y pueden ser manipuladas aplicando técnicas de visión por computador tales como Croma, Filtros Gaussianos, Segmentación, Detección de Bordes y Transformación de Distancias descriptas a continuación.

2.2.1. Modelo de Color RGB

El modelo de color RGB [5], consiste en la combinación de los tres colores primarios de luz: rojo, verde y azul, de donde se deriva su nombre RGB por las siglas del inglés de *Red*, *Green* y *Blue*. Los tres canales de color, R G y B, pueden ser combinados con diferentes intensidades generando una extensa paleta de colores. Los valores de intensidad de cada canal de color son representados de manera numérica generalmente con valores entre 0 y 1.

2.2.2. Croma

Croma, es una técnica de composición visual entre dos imágenes o frames, que consiste en la sustitución de una parte de la imagen por otra. Esta técnica es utilizada generalmente en cine y televisión para realizar una composición entre un individuo y un fondo ajeno a él.

Para lograr la sustitución de áreas de una imagen por otras, se utiliza de fondo una pantalla de color sólido y uniforme. Generalmente este fondo es de color verde o azul, el cual a través del uso de computadoras es sustituido digitalmente por otra imagen. Es importante que el área que se desea mantener no posea algún tono de color igual o muy parecido al color sólido utilizado en la pantalla, para evitar que se extraigan algunas de las regiones que se desean mantener en la imagen final.

A continuación, en la Figura 2.1, se muestra un ejemplo del uso de la técnica croma en televisión, donde un programa de noticias está siendo grabado sobre una pantalla verde, la cual al momento de su transmisión por televisión es sustituida por una imagen de fondo que complementa las noticias narradas.



Figura 2.1: Ejemplo del uso de Croma en un Noticiero de Televisión.

2.2.3. Convolución

La convolución de dos señales se define como un operador matemático $(*)$ entre dos funciones $f(x)$ y $g(x)$, escrita como $h(x) = f(x) * g(x)$, donde $h(x)$ representa la señal generada [5]. El valor que toma $h(x)$ en cada punto es la integral del producto de $f(x)$ con la función de filtro $g(x)$ del punto original.

2.2.3.1. Convolución Gaussiana

La convolución Gaussiana se caracteriza por su operador el cual tiene forma de campana de Gauss. En el procesamiento de imágenes, esta convolución es usada para filtrar el ruido presente en una imagen a través del suavizado del tono de los píxeles con respecto a sus vecinos. La intensidad del suavizado se determinará a través del tamaño de la máscara de filtro utilizada (función de filtro). La distribución Gaussiana bidimensional tiene la forma [5]:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.1)$$

En la Figura 2.2 se muestra un ejemplo de la aplicación de convolución Gaussiana con una máscara de 3x3 sobre una imagen, donde a la izquierda se encuentra la imagen original y a la derecha la imagen con la Convolución Gaussiana ya aplicada.

2.2.4. Segmentación

La segmentación consiste en el proceso de identificación de sub-conjuntos dentro de un todo, tomando en consideración una o más características que puedan diferenciar los elementos de dicho conjunto. En el área de procesamiento de imágenes, esta técnica es utilizada

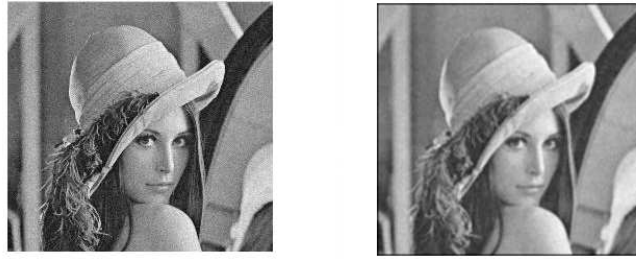


Figura 2.2: Ejemplo de la Aplicación de Convolución Gaussiana con una Máscara de 3x3.

para la separación de grupos de píxeles a través de la intensidad de tonos de color presentes en una imagen.

2.2.4.1. Segmentación Binaria

En el caso de la segmentación binaria de una imagen, la identificación de sub-conjuntos de píxeles separa a estos en dos categorías numéricas, unos y ceros. Esta identificación se realiza a través de la discriminación de áreas de tonos de color similar, donde al área representativa se le asignará uno de los valores posibles y al resto de las áreas el otro valor. Esta segmentación es importante para el estudio de objetos determinados e identificación de patrones dentro de una imagen.

A continuación, en la Figura 2.3 se muestra un ejemplo de la aplicación de segmentación binaria sobre una imagen, donde a la izquierda se encuentra la imagen original y a la derecha la imagen segmentada.

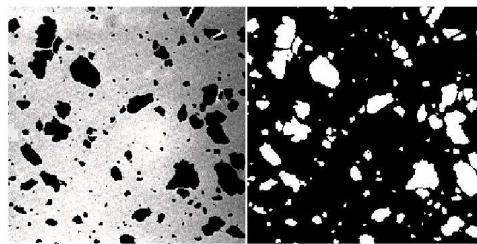


Figura 2.3: Ejemplo de la Aplicación de Segmentación Binaria.

2.2.5. Filtro de Extracción de Bordes Canny

Canny es considerado uno de los mejores métodos de detección de bordes, el cual está basado en la primera derivada empleando máscaras de convolución. Este filtro surge del trabajo

de J. Canny en su aplicación de trabajo de grado para una maestría en el MIT [4]. En una imagen, los puntos de borde pueden verse como zonas de píxeles en las que existe un cambio brusco de nivel de gris. Las máscaras utilizadas en el algoritmo de Canny representan aproximaciones de la definición de los bordes a considerar, donde la tolerancia de bordes dependerá del tamaño de la máscara.

A continuación, en la Figura 2.4 se muestra un ejemplo de la aplicación del filtro de extracción de bordes Canny sobre una imagen, donde a la izquierda se encuentra la imagen original y a la derecha la imagen con la extracción de bordes Canny.



Figura 2.4: Ejemplo de la Aplicación del Filtro de Extracción de Bordes Canny.

2.2.6. Transformación de Distancias

La Transformación de Distancias se realiza sobre una imagen en blanco y negro, donde los valores de los píxeles son 1 o 0 respectivamente. El valor para cada píxel de la imagen resultante representa la distancia calculada (i.e., distancia euclidiana) con respecto al píxel de valor 1 más cercano de la imagen original [3]. La Transformación de Distancias juega un papel importante en la comparación de imágenes, particularmente para la comparación de contornos.

A continuación, en la Figura 2.5 se muestra un ejemplo, tanto visual como su representación matricial, de la aplicación de transformación de distancias sobre una imagen.

2.3. Clasificación de Imágenes

Para el desarrollo del proyecto existe el interés de clasificar imágenes. Dentro del área de computación gráfica y de inteligencia artificial existen algoritmos con la finalidad de detectar similitudes o patrones mediante el uso de plantillas (Computación Gráfica), o mediante el uso de algoritmos de aprendizaje supervisado (Inteligencia Artificial). A continuación se definen

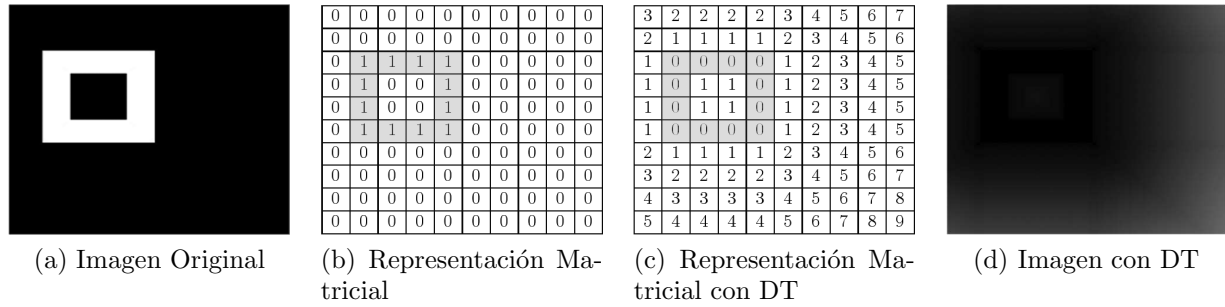


Figura 2.5: Ejemplo de la Aplicación de Transformación de Distancias.

los algoritmos de Clasificación por Plantillas y los algoritmos por Aprendizaje Supervisado estudiados para este trabajo.

2.3.1. Algoritmos de Clasificación por Plantillas

Los algoritmos de clasificación por plantillas se basan en comparación de datos modelo pre-existentes llamados plantillas. Las plantillas son creadas mediante un conjunto de datos, llamados datos de entrenamiento. La clasificación de este tipo de algoritmos consiste en buscar la plantilla acorde al dato a procesar, sacando la diferencia entre este dato y cada plantilla del conjunto, dando como resultado la clasificación correspondiente a la plantilla que tenga menor diferencia.

2.3.1.1. Clasificación de Formas a través de Distancias Internas

El Clasificador de Formas a través de Distancias Internas [9] busca la coincidencia de una cierta cantidad de puntos en el contorno del objeto en las imágenes a clasificar y las plantillas, creando a partir de estos las distancias internas entre ellos. Los cambios bruscos del contorno de la imagen son los candidatos a ser puntos de comparación entre la imagen y la plantilla, donde la distancia interna se define a través de estos puntos como la longitud más corta entre un punto del contorno de la imagen plantilla y un punto relativo, al contorno anterior, dentro de la imagen a clasificar.

2.3.2. Algoritmos por Aprendizaje Supervisado

Los algoritmos por Aprendizaje Supervisado consisten en generar una función capaz de clasificar, diferenciar o tomar decisiones sobre un conjunto de datos. Esta función se construye a partir de un entrenamiento previo el cual se debe realizar con un conjunto de datos suficientemente representativos al problema general.

Para el correcto desempeño de estos algoritmos, es recomendable realizar una normalización de los datos a evaluar con la finalidad de no generar errores que se pueden producir durante los cálculos realizados por estos algoritmos.

A continuación se presentan dos tipos de normalizaciones seguido de la descripción de los algoritmos de Clasificación por Aprendizaje Supervisado planteados en este trabajo.

2.3.2.1. Normalización por Varianza y Media

Este tipo de normalización se basa en la varianza y media de los datos a clasificar, donde los mismos pueden presentar rangos de valor variable. Con la finalidad de conseguir valores de rango similar en los datos a clasificar, a través de este tipo de normalización se logra que estos datos posean una media y varianza con valor 0 y 1 respectivamente. Dada esta transformación, los valores de estos datos se encontrarán en un rango entre -1 y 1.

2.3.2.2. Normalización por Mínimos y Máximos

La normalización por mínimos y máximos consiste en promediar los valores de un vector dados los valores máximo y mínimo del mismo, creando la cota de los valores posibles entre este rango transformando el resto de los valores espacialmente equidistantes a éste.

2.3.2.3. Perceptrón

Según T. Mitchell en [11], un perceptrón es una estructura que posee una neurona y su unidad básica de inferencia tiene forma de discriminador lineal. Esta estructura toma un vector de entradas y calcula una combinación lineal para éste. Esta salida es de valor 1 si el resultado es positivo ó -1 en caso contrario. En otras palabras, dado un vector de entradas

$\vec{x} = (x_1, x_2, \dots, x_n)$ la salida $o(\vec{x})$ de un perceptrón es:

$$o(x_1, \dots, x_n) = \begin{cases} 1 & \text{Si } w_1.x_1 + w_2.x_2 + \dots + w_n.x_n > 0 \\ -1 & \text{Si no} \end{cases} \quad (2.2)$$

donde \vec{w} es una matriz de pesos donde w_i es el aporte de la entrada x_i de la salida de dicho perceptrón.

Un perceptrón puede ser usado para resolver funciones booleanas, pero solo puede llegar a una solución si el problema es linealmente separable.

La regla de entrenamiento de un perceptrón es aumentar progresivamente, y en pequeños pasos, este vector de pesos \vec{w} . Dada la salida de la iteración anterior o , los objetivos t , el vector de entradas \vec{x} y una tasa de aprendizaje η , se calcula la variación de pesos que corresponde a la entrada i . En pocas palabras se calcula la diferencia del peso con respecto al objetivo de esa entrada, tal como se muestra en la Ecuación (2.3)

$$w_i \leftarrow w_i + \Delta w_i \quad (2.3)$$

donde:

$$\Delta w_i \leftarrow \eta.(t - o).x_i \quad (2.4)$$

la tasa de aprendizaje debe ser un número positivo y constante, el cual genera variaciones sobre Δw_i a través de las iteraciones. Los pesos son actualizados con la finalidad de reducir, a medida que se itera, el error cuadrático medio presente en la ecuación 2.5, donde t representa el objetivo, o la salida, D el conjunto de ejemplos de entrenamiento y \vec{w} el vector de pesos.

$$E(\vec{w}) = \sum_{d \in D} \frac{1}{2} (t_d - o_d)^2 \quad (2.5)$$

Para más información, el pseudo-código de Perceptrón se encuentra en el Apéndice A.1.

2.3.2.4. Red Neural

En este trabajo nos referiremos como Redes Neurales a los perceptrones multicapa, que a diferencia de los Perceptrones, son capaces de satisfacer problemas de decisión no lineal. Están conformadas por capas de neuronas enlazadas entre sí, donde la primera capa representa las entradas, la última capa posee las neuronas de salida, y entre ellas existe otro conjunto de capas denominadas capas de neuronas ocultas. La compleja estructura de las Redes Neurales es lo que permite satisfacer problemas de clasificación no lineal.

Entre las distintas formas de implementación existentes para Redes Neurales, la más común es denominada *Backpropagation*. El objetivo es minimizar el error cuadrático medio acumulado, producido por todas las neuronas 2.6. El nombre de *Backpropagation* surge debido a que el error acumulado es propagado hacia atrás por toda la red con la finalidad de actualizar los pesos de cada neurona.

$$E_d(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in \text{outputs}} (t_{kd} - o_{kd})^2 \quad (2.6)$$

Las Redes Neurales son métodos robustos para la clasificación de imágenes, reconocimiento de voz y problemas que requieran aprendizaje previo.

Para más información, el pseudo-código de una Red Neural con *Backpropagation* se encuentra en el Apéndice A.2.

2.3.2.5. SVM-Pegasos

Una Máquina de Vector de Soporte (SVM por sus siglas en inglés) es un algoritmo de aprendizaje supervisado que resuelve problemas de clasificación y regresión a través de la construcción de hiper-planos que maximizan el riesgo estructural a través de ejemplos de entrenamiento “muestras”, construyendo un modelo que prediga la clase de la nueva muestra.

Para hacer un análisis simple, iterativo y efectivo de este problema existe un método llamado Solucionador Fundamental de Sub-Gradientes Estimados para SVM (Pegasos por sus siglas en inglés) propuesto por S. Shalev-Shwartz et al. [18].

Para este método, a diferencia de un SVM, la función objetivo es una aproximación basada

en ejemplos de entrenamiento de forma (x, y) , resultado dicha función en:

$$f(\vec{w}) = \frac{\lambda}{2} \|\vec{w}\|^2 + \ell(\vec{w}; (x, y)) \quad (2.7)$$

donde \vec{w} representa el vector de pesos a alterar por cada iteración, a través de una variación basada en una tasa de aprendizaje predefinida.

El pseudo-código detallado de SVM-Pegasos se encuentra en el Apéndice A.3.

2.3.2.5.1. SVM-Pegasos por Lotes Pequeños

SVM-Pegasos por Lotes Pequeños considera un número k de ejemplos en cada iteración donde $1 \leq k \leq m$ es el nuevo parámetro de entrada al algoritmo. En cada iteración se escoge un subconjunto $A_t \subseteq S$, donde $|A_t| = k$, uniformemente aleatorio. Cuando $k = m$ la función maneja la función objetivo original. Al considerar lotes pequeños en este algoritmo se aproxima la función objetivo a:

$$f(\vec{w}; A_t) = \frac{\lambda}{2} \|\vec{w}\|^2 + \frac{1}{k} \sum_{i \in A_t} \ell(\vec{w}; (x_i, y_i)) \quad (2.8)$$

Como antes, se considera el sub-gradiente de la función objetivo dada por:

$$\nabla_t = \lambda w_t - \frac{1}{k} \sum_{i \in A_t} \theta[y_i \langle w_t, x_i \rangle < 1] y_i x_i \quad (2.9)$$

Se actualiza el peso $w_{t+1} \leftarrow w_t - \eta \nabla_t$ con una tasa de aprendizaje de $\eta_t = \frac{1}{\lambda t}$.

Para más información, el pseudo-código de SVM-Pegasos por Lotes Pequeños se encuentra en el Apéndice A.4.

2.3.2.5.2. SVM-Pegasos con Kernel

Según S. Shalev-Shwartz et al. en [18] una de las muchas ventajas de los SVMs es que pueden usar una función *kernel*, la cual separa los ejemplos de entrenamiento en un espacio de mayor dimensión para poder distinguir con mayor facilidad sus diferencias. Es posible el entrenamiento del SVM-Pegasos sin el acceso directo a los ejemplos de entrenamiento, usando los productos de salida de la función kernel aplicada. Para ello proponen el uso del SVM-

Pegasos con Kernel el cual su entrenamiento se dirige a la solución del problema minimizado:

$$\min_{\vec{w}} \frac{\lambda}{2} \|\vec{w}\|^2 + \frac{1}{m} \sum_{(x,y) \in S} \ell(\vec{w}; (\phi(x), y)) \quad (2.10)$$

donde,

$$\ell(\vec{w}; (\phi(x), y)) = \max\{0, 1 - y\langle \vec{w}, \phi(x) \rangle\}. \quad (2.11)$$

La función de transformación $\phi(\cdot)$ nunca es especificada explícitamente pero es expresada a través de un operador kernel $K(x, x') = \langle \phi(x), \phi(x') \rangle$. Para más información, el pseudo-código de SVM-Pegasos con Kernel se encuentra en el Apéndice A.5.

2.4. Estructuras Complementarias

Para este trabajo se requiere, además de la manipulación y clasificación de imágenes, el uso de estructuras y mecanismos que permitan el análisis y rápido acceso a conjuntos o agrupaciones de datos.

A continuación se describen las bases teóricas utilizados para la implementación de dichas estructuras.

2.4.1. Filtros Lineales

Un filtro lineal es un método que se usa para la eliminación del ruido dentro de una señal, a través del análisis y modificación de valores dentro de ésta. Además de la eliminación de ruido, los filtros lineales se emplean para el suavizado, detección o contraste de frecuencias dentro de las señales analizadas.

2.4.2. Ventana Deslizante

En control de flujo de datos, una Ventana Deslizante consiste en la estructura usada para controlar el flujo de paquetes entre un emisor y un receptor. Esta estructura permite al emisor procesar muchos paquetes antes de esperar por la respuesta de envío de éstos y permite al receptor esperar un conjunto de paquetes de manera no lineal.

2.4.3. Cluster

Un Cluster (del inglés conglomerado) consiste en la partición de datos con similitudes entre ellos. En el área de la computación se pueden usar Cluster para agrupar los datos según los valores de un atributo para buscar de manera más rápida los elementos de un conjunto por el atributo con el cual se realizó el Cluster.

2.4.4. Árbol de Decisión

Un árbol de decisión es un árbol de búsqueda el cual tiene como objetivo retornar una decisión a partir de una entrada que puede ser un objeto o una situación sobre la que se desea obtener una decisión a tomar [11].

Los árboles de decisión están conformados por nodos, ramas y hojas. Los nodos representan una prueba sobre el atributo asociado a él, mientras que las ramas del árbol representan uno de los valores del dominio del atributo correspondiente al nodo padre, por lo tanto las hojas serán el valor de decisión a ser retornado por el árbol.

2.4.5. Trie

El Trie consiste en una estructura tipo árbol que puede ser usado para representar diccionarios o conjuntos grandes de palabras, donde el recorrido del mismo a través de una rama específica representa el uso o adición (dependiendo de donde este sea aplicado) de un caracter en particular. A partir de la idea de que muchas palabras tienen los mismos prefijos se puede construir la estructura Trie con un conjunto de palabras, donde los caminos del árbol representan las palabras contenidas en el conjunto empleado para su construcción.

A continuación, en la Figura 2.6 se muestra un ejemplo de la representación visual de un trie.

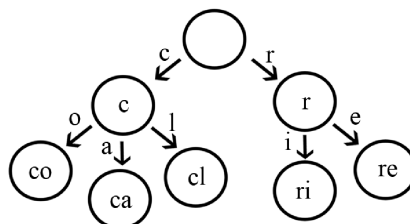


Figura 2.6: Ejemplo de la Representación Visual de un Trie

Capítulo 3

Etapas de Pre-Procesamiento (Manipulación de Frames)

La Etapa de Pre-Procesamiento del traductor de LSV-Castellano consiste en la manipulación de los frames del video que se desea traducir para obtener una representación adecuada de la seña para la Etapa de Procesamiento. La finalidad de esta etapa es eliminar toda aquella información innecesaria para el proceso de traducción y brindarle a la Etapa de Procesamiento el conjunto de datos de interés para la clasificación de cada uno de los frames.

En un video a traducir se aprecia una persona que recrea el deletreo de una palabra del Castellano a través del uso del LSV. Para lograr una extracción adecuada de la mano de la persona, y la futura traducción de las señas que realiza, se ha requerido del uso de un guante verde y ciertas condiciones de iluminación en el video. La iluminación del video es de color blanco dada por un bombillo fluorescente.

Para comenzar con la Etapa de Pre-procesamiento se separan los frames individuales que conforman el video en LSV, creando así una secuencia de frames. Luego, cada uno de estos frames son manipulados usando los métodos de Croma, Convolución Gaussiana y Canny descritos en la Sección 2.2. Dicha manipulación es realizada utilizando la siguiente secuencia de sub-etapas: Extracción de Áreas de Interés, Eliminación de Ruido y Recorte y Obtención de Líneas.

A continuación se describen cada una de las sub-etapas de Pre-Procesamiento:

3.1. Extracción de Áreas de Interés

La sub-etapa Extracción de Áreas de Interés, que en este trabajo se llamará EAI por sus siglas, es la encargada de extraer todas aquellas áreas del frame donde podría estar presente la mano con el guante. Para extraer las áreas de interés se emplea una derivación de la técnica Croma (Sección 2.2.2).

La derivación de la técnica Croma empleada, consiste en ubicar las zonas de color verde

en el frame mediante la evaluación de una función basada en rangos de valores RGB a cada píxel de la imagen, siendo esta función la que determina la pertenencia o no de cada uno de los píxeles en el área de interés. Al contrario de la técnica Croma, que descarta los píxeles de cierto color, se mantendrán los píxeles de color verde de la imagen, quienes serán evaluados como positivos al aplicar la Ecuación 3.1, y se descartarán todos aquellos píxeles que no sean valorados como verde.

$$f(R, G, B) = \begin{cases} 1 & \text{Si } ((G > R) \wedge (G > 30) \wedge (G > B) \wedge (R < \frac{G}{2} * 1,7) \wedge (B < \frac{G}{2} * 1,7)) \\ 0 & \text{Si no} \end{cases} \quad (3.1)$$

Para la evaluación de un píxel, la Ecuación 3.1 emplea los valores de cada canal de color (R, G y B) para determinar si el píxel corresponde al grupo de tonos verdes deseados. Los rangos de valor y las comparaciones entre canales, que conforman la ecuación, se determinaron experimentalmente. A modo de ilustración, se muestra en la Figura 3.1 un ejemplo de la aplicación de la derivación de la técnica Croma planteada a un frame.

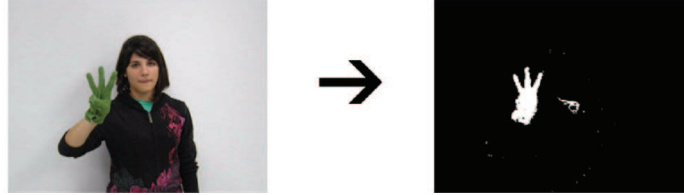


Figura 3.1: Ejemplo del Proceso de la sub-etapa Extracción de Áreas de Interés (EAI).

Mediante la sub-etapa EAI se obtiene una plantilla con valores binarios, donde los píxeles evaluados positivamente con la Ecuación 3.1 tendrán como valor uno y el resto tendrá el valor cero.

En los frames de video, además del guante, es común la aparición de varios elementos con tonos de color verde evaluados positivamente con la Ecuación 3.1. En la plantilla generada por la EAI se pueden encontrar zonas ajenas al guante a causa de dichos elementos, los cuales llamaremos “ruido”, que serán eliminadas en la siguiente sub-etapa del Pre-Procesamiento.

3.2. Eliminación de Ruido

La sub-etapa Eliminación de Ruido, que en este trabajo se denominará ER por sus siglas, es la encargada de continuar con el Pre-Procesamiento a partir de la plantilla resultante en

la sub-etapa EAI, con la finalidad de eliminar todo el ruido presente. El resultado será una imagen binaria donde los únicos píxeles con valor 1 serán aquellos que componen el área del guante. La Figura 3.2 muestra un ejemplo de la aplicación de los tres pasos que conforman la sub-etapa ER, que se describirá a continuación.

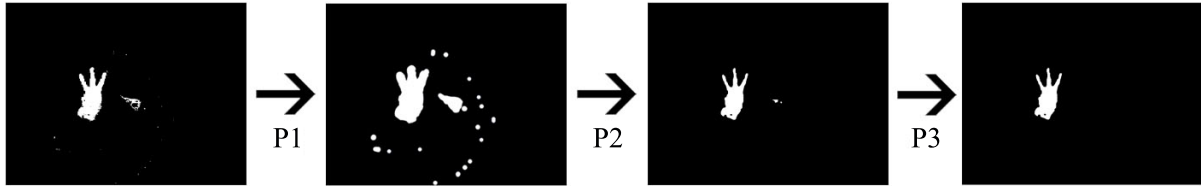


Figura 3.2: Ejemplo del Proceso de la sub-etapa Eliminación de Ruido (ER).

El primer paso (P1 en la Figura 3.2) consiste en la aplicación de una convolución basada en filtros de Gauss (Sección 2.2.3.1) a la imagen resultante de la sub-etapa EAI, resultando una imagen estilizada con bordes difuminados. La finalidad de aplicar este filtro es hacer que los píxeles aislados de ruido presentes en la plantilla tomen valores considerablemente inferiores al blanco, cuyo valor es 255.

Posteriormente se realiza una segmentación binaria (Sección 2.2.4.1) de la imagen (P2 en la Figura 3.2), tomando en cuenta como área de interés los píxeles que cumplan con un umbral pre-establecido (mayor a 200). Esto produce una imagen en blanco y negro con una disminución considerable del ruido, en comparación con la imagen inicial, y sin exceso de difuminado en los bordes del guante.

A partir de esta nueva imagen obtenida se procede a eliminar todas las áreas blancas excepto la de mayor tamaño (P3 en la Figura 3.2), que luego de todos los filtros aplicados sobre la imagen original, será el área correspondiente al guante utilizado por la persona.

Al aplicar las sub-etapas EAI y ER se obtiene una plantilla binaria del frame, donde se puede distinguir el área correspondiente al guante. Luego es necesaria la extracción de la información interna del guante empleando la imagen plantilla resultante de la sub-etapa ER. Para lograr esto es necesaria la sub-etapa Recorte y Obtención de Líneas que se describirá a continuación.

3.3. Recorte y Obtención de Líneas

La sub-etapa Recorte y Obtención de Líneas, que en este trabajo se denominará ROL por sus siglas, tiene como objetivo recortar la plantilla obtenida en la sub-etapa ER y a partir de ésta extraer del frame original el guante con su información interna, a diferencia de la plantilla que representa únicamente la forma del guante. La Figura 3.3 muestra un ejemplo de la aplicación de los tres pasos que conforman la sub-etapa ROL, que se describirán a continuación.

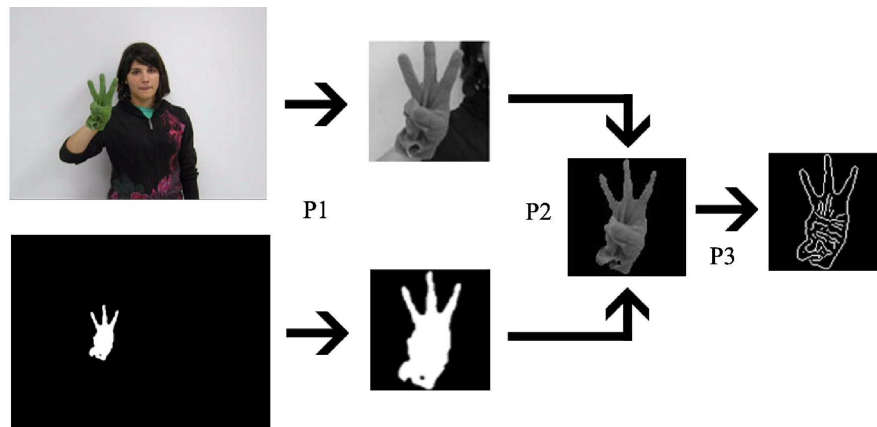


Figura 3.3: Ejemplo del Proceso de la sub-etapa Recorte y Obtención de Líneas (ROL).

El primer paso consiste en recortar el área del guante de la plantilla y a través de las posiciones utilizadas para hacer el recorte de ésta, realizar el mismo proceso sobre una copia del frame original en escala de grises (P1 en la Figura 3.3). El frame es transformado a escala de grises para facilitar su manipulación, ya que al convertir la imagen RGB a escala de grises su representación pasa a ser de una matriz numérica de tres dimensiones ($N \times M \times 3$) a una matriz de una dimensión ($N \times M$). A continuación las dos imágenes recortadas son re-escaladas a un tamaño preestablecido (100x100 px) que facilita el uso de la imagen en la Etapa de Procesamiento que se describe en la próxima sección.

El siguiente paso es generar una imagen con el guante en fondo negro. Esta imagen se logra haciendo una superposición de imágenes mediante la multiplicación entre la imagen binaria (obtenida de P3 en la Figura 3.2) y la imagen en escala de grises obtenidas de P1 (resultante de P2 en la Figura 3.3). Finalmente se obtiene la imagen del guante solo con las líneas del contorno y líneas internas a través de la aplicación del filtro Canny (P3 en la Figura 3.3).

Capítulo 4

Etapas de Procesamiento (Traducción de Videos)

La Etapa de Procesamiento del traductor de LSV-Castellano tiene como finalidad emplear los frames pre-procesados para que éstos sean evaluados a través de un método de clasificación que determine si pertenece a alguna seña del LSV. Posteriormente, una secuencia de “resultados” por frame serán analizados y refinados para generar como resultado final el deletreo de una palabra en lenguaje Castellano.

Para realizar el deletreo de una palabra en LSV, existe un conjunto de señas dirigidas específicamente al alfabeto Romano (ver Apéndice B.1), el cual está conformado por dos tipos: estáticas y dinámicas. Las señas estáticas se caracterizan en que su ejecución no requiere movimiento alguno, sino dejar la mano de una forma específica, por lo que se puede representar con una sola imagen. En cambio, las señas dinámicas se caracterizan en que su ejecución está conformada por una secuencia de movimientos y formas de la mano específicos, por lo cual se requiere más de una imagen para representarla. En este trabajo llamaremos “etapa de movimiento” a cada una de las imágenes que conforman una seña dinámica.

Con la finalidad de emplear la mejor técnica en el proceso de traducción, los métodos y algoritmos de aprendizaje empleados son: Transformación de Distancias, Normalizaciones, Perceptrón, Red Neural, SVM-Pegasos por Lotes Pequeños y SVM-Pegasos con Kernel.

El proceso de traducción consiste en utilizar el mejor método obtenido bajo un conjunto de frames pre-procesados, obteniendo una secuencia referente a las clasificaciones obtenidas. Estas secuencias son manipuladas a través de un proceso de refinamiento, resultando en una traducción aproximada o equivalente a la palabra objetivo.

A continuación se presentan dos secciones, donde se podrá encontrar la descripción del proceso de Etiquetado de Frames y el proceso base de Traducción de Video.

4.1. Etiquetado de Frames

El etiquetado de frames tiene como objetivo asociar a cada frame de video (pre-procesado), que corresponda a una seña estática del abecedario del LSV con una letra del abecedario Castellano o, de ser una seña dinámica, con una etapa de movimiento específica. Este etiquetado asocia a cada frame un número, positivo o negativo, que determina si lo contenido en el frame: es una seña estática ó una etapa específica de movimiento de alguna de las señas dinámicas (casos en los que el etiquetado será positivo), mientras que en cualquier otro caso el etiquetado será negativo.

La cardinalidad de valores positivos que la etiqueta puede tomar es igual a la cardinalidad del conjunto de señas del abecedario LSV a considerar, donde cada valor representa una única seña del abecedario LSV (valores positivos de etiquetado están listados en el Apéndice B.1). De ser negativo el etiquetado, el único valor que toma es -1, concluyendo que el frame corresponde a la ejecución de una transición entre letras de la palabra ejecutada ó que en el frame no aparece el guante. De esta manera, al etiquetar un conjunto de frames, se obtiene una secuencia de números que permite, en la traducción del video, eliminar todos los frames etiquetados con valor negativo y deducir con precisión la palabra deletreada en LSV.

Para lograr etiquetar con exactitud cada uno de los frames, se implementan y contrastan los resultados de diversos métodos de clasificación: Comparación de Plantillas con Transformación de Distancias, Clasificación de Formas a través de Distancias Internas, Perceptrón, Red Neural, SVM-Pegasos por Lotes Pequeños y SVM-Pegasos con Kernel explicados en la Sección 2.2. Los métodos enumerados fueron contrastados entre si con la finalidad de elegir el método que más se adapte al problema de esta etapa del proyecto.

Los métodos se agruparon en dos conjuntos según su naturaleza. El primer conjunto contiene métodos de clasificación basados en comparación de plantillas, mientras que el segundo conjunto agrupa métodos basados en clasificación por aprendizaje supervisado.

Para cada método particular se crea una instancia asociada a cada una de las letras del alfabeto LSV. Estas instancias son clasificadores binarios encargados de determinar la semejanza, representada numéricamente, del frame a procesar con respecto a la letra del

alfabeto que se está evaluando.

Una vez obtenidas las salidas de cada una de las instancias correspondientes a las letras del alfabeto, éstas son analizadas en conjunto considerando un umbral establecido experimentalmente para cada método. El umbral determina si hay suficiente semejanza entre el frame evaluado y alguna señal del abecedario con la finalidad de realizar su debido etiquetado. En caso de que más de un clasificador cumpla la condición del umbral, se tomará como letra clasificada aquella que posea el valor que represente mayor semejanza.

A modo de ilustración, se muestra en la Figura 4.1 un ejemplo del etiquetado de frames de video. En este ejemplo, puede verse un frame pre-procesado, donde el clasificador “A” indica una similitud de 0.01, el de “U” 0.03 y el de la “W” 0.9, por lo que este frame se clasifica como 24, donde 24 es la etiqueta de la señal “W”.

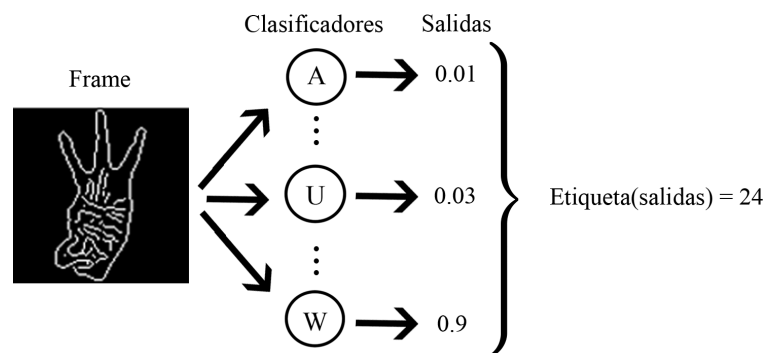


Figura 4.1: Ejemplo de etiquetado de un frame de video.

A continuación se detallan los dos conjuntos de algoritmos mencionados anteriormente describiendo la implementación de cada uno de los métodos que los conforman.

4.1.1. Algoritmos de Clasificación por Plantillas

Los Algoritmos de Clasificación por Plantillas se caracterizan por no requerir un entrenamiento previo a la ejecución. Estos métodos trabajan a partir de un conjunto o base de datos de imágenes, que en este trabajo se denominará “plantillas”. Para poder ejecutar los algoritmos, cada uno de ellos requiere que dichas plantillas posean ciertas características específicas, por lo que las imágenes empleadas para construir dicho conjunto, luego de haber sido pre-procesadas, deben ser nuevamente transformadas para cumplir con los requisitos del método en cuestión.

Cabe destacar que no hay una sola plantilla por cada seña del alfabeto Castellano, sino varias por cada una de ellas. La decisión de utilizar varias plantillas se debe a que estos métodos, al no tener un entrenamiento previo, solo tomarán como seña válida aquellas que sean exactamente igual a la plantilla. De esta manera se pueden abarcar múltiples variaciones que puedan existir para la ejecución de una misma seña.

A continuación se describen los Algoritmos de Clasificación por Plantillas que han sido implementados: Comparación de Plantillas con Transformación de Distancias y Clasificación de Formas a través de Distancias Internas.

4.1.1.1. Comparación de Plantillas con Transformación de Distancias

Como se menciona en la descripción de los Algoritmos de Clasificación por Plantillas, el método de Comparación de Plantillas con Transformación de Distancias (CPTD por sus siglas) requiere realizar una transformación adicional sobre su conjunto de plantillas (imágenes pre-procesadas). Esta transformación se realiza empleando la técnica Transformación de Distancias (DT por sus siglas en inglés) descrita en la Sección 2.2.6.

El método por CPTD se basa en la idea de G. Borgefors [3], que consiste en hacer la resta de los valores de los píxeles entre dos imágenes con DT aplicado, para obtener la correspondencia entre los bordes presentes en ellas a partir de las distancias generalizadas. En este trabajo se implementó este método con dos variaciones a considerar.

La primera variación consiste en trabajar las imágenes bajo formatos distintos. La imagen empleada como plantilla estará bajo el formato de salida del Pre-Procesamiento con la aplicación de DT, mientras que el frame a clasificar estará bajo el formato de salida de la etapa de Pre-Procesamiento.

La segunda variación consiste en que en vez de realizar una resta entre las imágenes, se realiza una multiplicación de los valores de los píxeles de mismas posiciones entre ambas. Esta modificación genera una imagen donde el valor de cada píxel, distinto a cero, representa la lejanía del borde de la imagen que se está evaluando con respecto al borde de la plantilla. La modificación se realiza con la finalidad de solo tomar en cuenta los pixeles correspondientes a los bordes del frame a evaluar, donde al tener una representación binaria, se obtendrán

unicamente las correspondencias al borde del mismo a través del producto entre las dos imágenes. Al sumar los píxeles de la imagen resultante se obtiene la diferencia total entre la plantilla y el frame en cuestión, donde a menor valor mayor será la semejanza y viceversa.

El método de CPTD consiste en buscar la plantilla de mayor semejanza con respecto al frame a evaluar. Una vez Pre-Procesado el frame a evaluar, por cada una de las seis plantillas correspondientes a una señal se calculará el número que representa la diferencia entre ambas descrita anteriormente. Este proceso se repetirá por cada señal del LSV, almacenando por cada una un grupo de valores de diferencias menores a un umbral ¹. A partir de estos grupos se extrae por cada señal el número de menor valor, y se elige como señal de mayor semejanza a aquella que posea el menor de todos. La cantidad de plantillas por señales y el umbral de selección de números de diferencias, se determinaron experimentalmente en el Apéndice C.1.

4.1.1.2. Clasificación de Formas a través de Distancias Internas

La Clasificación de Formas a través de Distancias Internas (SCUID por sus siglas en inglés) [9], al igual que el método de Comparación de Plantillas con Transformación de Distancias, emplea un conjunto de plantillas para ubicar la forma más parecida al frame a clasificar. Este método se caracteriza por realizar la comparación de dichas plantillas a través de la búsqueda de una cierta cantidad de puntos que coincidan entre el contorno de ambas imágenes.

Una vez pre-procesado el frame a evaluar, se le aplica una transformación adicional. Esta transformación requiere de dos imágenes, donde una de ellas es la generada del Pre-Procesamiento del frame, y la otra es una imagen de la mano completamente blanca obtenida de la sub-etapa de Pre-Procesamiento ROL. Una vez generadas estas dos imágenes, los valores de los píxeles son restados según sus posiciones, resultando una imagen de la mano blanca con las líneas internas del guante en negro. Este Pre-Procesamiento adicional se ilustra en la Figura (4.2).

Para este clasificador el conjunto de plantillas consta de seis imágenes por cada señal del alfabeto Castellano, cantidad determinada experimentalmente en el Apéndice C.2. El

¹Valor máximo de diferencia que puede existir, entre una imagen y una plantilla, para poder considerar que hay similitud entre ellas, cuyo valor es 8000

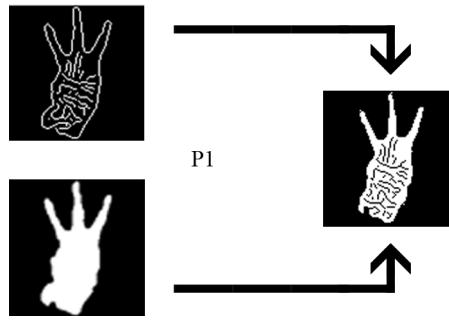


Figura 4.2: Ejemplo del Pre-Procesamiento adicional del método SCUID.

clasificador buscará entre las seis plantillas de cada letra del alfabeto aquella que posea la mayor cantidad de puntos en común con el frame que se está evaluando, considerando 105 puntos la mínima cantidad para definir que existe similitud entre la plantilla y el frame. Una vez comparado el frame con cada conjunto de plantillas, se elige la de mayor coincidencia de puntos como la plantilla con mayor semejanza con el frame. Dicha plantilla estará asociada a una seña específica con la cual se etiquetará el frame evaluado.

4.1.2. Algoritmos de Clasificación por Aprendizaje

Cada uno de los algoritmos de clasificación por aprendizaje inicia su procesamiento a partir de la imagen en Canny producida por la Etapa de Pre-Procesamiento (Capítulo 3).

El primer paso, antes de ejecutar los algoritmos de clasificación por aprendizaje, es transformar el valor de los píxeles de la imagen obtenida en la Etapa de Pre-Procesamiento, compuesta por unos y ceros, en valores más significativos para la clasificación de la imagen. Para realizar la transformación de la imagen y obtener valores más significativos, se aplica sobre la imagen el método de Transformación de Distancias (DT) (Sección 2.2.6).

El método DT le asignará a cada uno de los píxeles la distancia que hay entre el píxel en cuestión y la línea más cercana, siendo estas líneas píxeles de valor uno. Esta representación de la imagen es más rica en información dado que no solo se da a conocer donde se encontraban las líneas del guante (ahora píxeles de valor cero), sino que los otros píxeles que no aportaban información significativa, ahora indican que tan cerca se encuentra la línea de guante más cercana permitiendo así la clasificación de guantes con variaciones en la posición de dichas líneas.

Finalmente, como transformación previa a la clasificación, los valores de la imagen son normalizados entre cero y uno, bajo alguna de las siguientes normalizaciones mencionadas en la Sección 2.3: Normalización a partir de Mínimos y Máximos, y Normalización a partir de Varianza y Media. La normalización de los datos de entrada para los clasificadores presentados a continuación, se lleva a cabo con la finalidad de evitar problemas numéricos producidos por cálculos internos realizados por los clasificadores, como por ejemplo el calculo de la función sigmoidal de Redes Neurales.

Todos los clasificadores que se encuentran a continuación poseen ciertas características en común: 10000 entradas (cada una perteneciente a un píxel en específico), una única salida binaria, y los entrenamientos de cada clasificador se realizaron con 11 imágenes de la seña a clasificar y 4 imágenes por cada una de las señas restantes.

4.1.2.1. Perceptrón

Uno de los métodos de clasificación por aprendizaje supervisado es el Perceptrón (Sección 2.3.2.3). Adicionalmente a las características comunes de los clasificadores (número de entradas y salidas), los parámetros empleados son los siguientes:

- a) Se emplea la normalización por varianza y media de la imagen de entrada con Transformación de Distancias; esta elección se determina experimentalmente (ver Apéndice C.3).
- b) Una tasa de aprendizaje de 0,03; determinada experimentalmente (ver Apéndice C.3).
- c) Se inicializan los pesos en cero para el entrenamiento del clasificador.
- d) La condición de parada del entrenamiento es un número de iteraciones fijas (4000); determinada experimentalmente (ver Apéndice C.3).
- e) La salida es un número real entre -1 y 1, positiva cuando la imagen se clasifica como la seña en cuestión y negativa en caso contrario.

4.1.2.2. Red Neural

La Red Neural empleada como clasificador es una Red Neural *Backpropagation*, tal como fue descrito en la Sección 2.3.2.4. Adicionalmente a las características comunes de los

clasificadores (número de entradas y salidas), los parámetros empleados son los siguientes:

- a) 2 capas ocultas de 5 neuronas cada una. Esta configuración de capas fue determinada experimentalmente (ver Apéndice C.4).
- b) Se emplea la normalización por mínimos y máximos de la imagen de entrada con transformación de distancias; esta elección se determina experimentalmente (ver Apéndice C.4).
- c) Una tasa de aprendizaje de 0,03; determinada experimentalmente (ver Apéndice C.4).
- d) Se inicializan los pesos en cero para el entrenamiento del clasificador.
- e) La condición de parada del entrenamiento es un número de iteraciones fijas (4000); determinada experimentalmente (ver Apéndice C.4).
- f) La salida es un número real entre 0 y 1, cuando es mayor a 0,7 la imagen se clasifica como la seña asociada al clasificador, y cuando es menor o igual a 0,7 no es clasificada.

4.1.2.3. SVM-Pegasos por Lotes Pequeños

Para la clasificación se emplea un clasificador SVM-Pegasos por Lotes Pequeños tal como fue descrito en la Sección 2.3.2.5.1. Adicionalmente a las características comunes de los clasificadores (número de entradas y salidas), los parámetros empleados son los siguientes:

- a) Se emplea la normalización por varianza y media de la imagen de entrada con Transformación de Distancias; esta elección se determina experimentalmente (ver Apéndice C.5).
- b) La condición de parada del entrenamiento es un número de iteraciones fijas, valor calculado a partir de la Ecuación 4.1 extraído del artículo [18], donde n representa el tamaño del vector de pesos \vec{w} , ϵ representa la exactitud de la solución y λ es el parámetro de regularización de los SVM. El valor de λ empleado es de 2^{-3} y el ϵ es $0,5^2$, valores determinados experimentalmente (ver Apéndice C.5).

$$\frac{n}{\epsilon \lambda} \tag{4.1}$$

- c) La salida es un número real entre -1 y 1, positiva cuando la imagen se clasifica como la seña en cuestión y negativa en caso contrario.

4.1.2.4. SVM-Pegasos con Kernel

Se emplea un clasificador SVM-Pegasos con Kernel tal como fue descrito en la Sección 2.3.2.5.2. Adicionalmente a las características comunes de los clasificadores (número de entradas y salidas), los parámetros empleados son los siguientes:

- a) Se emplea la normalización por mínimos y máximos de la imagen de entrada con Transformación de Distancias; esta elección se determina experimentalmente (ver Apéndice C.6).
- b) La condición de parada del entrenamiento es un número de iteraciones fijas, valor calculado a partir de la Ecuación 4.1 extraído del artículo [18], donde n representa el tamaño del vector de pesos \vec{w} , ϵ representa la exactitud de la solución y λ es el parámetro de regularización de los SVM. El valor de λ empleado es de 2^3 y el ϵ es $0,5^2$, valores determinados experimentalmente (ver Apéndice C.6).
- c) Un kernel de tipo RBF (Radial Basic Function, por sus siglas en inglés) representado en la Ecuación 4.2, donde σ representa la fuerza de separación. El valor de σ empleado es de 2^{-3} , determinado experimentalmente (ver Apéndice C.6).

$$RBF = e^{-(\|x_i - x_j\|^2)/(2\sigma^2)} \quad (4.2)$$

- d) La salida es un número real entre -1 y 1, positiva cuando la imagen se clasifica como la seña en cuestión y negativa en caso contrario.

4.2. Traducción de Video

Durante el deletreo de una palabra en LSV existen dos fases: la ejecución de la seña de determinada letra, y la transición entre la ejecución de dos señas. Para afirmar con certeza si una seña se ejecuta en el video, no basta con analizar un solo frame del video, sino un conjunto cercano de los mismos y así determinar cuál de las fases de deletreo (seña o transición) se

analiza. Si la fase que se analiza es de ejecución, se determina cuál letra del alfabeto es ejecutada en dicho momento.

El proceso de traducción tiene como objetivo la generación de cadenas de caracteres (palabra traducida), al diferenciar entre las señas ejecutadas y las transiciones entre la ejecución de dos señas. Para esto se emplean técnicas basadas en filtros lineales y análisis de ventanas deslizantes basadas en el control de flujo de datos.

A continuación se describe el proceso de traducción implementado.

4.2.1. Proceso de Traducción

Se considerará un video como una secuencia numérica, donde los números corresponden al etiquetado obtenido a través de la clasificación individual de los frames (explicada en la Sección 4.1). A través del análisis de esta secuencia numérica se podrá diferenciar con mayor facilidad los conjuntos de frames que componen una transición entre la ejecución de señas del abecedario del LSV, además de aquellos conjuntos de frames que podrían formar parte de la ejecución de una seña. Estos dos conjuntos serán llamados “momento de transición” y “momento de ejecución” respectivamente.

En este trabajo se denomina “Ventana Deslizante” a la técnica basada en filtros lineales que tiene como objetivo la eliminación de aquellos momentos de transición que se producen al analizar un conjunto de frames de tamaño fijo (que será llamado “ventana”) con algún método de clasificación del Etiquetado de Frames. De esta manera se obtiene una mínima secuencia de momentos de ejecución que al ser refinados representarán la traducción final de cierta palabra. Esta “ventana” se asemeja a las ventanas deslizantes del área de Redes, específicamente al control de flujo de datos.

La aplicación de Ventana Deslizante sobre un video, inicia con la creación de una ventana (de tamaño 10) a partir de la selección de un conjunto de frames. Los frames que conforman la ventana son sometidos a la Etapa de Pre-Procesamiento y al Etiquetado de Frames, resultando una secuencia numérica. Luego es analizado el contenido de la ventana, procedimiento que será descrito posteriormente tomando en cuenta el número de apariciones de las distintas señas de la ventana actual, para así determinar si ésta representa un momento de ejecución

o de transición.

Si la ventana representa un momento de ejecución y del análisis de la ventana se deriva que se ejecutó una seña estática, se añade a la traducción la letra del abecedario asociada a dicha seña. Por otro lado, si la ventana representa un momento de ejecución y del análisis de la ventana se concluye que se ejecutó la primera etapa de movimiento de una seña dinámica, se realizará un proceso adicional de tipo Ventana Deslizante con el propósito de ubicar la ejecución de las etapas de movimiento restantes. De encontrarse las etapas restantes se añade a la traducción la letra del abecedario asociada a dicha seña dinámica, sino se continúa el proceso de traducción desde la ventana donde se ubicó la primera etapa.

Finalmente, se inicializará una nueva ventana a partir del desplazamiento de la actual, pudiendo ésta contener algunos de los frames de la ventana anterior y frames subsiguientes a los ya analizados, para repetir el proceso descrito hasta llegar al final del video. El desplazamiento de la ventana podrá ser de 10 frames si en los últimos 5 frames de la ventana anterior no hay más de 2 con la misma etiqueta positiva; en caso contrario el desplazamiento será de 5 frames, si la etiqueta positiva no coincide con la que posea la mayor cantidad de apariciones en la primera porción de la ventana (ver Figura 4.3). Esta diferencia se debe a que, de no haber más de 2 frames con el mismo etiquetado y al hacer un desplazamiento de 5 frames, las probabilidades para que esa nueva ventana sea momento de ejecución son muy bajas, siendo en la mayoría de los casos innecesario invertir tiempo de ejecución evaluando una ventana ya analizada anteriormente.

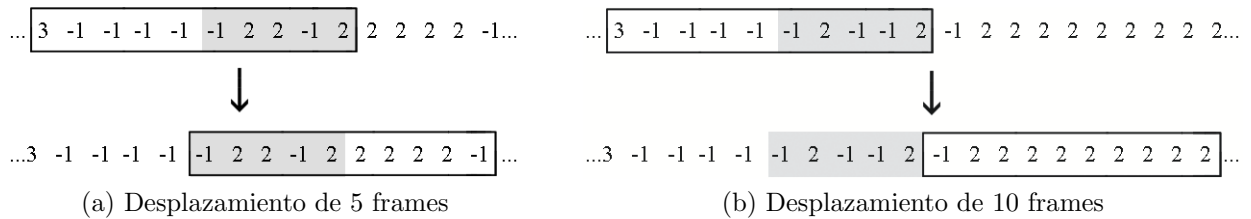


Figura 4.3: Ejemplo de Desplazamiento de Ventanas de 5 y 10 frames.

Cabe destacar que el proceso adicional de Ventana Deslizante, para la ubicación de las etapas de movimiento de una seña dinámica en particular, requiere de una ventana de menor tamaño (de 5 frames) con un desplazamiento constante de 3 frames. Esta variación se debe a que el tiempo donde aparece una etapa de movimiento es breve, a comparación de una

seña estática, debido a que son señas que involucran movimiento y el tiempo requerido para realizarlas es corto.

Exceptuando la primera ventana, la decisión del tipo de momento que representa dependerá de los resultados del análisis de las ventanas previas. Dentro de este análisis, se dirá que una seña del abecedario del LSV “predomina” si aparece al menos un número determinado de veces (7) en la ventana. Para determinar el tipo de momento, se toman en cuenta las siguientes condiciones:

- a) Cuál es la seña que predomina en la ventana a analizar, y la cantidad de apariciones dentro de la ventana.
- b) El tipo de momento de la ventana anteriormente analizada. En caso que la ventana previa sea un momento de ejecución, es necesario considerar cuál seña predominó y cuantas apariciones consecutivas lleva esta seña.

Al analizar una ventana se pueden presentar las siguientes cuatro situaciones:

1. Si en la ventana actual no predomina ninguna seña del abecedario del LSV, se considera que ésta representa un momento de transición. Ver ejemplo en la Figura 4.4 (a).
2. Si la ventana previa representa un momento de ejecución donde la seña predominante es diferente a la seña que predomina en la actual, entonces se considera que la ventana actual representa un momento de ejecución. Ver ejemplo en la Figura 4.4 (b).
3. Si la ventana previa representa un momento de ejecución, donde la seña predominante es igual a la que predomina en la actual, y si la seña en cuestión no ha superado un cierto número de apariciones consecutivas, entonces se considerará que la ventana actual representa un momento de transición. Este caso se considera como momento de transición en vez de ejecución para evitar la aparición consecutiva de una misma letra del abecedario Castellano en la traducción. Ver ejemplo en la Figura 4.4 (c).
4. Si la ventana previa representa un momento de ejecución, donde la seña predominante es igual a la que predomina en la actual, y si la seña en cuestión ya ha aparecido cierto número de veces consecutivas, entonces se considerará que la ventana actual representa un momento de ejecución. Este caso representa la traducción de palabras que contengan

dos letras iguales consecutivas durante su deletreo. Ver ejemplo en la Figura 4.4 (d).

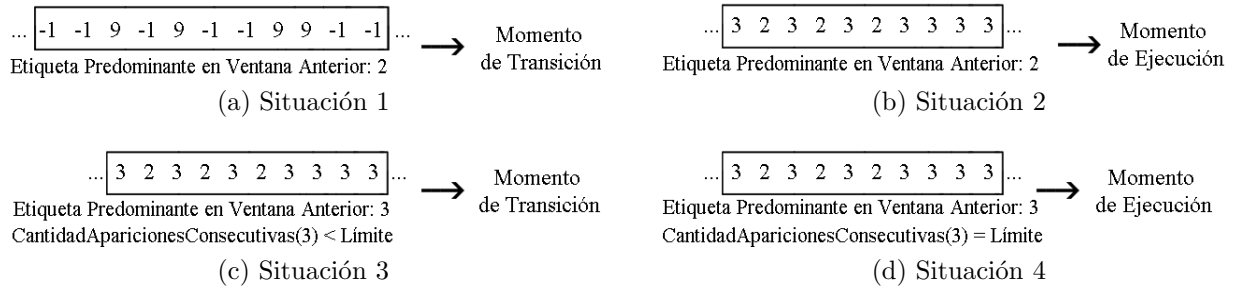


Figura 4.4: Ejemplos de las diferentes situaciones en el análisis de una Ventana.

Evaluando las situaciones descritas anteriormente y concatenando las letras correspondientes al análisis de las ventanas, momentos de ejecución, se obtiene la palabra traducida de un video.

A continuación se presenta el pseudocódigo asociado al proceso base de traducción implementado.

Algoritmo 1 Pseudocódigo del Proceso Base de Traducción

```

1: procedure procesoBaseTraducción( $V, S, n, d$ )
2:   -  $V$  es la secuencia de frames de un video.
3:   -  $S$  son el conjunto de señas a buscar.
4:   -  $n$  es el tamaño de la ventana a emplear.
5:   -  $d$ : si no es vacío representa la cantidad de frames a desplazar la ventana.
6:   Inicializar Traducción y Señanterior como vacíos.
7:   Inicializar Ventana con secuencia de los primeros  $n$  frames de  $V$ .
8:   Inicializar Cont como 0. Contador de señas predominantes consecutivas.
9:   while Queden frames en  $V$  por analizar do
10:    Sustituir en Ventana los frames por etiquetas. Por cada uno se evalúa primero el
    clasificador de Señanterior, de ser este  $-1$  se evalúan los restantes en  $S$ .
11:    Señanterior = predomina(Ventana), donde predomina asigna la etiqueta de la seña
    predominante en la ventana de entrada.
12:    if Señanterior =  $-1$  then
13:      Cont = 0.
14:    else if Señanterior = Señanterior y Cont < 8 then
15:      Cont = Cont + 1.
16:    else if Señanterior = Señanterior y Cont = 8 then
17:      Se añade a Traducción la letra correspondiente a Señanterior.
18:      Cont = 0.
19:    else if Señanterior  $\neq$  Señanterior then
20:      if esEstática(Señanterior) then
21:        Se añade a Traducción la letra correspondiente a Señanterior.

```

```

22:         else
23:              $t = \text{procesoBaseTraducción}(v, s, 5, 3)$ , donde  $v$  es una porción de frames
                subsiguientes a  $Ventana$ ,  $s$  son las etapas de movimiento de  $Seña$ , 5 es la cantidad de
                frames de la ventana a emplear y 3 es la cantidad de frames a desplazar.
24:             if  $t$  posee todas las etapas de movimiento de  $Seña$  then
25:                 Se añade a  $Traducción$  la letra correspondiente a  $Seña$ .
26:                  $SeñaAnterior = Seña$ .
27:             else
28:                  $SeñaAnterior = -1$ .
29:             end if
30:         end if
31:          $Cont = 0$ .
32:     end if
33:     if  $esVacio(d)$  then
34:         Inicializar  $p$  y  $c$  con la etiqueta de la seña que predomina en los últimos 5
                frames de  $Ventana$  y su cantidad de apariciones repectivamente.
35:         if  $p \neq -1$ ,  $p \neq Seña$  y  $c < 3$  then
36:             Desplazar  $Ventana$  por los 10 frames subsiguientes.
37:         else
38:             Desplazar  $Ventana$  por los 5 frames subsiguientes.
39:         end if
40:     else
41:         Desplazar  $Ventana$  por los  $d$  frames subsiguientes.
42:     end if
43: end while
44: return  $Traducción$ 
45: end procedure

```

Capítulo 5

Mejoras al Proceso de Traducción de Videos

Al analizar las señas del LSV (ver Apéndice B.1) se pueden observar similitudes entre algunas de ellas, por ejemplo el caso de las letras “U” y “R” que son muy difíciles de diferenciar. Dicho problema se refleja al evaluar los clasificadores sobre las señas, ya que se presentan casos donde estas se confunden entre sí y producen traducciones erróneas, tal como se ilustra en la Figura 5.1.

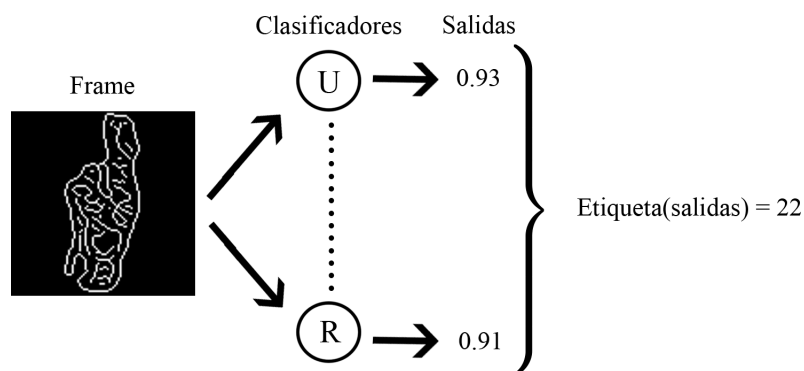


Figura 5.1: Ejemplo de confusión entre las señas U y R, donde la seña a clasificar es una R y es etiquetada como U debido al parecido entre éstas.

Dado que esta confusión se puede presentar en el proceso de traducción descrito en la Sección 4.2.1, no se puede garantizar que la traducción obtenida sea una palabra existente en el Castellano. Además, como en el proceso de traducción se emplea la técnica Ventana Deslizante, al momento de analizar una ventana puede aparecer una seña ajena a la palabra deletreada dado que ésta predomina en la ventana. Es por esto que se realizarán mejoras a la traducción de video aplicando nuevos métodos: Diccionario Trie y Árboles de Decisión de Clusters; los cuales, al ser integrados con el proceso de traducción mencionado anteriormente, simplifican la decisión de cuál seña es ejecutada dentro de cierto período de tiempo del video.

A continuación se describen las mejoras integrando los métodos ya mencionados.

5.1. Empleo de Árbol de Decisión de Clusters

El Árbol de Decisión de Clusters tiene la finalidad de mejorar la precisión de la clasificación de los frames, además de reducir los tiempos de ejecución en la Etapa de Procesamiento.

La idea inicial es agrupar las señas del abecedario LSV en pequeños clusters de señas. Para que sean útiles estos clusters es necesario poder determinar, dado un frame Pre-Procesado, a que cluster pertenece y de esta manera reducir la cantidad de posibles clasificadores a evaluar en el Etiquetado de Frames 4.1. La cantidad de clasificadores a evaluar se reducirá del total de letras del abecedario, a la cantidad de letras que forman parte del cluster al que fue asociado el frame.

Para poder determinar a cuál cluster está asociado el frame, se implementa un árbol de decisión (Sección 2.4.4). Cada nodo del árbol representa un atributo de la instancia, mientras que cada rama que sale de un nodo en particular corresponde a los diferentes valores que puede tomar el atributo en cuestión. El valor de los atributos para un frame en particular se determina a partir de la ejecución de clasificadores binarios, uno por cada valor del dominio de ambos atributos. Los atributos utilizados para asociar un frame a alguno de los clusters finales son: Forma y Lateral.

El atributo Forma separa el conjunto de señas del abecedario LSV en tres clusters que poseen señas con formas semejantes. Estas agrupaciones se basan en el contorno del guante sin tomar en cuenta la información interna del mismo (ver Tabla E.1), por lo que el frame a evaluar es manipulado con un proceso extra en la sub-etapa Recorte y Obtención de Líneas de la Etapa de Pre-Procesamiento (Sección 3.3). Este proceso consiste en rellenar las áreas negras internas al guante con píxeles de color blanco con la finalidad de eliminar los huecos internos antes de aplicar Canny (P4 en la Figura 5.2). De esta manera se obtendrán únicamente las líneas correspondientes al contorno del guante (P5 en la Figura 5.2).

Para determinar el valor del atributo Forma para un frame en particular, según las tres posibles agrupaciones, se utilizan tres clasificadores basados en una variación del método de Comparación de Plantillas con Transformación de Distancias (Sección 4.1.1.1). A diferencia del método original donde los clasificadores utilizan un conjunto de plantillas para cada una

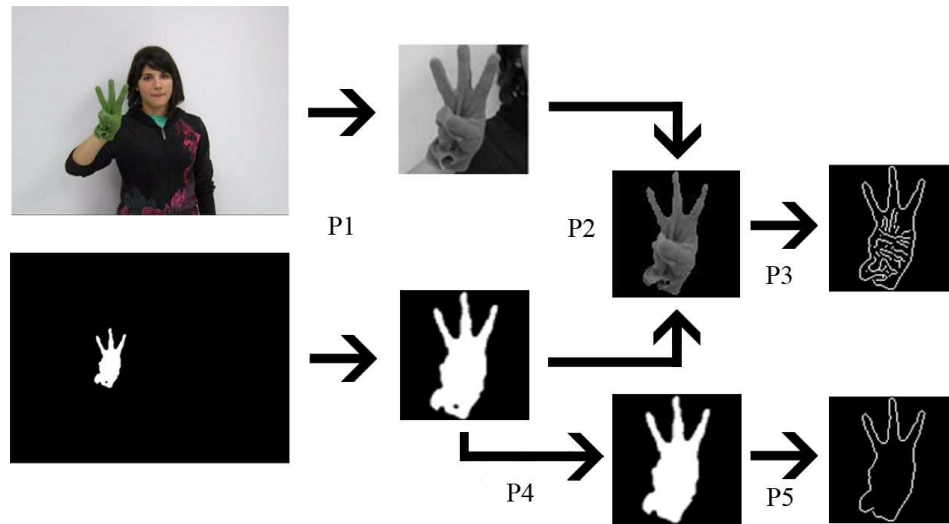


Figura 5.2: Representación de la sub-etapa Recorte y Obtención de Líneas (ROL) de la Etapa de Pre-Procesamiento con proceso extra.

de las letras del abecedario Castellano, cada uno de los tres clasificadores posee un conjunto de plantillas que describen y engloban aquellas señas de contorno similar. Por ejemplo letras como: la “A” y la “S” ó la “U” y la “R”, que son muy similares entre si, son representadas por una sola plantilla de contorno.

En la Figura 5.3 se muestra una representación del Árbol de Decisión de Clusters con el atributo Forma.

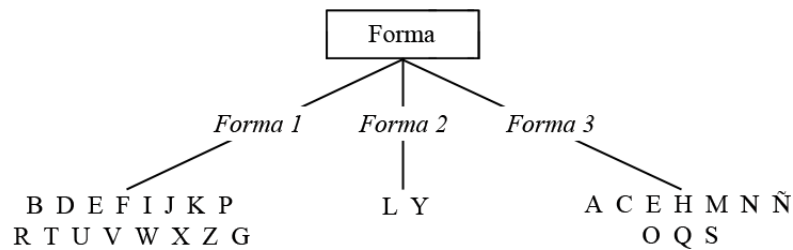


Figura 5.3: Representación del Árbol de decisión de Clusters con el atributo Forma.

Con la finalidad de generar clusters de menor tamaño, se procede a utilizar un segundo atributo llamado Lateral, que divide los tres clusters obtenidos a través del atributo Forma en clusters de menor tamaño, donde cada uno posee a lo sumo once señas. La separación en clusters de menor tamaño brindará mejoras considerables en el Proceso de Traducción por la reducción de clasificadores de señas candidatos a evaluar en el proceso de Traducción de Video LSV.

El atributo Lateral determina si el área visible en la seña es o no la parte lateral de la

mano. Debido a que este atributo solo indica si la seña es lateral o no, los valores que puede tomar son: Si o No. Para determinar el valor del atributo para un frame en particular se utiliza un clasificador binario para cada cluster obtenido a través del atributo Forma, menos para el de valor Forma 2 (Figura 5.3) el cual posee solo dos posibles señas del conjunto y ambas son señas no laterales. En la Figura 5.4 se muestra la representación del Árbol de Decisión de Clusters con los atributo Forma y Lateral.

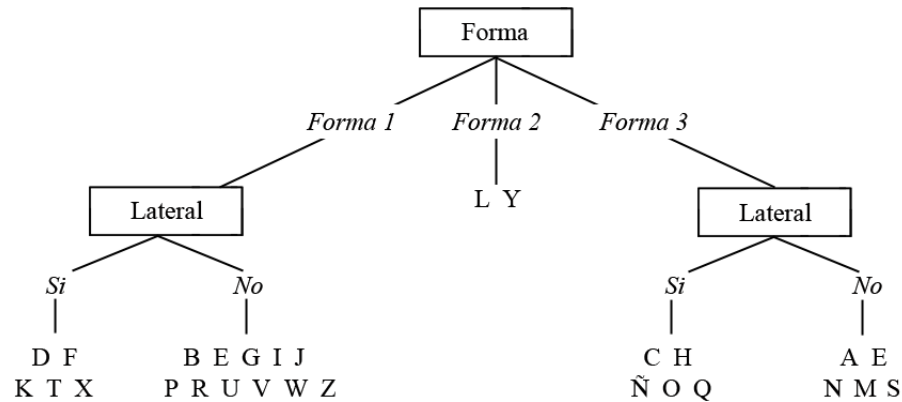


Figura 5.4: Representación del Árbol de Decisión de Clusters con los atributos Forma y Lateral.

Para determinar el valor de Lateral se utilizaron dos Redes Neuronales, una para clasificar las señas de Forma 1 y otra para las de Forma 3. Dichas Redes Neuronales son diferentes entre sí, donde sus configuraciones corresponden a los mejores resultados obtenidos experimentalmente (ver Apéndice E).

Una vez cargada la estructura del Árbol de Decisión de Clusters con los clasificadores correspondientes a cada uno de los nodos, se crea una ventana a partir de la selección de un conjunto de frames del video. Los frames que conforman la ventana creada son sometidos a la Etapa de Pre-Procesamiento con la variación descrita anteriormente (Figura 5.2), obteniendo un conjunto de imágenes binarias en formato Canny que representan el contorno de la mano. Luego, el árbol es recorrido desde la raíz hasta alguno de las hojas cluster, mediante la aplicación de los clasificadores correspondientes al atributo de cada nodo sobre cada imagen, obteniendo de esta manera un conjunto reducido de clasificadores de señas a ser evaluados en el proceso de Etiquetado de Frames. Finalmente, al terminar el proceso de etiquetado para cada uno de los frames que conforman la ventana, se obtiene una secuencia numérica

de etiquetas con la cual se determinará a través del análisis de ventanas si la ventana actual se trata de un momento de ejecución o de transición, análisis descrito en la Sección 4.2.1. Una vez analizada la ventana actual se crea una nueva con los siguientes frames, y se repite el procedimiento hasta que culmine el video.

A través del Árbol de Decisión de Clusters se reducen la cantidad de señas candidatas a evaluar para cada frame. Con ésto se reducirán considerablemente las posibilidades de error, ya que son menos la cantidad de clasificadores que se van a probar. Además por ser menor la cantidad de clasificadores a evaluar por frame será menor el tiempo empleado en la traducción del video.

A continuación se presenta el pseudocódigo asociado al proceso base de traducción implementado empleando un Árbol de Decisión de Clusters.

Algoritmo 2 Pseudocódigo del Proceso Base de Traducción con Árbol de Decisión de Clusters

```

1: procedure procesoTraducciónArbolClusters( $V, S, n, d, arbolC$ )
2:   -  $V$  es la secuencia de frames de un video.
3:   -  $S$  son el conjunto de señas a buscar.
4:   -  $n$  es el tamaño de la ventana a emplear.
5:   -  $d$  : si no es vacío representa la cantidad de frames a desplazar la ventana.
6:   -  $arbolC$  : estructura Árbol de Decisión de Clusters.
7:   Inicializar Traducción y SeñaAnterior como vacíos.
8:   Inicializar Ventana con secuencia de los primeros  $n$  frames de  $V$ .
9:   Inicializar Cont como 0. Contador de señas predominantes consecutivas.
10:  while Queden frames en  $V$  por analizar do
11:    Sustituir en Ventana los frames por etiquetas. Por cada uno se evalúa primero el
    clasificador de SeñaAnterior, de ser este  $-1$  se evalúan las señas candidatas al evaluar
    el frame en arbolC.
12:     $Seña = predomina(Ventana)$ , donde predomina asigna la etiqueta de la seña
    predominante en la ventana de entrada.
13:    if  $Seña = -1$  then
14:       $Cont = 0$ .
15:    else if  $SeñaAnterior = Seña$  y  $Cont < 8$  then
16:       $Cont = Cont + 1$ .
17:    else if  $SeñaAnterior = Seña$  y  $Cont = 8$  then
18:      Se añade a Traducción la letra correspondiente a  $Seña$ .
19:       $Cont = 0$ .
20:    else if  $SeñaAnterior \neq Seña$  then
21:      if esEstática( $Seña$ ) then

```

```

22:           Se añade a Traducción la letra correspondiente a Seña.
23:       else
24:            $t = \text{procesoTraducciónArbolClusters}(v, s, 5, 3, \text{arbolC})$ , donde  $v$  es una
           porción de frames subsiguientes a Ventana y  $s$  son las etapas de movimiento de Seña, 5
           es la cantidad de frames de la ventana a emplear, 3 es la cantidad de frames a desplazar
           y arbolC la estructura tipo árbol a emplear.
25:           if  $t$  posee todas las etapas de movimiento de Seña then
26:               Se añade a Traducción la letra correspondiente a Seña.
27:                $\text{SeñaAnterior} = \text{Seña}$ .
28:           else
29:                $\text{SeñaAnterior} = -1$ .
30:           end if
31:       end if
32:        $\text{Cont} = 0$ .
33:   end if
34:   if  $\text{esVacio}(d)$  then
35:       Inicializar  $p$  y  $c$  con la etiqueta de la seña que predomina en los últimos 5
       frames de Ventana y su cantidad de apariciones repectivamente.
36:       if  $p \neq -1$ ,  $p \neq \text{Seña}$  y  $c < 3$  then
37:           Desplazar Ventana por los 10 frames subsiguientes.
38:       else
39:           Desplazar Ventana por los 5 frames subsiguientes.
40:       end if
41:   else
42:       Desplazar Ventana por los  $d$  frames subsiguientes.
43:   end if
44: end while
45: return Traducción
46: end procedure

```

5.2. Empleo de Diccionario Trie

En este trabajo se denomina Diccionario Trie a una estructura basada en Trie (Sección 2.4.5), la cual da a conocer las posibles letras a considerar basándose en las anteriormente traducidas. Esta estructura es de tipo árbol, y se emplea para la construcción de palabras presentes en un diccionario. Todo estado de construcción de una palabra está asociado a un nodo, y a su vez estos nodos están asociados a nuevos nodos a los que se les asocia futuros estados de construcción.

Los nodos del Diccionario Trie contendrán toda la información referente a cada posible futuro estado de construcción, información que estará comprendida por lo siguiente:

- a) Señal o letra que al añadirse al recorrido produce el siguiente estado de construcción.
- b) Un apuntador al siguiente estado de construcción (arcos del Diccionario Trie).
- c) Un indicador que define si el siguiente estado de construcción forma una palabra del diccionario.
- d) La probabilidad de uso de dicha señal, basada en las palabras de un diccionario.

A modo de ilustración, se muestra en la Figura 5.5 la representación de un Diccionario Trie.

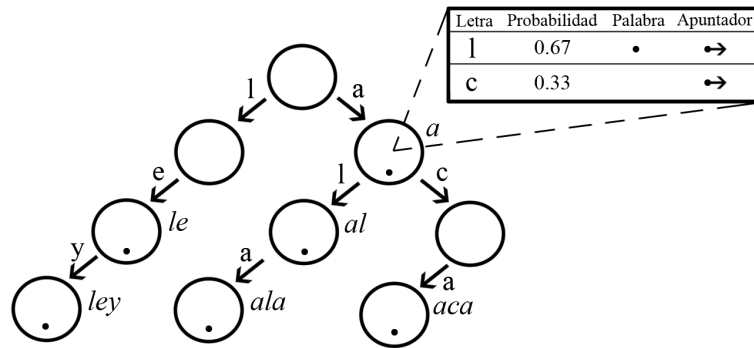


Figura 5.5: Ejemplo de la representación visual del Diccionario Trie implementado

Gracias a esta estructura, solo se pueden traducir palabras contenidas en el diccionario utilizado [8]. Esto se logra al evaluar los clasificadores correspondientes a letras del alfabeto Castellano que estén listadas como opción de traducción correspondientes al estado de la palabra actual, previniendo algunas confusiones entre señales. Además, como se evalúa un número menor o igual al total de clasificadores existentes, el tiempo que requiere la traducción del video disminuye según la cantidad de clasificadores evaluados.

Al integrar el Diccionario Trie con el proceso de traducción de la Sección 4.2.1, se utilizarán a lo largo del proceso los siguientes tres datos: nodo del Diccionario Trie, estado de la palabra en traducción y última palabra existente completada. Inicialmente el primer parámetro se asocia a la raíz del trie, y los otros se inicializan vacíos.

Una vez cargado el Diccionario Trie y inicializados los datos descritos, se crea una ventana a partir de la selección de un conjunto de frames del video. Los frames que conforman la ventana son sometidos a la Etapa de Pre-Procesamiento, resultando en un conjunto de imágenes binarias en formato Canny. Luego, a diferencia del Proceso de Traducción (Sección 4.2.1), dichos frames Pre-Procesados son etiquetados, donde solo se tomarán en cuenta los

clasificadores de las señas presentes como posibles candidatos en el nodo del trie, generando una secuencia numérica. Esta secuencia, o ventana, es analizada para determinar si representa un momento de ejecución o de transición (Sección 4.2.1).

Si la ventana analizada representa un momento de ejecución se realiza lo siguiente: se añade al estado de la palabra en traducción la letra del abecedario asociada a la seña derivada del análisis, se actualiza la última palabra existente si el Diccionario Trie indica que el uso de esta seña completa una palabra existente, y se actualiza el nodo del Diccionario Trie siguiendo el apuntador de la seña recién traducida.

Finalmente, se elige una nueva ventana a partir de un conjunto de frames subsiguientes a los ya analizados, para repetir el proceso descrito hasta llegar al final del video, donde una vez completado el proceso de traducción se obtienen dos parámetros. El primero es la última palabra existente completada, la cual indica cual es la traducción obtenida del video asegurando que ésta es una palabra existente. La segunda es el último estado de la palabra en traducción, que indica la parte inicial de palabras existentes.

A continuación se presenta el pseudocódigo asociado al proceso base de traducción implementado empleando un Diccionario Trie.

Algoritmo 3 Pseudocódigo del Proceso Base de Traducción con Diccionario Trie

```

1: procedure procesoTraducciónTrie( $V, S, n, d, trie$ )
2:   -  $V$  es la secuencia de frames de un video.
3:   -  $S$  son el conjunto de señas a buscar.
4:   -  $n$  es el tamaño de la ventana a emplear.
5:   -  $d$  : si no es vacío representa la cantidad de frames a desplazar la ventana.
6:   -  $trie$  : estructura Diccionario Trie.
7:   Inicializar Traducción, PalabraDicc y SeñaAnterior como vacíos.
8:   Inicializar Ventana con secuencia de los primeros  $n$  frames de  $V$ .
9:   Inicializar Cont como 0. Contador de señas predominantes consecutivas.
10:  Inicializar Nodo con el apuntador al nodo raíz de trie.
11:  while Queden frames en  $V$  por analizar do
12:    Sustituir en Ventana los frames por etiquetas. Por cada uno se evalúa primero el
    clasificador de SeñaAnterior, de ser este  $-1$  se evalúan las señas candidatas en Nodo.
13:     $Seña = predomina(Ventana)$ , donde predomina asigna la etiqueta de la seña
    predominante en la ventana de entrada.
14:    if  $Seña = -1$  then
15:       $Cont = 0$ .
16:    else if  $SeñaAnterior = Seña$  y  $Cont < 8$  then
```

```

17:       $Cont = Cont + 1.$ 
18:    else if  $SeñaAnterior = Seña$  y  $Cont = 8$  then
19:      if  $Nodo$  contiene como candidato a  $Seña$  then
20:        Se añade a  $Traducción$  la letra correspondiente a  $Seña$ .
21:        A  $Nodo$  se le asigna el nodo correspondiente al uso de  $Seña$ .
22:        if En  $Nodo$  el uso de  $Seña$  genera una palabra. then
23:           $PalabraDicc = Traducción.$ 
24:        end if
25:         $Cont = 0.$ 
26:      end if
27:    else if  $SeñaAnterior \neq Seña$  then
28:      if  $esEstática(Seña)$  then
29:        Se añade a  $Traducción$  la letra correspondiente a  $Seña$ .
30:        A  $Nodo$  se le asigna el nodo correspondiente al uso de  $Seña$ .
31:        if En  $Nodo$  el uso de  $Seña$  genera una palabra. then
32:           $PalabraDicc = Traducción.$ 
33:        end if
34:      else
35:         $t = procesoTraducciónTrie(v, s, 5, 3, trie)$ , donde  $v$  es una porción de frames subsiguientes a  $Ventana$  y  $s$  son las etapas de movimiento de  $Seña$ , 5 es la cantidad de frames de la ventana a emplear, 3 es la cantidad de frames a desplazar y  $trie$  la estructura trie a emplear.
36:        if  $t$  posee todas las etapas de movimiento de  $Seña$  then
37:          Se añade a  $Traducción$  la letra correspondiente a  $Seña$ .
38:          A  $Nodo$  se le asigna el nodo correspondiente al uso de  $Seña$ .
39:          if En  $Nodo$  el uso de  $Seña$  genera una palabra. then
40:             $PalabraDicc = Traducción.$ 
41:          end if
42:           $SeñaAnterior = Seña.$ 
43:        else
44:           $SeñaAnterior = -1.$ 
45:        end if
46:      end if
47:       $Cont = 0.$ 
48:    end if
49:    if  $esVacio(d)$  then
50:      Inicializar  $p$  y  $c$  con la etiqueta de la seña que predomina en los últimos 5 frames de  $Ventana$  y su cantidad de apariciones repectivamente.
51:      if  $p \neq -1$ ,  $p \neq Seña$  y  $c < 3$  then
52:        Desplazar  $Ventana$  por los 10 frames subsiguientes.
53:      else
54:        Desplazar  $Ventana$  por los 5 frames subsiguientes.
55:      end if
56:    else
57:      Desplazar  $Ventana$  por los  $d$  frames subsiguientes.

```

```

58:     end if
59:   end while
60: return Traducción
61: end procedure

```

5.3. Empleo de Integración de Árboles de Decisión de Clusters y Diccionario Trie

El empleo de Árboles de Decisión de Clusters (Sección 5.1) garantiza una mayor precisión en el proceso de traducción al disminuir la cantidad de clasificadores a evaluar por cada frame de video, lo cual genera una disminución en el tiempo de ejecución. Pero esta mejora no garantiza que la traducción propuesta de la palabra sea la correcta, a diferencia del empleo de Diccionario Trie (Sección 5.2) que sí lo garantiza. Además el Diccionario Trie puede disminuir la cantidad de clasificadores a evaluar dependiendo del estado de la traducción de la palabra, y así reducir el tiempo de ejecución.

Dado que el empleo individual de ambas estructuras aportan beneficios significativos al proceso de traducción, se propone la integración de ambas estructuras para eliminar las desventajas individuales que éstas poseen. Dicho proceso de traducción con la integración de ambas estructuras se describe a continuación.

Al igual que en proceso descrito en la Sección 5.2, se mantendrán los siguientes tres datos inicializados de la misma forma: nodo del Diccionario Trie, estado de la palabra en traducción y última palabra existente completada. Además, se cargan las estructuras Diccionario Trie y Árbol de Decisión de Cluster.

Una vez cargadas e inicializadas las estructuras y datos mencionados, se crea una ventana a partir de la selección de un conjunto de frames del video. Luego se extraen la cantidad de señas candidatas, según el nodo actual del Diccionario Trie, y dependiendo de un umbral pre-establecido (17 señas) se realizará uno de los siguientes escenarios:

- a) Si la cantidad de señas es menor al umbral: a los frames de la ventana en cuestión se les realizará el Pre-Procesamiento del Capítulo 3. Luego, al igual que en la Sección 5.2, se realizará el etiquetado de los frames tomando en cuenta sólo aquellos candidatos del nodo del Diccionario Trie.

- b) Si la cantidad de señas es mayor o igual al umbral: a los frames de la ventana en cuestión se realizará la derivación del Pre-Procesamiento mencionado en la Sección 5.1. Luego se realizará el etiquetado de los frames tomando en cuenta sólo aquellas señas candidatas que surjan del recorrido del frame sobre el Árbol de Decisión de Clusters, y además aparezcan como candidatos en el nodo actual del Diccionario Trie.

Una vez etiquetados todos los frames de la ventana se procederá a analizar dicha ventana (Sección 4.2.1), para determinar el tipo de momento que representa. Finalmente, se elige una nueva ventana a partir de un conjunto de frames subsiguientes a los ya analizados, para repetir el proceso descrito hasta llegar al final del video, donde una vez completado el proceso de traducción se obtienen dos datos. El primero es la última palabra existente completada, la cual indica cual es la traducción obtenida del video asegurando que ésta es una palabra existente. La segunda es el último estado de la palabra en traducción, que indica la parte inicial de una palabra existente.

A continuación se presenta el pseudocódigo asociado al proceso base de traducción implementado, integrando un Árbol de Decisión de Clusters y un Diccionario Trie.

Algoritmo 4 Pseudocódigo del Proceso Base de Traducción con Diccionario Trie y Árbol de Decisión de Clusters

```

1: procedure procesoTraducciónArbolClusterTrie( $V, S, n, d, trie, arbolC$ )
2:   -  $V$  es la secuencia de frames de un video.
3:   -  $S$  son el conjunto de señas a buscar.
4:   -  $n$  es el tamaño de la ventana a emplear.
5:   -  $d$  : si no es vacío representa la cantidad de frames a desplazar la ventana.
6:   -  $trie$  : estructura Diccionario Trie.
7:   -  $arbolC$  : estructura Árbol de Decisión de Clusters.
8:   Inicializar Traducción, PalabraDicc y SeñaAnterior como vacíos.
9:   Inicializar Ventana con secuencia de los primeros  $n$  frames de  $V$ .
10:  Inicializar Cont como 0. Contador de señas predominantes consecutivas.
11:  Inicializar Nodo con el apuntador al nodo raíz de  $trie$ .
12:  while Queden frames en  $V$  por analizar do
13:    for all frames no analizados en Ventana do
14:      Evaluar el clasificador de SeñaAnterior.
15:      if evaluación es -1 y la cantidad de señas candidatas es menor a 17 then
16:        Se evalúan los clasificadores de las señas candidatas en Nodo.
17:      else if evaluación es -1 y la cantidad de señas candidatas es mayor a 16 then

```

```

18:         Se evalúan los clasificadores candidatos que surjan de la evaluación del
        frame en arbolC y además estén presentes como candidatos en Nodo.
19:     else
20:         Se sustituye el frame en Ventana con la etiqueta de la evaluación.
21:     end if
22: end for
23:     Seña = predomina(Ventana), donde predomina asigna la etiqueta de la seña
    predominante en la ventana de entrada.
24:     if Seña = -1 then
25:         Cont = 0.
26:     else if SeñaAnterior = Seña y Cont < 8 then
27:         Cont = Cont + 1.
28:     else if SeñaAnterior = Seña y Cont = 8 then
29:         if Nodo contiene como candidato a Seña then
30:             Se añade a Traducción la letra correspondiente a Seña.
31:             A Nodo se le asigna el nodo correspondiente al uso de Seña.
32:             if En Nodo el uso de Seña genera una palabra. then
33:                 PalabraDicc = Traducción.
34:             end if
35:             Cont = 0.
36:         end if
37:     else if SeñaAnterior ≠ Seña then
38:         if esEstática(Seña) then
39:             Se añade a Traducción la letra correspondiente a Seña.
40:             A Nodo se le asigna el nodo correspondiente al uso de Seña.
41:             if En Nodo el uso de Seña genera una palabra. then
42:                 PalabraDicc = Traducción.
43:             end if
44:         else
45:             t = procesoTraducciónArbolClusterTrie(v, s, 5, 3, trie, arbolC), donde v
            es una porción de frames subsiguientes a Ventana y s son las etapas de movimiento de
            Seña, 5 es la cantidad de frames de la ventana a emplear, 3 es la cantidad de frames a
            desplazar, trie la estructura trie a emplear y arbolC la estructura tipo árbol a emplear.
46:             if t posee todas las etapas de movimiento de Seña then
47:                 Se añade a Traducción la letra correspondiente a Seña.
48:                 A Nodo se le asigna el nodo correspondiente al uso de Seña.
49:                 if En Nodo el uso de Seña genera una palabra. then
50:                     PalabraDicc = Traducción.
51:                 end if
52:                 SeñaAnterior = Seña.
53:             else
54:                 SeñaAnterior = -1.
55:             end if
56:         end if
57:     Cont = 0.

```

```
58:      end if
59:      if esVacio(d) then
60:          Inicializar  $p$  y  $c$  con la etiqueta de la seña que predomina en los últimos 5
            frames de Ventana y su cantidad de apariciones repectivamente.
61:          if  $p \neq -1$ ,  $p \neq \textit{Seña}$  y  $c < 3$  then
62:              Desplazar Ventana por los 10 frames subsiguientes.
63:          else
64:              Desplazar Ventana por los 5 frames subsiguientes.
65:          end if
66:      else
67:          Desplazar Ventana por los  $d$  frames subsiguientes.
68:      end if
69:  end while
70: return Traducción
71: end procedure
```

Capítulo 6

Evaluación Experimental

En este capítulo se presentan los experimentos realizados junto con los resultados obtenidos en los mismos, además de las conclusiones que se pueden obtener en el contraste de los resultados de dichos experimentos.

Recordemos que el objetivo principal de este trabajo es la traducción de palabras deletreadas en el Lenguaje de Señas Venezolano (LSV) a texto de alfabeto Romano, implementando diversos métodos de procesamiento de video los cuales serán comparados en cuanto a precisión y velocidad para determinar el mejor.

Para la ejecución de los experimentos que se describirán en este capítulo, se requiere un conjunto de datos en video e imagen, donde la persona que ejecuta las señas posea un guante verde de cierta tonalidad, capturados bajo una iluminación artificial de color blanco. Dado que, previamente a la realización de este trabajo, no se poseía un conjunto de datos bajo las condiciones requeridas, se realizó una recolección de los mismos empleando una cámara digital de 10MP de zoom 18x (óptico) y 4x (digital).

El conjunto de datos de tipo imagen recolectado está conformado por un total de 589 imágenes a color de 640x480 píxeles. El conjunto de datos recolectado, de tipo video, está conformado por un total de 37 videos formato AVI, de 640x480 píxeles, grabados a 30 frames por segundo. Dentro de este conjunto de videos se deletrean 35 palabras del Castellano de 4 a 10 letras de longitud, donde la ejecución de cada seña tiene un tiempo de duración aproximado de 2 segundos. Los conjunto de datos de ambos tipos se pueden encontrar en [8].

Todos los experimentos fueron desarrollados en el lenguaje de programación Matlab, y ejecutados con *Matlab R2008a (Versión 7.6.0.324)* en una computadora de procesador *Intel(R) Core(TM) i7 1.60GHz* con 6MB de cache de 3 niveles y 6GB de memoria RAM, bajo una arquitectura de 64bits y sistema operativo *Windows 7 Home Premium 64-bit*.

Durante el desarrollo del Capítulo 6 se presentarán dos experimentos. El primero tiene como objetivo determinar el mejor método de clasificación de frames individuales. El segundo tiene como objetivo determinar cuál de los Procesos de Traducción de Video, descritos en los Capítulos 4 y 5, genera las mejores traducciones en el menor tiempo posible.

6.1. Experimento #1: Determinar el Mejor Método de Clasificación

En este experimento se quiere determinar el mejor método de clasificación de frames individuales. Para lograr el objetivo de este experimento, se ejecutan sobre un conjunto de prueba los siguientes métodos: Comparación de Plantillas con Transformación de Distancias, Clasificación de Formas a través de Distancias Internas (SCUID por sus siglas en inglés), Perceptrón, Red Neural, SVM-Pegasos por Lotes Pequeños y SVM-Pegasos con Kernel.

6.1.1. Condiciones y Parámetros

Para realizar este experimento se selecciona por cada uno de los métodos de clasificación, ya mencionados, las configuraciones que produjeron mejores resultados en los Experimentos de Clasificación de Señas Estáticas (ver Apéndice C). En el caso de los métodos por aprendizaje supervisado se entrenaron para cada uno de ellos 27 clasificadores, correspondientes a las señas estáticas y a las etapas iniciales de movimiento de las señas dinámicas del alfabeto LSV. No se emplean clasificadores asociados a las etapas no iniciales de movimiento de las señas dinámicas, ya que para clasificar una seña de este tipo se requiere el análisis de más de un frame, lo cual no es el objetivo de este experimento. En el caso de los métodos de clasificación por comparación de plantillas, dentro del conjunto de éstas se encontrarán las correspondientes a las señas estáticas y a las etapas iniciales de movimiento mencionadas.

Cabe destacar que para el entrenamiento de un clasificador en particular se emplearon 11 imágenes de la seña del clasificador en cuestión, y 4 por cada una de las señas restantes, resultando en un total de 297 imágenes.

Adicionalmente se empleará como conjunto de prueba 31 videos del conjunto de datos. Debido a que la finalidad de este experimento es determinar el mejor método de clasificación de frames individuales, el conjunto de datos de prueba no contendrá videos con la ejecución

de señas dinámicas ya que para lograr su clasificación se requiere del análisis de más de un frame.

6.1.2. Resultados y Análisis

Para determinar el mejor método de clasificación, cada video es sometido a los algoritmos de clasificación mencionados. Los resultados serán evaluados usando las siguientes métricas: porcentaje de clasificación de frames y frames procesados por segundo (fps), empleando el método en cuestión. El porcentaje de clasificación de frames toma en consideración la clasificación correcta de frames dentro de los momentos de ejecución y el correcto descarte de frames en los momentos de transición. En la Figura 6.1 se muestra el comportamiento de los métodos empleados a través de los parámetros obtenidos como resultados, los cuales conformarán los ejes de la gráfica. En la gráfica, cada figura de color representa un método específico empleado sobre uno de los distintos casos de prueba (videos con el deletreo en LSV de una palabra del Castellano).

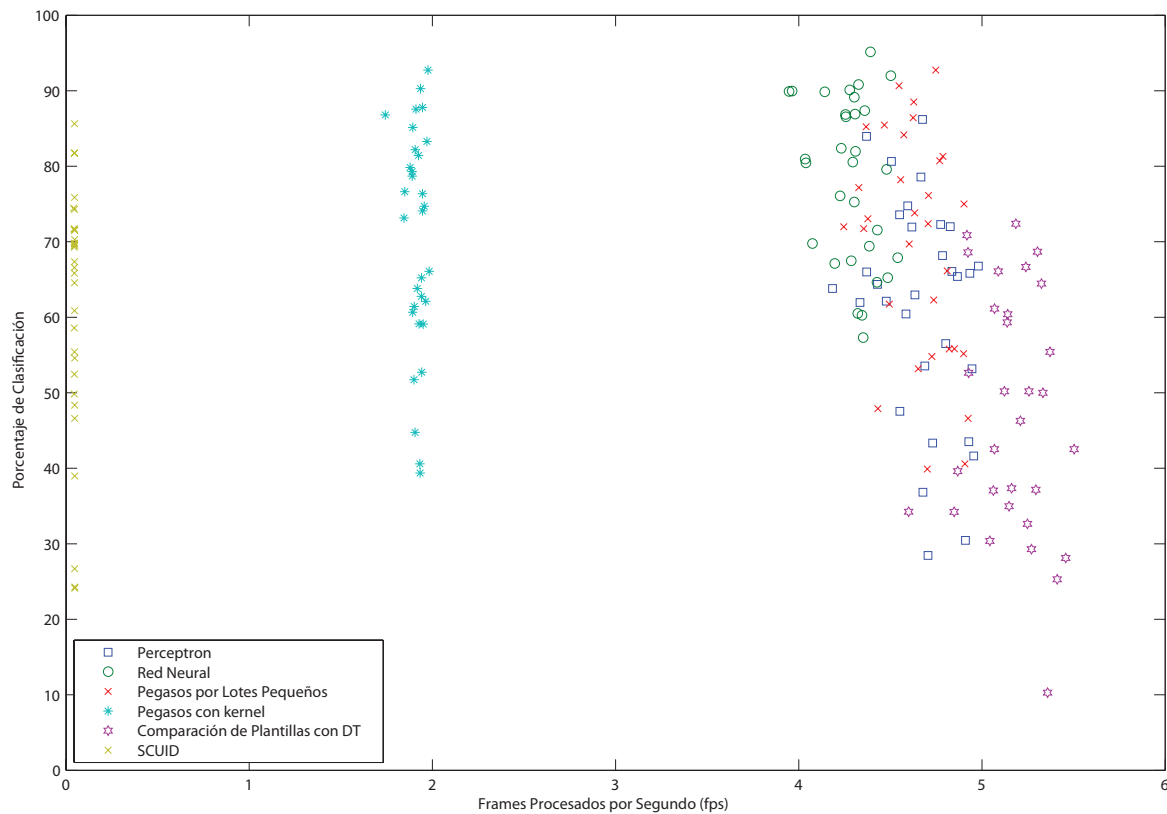


Figura 6.1: Gráfica de comparación entre los métodos de clasificación estudiados, a través de la precisión y la cantidad de frames procesados por segundo (fps)

A partir de la gráfica anterior, se puede observar que los métodos de clasificación SCUID y SVM-Pegasos con Kernel poseen un fps bajo a comparación del resto. SCUID es el que posee el fps más bajo de todos (menor a 0,1 fps). En cambio, SVM-Pegasos con Kernel posee fps intermedio entre el método anterior y el resto (entre 1,75 y 2 fps). El grupo de métodos restantes poseen valores de fps entre 3,9 y 5, siendo estos los más rápidos de todos.

Adicionalmente, se puede observar en la gráfica anterior, la presencia de aglomeraciones de valores, donde clasifican entre 10 % y 95 % de los frames de cada video. En primer aglomerado, método SCUID, se puede apreciar una concentración de algunos videos alrededor del 70 % de clasificación, pero los porcentajes de los videos restantes se encuentran muy dispersos en el rango mencionado anteriormente. En cambio en el segundo, SVM-Pegasos con Kernel, se presentan dos áreas de concentración, una alrededor del 75 % y 95 % y la segunda cercanos al 65 %.

Para realizar un análisis específico de los métodos restantes, se calcularon estadísticas sobre todos los resultados obtenidos. La Tabla 6.1 representa el comportamiento de cada método con respecto a los frames procesados por segundo y porcentajes de clasificación, describiendo para cada uno de éstos los valores estadísticos de media (MED), mínimos (MIN), máximos (MAX) y desviación estándar (DE), calculados a partir de los resultados del experimento.

Método	Frames Procesados por Segundo				Porcentaje de Clasificación			
	MED	MIN	MAX	DE	MED	MIN	MAX	DE
Perceptron	4,67	4,18	4,98	0,2	61,38	28,45	86,2	14,65
Red Neural	4,28	3,94	4,53	0,14	78,48	57,32	95,15	10,82
SVM-Pegasos por Lotes Pequeños	4,64	4,24	4,92	0,18	69,49	39,9	92,72	17,77
SVM-Pegasos con Kernel	1,91	1,74	1,98	0,04	70,3	39,36	92,72	14,55
Comparación de Plantillas con DT	5,19	4,79	5,19	0,17	47,06	10,28	72,38	15,65
SCUID	0,21	0,042	0,047	$1,39e^{-6}$	61,67	24,13	85,63	16,01

Tabla 6.1: Estadísticas de los Métodos de Clasificación

Al analizar los porcentajes de clasificación de la Tabla 6.1 se puede apreciar que la Red Neural es la que posee la media más alta (78,48 %) seguido del SVM-Pegasos por Lotes Pequeños (69,49 %), Perceptrón (61,38 %) y Comparación de Plantillas en DT (47,06 %). Dentro de los rangos de valor encontramos que nuevamente la Red Neural es la que posee los valores más altos, tanto en mínimo y máximo (57,32 %-95,15 %). A pesar que el resto de los

métodos poseen valores máximos altos, la dispersión de los valores es muy amplia, lo cual se puede observar en la desviación estándar de estos métodos (con valores mayores a 14). En cambio la Red Neural tiene la menor DE (10,82); lo que indica que este método posee los resultados más estables.

Con respecto a las estadísticas de los fps de la Tabla 6.1, se puede observar que del último grupo de métodos mencionado, la Red Neural es la que posee valores más estables dado que es la que presenta la menor desviación estándar de los tres métodos (0,14). A pesar de esto, la Comparación de Plantillas con DT es el que posee la mejor media (5,19 fps) y la menor dispersión de todas (4,79-5,19 fps).

Tomando en cuenta que el objetivo principal es la generación de traducciones precisas en el menor tiempo posible, se concluye que el mejor método de clasificación de frames individuales es la Red Neural. Aunque la Comparación de Plantillas con DT es el método con mayor cantidad de fps promedio, se elige la Red Neural debido a que la diferencia entre los fps de ésta con respecto a la Comparación de Plantillas con DT (0,91 fps), es menor en contraste con la diferencia en porcentaje de clasificación entre éstos dos métodos (31,42 %). Por lo tanto, es conveniente sacrificar la mínima diferencia de tiempo y obtener clasificaciones más precisas.

6.2. Experimento #2: Determinar el Mejor Proceso de Traducción de Video

El Experimento #2 tiene como objetivo determinar cuál de los Procesos de Traducción de Video, descritos en los Capítulos 4 y 5, genera las mejores traducciones en el menor tiempo posible. Para cumplir su objetivo, este experimento realiza cuatro pruebas, una por cada uno de los procesos de traducción: Proceso Base de Traducción, Proceso de Traducción empleando un Árbol de Decisión de Clusters, Proceso de Traducción empleando un Diccionario Trie, Proceso de Traducción integrando un Árbol de Decisión de Clusters y un Diccionario Trie.

Cada una de las pruebas generan como resultados la traducción de la palabra deletreada y dos indicadores que reflejan la cantidad de frames procesados por segundo (fps). El primer indicador es calculado tomando en cuenta el tiempo del Pre-Procesamiento y Procesamiento, mientras que el segundo se calcula tomando en cuenta únicamente el tiempo del Procesa-

miento. Se realizan los cálculos de los fps con y sin Pre-Procesamiento para observar con qué grado éste puede afectar el tiempo de ejecución. Adicionalmente a los resultados mencionados, aquellos experimentos que empleen la estructura Diccionario Trie generan una salida adicional que corresponde a la palabra más larga, existente en el diccionario empleado para la estructura, a partir de la traducción obtenida durante dicho proceso.

Cabe destacar que la traducción generada de un video se puede categorizar como alguno de los siguientes cuatro tipos:

- a) Traducción Vacía: cadena vacía de caracteres.
- b) Traducción Errada: al menos uno de los caracteres no coincide con la palabra objetivo.
- c) Traducción Parcial: la palabra objetivo coincide parcialmente con la cadena de caracteres obtenida.
- d) Traducción Correcta: la palabra objetivo es idéntica a la traducción generada.

Adicionalmente, las Traducciones Erradas y Parciales pueden presentar los siguientes casos:

- a) Repetición de Letras: aparición de letras duplicadas en la palabra generada. Esto ocurre cuando el tiempo de ejecución de la seña es muy largo y la palabra objetivo no presenta la repetición continua de la letra asociada a la seña en cuestión.
- b) Omisión de Letras: la palabra objetivo posee letras que no se encuentran en la palabra generada.
- c) Confusión de Señas: entre la palabra objetivo y la palabra generada se presentan pares de letras diferentes, dado que sus señas del LSV son parecidas entre si.
- d) Inserción Errada: la palabra generada posee letras que no se encuentran en la palabra objetivo.

Dado que el objetivo del Traductor LSV-Castellano es obtener las traducciones más cercanas al resultado objetivo, en este experimento se busca lograr la mayor cantidad posible de Traducciones Correctas y Parciales (producida por omisión de letras), y evitar por completo la obtención de Traducciones Erradas y Vacías.

6.2.1. Condiciones y Parámetros

Para cada una de las corridas se emplea un conjunto de prueba de 37 videos, del conjunto de datos, donde 6 de ellos contendrán la ejecución de señas dinámicas. Las palabras deletreadas en estos videos se pueden encontrar en las tablas del Apéndice F.

Para realizar cada una de las corridas se emplea la Red Neural de mejor configuración según los resultados obtenidos en los experimentos de clasificación de señas estáticas y dinámicas (ver Apéndice D).

Las ventanas utilizadas para el proceso de traducción son de 10 frames cada una, con un desplazamiento de 5 a 10 frames. En el caso de procesar las fases que conforman una seña dinámica, la ventana será de 5 frames con un desplazamiento fijo de 3 frames.

La estructura Diccionario Trie fue creada a través de un conjunto de palabras de un diccionario Castellano que se puede encontrar en [8], y la estructura Árbol de Decisión de Clusters, con la configuración que produjo los mejores resultados en los experimentos presentados en el Apéndice E.

6.2.2. Resultados y Análisis

En la Tabla 6.2 se muestran el número de ocurrencias de cada uno de los distintos tipos de traducción que se presentaron por cada proceso de traducción. En el Apéndice F se presentan tablas detalladas de los resultados obtenidos a partir de la ejecución de cada proceso de traducción.

Proceso de Traducción	Tipo de Traducción			
	Vacía	Errada	Parcial	Correcta
Base	0	16	5	16
Árbol de Decisión de Clusters	0	13	6	18
Diccionario Trie	0	1	12	24
Árbol de Decisión de Clusters y Diccionario Trie	0	1	12	24

Tabla 6.2: Tipos de traducción y la cantidad de ocurrencias en los resultados obtenidos bajo cada uno de los procesos de traducción empleados

A partir de la Tabla 6.2, se puede observar que ninguno de los procesos de traducción generan traducciones vacías. Además, se aprecia que el proceso base es el que genera la mayor cantidad las traducciones erradas (dieciséis), seguido del proceso de traducción que emplea

un Árbol de Decisión de Clusters, con trece traducciones, y en los procesos que emplean un Diccionario Trie, se aprecia una sola aparición de este tipo. En cambio, se observa que en ambos procesos que usan un Diccionario Trie se generan la mayor cantidad de traducciones parciales y correctas, doce y veinticuatro respectivamente, seguido del proceso de traducción empleando un Árbol de Decisión de Clusters, con seis traducciones parciales y dieciocho correctas, y del proceso base, cinco y dieciséis.

Cabe destacar que al analizar la salida de palabras existentes, presentes en los dos procesos donde se usa Diccionario Trie, se aprecia que ambos procesos generaron los mismos resultados frente a los videos procesados. Para ambos procesos se generaron cuatro traducciones vacías, ninguna errada, nueve parciales (con respecto a la palabra objetivo) y veinticuatro correctas. Se puede observar que el aumento en la cantidad de palabras vacías para esta salida, corresponde a la traducción errónea mas tres de las traducciones parciales obtenidas en la salida anteriormente analizada. Gracias a esta observación, se puede afirmar que la pérdida completa o el fraccionamiento de las traducciones parciales de palabras existentes generan palabras vacías o palabras existentes pero parciales a la objetivo.

Para poder juzgar sobre las mejoras al proceso base de traducción en cuanto a la precisión sobre los resultados obtenidos en la salida de traducciones generadas se realiza un estudio estadístico llamado la prueba de McNemar, propuesta por Q. McNemar en [10]. Esta prueba calcula la diferencia entre dos proporciones x y y dando como resultado un p-valor, que representa el nivel de significación del contraste. Si el p-valor es mayor a 0,05 no se puede afirmar que su diferencia es significativa, de lo contrario se afirma que x posee una diferencia significativa a y . En este trabajo la prueba de McNemar es utilizada para decidir si puede o no aceptarse que la estructura empleada induce un cambio significativo en la precisión de la respuesta de clasificación.

Como se mencionó anteriormente, los métodos donde se utilizan el Diccionario Trie y la integración Diccionario Trie con el Árbol de Decisión de Clusters, poseen resultados idénticos, por lo cual se tomarán como uno a la hora de hacer esta prueba. La Prueba de McNemar realizada entre estos dos métodos y el proceso base de traducción dio un p-valor de 0,01.

Siendo este p-valor menor a 0,05 se puede afirmar que el uso de la estructura Diccionario Trie y la integración del Diccionario Trie y el Árbol de Decisión de Clusters son mejoras significativas al proceso base de traducción.

Comparando en la prueba de McNemar el uso de la integración del Diccionario Trie y el Árbol de Decisión de Clusters con respecto a la estructura Árbol de Decisión de Clusters dio un p-valor de 0,07, por lo cual no se puede concluir que la diferencia entre estas dos estructuras es significativa, sin embargo se observa que el p-valor se acerca bastante al umbral.

El uso de la estructura Árbol Cluster mejora la respuesta del proceso base de traducción en dos videos, siendo esta cantidad mucho menor a la comparación del análisis anterior, donde se concluyó que dicha mejora no era significativa. Por lo cual no se puede concluir que esta mejora es significativa con respecto al proceso base de traducción. Finalmente, con estos análisis se puede concluir que la estructura que ayuda significativamente en precisión de resultados es el Diccionario Trie.

En la Tabla 6.3 se podrán observar las estadísticas obtenidas en este experimento en base a los dos tipos de fps calculados bajo cada uno de los procesos de traducción ejecutados.

Proceso de Traducción	Tipo de fps							
	Con Pre-Procesamiento				Sin Pre-Procesamiento			
	MED	MIN	MAX	DE	MED	MIN	MAX	DE
Base	4,66	3,98	6,03	0,34	12,79	10,19	18,53	1,55
Árbol de Decisión de Clusters	4,59	4,15	5,98	0,33	14,88	12,63	20,49	1,65
Diccionario Trie	5,2	4,46	7,09	0,47	16,83	12,71	31,29	5
Árbol de Decisión de Clusters y Diccionario Trie	5,16	4,43	6,99	0,47	17,12	13,2	23,49	2,14

Tabla 6.3: Estadísticas de fps a partir de los resultados obtenidos bajo cada uno de los procesos de traducción empleados

A partir de estas estadísticas se puede observar que, al considerar la cantidad de frames procesados por segundo con Pre-Procesamiento, el mejor de todos es el proceso de traducción empleando la estructura Diccionario Trie, con una media de 5,2 fps. Pero si se considera la cantidad de frames procesados por segundo sin Pre-Procesamiento, el mejor de todos es el proceso de traducción integrado a un Árbol de Decisión de Clusters y Diccionario Trie, con una media de 17,12 fps.

Adicionalmente, se aprecia que el uso de Árbol de Decisión de Clusters disminuye leve-

mente los fps con Pre-Procesamiento, al ser empleado tanto con el proceso base como con el proceso integrado con Diccionario Trie, pero con respecto a los fps sin Pre-Procesamiento el uso de esta estructura genera mejores resultados. Esta diferencia se debe a que, a nivel de Pre-Procesamiento, el uso de Árbol de Decisión de Clusters implica manipulaciones adicionales sobre los frames a procesar. Por lo tanto se puede comprobar al contrastar la disminución de fps (fps sin incluir Pre-Procesamiento menos fps incluyendo Pre-Procesamiento) entre los procesos que usan dicha estructura contra los que no, siendo esta diferencia el costo de realizar dichas manipulaciones adicionales.

Gracias a las conclusiones obtenidas a partir de las pruebas de McNemar, se puede afirmar que los mejores procesos de traducción, de los implementados, son aquellos dos que emplean un Diccionario Trie. Pero gracias al análisis de los frames procesados por segundo, sin Pre-Procesamiento, se puede afirmar que el mejor de ambos es el que integra tanto el Diccionario Trie como el Árbol de Decisión de Clusters en el proceso de traducción de videos dado su alto valor de fps.

Capítulo 7

Conclusiones y Recomendaciones

Se logró el desarrollo de un reconocedor base de lenguaje de señas, cuya función es traducir un video del deletreo de una palabra en LSV al Castellano. Para lograrlo se contrastaron varios métodos de clasificación para evaluar la calidad de etiquetado de frames individuales: el Clasificador de Formas a través de Distancias Internas, Comparación de Plantillas con Transformación de Distancias, Perceptrón, Red Neural, SVM-Pegasos por Lotes Pequeños y SVM-Pegasos con Kernel. La Red Neural fue el método que presentó el mejor desempeño, donde logra su propósito con una precisión de 78,48 % sobre el conjunto de prueba usado y con una tasa promedio de 4,28 frames procesados por segundo.

Para el procesamiento de video se desarrolló un método, proceso base de traducción, basado en ventanas deslizantes y filtros lineales. Este método logró su objetivo traduciendo correctamente 16 videos de un total de 37 videos. Adicionalmente, se desarrollaron tres mejoras a este procesamiento: el empleo de una estructura Árbol de Decisión de Clusters, el empleo de una estructura Diccionario Trie y la integración estas dos, las cuales logran su propósito en el mecanismo de traducción. El empleo de un Árbol de Decisión de Clusters logró su objetivo traduciendo correctamente 18 videos mientras que el uso del Diccionario Trie y la integración de las dos estructuras son los que generan los mejores resultados obteniendo un total de 24 videos correctamente clasificados y 12 de traducción parcial, donde la integración de las estructuras es el mecanismo que procesa la mayor cantidad de frames por segundo. Gracias a estos resultados se concluye que el empleo de las estructuras integradas, es el método que genera mejoras estadísticamente significativas al proceso de traducción.

Direcciones futuras:

Como dirección futura principal, se propone la extensión de este trabajo mediante la implementación de un módulo adicional capaz de traducir el universo de señas del LSV ajenas

a las utilizadas para el deletreo de palabras. Con este nuevo módulo además de traducir videos donde se deletrean palabras aisladas, se podrán traducir conversaciones completas en LSV.

Adicionalmente a la propuesta del módulo de extensión, se proponen a continuación algunas mejoras a considerar sobre el traductor del deletreo de palabras LSV presentado en este trabajo, con al finalidad de obtener mayor precisión y menores tiempos de ejecución:

- Con la finalidad de flexibilizar las condiciones para la captura del video a traducir, se podría estudiar la posibilidad de detectar la mano sin requerir el uso de un guante de color.
- Uno de los procesos que requiere más tiempo es el pre-procesamiento de frames a clasificar. Es por esto que se podría plantear como objetivo la disminución de este tiempo utilizando otras técnicas de manipulación de imágenes por computador.
- Una de las estructuras propuestas para mejorar el tiempo de procesamiento es el Árbol de Decisión de Clusters, el cual necesita un pre-procesamiento adicional al frame a ser clasificado, aumentando el tiempo de pre-procesamiento de los mismos. Para solucionar el problema se podría idear una forma de utilizar esta estructura sin las necesidad de utilizar este pre-procesamiento adicional.
- Como se describe en la Sección 5.1, la finalidad del Árbol de Decisión de Clusters es reducir la cantidad de clasificadores a evaluar. Con la finalidad de obtener una menor cantidad de clasificadores en las hojas del árbol, se podría estudiar el uso de atributos adicionales que dividan aun más los clusters hoja.
- Tal como se describe en la Sección 5.2, los nodos del Diccionario Trie almacenan la probabilidad de ocurrencia de sus letras candidatas. Para una posible mejora a nivel de precisión, se podría estudiar un mecanismo para integrar el uso de estos datos junto con las salidas de los clasificadores para otorgar un peso adicional a las letras candidatas con mayor probabilidad de aparición.
- Para lograr un mayor acercamiento a la traducción de videos a tiempo real sin perjudicar su precisión, se podría estudiar la posibilidad de no procesar todos los frames que componen el video. Una prueba interesante a realizar sería generar ventanas compuestas por frames no adyacentes entre si, por ejemplo: considerar unicamente los frames impares.

- Para mejorar el proceso de etiquetado de frames, y con esto lograr un mejor desempeño del traductor, se podría estudiar la posibilidad de eliminar las confusiones generadas al momento de evaluar clasificadores correspondientes a señas similares entre si. Una propuesta para resolver este problema es entrenar nuevos clasificadores particulares para éstos casos, que sean capaces de distinguir las señas semejantes. Otra sugerencia es el re-entrenamiento de los clasificadores actuales, a través de un conjunto de entrenamiento con mayor cantidad de imágenes correspondientes a las señas que pueden crear confusión entre si.

Bibliografía

- [1] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.
- [2] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *IEEE Conference on Computer Vision*, pages 1395–1402, 2005.
- [3] Gunilla Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:849–865, 1988.
- [4] John F. Canny. A computational approach to edge detection. *IEEE Conference on Computer Vision*, 6:679–698, 1986.
- [5] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, MA, 2. edition, 1990.
- [6] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.
- [7] IBM. Sisi - say it, sign it. Website, 2010. <http://mqtt.org/SiSi/>.
- [8] Rodolfo Sánchez Juan Cifuentes, Leonardo Da Costa. Conjunto de datos del traductor lsv-castellano. Website, 2010. <http://www.gia.usb.ve/~rodolfo/RLS/>.
- [9] Haibin Ling and David W. Jacobs. Shape classification using the inner-distance. *Center for Automation Research and Computer Science Department, University of Maryland, College Park, IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.
- [10] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12:153–157, 1947. 10.1007/BF02295996.

- [11] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.
- [12] NeoEssential. Video diccionario de lenguaje de señas venezolana. Website. http://www.neoessentia.org/video_dic01_home.htm.
- [13] World Federation of the Deaf. About. Website, 2009. <http://www.wfdeaf.org/>.
- [14] World Health Organization. Deafness and hearing impairment. Website, 2010. <http://www.who.int/mediacentre/factsheets/fs300/en/>.
- [15] Lewis M. Paul. Ethnologue: Languages of the world. Website, 2009. http://www.ethnologue.com/show_country.asp?name=VE.
- [16] Lewis M. Paul. Venezuelan sign language. Website, 2009. http://www.ethnologue.com/show_language.asp?code=vs1.
- [17] Yolanda Pérez. *La Norma en la Lengua de Señas Venezolana*, volume 8. Sapiens, Universidad Pedagógica Experimental Libertador Caracas, Venezuela pp. 105-121, 2007.
- [18] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Appearing in Proceedings of the 24 th International Conference on Machine Learning, Corvallis, OR, 2007*.
- [19] Luis Terán. *Software Educativo para el uso del Alfabeto Manual como herramienta de enseñanza del español como segunda lengua en niños sordos del subsistema de educación inicial*. Caracas, Universidad Católica Andrés Bello, 2008.
- [20] Joachim Weickert. *Anisotropic Diffusion in Image Processing*. Department of Computer Science University of Copenhagen Copenhagen, Denmark. B.G. Teubner Stuttgart, 1998.

Apéndice A

Pseudo-Códigos Implementados

A.1. Perceptrón

Algoritmo 5 Pseudocódigo de un Perceptrón [11]

```
1: procedure perceptron(Ejemplos de entrenamiento,  $\eta$ )
2:   - Cada ejemplo de entrenamiento es de la forma  $\langle \vec{x}, t \rangle$  donde  $\vec{x}$  es el vector de
   valores de entrada y  $t$  el valor objetivo.
3:   -  $\eta$  es la tasa de aprendizaje (ej. 0,05).
4:   Inicializar todos los  $w_i$  en un número aleatorio cercano a 0.
5:   while no se cumpla la condición de parada do
6:     Inicializar los  $\Delta w_i$  en 0.
7:     for all  $\langle \vec{x}, t \rangle$  dentro de los ejemplos de entrenamiento do
8:       Calcular la salida  $o$  para la instancia del vector de entrada  $\vec{x}$ 
9:       for all unidad lineal de pesos  $w_i$  do
10:         $\Delta w_i \leftarrow \eta \cdot (t - o) \cdot x_i$ 
11:      end for
12:    end for
13:    for all unidad lineal de pesos  $w_i$  do
14:       $w_i \leftarrow w_i + \Delta w_i$ 
15:    end for
16:  end while
17: return Vector de pesos  $\vec{w}$  resultante del entrenamiento
18: end procedure
```

A.2. Red Neural

Algoritmo 6 Pseudocódigo de la Red Neural por Backpropagation [11]

```

1: procedure redNeural(Ejemplos de entrenamiento,  $\eta$ ,  $n_e$ ,  $n_s$ ,  $n_o$ )
2:   - Cada ejemplo de entrenamiento es de la forma  $\langle \vec{x}, \vec{t} \rangle$  donde  $\vec{x}$  es el vector de
     valores de entrada y  $\vec{t}$  el vector de valores objetivo.
3:   -  $\eta$  es la tasa de aprendizaje (ej. 0,05).
4:   -  $n_e$  es la cantidad de entradas de la red.
5:   -  $n_s$  es la cantidad de unidades de salidas de la red.
6:   -  $n_o$  es la cantidad de unidades en la capa oculta de la red.
7:   - La entrada de una unidad  $i$  a una unidad  $j$  se denota  $x_{ji}$ , y el peso de la unidad  $i$ 
     a la unidad  $j$  se denota  $w_{ji}$ .
8:   Crear una Red Feed-Forward con  $n_e$  entradas,  $n_o$  unidades ocultas y  $n_s$  unidades de
     salida.
9:   Inicializar todos los pesos de la red en pequeños números aleatorios (ej. entre -0.05 y
     0.05).
10:  while no se cumpla la condición de parada do
11:    for all  $\langle \vec{x}, \vec{t} \rangle$  dentro de los ejemplos de entrenamiento do
12:      Propagar hacia adelante la entrada a través de la red:
13:      Introducir en la red la instancia  $\vec{x}$  y calcular la salida  $o_u$  de cada unidad  $u$  en
        la red.
14:      Propagar los errores hacia atrás a través de la red :
15:      for all Unidad de salida  $k$  de la red do
16:         $\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$ 
17:      end for
18:      for all Unidad  $h$  oculta do
19:         $\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in \text{salidas}} w_{kh} \delta_k$ 
20:      end for
21:      for all Peso  $w_{ji}$  de la red do
22:         $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$ 
23:        Donde:
24:         $\Delta w_{ji} = \eta \delta_j x_{ji}$ 
25:      end for
26:    end for
27:  end while
28: return Vector de pesos  $\vec{w}$  resultante del entrenamiento
29: end procedure

```

A.3. SVM-Pegasos

Algoritmo 7 Pseudocódigo de SVM-Pegasos [18]

```

1: procedure pegasos( $S, \lambda, T$ )
2:   -  $S$  es el conjunto de ejemplos de entrenamiento.
3:   -  $\lambda$  parámetro de regulación.
4:   -  $T$  es el número de iteraciones empleado como condición de parada.
5:   Inicializar  $w_1 = 0$ .
6:   for  $t = 1, 2, \dots, T$  do
7:     Escoger  $i_t \in \{1, \dots, |S|\}$ , uniformemente aleatorio
8:     Inicializar  $\eta_t = \frac{1}{\lambda t}$ 
9:     if  $y_{i_t} \langle w_t, x_{i_t} \rangle < 1$  then
10:       Inicializar  $w_{t+1} \leftarrow (1 - \eta_t \lambda) w_t + \eta_t y_{i_t} x_{i_t}$ 
11:     else
12:       Inicializar  $w_{t+1} \leftarrow (1 - \eta_t \lambda) w_t$ 
13:     end if
14:     [Opcional :  $w_{t+1} \leftarrow \min\{1, \frac{1/\sqrt{\lambda}}{\|w_{t+1}\|}\} w_{t+1}$ ]
15:   end for
16: return  $w_{T+1}$ 
17: end procedure

```

A.4. SVM-Pegasos por Lotes Pequeños

Algoritmo 8 Pseudocódigo de SVM-Pegasos por Lotes Pequeños [18]

```

1: procedure pegasosPorLotes( $S, \lambda, T, k$ )
2:   -  $S$  es el conjunto de ejemplos de entrenamiento.
3:   -  $\lambda$  parámetro de regulación.
4:   -  $T$  es el número de iteraciones empleado como condición de parada.
5:   -  $k$  cardinalidad del subconjunto  $A_t$  de ejemplos de entrenamiento.
6:   Inicializar  $w_1 = 0$ .
7:   for  $t = 1, 2, \dots, T$  do
8:     Escoger  $A_t \subseteq S$ , donde  $|A_t| = k$ , uniformemente aleatorio
9:     Inicializar  $A_t^+ = \{i \in A_t : y_i \langle w_t, x_i \rangle < 1\}$ 
10:    Inicializar  $\eta_t = \frac{1}{\lambda t}$ 
11:    Inicializar  $w_{t+1} \leftarrow (1 - \eta_t \lambda)w_t + \frac{\eta_t}{k} \sum_{i \in A_t^+} y_i x_i$ 
12:    [Opcional :  $w_{t+1} \leftarrow \min\{1, \frac{1/\sqrt{\lambda}}{\|w_{t+1}\|}\} w_{t+1}$ ]
13:  end for
14: return  $w_{T+1}$ 
15: end procedure

```

A.5. SVM-Pegasos con Kernel

Algoritmo 9 Pseudocódigo de SVM-Pegasos con Kernel [18]

```

1: procedure pegasosKernel( $S, \lambda, T$ )
2:   -  $S$  es el conjunto de ejemplos de entrenamiento.
3:   -  $\lambda$  parámetro de regulación.
4:   -  $T$  es el número de iteraciones empleado como condición de parada.
5:   -  $k$  cardinalidad del subconjunto  $A_t$  de ejemplos de entrenamiento.
6:   Inicializar  $\alpha_1 = 0$ .
7:   for  $t = 1, 2, \dots, T$  do
8:     Escoger  $i_t \in \{0, \dots, |S|\}$  uniformemente aleatorio.
9:     for all  $j \neq i_t$  do
10:      Inicializar  $\alpha_{t+1}[j] = \alpha_t[j]$ 
11:    end for
12:    if  $y_{i_t} \frac{1}{\lambda_t} \sum_j \alpha_t[j] y_{i_t} K(x_{i_t}, x_j) < 1$  then
13:      Inicializar  $\alpha_{t+1}[i_t] = \alpha_t[i_t] + 1$ 
14:    else
15:      Inicializar  $\alpha_{t+1}[i_t] = \alpha_t[i_t]$ 
16:    end if
17:  end for
18: return  $\alpha_{T+1}$ 
19: end procedure

```

Apéndice B

Lista de Tablas



















LETRA	RGB	CANNY	ETIQUETA
A			1
B			2
C			3
D			4
E			5
F			6
G			7
	↓ 	↓ 	28
H			8

Tabla B.1: Abecedario en Lenguaje de Señas Venezolano bajo su representación en RGB, Canny y Etiquetado.







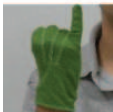













LETRA	RGB	CANNY	ETIQUETA
I			9
J			10
	↓	↓	
			29
	↓	↓	
			30
K			11
L			12
M			13
N			14
O			16
P			17

Tabla B.1: Abecedario en Lenguaje de Señas Venezolano bajo su representación en RGB, Canny y Etiquetado.
























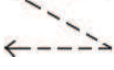

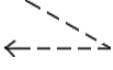
LETRA	RGB	CANNY	ETIQUETA
Q			18
	 	 	31
R			19
S			20
T			21
U			22
V			23
W			24
X			25
Y			26
Z	 	 	27

Tabla B.1: Abecedario en Lenguaje de Señas Venezolano bajo su representación en RGB, Canny y etiquetado.

Apéndice C

Experimentos de Clasificación de Señas Estáticas

En este Apéndice se describen los experimentos de clasificación de señas estáticas por cada uno de los algoritmos de clasificación, tanto por plantillas como por aprendizaje. Estos se realizan con el objetivo de encontrar la mejor configuración posible para cada método.

Cabe destacar que, para la preparación y ejecución de estos experimentos, se empleó un conjunto de datos de tipo imagen conformado por un total de 418 imágenes a color de 640x480 píxeles asociadas a las 22 señas estáticas del alfabeto del LSV. Este conjunto de datos fue dividido en dos sub-conjuntos: entrenamiento y prueba, de los cuales el primero posee un 57,9 % de las imágenes y el segundo las restantes. Cabe destacar que en los algoritmos de aprendizaje supervisado, para el entrenamiento de un clasificador de una seña en particular se emplearon 11 imágenes de la seña en cuestión y 2 por cada una de las seña estáticas restantes, imágenes que se tomadas del conjunto de entrenamiento.

C.1. Experimento #1: Clasificador por Comparación de Plantillas con Transformación de Distancias

Como se describe en la Sección 4.1.1.1, los clasificadores por Comparación de Plantillas con Transformación de Distancias (DT) emplean un conjunto de imágenes llamadas plantillas, asociadas a cada una de las señas que se quieren etiquetar, las cuales serán comparadas contra un conjunto de prueba.

Para determinar la mejor configuración de este clasificador se realizaron 5 corridas, variando en cada una la cantidad de imágenes por seña estática que conformarán el conjunto total de plantillas a utilizar. Estas 5 corridas fueron realizadas bajo las siguientes cantidades de plantillas por cada seña estática: 1, 3, 6, 9 y 11, y para todas las corridas se empleó el mismo conjunto de prueba.

Para determinar la mejor configuración de este clasificador se tomó en cuenta los siguientes factores: porcentaje de clasificación (imágenes correctamente clasificadas) y la cantidad de imágenes que este clasificador puede procesar por segundo.

En la Figura C.1 se muestra una gráfica de barras que contrasta los porcentajes de clasificación para cada corrida realizada.

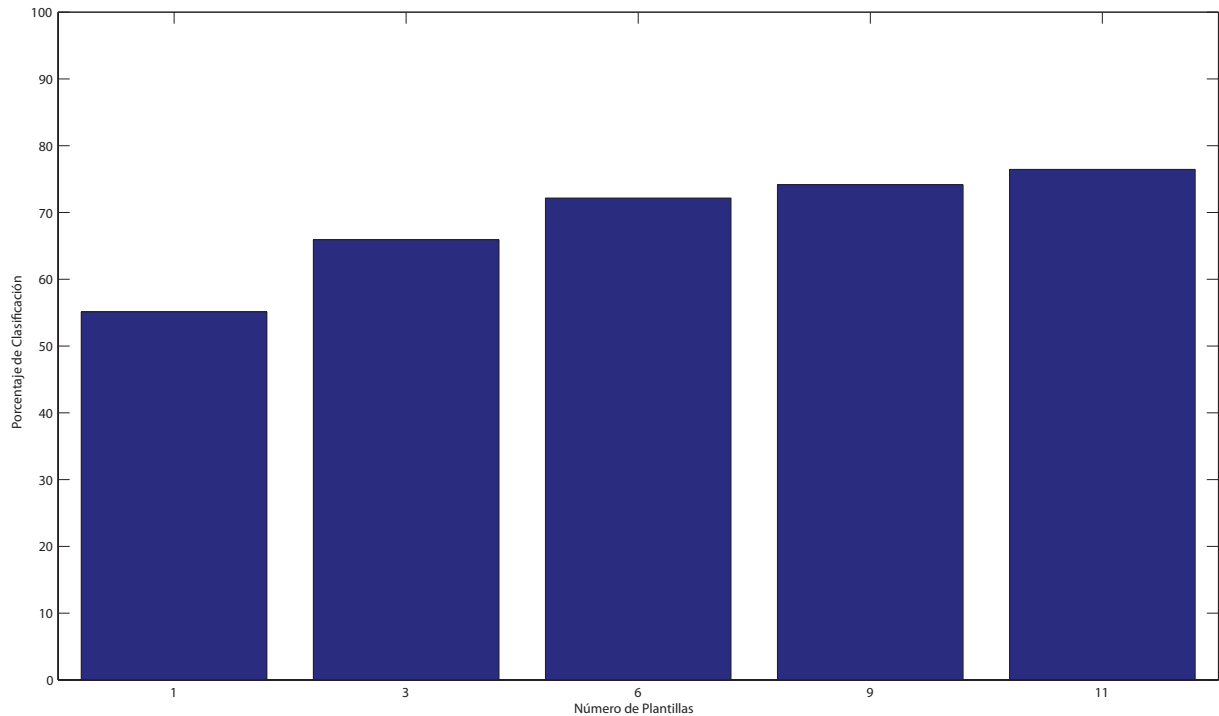


Figura C.1: Número de Plantillas Vs Porcentaje de Clasificación empleando Comparación de Plantillas con DT

Se puede observar que el porcentaje de clasificación más alto (76,44 %), se obtiene empleando 11 plantillas por señal. Cabe destacar que la diferencia de porcentajes, contra la configuración anterior, es de a lo sumo 4 % empleando las configuraciones de 6 y 9 plantillas, por lo cual se toma en consideración el tiempo de ejecución para decidir cuál de estas tres configuraciones usar. En la Tabla C.1 se muestra el tiempo total de ejecución y la cantidad de imágenes procesadas por segundo, por cada una de las configuraciones planteadas.

Con respecto a los tiempos de ejecución, se aprecia que a medida que se emplea una mayor cantidad de plantillas por señal, los tiempos de ejecución aumentan. Esto sucede porque por cada imagen a clasificar se realiza un número mayor de comparaciones, afectando negativamente la cantidad de imágenes que se pueden procesar por segundo.

Configuración (plantillas por señal)	Tiempo de Ejecución (seg)	Imágenes Procesadas por Segundo
1	0,73	239,14
3	1,34	130,57
6	2,21	79,32
9	3,05	57,58
11	3,53	49,75

Tabla C.1: Tiempos de Ejecución por Configuración

Tomando en cuenta que las configuraciones empleando de 6 a 11 plantillas por señal producen los mejores porcentajes de clasificación, la configuración que se empleará será la de mejor tiempo de ejecución. Al comparar los tiempos de ejecución (Tabla C.1) se puede observar que el uso de 6 plantillas por señal procesa las imágenes más rápido que las otras 2 configuraciones (2,21 seg), siendo este procesamiento de 79 imágenes por segundo aproximadamente. Es por esto que se puede afirmar que, dentro de las configuraciones consideradas, la mejor es el empleo de 6 plantillas por señal.

C.2. Experimento #2: Clasificador de Formas a través de Distancias Internas

Tal como se describe en la Sección 4.1.1.2, los clasificadores de este método emplean un conjunto de imágenes llamadas plantillas, asociadas a cada una de las señales que se quieren etiquetar, las cuales serán comparadas contra un conjunto de prueba.

Para determinar la mejor configuración de este clasificador se realizaron 5 corridas, variando en cada una la cantidad de imágenes por señal estática que conformarán el conjunto total de plantillas a utilizar. Estas 5 corridas fueron realizadas bajo las siguientes cantidades de plantillas por cada señal estática: 1, 3, 6, 9 y 11, y para todas las corridas se empleó el mismo conjunto de prueba.

Para determinar la mejor configuración de este clasificador se tomaron en cuenta los siguientes factores: porcentaje de clasificación (imágenes correctamente clasificadas) y la cantidad de imágenes que este clasificador puede procesar por segundo.

En la Figura C.2 se muestra una gráfica de barras que contrasta los porcentajes de clasificación para cada corrida realizada.

En la gráfica de barras se puede observar que el porcentaje de clasificación más alto (90.9 %) es empleando 11 plantillas por señal. Cabe destacar que la diferencia de porcentajes,

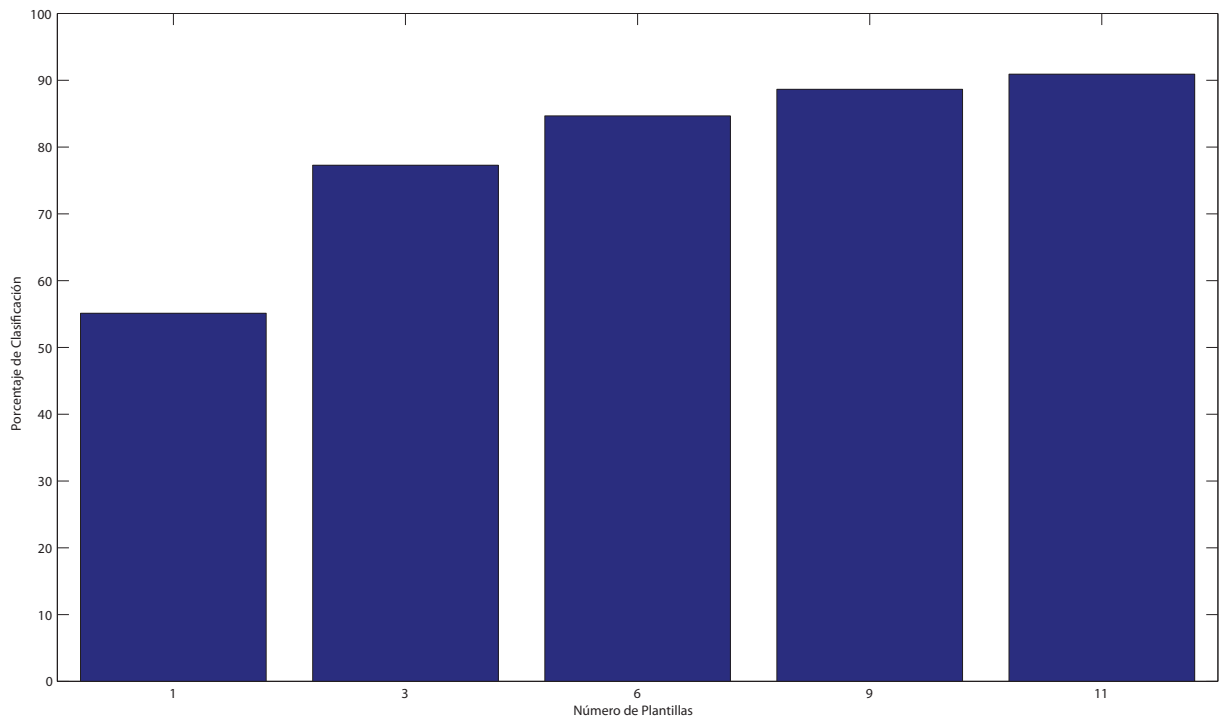


Figura C.2: Número de Plantillas Vs Porcentaje de Clasificación empleando SCUID

contra la configuración anterior, es un aproximado de 5.4 % empleando una cantidad de 6 y 9 de plantillas, por lo cual se toma en consideración el tiempo de ejecución para decidir cuál de éstas usar. En la Tabla C.2 se muestra el tiempo total de ejecución y la cantidad de imágenes procesadas por segundo, por cada una de las configuraciones planteadas.

Configuración (plantillas por seña)	Tiempo de Ejecución (seg)	Imágenes Procesadas por Segundo
1	576,47	0,31
3	1663,3	0,11
6	3322,9	0,05
9	5058,2	0,03
11	6191,81	0,03

Tabla C.2: Tiempos de Ejecución por Configuración del método SCUID

Con respecto a los tiempos de ejecución, se aprecia que a medida que se emplea una mayor cantidad de plantillas por seña, los tiempos de ejecución aumentan. Esto sucede porque por cada imagen a clasificar se realiza una mayor cantidad de plantillas por comparación, afectando negativamente la cantidad de imágenes que se pueden procesar por segundo.

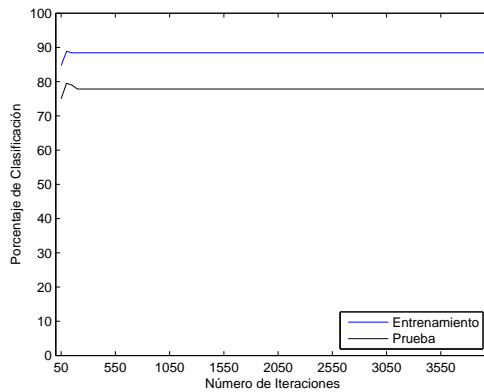
Las configuraciones empleando de 6 a 11 plantillas por seña producen los mejores porcentajes de clasificación, sobre estas configuraciones se tomará en consideración el tiempo de ejecución para decidir sobre la mejor configuración. Al comparar los tiempos de ejecución

(Tabla C.2) se puede observar que el uso de 6 plantillas por seña procesa las imágenes más rápido que las otras dos configuraciones (3322,9 seg), siendo este procesamiento de 0,05 imágenes por segundo. Gracias a este análisis se puede afirmar que, dentro de las configuraciones consideradas, la cantidad de plantillas que se usará es de 6 plantillas por seña.

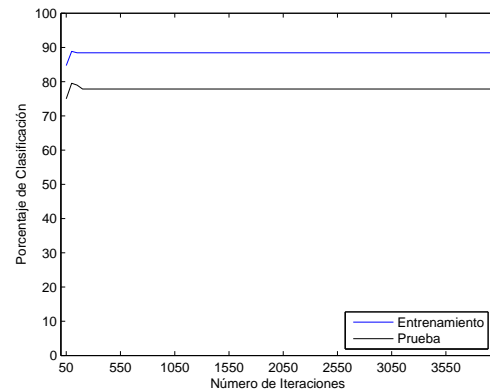
C.3. Experimento #3: Clasificador Perceptrón

El objetivo de este experimento es determinar la mejor configuración de Perceptrón para la clasificación de señas estáticas. Para esto se realizarán diversas corridas contrastando los porcentajes de clasificación generados a partir de la combinación de diversos valores de tasas de aprendizaje y tipos de normalización. Las tasas de aprendizaje empleadas en este experimento son: 0,01, 0,03, 0,05, 0,07, 0,1 y 0,2, y los tipos de normalización a considerar son: por mínimos y máximos, y por varianza y media.

A continuación se presentan gráficas que contrastan los porcentajes de clasificación sobre el conjunto de prueba contra el de entrenamiento bajo cada una de las normalizaciones, por cada tasa de aprendizaje empleada. Las normalizaciones empleadas son: por máximos y mínimos (ver Figuras C.3 y C.4), y por varianza y media (ver Figuras C.5 C.6).

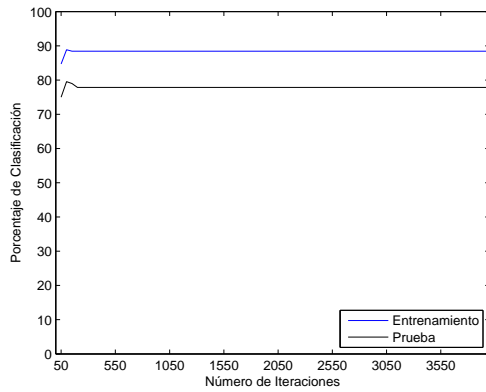


(a) Tasa de Aprendizaje: 0,01

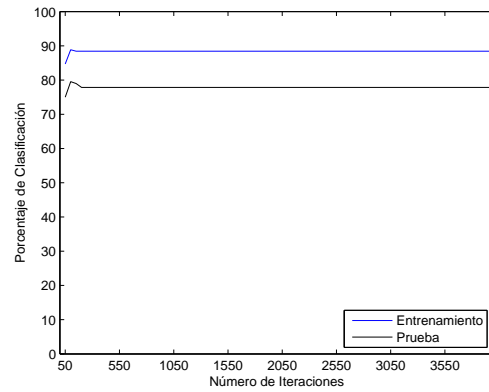


(b) Tasa de Aprendizaje: 0,03

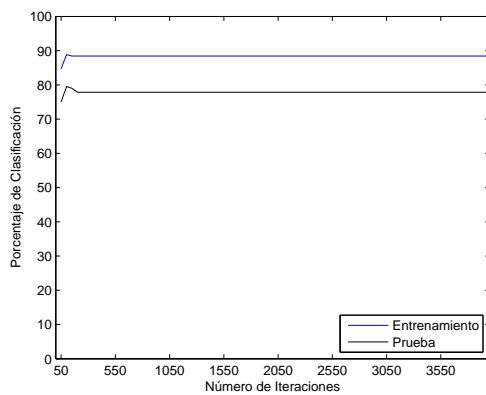
Figura C.3: Gráficas de Porcentajes de Clasificación de Perceptrones, bajo una Normalización por Mínimos y Máximos variando la tasa de aprendizaje y Tasas de Aprendizaje 0,01 y 0,03



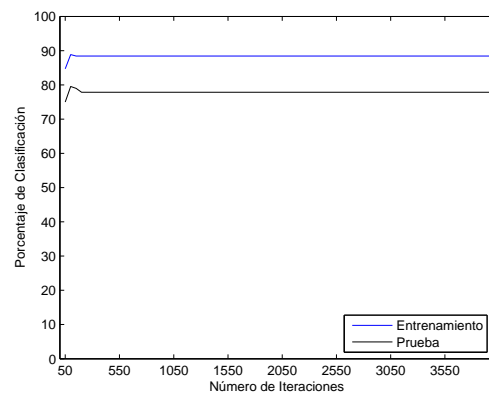
(a) Tasa de Aprendizaje: 0,05



(b) Tasa de Aprendizaje: 0,07

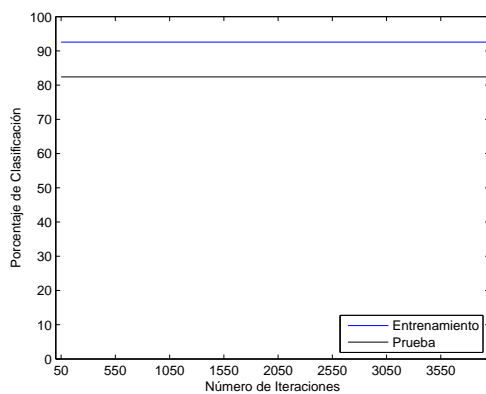


(c) Tasa de Aprendizaje: 0,1

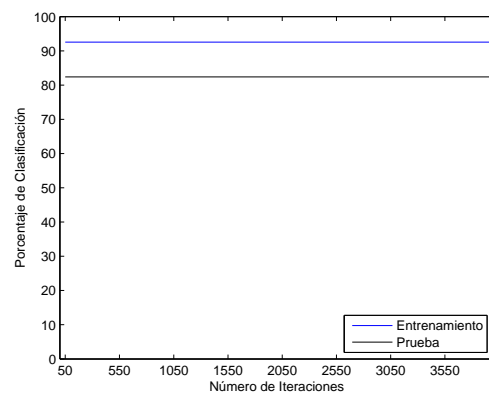


(d) Tasa de Aprendizaje: 0,2

Figura C.4: Gráficas de Porcentajes de Clasificación de Perceptrones, bajo una Normalización por Mínimos y Máximos variando la tasa de aprendizaje y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2

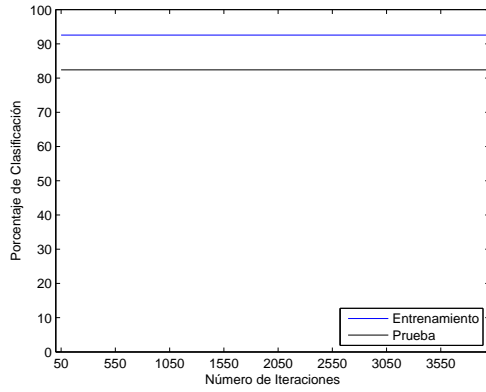


(a) Tasa de Aprendizaje: 0,01

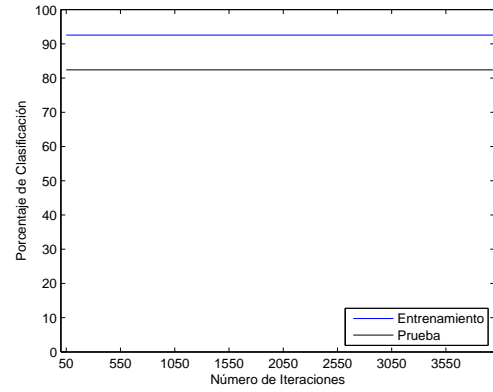


(b) Tasa de Aprendizaje: 0,03

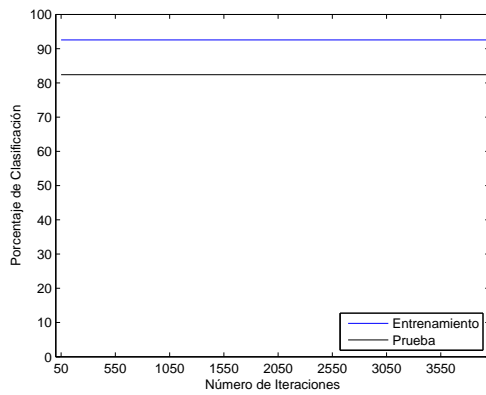
Figura C.5: Gráficas de Porcentajes de Clasificación de Perceptrones, bajo una Normalización por Varianza y Media variando la tasa de aprendizaje y Tasas de Aprendizaje 0,01 y 0,03



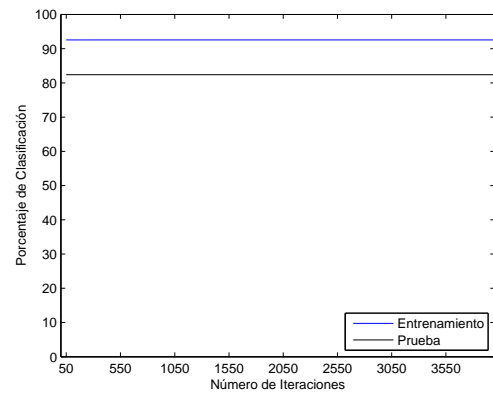
(a) Tasa de Aprendizaje: 0,05



(b) Tasa de Aprendizaje: 0,07



(c) Tasa de Aprendizaje: 0,1



(d) Tasa de Aprendizaje: 0,2

Figura C.6: Gráficas de Porcentajes de Clasificación de Perceptrones, bajo una Normalización por Varianza y Media variando la tasa de aprendizaje y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2

En ambas gráficas se puede observar que a medida que las iteraciones avanzan los cambios de valor en los porcentajes de clasificación para el conjunto de entrenamiento y prueba son parecidos entre si, a pesar que los porcentajes sean distintos. Esto permite concluir que el conjunto de entrenamiento es representativo con respecto al conjunto de prueba. Adicionalmente se observa que, para todas las configuraciones, a partir de la iteración 150 los porcentajes de clasificación se mantienen constantemente bajo el mismo valor.

A partir de los resultados obtenidos (representados en las gráficas anteriores), en la Tabla C.3 se presentan los porcentajes de mayor valor bajo cada una de las combinaciones mencionadas.

En la tabla anterior, se puede observar que para ambas normalizaciones el uso de cualquier tasa de aprendizaje generará como máximo el mismo porcentaje de clasificación sobre el

Tasa de Aprendizaje	Mínimos y Máximos				Varianza y Media			
	Entrenamiento		Prueba		Entrenamiento		Prueba	
	Max %	Iteración	Max %	Iteración	Max %	Iteración	Max %	Iteración
0,01	88,84 %	100	79,54 %	100	92,56 %	50	82,38 %	50
0,03	88,84 %	100	79,54 %	100	92,56 %	50	82,38 %	50
0,05	88,84 %	100	79,54 %	100	92,56 %	50	82,38 %	50
0,07	88,84 %	100	79,54 %	100	92,56 %	50	82,38 %	50
0,1	88,84 %	100	79,54 %	100	92,56 %	50	82,38 %	50
0,2	88,84 %	100	79,54 %	100	92,56 %	50	82,38 %	50

Tabla C.3: Porcentajes e Iteraciones Corredpondientes a los Mejores Resultados por Configuración del Perceptrón

conjunto de prueba, por mínimos y máximos 79,54 % y por varianza y media 82,38 %. A partir de esto se observa que la normalización por varianza y media es la que ayuda a la obtención de mayores porcentajes de clasificación. Finalmente, dado que cualquiera de las tasas de aprendizaje generan los mejores resultados, se decide emplear una de valor 0,03.

C.4. Experimento #4: Clasificador Red Neural

El objetivo de este experimento es generar la mejor configuración de la Red Neural para la clasificación de señas estáticas. Para esto, primero se contrastan diferentes configuraciones de cantidad de neuronas y capas ocultas. Con la mejor configuración obtenida, se evalúa qué combinación de normalización (por Máximos y Mínimos, o por Varianza y Media) y tasa de aprendizaje genera los mejores resultados como configuración global.

Para la determinación de la configuración de cantidad de neuronas y capas ocultas, se comparan los resultados con redes de 1 capa oculta de 5, 10 y 15 neuronas y redes de 2 capas ocultas de 5 o 10 neuronas. A continuación se presenta en la Figura C.7 una gráfica de barras que contrasta los resultados bajo las configuraciones planteadas.

Se puede observar que el uso de una red con 1 capa oculta de 15 neuronas, otra con 2 capas ocultas de 5 neuronas cada u otra de 2 capas ocultas de 10 neuronas son las que generaron los mejores porcentajes de clasificación, donde el mayor porcentaje (78,4 %) corresponde a las últimas dos redes. Debido a este empate se decide emplear la red de menor costo, la cual es de 2 capas ocultas de 5 neuronas cada una.

Una vez determinada que la mejor configuración es la combinación de 2 capas ocultas de 5 neuronas cada una, se procede a determinar el tipo de normalización, a aplicar sobre las

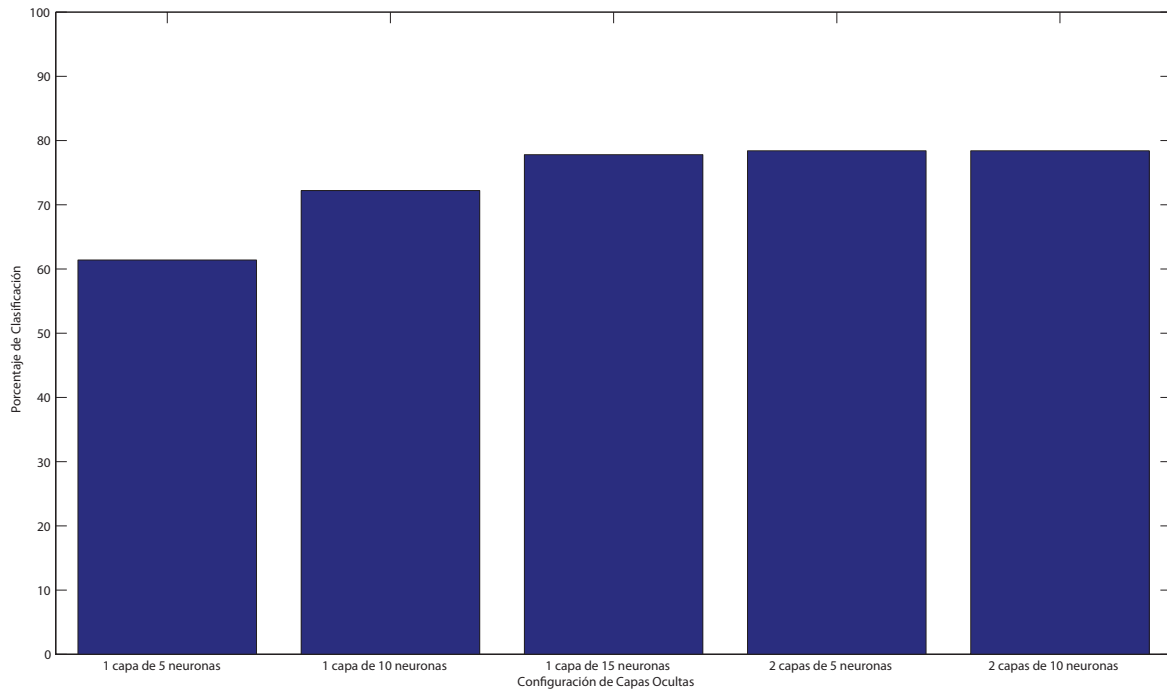
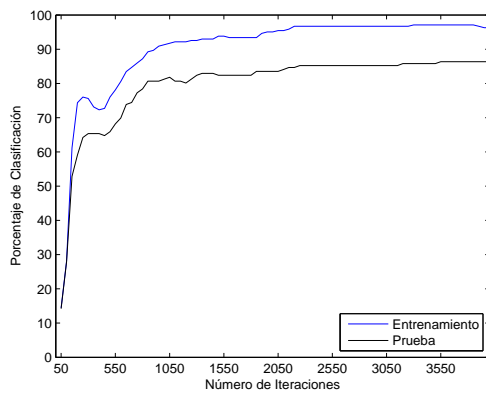


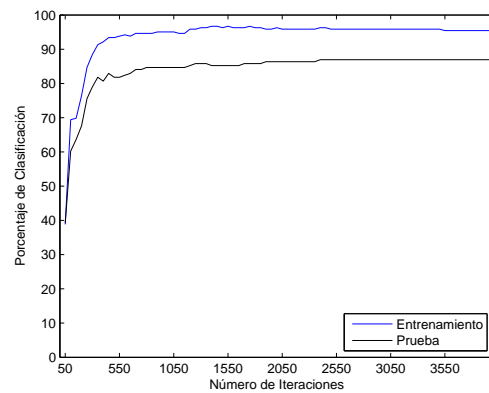
Figura C.7: Porcentajes de Clasificación de una Red Neural Modificando la Configuración de sus Capas Ocultas

imágenes del conjunto de prueba y entrenamiento, bajo una tasa de aprendizaje específica a ser usada en el entrenamiento de la red.

A continuación se presentan gráficas que contrastan los porcentajes de clasificación sobre el conjunto de prueba contra el de entrenamiento bajo cada una de las normalizaciones, por cada tasa de aprendizaje empleada. Las tasas de aprendizaje empleadas son: 0,01, 0,03, 0,05, 0,07, 0,1 y 0,2, y las normalizaciones empleadas son: por máximos y mínimos (ver Figuras C.8 y C.9), y por varianza y media (ver Figuras C.10 y C.11).

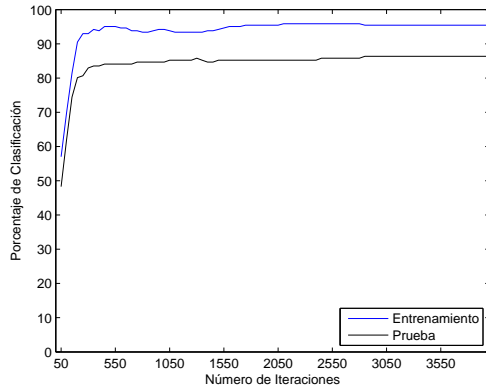


(a) Tasa de Aprendizaje: 0,01

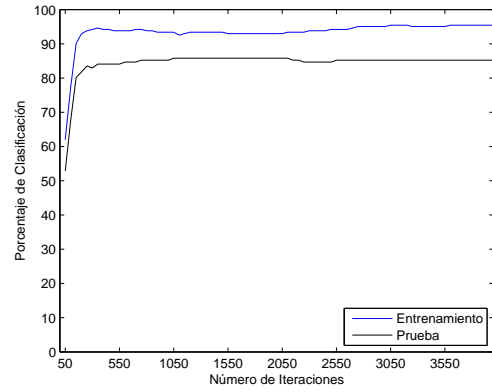


(b) Tasa de Aprendizaje: 0,03

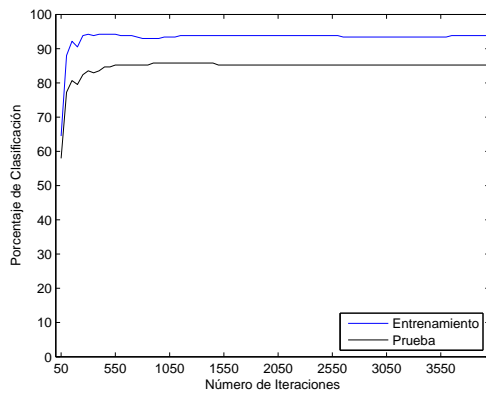
Figura C.8: Gráfica de Porcentajes de Clasificación de Red Neural bajo una Normalización por Mínimos y Máximos variando la tasa de aprendizaje y Tasas de Aprendizaje 0,01 y 0,03



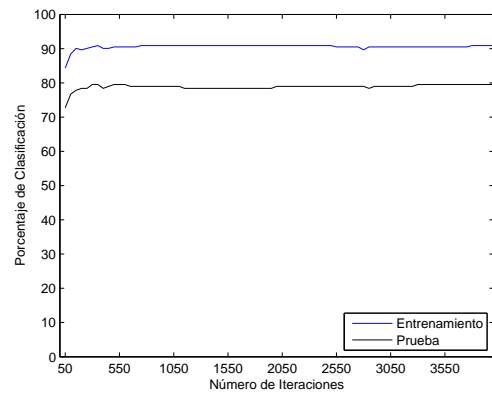
(a) Tasa de Aprendizaje: 0,05



(b) Tasa de Aprendizaje: 0,07

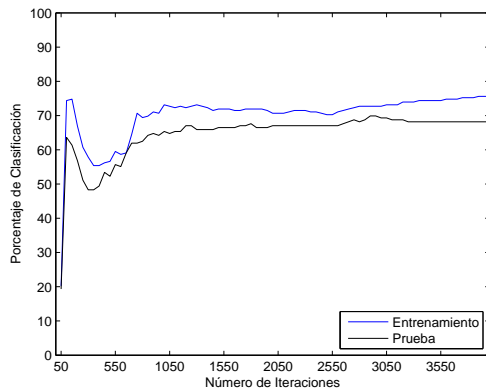


(c) Tasa de Aprendizaje: 0,1

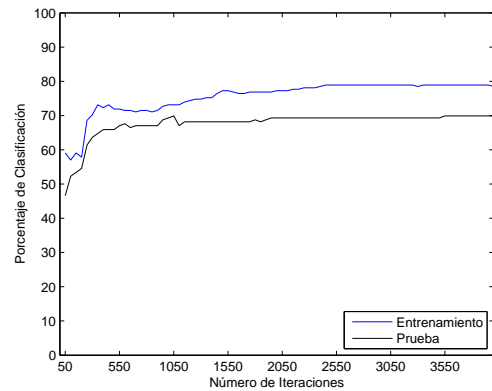


(d) Tasa de Aprendizaje: 0,2

Figura C.9: Gráfica de Porcentajes de Clasificación de Red Neural bajo una Normalización por Mínimos y Máximos variando la tasa de aprendizaje y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2

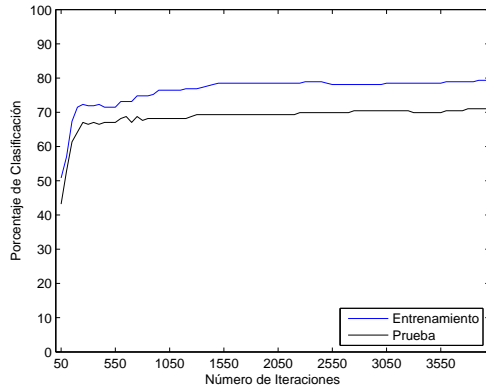


(a) Tasa de Aprendizaje: 0,01

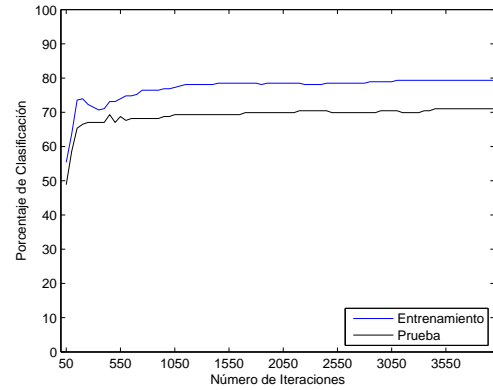


(b) Tasa de Aprendizaje: 0,03

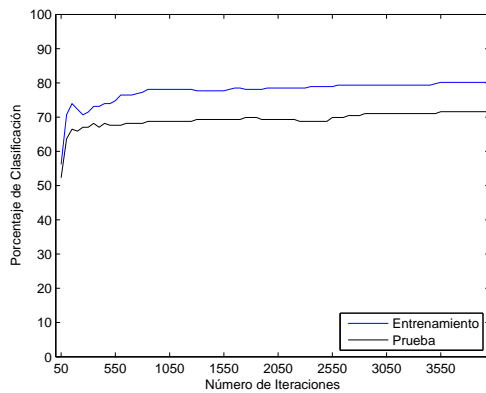
Figura C.10: Gráfica de Porcentajes de Clasificación de Red Neural bajo una Normalización por Varianza y Media variando la tasa de aprendizaje y Tasas de Aprendizaje 0,01 y 0,03



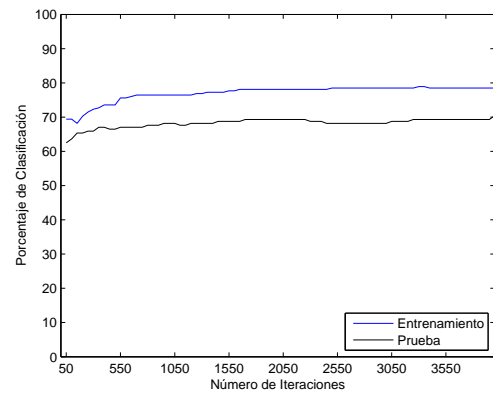
(a) Tasa de Aprendizaje: 0,05



(b) Tasa de Aprendizaje: 0,07



(c) Tasa de Aprendizaje: 0,1



(d) Tasa de Aprendizaje: 0,2

Figura C.11: Gráfica de Porcentajes de Clasificación de Red Neural bajo una Normalización por Varianza y Media variando la tasa de aprendizaje y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2

A partir de los resultados representados en las gráficas anteriores, en la Tabla C.4 se presenta los porcentajes de mayor valor bajo cada una de las combinaciones mencionadas.

Tasa de Aprendizaje	Mínimos y Máximos				Varianza y Media			
	Entrenamiento Max %	Iteración	Prueba Max %	Iteración	Entrenamiento Max %	Iteración	Prueba Max %	Iteración
0,01	97,11	3300	86,36	3550	75,62	3900	69,89	2900
0,03	96,69	1400	86,93	2400	78,93	2450	69,89	1050
0,05	95,87	2100	86,36	2850	79,34	3900	71,02	3800
0,07	95,45	3050	85,8	1050	79,34	3100	71,02	3450
0,1	94,21	400	85,8	900	80,17	3550	71,59	3550
0,2	90,91	350	79,55	500	78,93	3300	70,45	4000

Tabla C.4: Porcentajes e Iteraciones Correspondientes a los Mejores Resultados por Configuración de la Red neural

Se puede observar que en el transcurso de las iteraciones los cambios de valor en los porcentajes de clasificación, sobre la evaluación de los conjuntos de entrenamiento y prueba,

son muy parecidos. Esto permite afirmar que el conjunto de entrenamiento es representativo con respecto al de prueba.

Cabe acotar que con una normalización por mínimos y máximos se generan los porcentajes más altos. Con respecto a las tasas de aprendizaje bajo la normalización antes mencionada, se observa que los mejores resultados bajo el conjunto de prueba son de 86,96 % con una tasa de aprendizaje de 0,03, siendo esta tasa la que genera el segundo mejor porcentaje bajo el conjunto de entrenamiento.

Es por los resultados y análisis anteriores, que se puede afirmar que la configuración de Red Neural que generó los mejores resultados es de 2 capas ocultas de 5 neuronas cada una, con una tasa de aprendizaje de 0,03 y una normalización de las imágenes de entrada por mínimos y máximos.

C.5. Experimento #5: Clasificador SVM-Pegasos por Lotes Pequeños

El objetivo de este experimento es determinar la mejor configuración del SVM-Pegasos por Lotes Pequeños para la clasificación de señas estáticas. Para esto se realizan diversas corridas variando la normalización empleada y el parámetro λ de regulación de SVMs, el cual influye en dos aspectos del clasificador: el límite de iteraciones de entrenamiento y la tasa de aprendizaje. El límite de iteraciones se determina por la Ecuación C.1, donde n representa la cantidad de entradas (píxeles de la imagen), ϵ la ϵ -precisión de la solución (valor empleado 0,5²). La tasa de aprendizaje se calcula por cada iteración y se determina por la Ecuación C.2, donde t representa la iteración en que se calcula dicha tasa.

$$iteraciones = \frac{n}{\epsilon \lambda} \quad (C.1)$$

$$N = \frac{1}{t\lambda} \quad (C.2)$$

Los tipos de normalización empleados fueron: por mínimos y máximos, y por varianza y media. Los valores que se emplearon como λ se determinaron por el cálculo de: 2^n , donde n toma los valores: -5, -3, -1, 1, 3, 5, 7, 9, 11, 13 y 15, tomados de [6].

A continuación se presenta en la Figura C.12 una gráfica de barras que presenta los

resultados bajo normalización por mínimos y máximos empleando los valores de λ planteados.

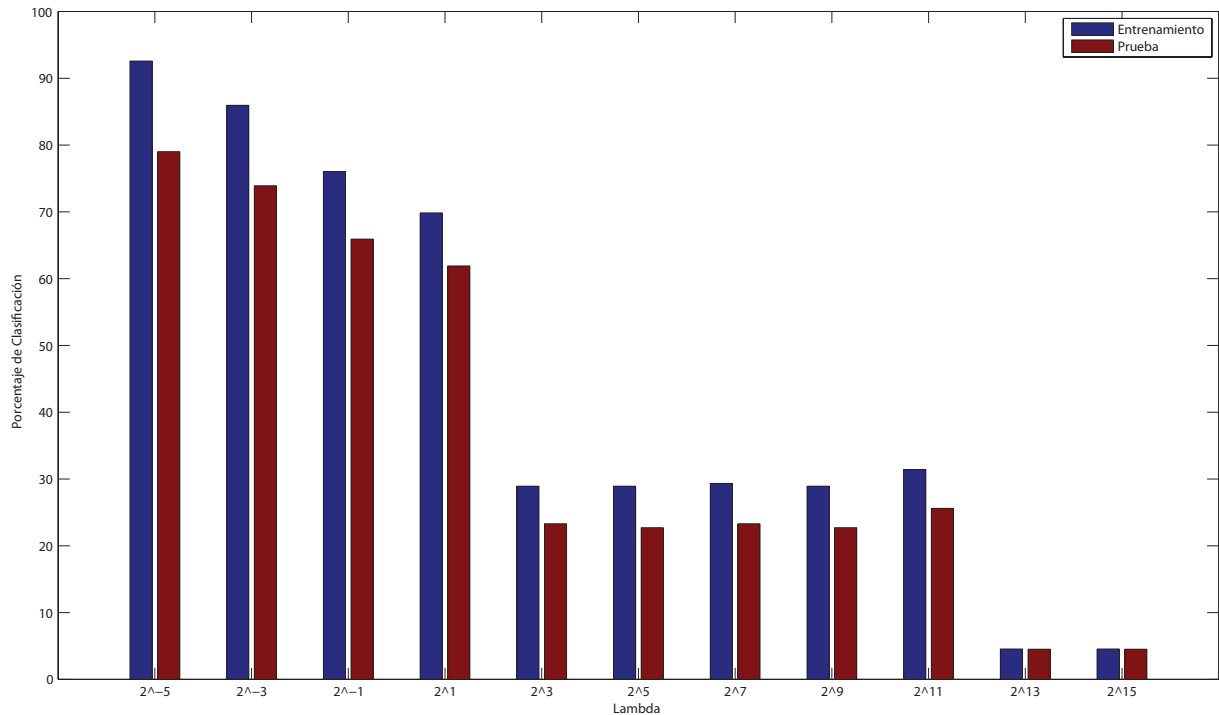


Figura C.12: Porcentajes de Clasificación bajo valores λ de SVM-Pegasos por Lotes Pequeños con una normalización por Mínimos y Máximos

En esta gráfica se puede observar que el valor λ influye considerablemente en el porcentaje de clasificación, donde el menor valor λ genera el porcentaje de acierto es más elevado (79 % sobre el conjunto de prueba y 95,2 % con entrenamiento con $\lambda 2^{-5}$) y a medida que el λ aumenta los porcentajes decaen hasta un 4,5 %. Esto se debe a que mientras más elevado sea el valor λ menos iteraciones de entrenamiento se realizarán. Además la tasa de aprendizaje inicial será menor, y a medida que itere el entrenamiento decrementará aun más, realizando saltos de menor tamaño. Es por esto que se puede concluir que con una normalización de mínimos y máximos el mejor valor de λ empleado es de 2^{-5} .

A continuación se presenta en la Figura C.13 una gráfica de barras que presenta los resultados bajo normalización por varianza y media empleando los valores de λ planteados.

En esta gráfica se puede observar el mismo comportamiento ocurrido con la normalización de mínimos y máximos, donde a menos valor λ menor será el porcentaje de acierto. En este caso el mayor porcentaje (85,2 % sobre el conjunto de prueba) se da con un λ de valor 2^{13} .

En las dos gráficas se puede observar un comportamiento similar al contrastar los porcen-

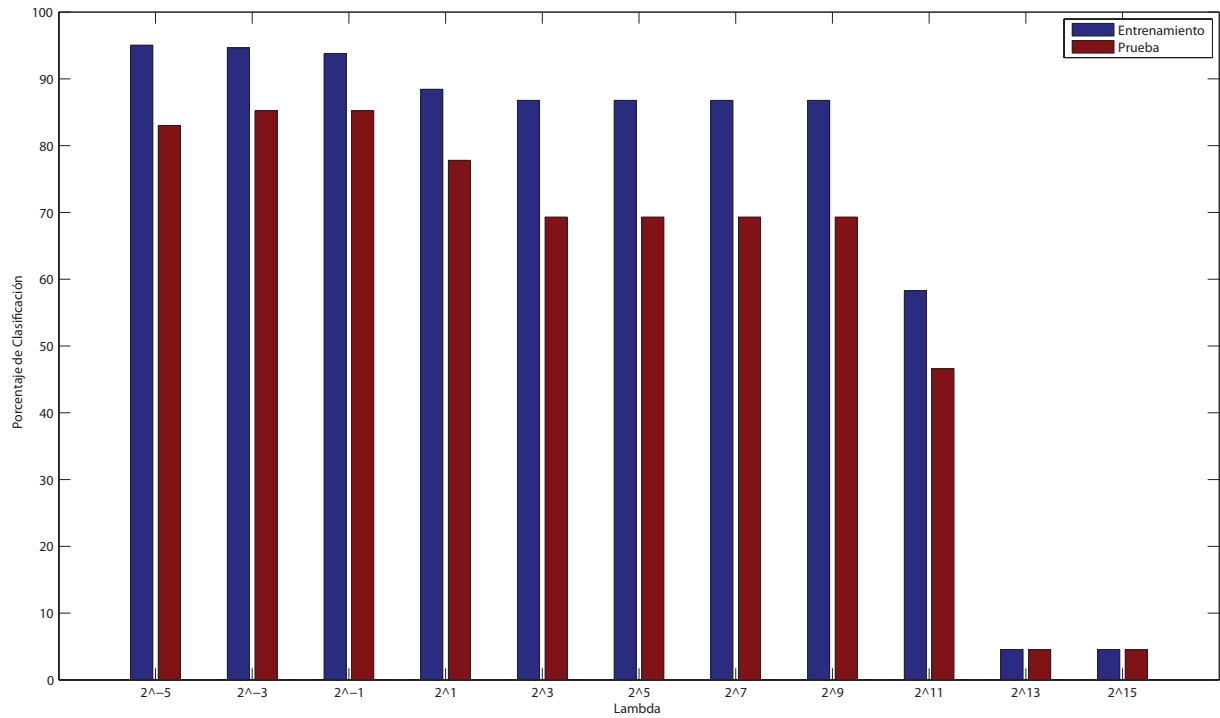


Figura C.13: Gráfica de barras de las configuraciones λ del SVM-Pegasos por Lotes Pequeños con una normalización de Varianza y Media

tajes de clasificación de entrenamiento y prueba de mismo λ , donde los porcentajes generados son cercanos entre si. Es por esto que se puede concluir que el conjunto de entrenamiento es representativo sobre el conjunto de prueba usado. Debido a que ambas configuraciones cumplen lo antes mencionado, se decide emplear la configuración que ofrece los mayores porcentajes de clasificación: normalización por varianza y media con un λ de valor 2^{-3} .

C.6. Experimento #6: Clasificador SVM-Pegasos con Kernel

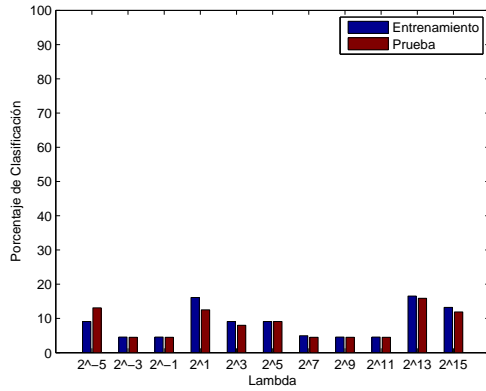
El objetivo de este experimento es determinar la mejor configuración del SVM-Pegasos por Lotes Pequeños para la clasificación de señas estáticas. Para esto se realizan diversas corridas variando la normalización empleada, σ de la función Kernel RBF (Radial Basic Function, por sus siglas en inglés Ecuación. C.3) y el parámetro λ de regulación de SVMs, el cual influye en dos aspectos del clasificador: el límite de iteraciones de entrenamiento y la tasa de aprendizaje. El límite de iteraciones se determina por la Ecuación C.1, donde n representa la cantidad de entradas (píxeles de la imagen), ϵ la ϵ -precisión de la solución (valor empleado 0,5²). La tasa de aprendizaje se calcula por cada iteración y se determina por la Ecuación C.2 (Sección C.5), donde t representa la iteración en que se calcula dicha

tasa.

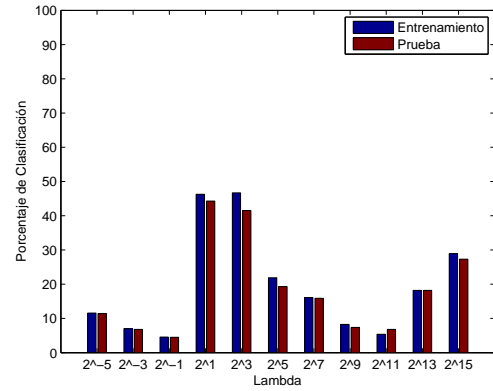
$$RBF = e^{-(\|x_i - x_j\|^2)/(2\sigma^2)} \quad (C.3)$$

Los tipos de normalización empleados fueron: por mínimos y máximos, y por varianza y media. Los valores que se emplearon como λ se determinaron por el cálculo de 2^n , donde n toma los valores: -5, -3, -1, 1, 3, 5, 7, 9, 11, 13 y 15, y como σ por el mismo cálculo anterior donde n toma los valores: -15, -13, -11, -9, -7, -5, -3, -1, 1 y 3, tomados de [6].

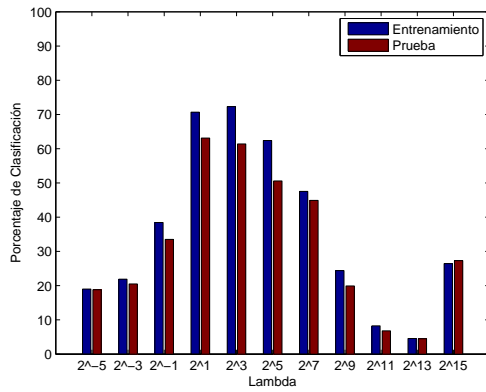
A continuación se presentan las gráficas que contrastan los porcentajes de clasificación sobre el conjunto de prueba contra el de entrenamiento bajo cada una de las combinaciones mencionadas bajo una normalización específica: por máximos y mínimos (ver Figuras C.14 y C.15), y por varianza y media (ver Figuras C.16 y C.17).



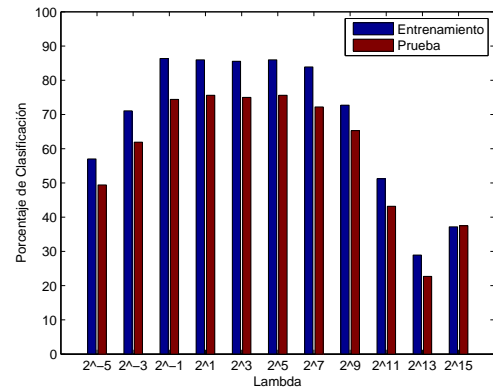
(a) Sigma: 2^{-15}



(b) Sigma: 2^{-13}



(c) Sigma: 2^{-11}



(d) Sigma: 2^{-9}

Figura C.14: Porcentajes de clasificación de un SVM-Pegasos con Kernel con Normalización por Mínimos y Máximos, bajo los valores de λ evaluados y σ : 2^{-15} 2^{-13} 2^{-11} 2^{-9}

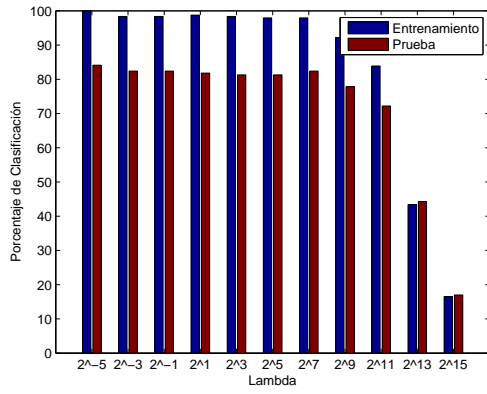
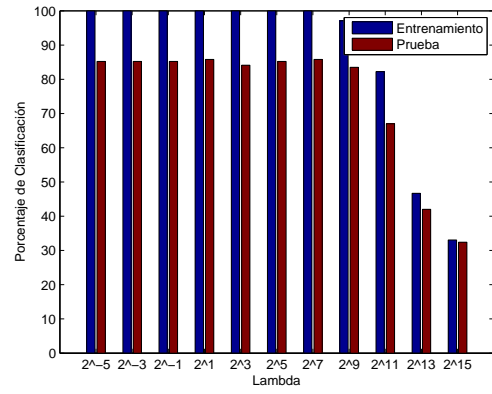
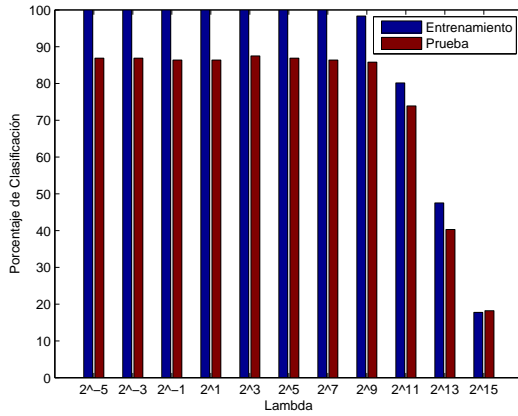
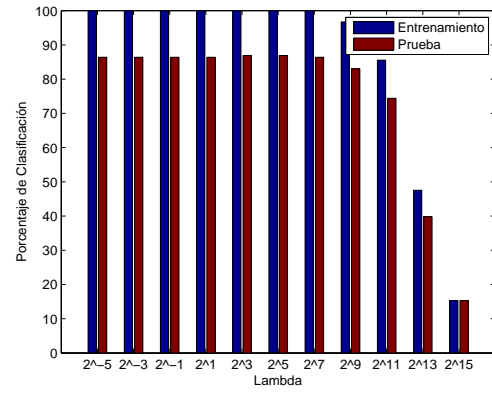
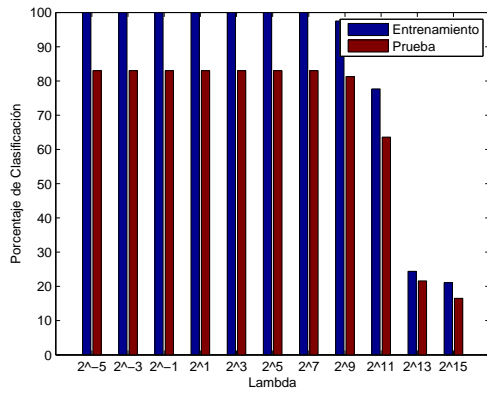
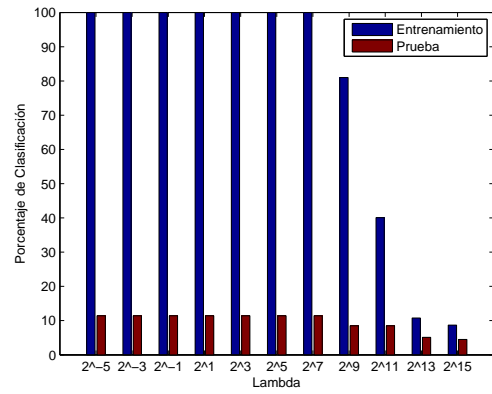
(a) Sigma: 2^{-7} (b) Sigma: 2^{-5} (c) Sigma: 2^{-3} (d) Sigma: 2^{-1} (e) Sigma: 2^1 (f) Sigma: 2^3

Figura C.15: Porcentajes de clasificación de un SVM-Pegasos con Kernel con Normalización por Mínimos y Máximos, bajo los valores de λ evaluados y σ : 2^{-7} 2^{-5} 2^{-3} 2^{-1} 2^1 2^3

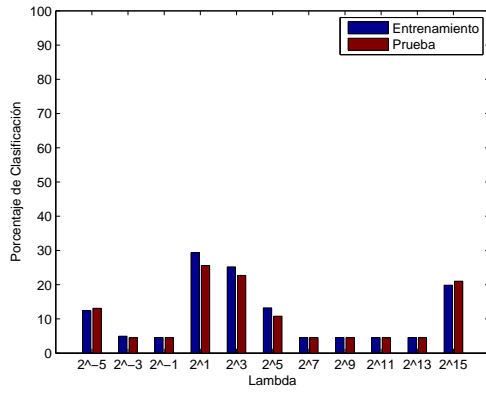
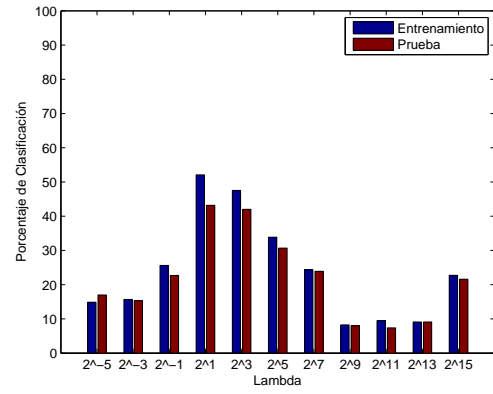
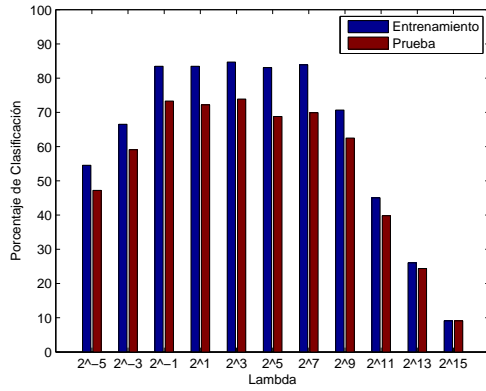
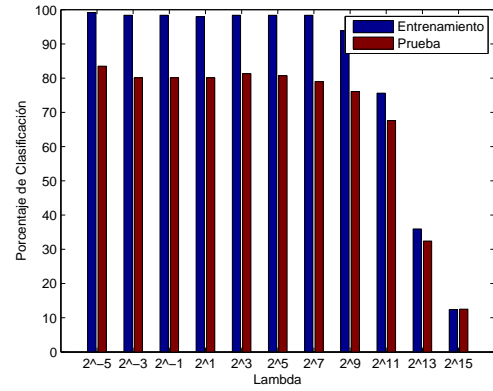
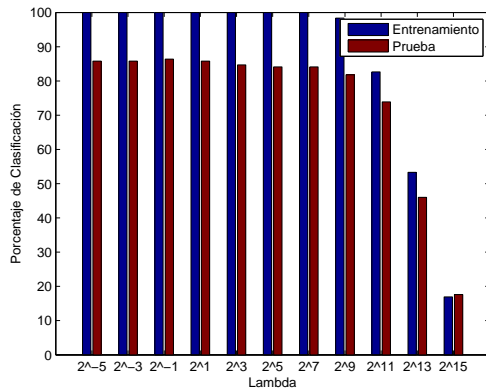
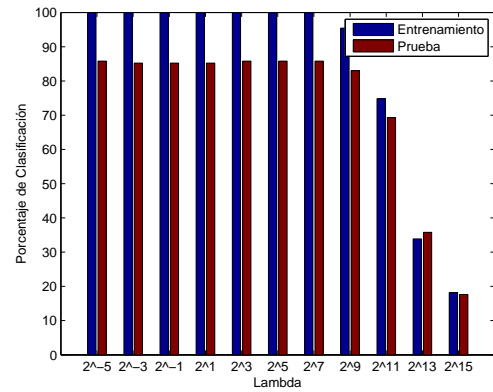

 (a) Sigma: 2^{-15}

 (b) Sigma: 2^{-13}

 (c) Sigma: 2^{-11}

 (d) Sigma: 2^{-9}

 (e) Sigma: 2^{-7}

 (f) Sigma: 2^{-5}

 Figura C.16: Porcentajes de clasificación de un SVM-Pegasos con Kernel con Normalización por Varianza y Media, bajo los valores de λ evaluados y σ : 2^{-15} 2^{-13} 2^{-11} 2^{-9} 2^{-7} 2^{-5}

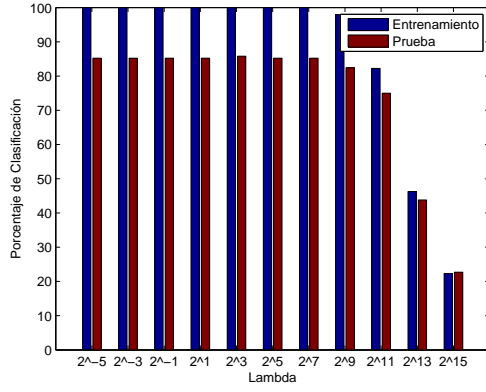
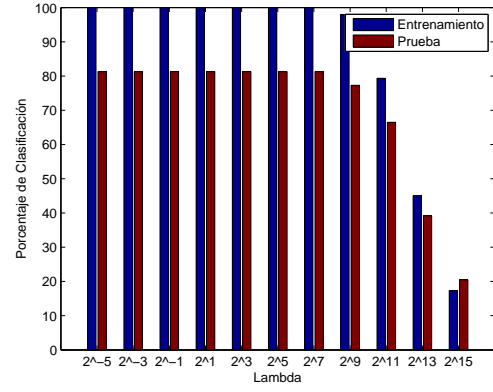
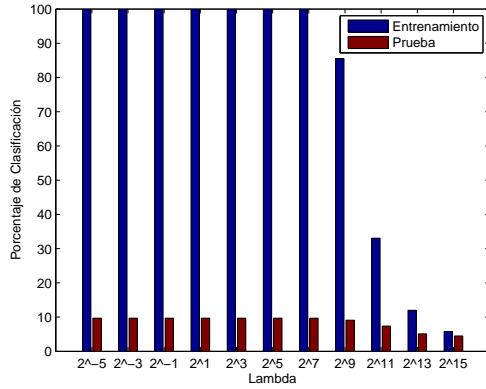
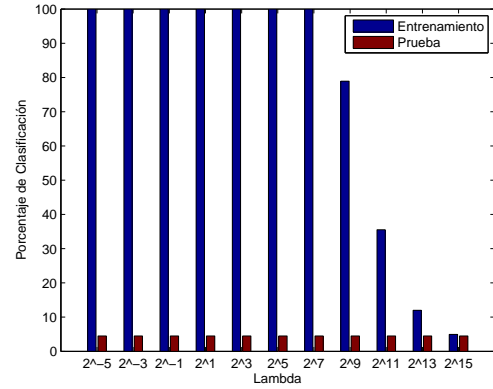
(a) Sigma: 2^{-3} (b) Sigma: 2^{-1} (c) Sigma: 2^1 (d) Sigma: 2^3

Figura C.17: Porcentajes de clasificación de un SVM-Pegasos con Kernel con Normalización por Varianza y Media, bajo los valores de λ evaluados y σ : 2^{-3} 2^{-1} 2^1 2^3

A partir de los resultados representados en las gráficas anteriores, en la Tabla C.5 se presentan los porcentajes de mayor valor bajo cada una de las combinaciones mencionadas.

Sigma (σ)	Mínimos y Máximos				Varianza y Media			
	Entrenamiento Max %	λ	Prueba Max %	λ	Entrenamiento Max %	λ	Prueba Max %	λ
2^{-15}	16.5	2^{13}	15,9	2^{13}	29.33	2^1	25,6	2^1
2^{-13}	46.69	2^3	44,3	2^1	52.06	2^1	43,2	2^1
2^{-11}	72.31	2^3	63,1	2^1	84.71	2^3	73,9	2^3
2^{-9}	86.36	2^{-1}	75,6	2^1	99.17	2^{-5}	83,5	2^{-5}
2^{-7}	100	2^{-5}	84,1	2^{-5}	100	2^{-5}	86,4	2^{-1}
2^{-5}	100	2^{-5}	85,8	2^1	100	2^{-5}	85,8	2^{-5}
2^{-3}	100	2^{-5}	87,5	2^3	100	2^{-5}	85,8	2^3
2^{-1}	100	2^{-5}	86,9	2^3	100	2^{-5}	81,3	2^{-5}
2^1	100	2^{-5}	83	2^{-5}	100	2^{-5}	9,7	2^{-5}
2^3	100	2^{-5}	11,4	2^{-5}	100	2^{-5}	4,5	2^{-5}

Tabla C.5: Tabla resumen de las configuraciones de σ del SVM-Pegasos con Kernel

En esta tabla se puede apreciar que el máximo valor de porcentaje de clasificación, con la normalización de mínimos y máximos, se encuentra en el valor σ de 2^{-3} con un 87,5 % de clasificación sobre el conjunto de prueba. Con la normalización de varianza y media se observa que el máximo se encuentra en el σ de valor 2^{-7} con un total de 86,4 % de clasificación sobre prueba.

Cabe acotar que sobre el conjunto de entrenamiento el porcentaje de clasificación es igual en los máximos de cada normalización con un 100 %, pero sobre el conjunto de prueba se destaca la normalización por mínimos y máximos que generan los porcentajes más altos. Siendo una diferencia de los máximos de 1,1 % con respecto a la normalización por varianza y media, siendo el porcentaje de clasificación por máximos y mínimos de un 87,5 %.

En las dos gráficas se puede observar un comportamiento similar al contrastar los porcentajes de clasificación de entrenamiento y prueba del mismo σ , donde los porcentajes generados son cercanos entre si. Es por esto que se puede concluir que el conjunto de entrenamiento es representativo sobre el conjunto de prueba usado. Debido a que ambas configuraciones cumplen lo antes mencionado, se decide emplear la configuración que ofrece los mayores porcentajes de clasificación: normalización por mínimos y máximos con un σ de valor 2^{-3} y un λ de 2^3 ya que presenta una mejor configuración.

Apéndice D

Entonación de las Redes Neuronales

D.1. Mejora del Método Elegido

Este Apéndice describe la generación del mejor conjunto de clasificadores de señas, tanto estáticas como dinámicas. Para ello se emplea el clasificador por Redes Neuronales, del cual se obtuvieron los mejores resultados en: Experimento #1: Determinar del Mejor Método de Clasificación (Sección 6.1). Para realizar esta entonación se emplearon las Redes Neuronales mencionadas en el experimento anterior.

A continuación, en la Figura D.1, se presenta una gráfica con los porcentajes de clasificación (imágenes correctamente clasificadas) obtenidos a través del uso de los 27 clasificadores en conjunto bajo una misma iteración de entrenamiento, valor de iteración que varía entre 50 y 4000. Estos porcentajes son generados a partir de un conjunto de prueba conformado por 8 imágenes de cada clasificador, dando un total de 248 imágenes.

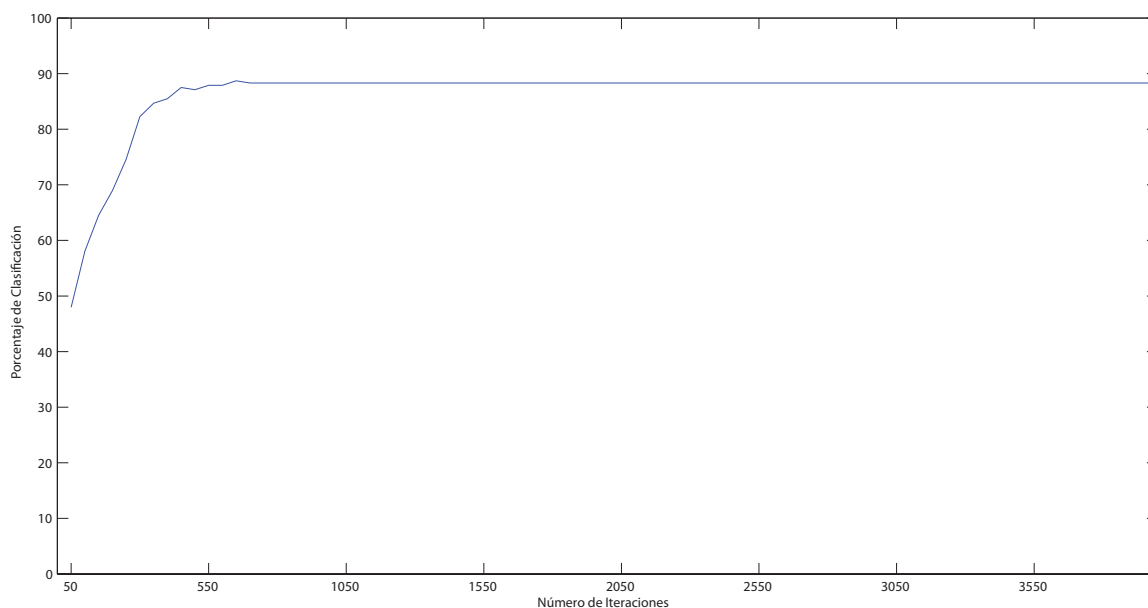


Figura D.1: Porcentajes de Clasificación de la Red Neural Variando las Iteraciones de Entrenamiento

A partir de los resultados obtenidos, se puede observar que en la iteración 650 se generan los mejores resultados donde todos los clasificadores emplean una misma iteración, por lo cual esta iteración será utilizada como base para los experimentos de Proceso de Traducción de Video.

Apéndice E

Configuración del Árbol de Decisión de Clusters

En este Apéndice se presenta la configuración estructura de Árbol de Decisión de Clusters empleada en el diseño e implementación del traductor LSV-Castellano. El apéndice se presenta dividido en dos puntos, donde el primero se encuentra bajo el nombre Nivel #1 y presenta las plantillas utilizadas por el clasificador, basado en comparación de plantillas, utilizado para clasificar la forma de la mano. El segundo punto, llamado Nivel #2, posee los experimentos realizados para determinar la mejor configuración del clasificador de aprendizaje supervisado, utilizado para determinar si la mano se encuentra en posición lateral.

E.1. Nivel #1: Descripción de las plantillas utilizadas

Como se presentó en la Sección 5.1, la idea de construir un árbol de decisión de conglomerados nace con la necesidad de disminuir la cantidad de clasificadores totales a evaluar por frame. Se planteó utilizar inicialmente un atributo de nombre Forma el cual determinará a cual de los tres grupos: Forma1, Forma2 o Forma3, pertenece. El valor de este atributo para una seña en particular es calculado utilizando el clasificador de Comparación de Plantillas con Transformación Distancias, con el cual se construye el primer nivel del árbol generando tres clusters iniciales.

Como fue explicado en la Sección 5.1, se utilizaron un conjunto de plantillas por cada tipo de Forma, las cuales se encargan de determinar que tan parecida es la seña a esa forma. A continuación, en la tabla E.1 se muestran las plantillas utilizadas para determinar cada valor del atributo Forma.

Una vez determinado el conjunto Forma al que pertenece la mano, se procede a evaluar el atributo Lateral que permitirá obtener clusters de menor tamaño. A continuación se muestra la configuración elegida para los clasificadores encargados de definir el valor de este atributo









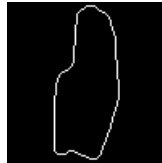









Tabla de Platillas del Atributo Forma.				
Forma 1				
				
				
Forma 2				
				
Forma 3				
				
				

Tabla E.1: Representación de las plantillas de contorno que definen los tres valores posibles del atributo Forma.

para una seña dada.

E.2. Nivel #2: Descripción de las plantillas utilizadas

Para determinar si una seña es lateral o no se utilizó el clasificador basado en aprendizaje Red Neural, tal como se presentó en la Sección 5.1. La decisión de utilizar este clasificador se debe a los buenos resultados obtenidos por él en los experimentos de la Sección 6.1. Se utilizará una Red Neural binaria asociada a cada cluster, menos al cluster de Forma 2 el cual

solo posee dos posibles señas, resultando un total de dos Redes Neurales. En esta sección del apéndice se presentan los experimentos realizados con la finalidad de obtener la mejor configuración para estos dos clasificadores.

Los experimentos fueron realizados empleando un conjunto de datos tipo imagen conformados por un total de 420 imágenes a color de 640x480 píxeles, asociadas a las 22 señas estáticas del alfabeto del LSV y 5 señas correspondientes a la primera posición de las señas dinámicas del alfabeto LSV. Para el entrenamiento de cada Red Neural se utilizaron 110 imágenes, donde 55 imágenes son ejemplos de Lado y 55 de no Lado. Para el Test se utilizaron las 200 imágenes restantes, evaluando 120 para el primer clasificador y 80 para el segundo.

El primer experimento consiste en conseguir la mejor combinación de cantidades de capas y neuronas ocultas para las redes neurales encargadas de determinar si una seña es Lateral o no. Para conseguir la mejor combinación se compararon los resultados con redes de 1 capa oculta de 5, 10 y 15 neuronas y redes de 2 capas ocultas de 5 y 10 neuronas. A continuación se muestran en la Tabla E.2, para el cluster de Forma 1, y en la Tabla E.3, para el cluster de Forma 3, las configuraciones evaluadas y el porcentaje de clasificación sobre el conjunto de prueba obtenido.

Configuración	Porcentaje de Clasificación
1 Capa de 5 Neuronas	94,17 %
1 Capa de 10 Neuronas	95 %
1 Capa de 15 Neuronas	92,5 %
2 Capas de 5 Neuronas	92,5 %
2 Capas de 10 Neuronas	94,17 %

Tabla E.2: Porcentajes de Clasificación de la Red Neural Lateral para el Cluster de Forma 1 Modificando la Configuración de sus Capas Ocultas

Configuración	Porcentaje de Clasificación
1 Capa de 5 Neuronas	97,5 %
1 Capa de 10 Neuronas	97,5 %
1 Capa de 15 Neuronas	97,5 %
2 Capas de 5 Neuronas	97,5 %
2 Capas de 10 Neuronas	97,5 %

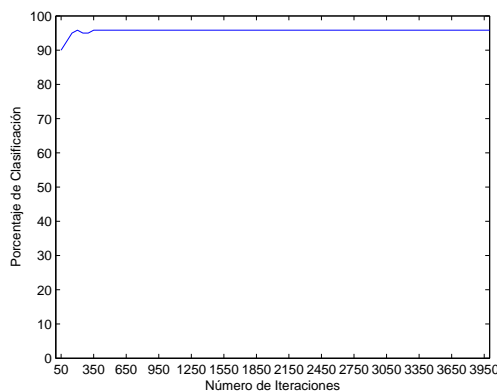
Tabla E.3: Porcentajes de Clasificación de la Red Neural Lateral para el Cluster de Forma 3 Modificando la Configuración de sus Capas Ocultas

Con respecto a la red neural asociada al cluster de Forma 1, se puede ver que todos los

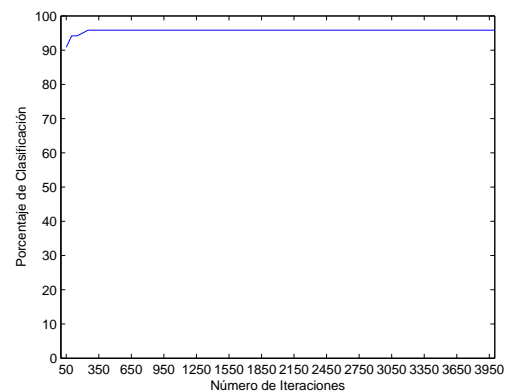
porcentajes de clasificación son bastante altos, apareciendo en un rango entre 92 % y 95 %, siendo el más alto la estructura de 1 capa de 10 neuronas oculta la cual será utilizada como estructura final para la red. Con respecto a la red neural asociada al cluster de Forma 3 podemos ver que el porcentaje de clasificación es siempre el mismo para las configuraciones de capas y neuronas evaluadas en el experimento. Debido al comportamiento observado en las redes neurales del cluster Forma 3 se decide utilizar la representación más simple, 1 capa de 5 neuronas, donde al tener menor cantidad de capas y neuronas el costo de utilizar la red neural sera considerablemente menor.

Una vez elegida la mejor estructura para las dos redes, con respectos a capas y neuronas ocultas, se procedió a evaluar en cuál iteración, tasa de aprendizaje y normalización se obtiene el mayor porcentaje de clasificación y de esta manera elegir la configuración final de la red.

A continuación se presentan gráficas, tanto para el cluster Forma 1 y Forma 3, que contrastan los porcentajes de clasificación sobre el conjunto de prueba bajo cada una de las normalizaciones por cada tasa de aprendizaje empleada. Las tasas de aprendizaje empleadas son: 0,01, 0,03, 0,05, 0,07, 0,1 y 0,2, y las normalizaciones empleadas son: por máximos y mínimos (para cluster Forma 1 ver Figuras E.1 y E.2, para cluster Forma 3 ver Figuras E.5 y E.6), y por varianza y media (para cluster Forma 1 ver Figuras E.3 y E.4, para cluster Forma 3 ver Figuras E.7 y E.8).

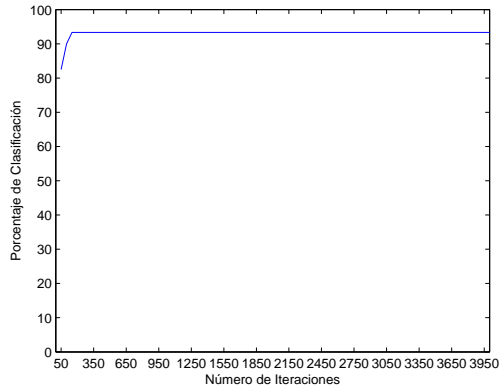


(a) Tasa de Aprendizaje: 0,01

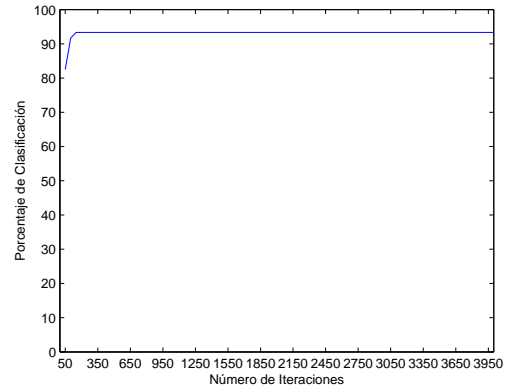


(b) Tasa de Aprendizaje: 0,03

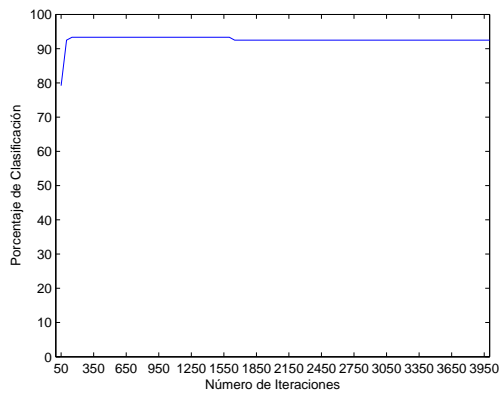
Figura E.1: Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 1, bajo una Normalización por Mínimos y Máximos y Tasas de Aprendizaje 0,01 y 0,03



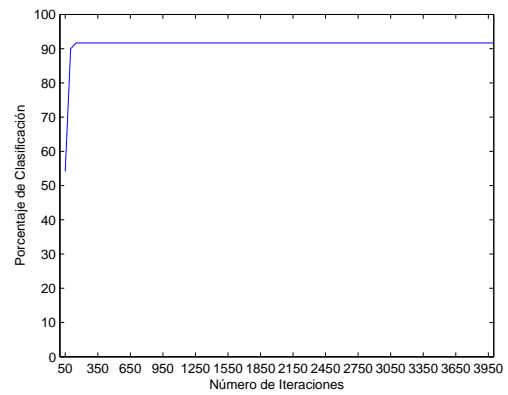
(a) Tasa de Aprendizaje: 0,05



(b) Tasa de Aprendizaje: 0,07

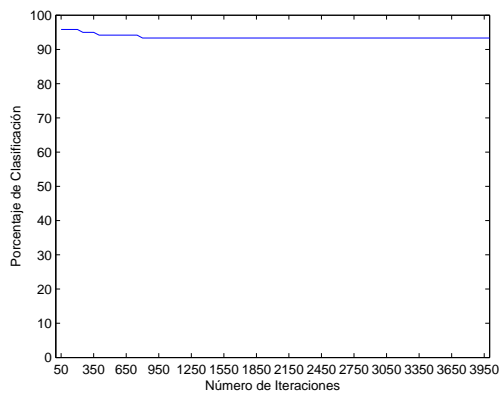


(c) Tasa de Aprendizaje: 0,1

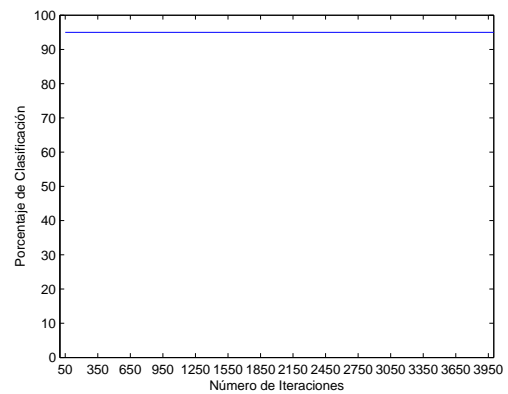


(d) Tasa de Aprendizaje: 0,2

Figura E.2: Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 1, bajo una Normalización por Mínimos y Máximos y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2

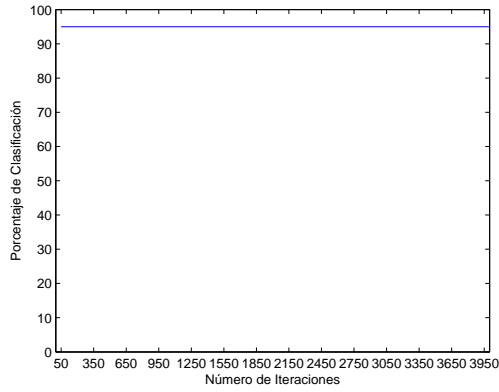


(a) Tasa de Aprendizaje: 0,01

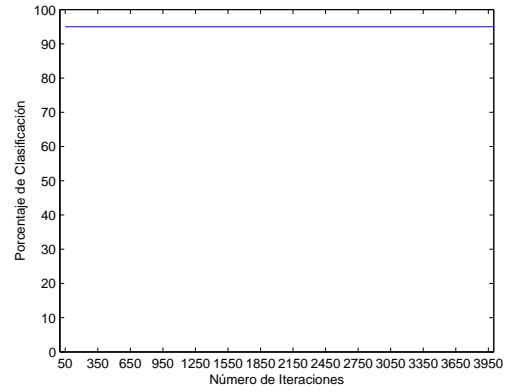


(b) Tasa de Aprendizaje: 0,03

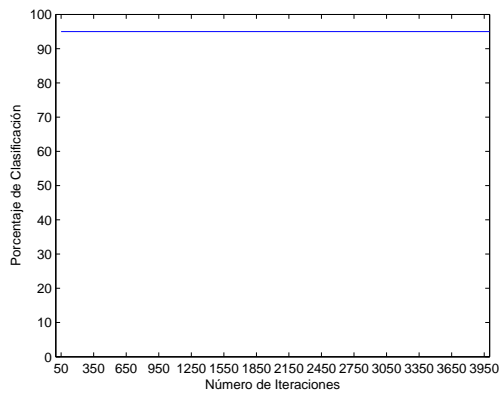
Figura E.3: Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 1, bajo una Normalización por Varianza y Media y Tasas de Aprendizaje 0,01 y 0,03



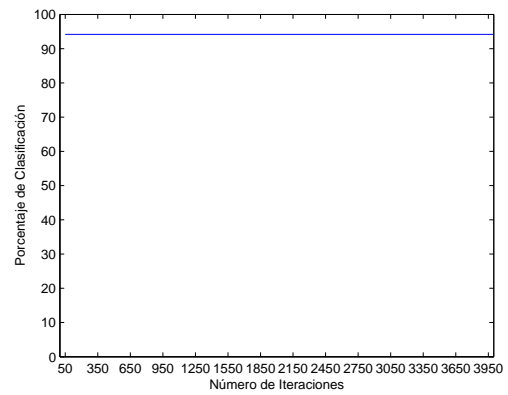
(a) Tasa de Aprendizaje: 0,05



(b) Tasa de Aprendizaje: 0,07

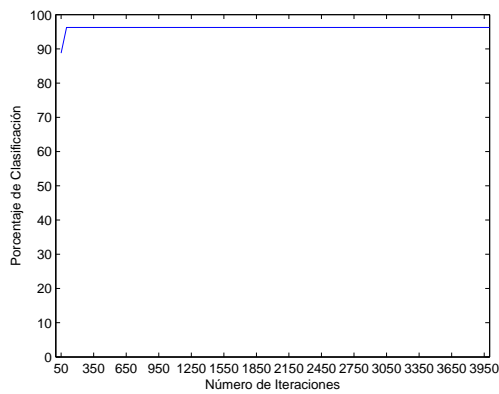


(c) Tasa de Aprendizaje: 0,1

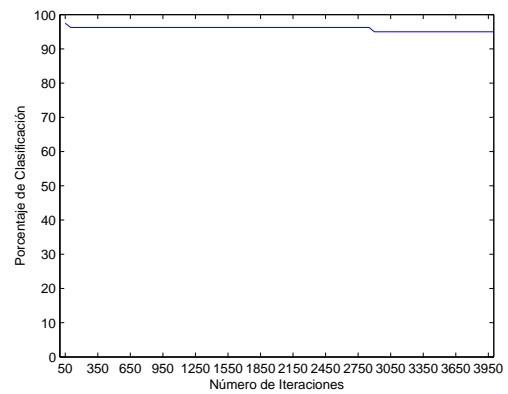


(d) Tasa de Aprendizaje: 0,2

Figura E.4: Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 1, bajo una Normalización por Varianza y Media y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2

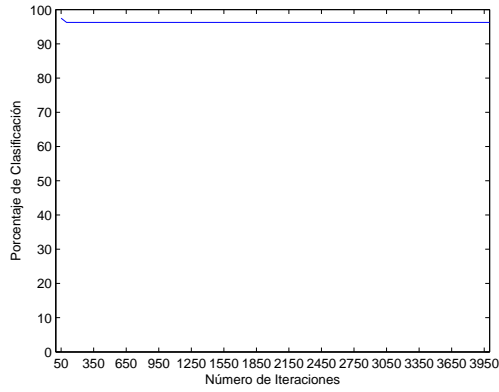


(a) Tasa de Aprendizaje: 0,01

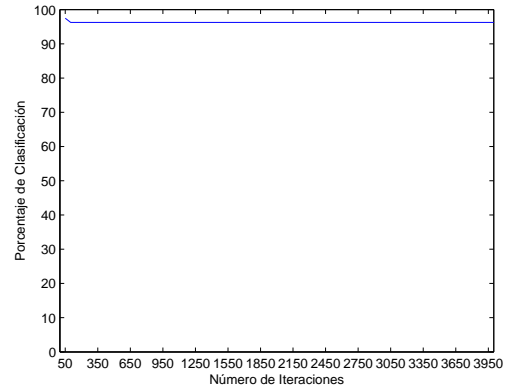


(b) Tasa de Aprendizaje: 0,03

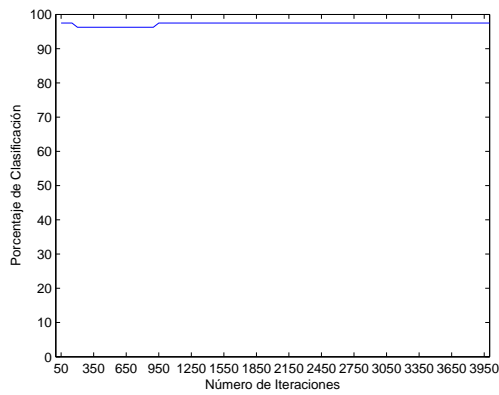
Figura E.5: Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 3, bajo una Normalización por Mínimos y Máximos y Tasas de Aprendizaje 0,01 y 0,03



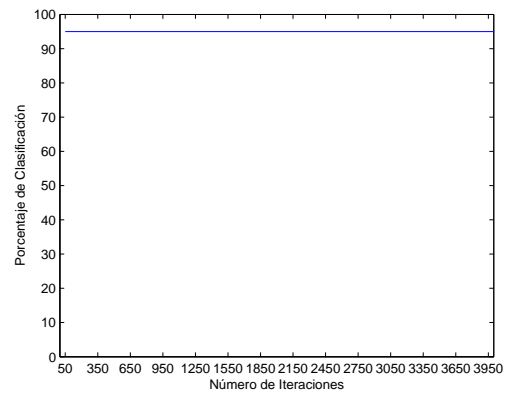
(a) Tasa de Aprendizaje: 0,05



(b) Tasa de Aprendizaje: 0,07

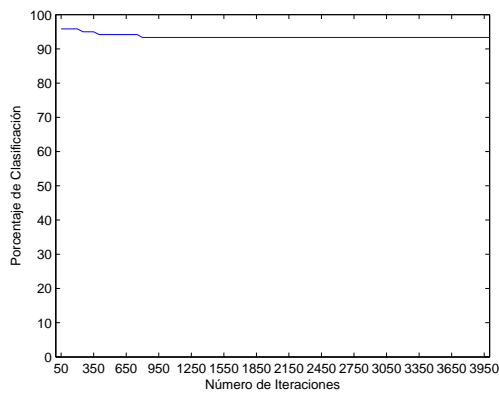


(c) Tasa de Aprendizaje: 0,1

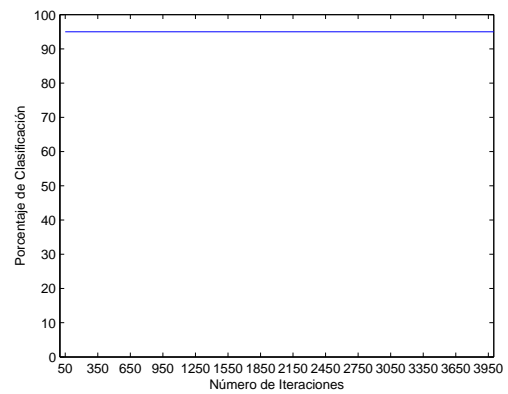


(d) Tasa de Aprendizaje: 0,2

Figura E.6: Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 3, bajo una Normalización por Mínimos y Máximos y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2

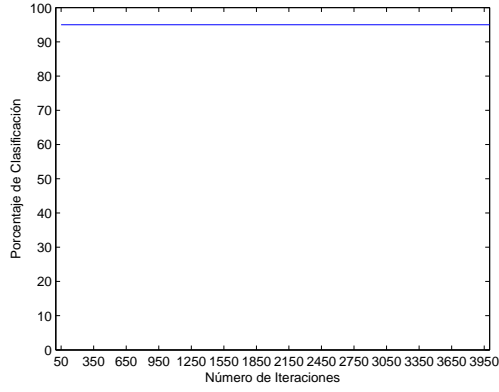


(a) Tasa de Aprendizaje: 0,01

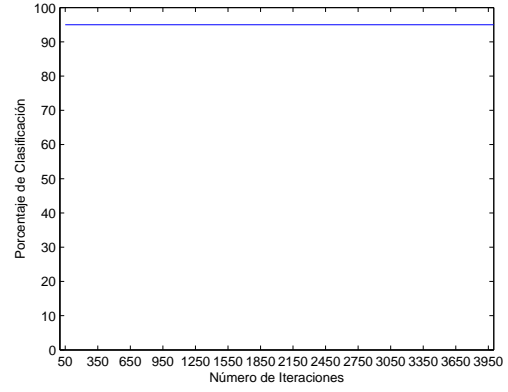


(b) Tasa de Aprendizaje: 0,03

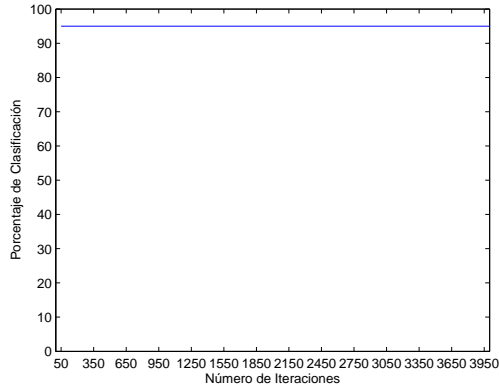
Figura E.7: Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 3, bajo una Normalización por Varianza y Media y Tasas de Aprendizaje 0,01 y 0,03



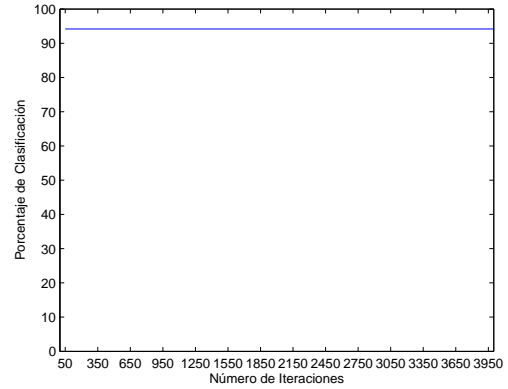
(a) Tasa de Aprendizaje: 0,05



(b) Tasa de Aprendizaje: 0,07



(c) Tasa de Aprendizaje: 0,1



(d) Tasa de Aprendizaje: 0,2

Figura E.8: Gráfica de Porcentajes de Clasificación de la Red Neural de Cluster Forma 3, bajo una Normalización por Varianza y Media y Tasas de Aprendizaje 0,05, 0,07, 0,1 y 0,2

A partir de los resultados representados en las gráficas anteriores, en la Tabla E.4 se presenta los porcentajes de mayor valor bajo cada una de las combinaciones mencionadas con respecto a la red neural asociada al cluster de Forma 1, mientras que en la Tabla E.5 se muestran los mismos resultados con respecto a la red neural asociada al cluster de Forma 3.

Tasa de Aprendizaje	Mínimos y Máximos		Varianza y Media	
	Prueba		Prueba	
	Max %	Iteración	Max %	Iteración
0,01	95,83	200	95,83	50
0,03	95,83	250	95	50
0,05	93,33	150	95	50
0,07	93,33	150	95	50
0,1	93,33	150	95	50
0,2	91,67	150	94,17	50

Tabla E.4: Porcentajes e Iteraciones Correspondientes a los Mejores Resultados por Configuración para Red Neural del Cluster Forma 1

Tasa de Aprendizaje	Mínimos y Máximos		Varianza y Media	
	Prueba		Prueba	
	Max %	Iteración	Max %	Iteración
0,01	96,25	100	97,5	100
0,03	97,5	50	97,5	50
0,05	97,5	50	97,5	50
0,07	97,5	50	97,5	50
0,1	97,5	50	97,5	50
0,2	95	50	97,5	50

Tabla E.5: Porcentajes e Iteraciones Corredpondientes a los Mejores Resultados por Configuración para Red Neural del Cluster Forma 3

Para la red neural asociada al cluster de Forma 1 se puede ver que ambas normalizaciones alcanzan el mismo porcentaje máximo de clasificación 95,83 %, sin embargo el comportamiento de ese porcentaje de clasificación, para las iteraciones evaluadas, es más estable utilizando la Normalización de Mínimos y Máximos (Ver Figuras E.1 E.2, E.3 y E.4), por lo tanto el tipo de normalización elegido para esta red es la Normalización de Mínimos y Máximos. Para la tasa de aprendizaje se elige el valor de 0,03 el cual junto a 0,01 fueron los que obtuvieron el porcentaje de clasificación mas alto 95,83 % con la normalización elegida.

Con respecto a la red neural asociada al cluster de Forma 3 se puede notar que ambas normalizaciones, al igual que en el caso anterior, alcanzan el mismo porcentaje máximo de clasificación 97,5 %, sin embargo en este caso es la Normalización de Varianza y Media la que posee un comportamiento más estable con respecto al porcentaje de clasificación máximo a través de las iteraciones evaluadas (Ver Figuras E.5, E.6, E.7 y E.8). Al analizar las tasas de aprendizaje para la normalización elegida se puede ver que el comportamiento es exactamente el mismo en todos menos con 0.01 el cual tiene un inicio con un porcentaje menor. Dado que el comportamiento es igual para todas las tasas de aprendizaje se utiliza el valor de 0.03 como en la red neural anterior.

A continuación se muestra en la Tabla E.6 un resumen con la configuración final para la red neural encargada de definir el valor del atributo Lateral sobre el cluster de Forma 1 y la red neural encargada de definir el valor del atributo Lateral sobre el cluster de Forma 3.

Red Neural	# Capas Ocultas	# Neuronas Ocultas	Normalización	Tasa de Aprendizaje
Cluster Forma 1	1	10	Mínimos y Máximos	0,03
Cluster Forma 3	1	5	Varianza y Media	0,03

Tabla E.6: Configuración final de las redes neurales encargadas de definir el atributo Lateral sobre cada uno de los clusters correspondientes a Forma 1 y Forma 3

Apéndice F

Tablas de Traducciones de Videos

Palabra Objetivo	Traducción	Frames Procesados por Segundo (fps)	
		Con Pre-Procesamiento	Sin Pre-Procesamiento
ACAMPAR	ACAMPAR	4.29	12.98
AQUI	AQUI	5.05	18.53
BULTO	BULXO	4.67	12.70
CAMBIO	CAMBIO	4.65	12.48
CAMBUR	CAMBRR	4.39	12.49
CARRO	CARRC	5.04	13.65
COCODRILO	COCODRILO	4.34	11.46
DADO	DADO	5.06	13.73
FAMILIA	FAMILA	4.45	12.39
GEMELO	GEMELCO	6.03	15.67
HABIAN	HABIA	4.53	11.96
HIPOPOTAMO	HBIPOPOXAS	4.87	12.47
IDEA	IDEA	4.53	12.34
JARABE	JAUABE	5.12	14.50
KILO	KILO	4.61	12.68
KILOMETRO	KILOS MERC	4.55	11.26
LIBRO	LIBRO	4.53	12.48
LLANO	LLAMO	4.60	13.47
MALO	MSALO	4.78	12.01
MOCHILERO	MOCHILEUO	4.19	11.43
NIÑO	NNINO	4.97	16.69
PELO	PELO	4.85	13.23
PERIODICO	PERIODICO	4.35	12.18
PROPINA	PROPI	4.83	11.34
REVISTA	REVISTA	4.51	12.21
SOBRE	SOBRE	4.72	12.97
TIJERA	TIJOERA	4.55	13.07
TURPIAL	TURPIL	4.22	10.19
TURPIAL	TTURPIAL	4.62	11.85
VIVIR	VIVIR	3.98	10.79
VUELTA	VUELTS	4.68	12.19
WATERPOLO	WATERPOLO	4.94	13.11
WATERPOLO	WATEERBPOLO	4.46	11.21
XILOFONO	XILOFONO	4.53	12.39
YEMA	YEMA	4.67	13.82
YEYUNO	YYEYUC	4.81	12.46
ZAPATA	ZAPAA	4.58	12.75

Tabla F.1: Resultados del Proceso Base de Traducción sobre 37 videos

Palabra Objetivo	Traducción	Frames Procesados por Segundo (fps)	
		Con Pre-Procesamiento	Sin Pre-Procesamiento
ACAMPAR	ACAMPAR	4.39	17.79
AQUI	AQUI	4.87	20.49
BULTO	BULXO	4.61	15.23
CAMBIO	CAMBIO	4.63	14.45
CAMBUR	CAMBRR	4.29	15.12
CARRO	CARRC	4.82	14.96
COCODRILO	COCODRILO	4.31	13.31
DADO	DADO	4.88	15.75
FAMILIA	FAMMILA	4.31	14.00
GEMELO	GEMELCO	5.98	18.71
HABIAN	HABIA	4.38	13.34
HIPOPOTAMO	HIPOPOXAS	4.85	15.09
IDEA	IDEA	4.37	13.54
JARABE	JARABE	4.93	15.75
KILO	KILO	4.81	15.25
KILOMETRO	KILOERC	4.58	13.91
LIBRO	LIBRO	4.37	13.63
LLANO	LLAMO	4.43	15.15
MALO	MSLO	4.72	14.58
MOCHILERO	MOCHILEUO	4.19	14.04
NINO	NNINO	4.82	18.90
PELO	PELO	4.66	14.72
PERIODICO	PERIODICO	4.36	14.42
PROPINA	PROPI	4.91	13.88
REVISTA	REVISTA	4.33	13.69
SOBRE	SOBRE	4.53	14.81
TIJERA	TIERA	4.15	13.38
TURPIAL	TURPIL	4.28	12.63
TURPIAL	TURPIAL	4.87	14.45
VIVIR	VIVIR	4.21	12.73
VUELTA	VUELTS	4.64	14.38
WATERPOLO	WATERPOLO	4.72	14.79
WATERPOLO	WATERPOLO	4.38	13.56
XILOFONO	XILOFON	4.47	14.49
YEMA	YEMA	4.52	15.69
YEYUNO	YYEYUC	4.73	14.78
ZAPATA	ZAPAA	4.59	15.00

Tabla F.2: Resultados del Proceso Base de Traducción empleando Árbol de Decisión de Clusters sobre 37 videos

Palabra Objetivo	Traducción de Palabra Parcial	Traducción de Palabra Existente	Frames Procesados por Segundo (fps)	
			Con Pre-Procesamiento	Sin Pre-Procesamiento
ACAMPAR	ACAMPAR	ACAMPAR	4.82	17.15
AQUI	AQUI	AQUI	5.48	20.47
BULTO	BULO	BULO	5.30	15.27
CAMBIO	CAMBIO	CAMBIO	5.47	15.29
CAMBUR	CAMBR	CA	4.46	13.86
CARRO	CARR	CA	5.16	14.48
COCODRILO	COCODRILO	COCODRILO	5.57	15.55
DADO	DADO	DADO	5.24	14.86
FAMILIA	FAMIL		4.92	14.89
GEMELO	GEMELO	GEMELO	7.09	30.11
HABIAN	HABIA	HABIA	5.08	14.32
HIPOPOTAMO	HIPOPOTA	HIPO	5.04	14.57
IDEA	IDEA	IDEA	4.59	12.71
JARABE	JARABE	JARABE	5.25	25.55
KILO	KILO	KILO	5.22	15.01
KILOMETRO	KILOME	KILO	5.26	14.98
LIBRO	LIBRO	LIBRO	5.82	16.44
LLANO	LLAM		4.71	14.62
MALO	MALO	MALO	4.99	13.49
MOCHILERO	MOCHILE	MOCHIL	4.70	14.17
NIÑO	NIÑO	NIÑO	5.88	31.29
PELO	PELO	PELO	4.91	13.57
PERIODICO	PERIODICO	PERIODICO	4.77	14.42
PROPINA	PROPI	PRO	4.97	12.96
REVISTA	REVISTA	REVISTA	4.96	14.22
SOBRE	SOBRE	SOBRE	4.87	14.12
TIJERA	TIJERA	TIJERA	4.97	28.02
TURPIAL	TURPIAL	TURPIAL	5.35	14.40
TURPIAL	TURPI	TU	4.95	13.40
VIVIR	VIVIR	VIVIR	5.47	15.60
VUELTA	VUEL		4.98	13.97
WATERPOLO	WATERPOLO	WATERPOLO	5.56	16.04
WATERPOLO	WATERPOLO	WATERPOLO	5.19	15.29
XILOFONO	XILOFONO	XILOFONO	5.15	15.20
YEMA	YEMA	YEMA	6.04	18.53
YEYUNO	YEYU		5.17	14.69
ZAPATA	ZAPATA	ZAPATA	4.89	29.19

Tabla F.3: Resultados del Proceso Base de Traducción empleando Diccionario Trie sobre 37 videos

Palabra Objetivo	Traducción de Palabra Parcial	Traducción de Palabra Existente	Frames Procesados por Segundo (fps)	
			Con Pre-Procesamiento	Sin Pre-Procesamiento
ACAMPAR	ACAMPAR	ACAMPAR	4.76	20.24
AQUI	AQUI	AQUI	5.42	23.49
BULTO	BULO	BULO	5.22	18.90
CAMBIO	CAMBIO	CAMBIO	5.41	18.54
CAMBUR	CAMBR	CA	4.43	17.41
CARRO	CARR	CA	5.06	16.54
COCODRILO	COCODRILO	COCODRILO	5.56	17.27
DADO	DADO	DADO	5.31	17.13
FAMILIA	FAMIL		4.89	18.47
GEMELO	GEMELO	GEMELO	6.99	21.10
HABIAN	HABIA	HABIA	5.04	16.34
HIPOPOTAMO	HIPOPOTA	HIPO	5.01	15.16
IDEA	IDEA	IDEA	4.55	13.20
JARABE	JARABE	JARABE	5.12	16.50
KILO	KILO	KILO	5.21	16.20
KILOMETRO	KILOME	KILO	5.26	16.71
LIBRO	LIBRO	LIBRO	5.81	17.65
LLANO	LLAM		4.73	16.18
MALO	MALO	MALO	5.00	15.16
MOCHILERO	MOCHILE	MOCHIL	4.71	16.39
NINO	NINO	NINO	5.81	21.12
PELO	PELO	PELO	4.84	15.16
PERIODICO	PERIODICO	PERIODICO	4.70	15.67
PROPINA	PROPI	PRO	4.93	14.49
REVISTA	REVISTA	REVISTA	4.96	16.39
SOBRE	SOBRE	SOBRE	4.67	14.66
TIJERA	TIJERA	TIJERA	4.92	17.21
TURPIAL	TURPI	TU	4.94	15.18
TURPIAL	TURPIAL	TURPIAL	5.46	16.22
VIVIR	VIVIR	VIVIR	5.35	16.20
VUELTA	VUEL		4.97	15.84
WATERPOLO	WATERPOLO	WATERPOLO	5.56	17.93
WATERPOLO	WATERPOLO	WATERPOLO	5.23	18.56
XILOFONO	XILOFONO	XILOFONO	4.96	15.22
YEMA	YEMA	YEMA	6.12	21.56
YEYUNO	YEYU		5.16	16.53
ZAPATA	ZAPATA	ZAPATA	4.77	16.80

Tabla F.4: Resultados del Proceso Base de Traducción empleando Árbol de Decisión de Clusters y Diccionario Trie sobre 37 videos