Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Integración de Redes Neuronales en Tareas de Reconocimiento de Escritura Manuscrita

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Puigcerver Ibáñez, Joan

Director: Gómez Adrián, Jon Ander

Curso 2015 - 2016

Agradecimientos

Quisiera manifestar mi agradecimiento a todas las personas que me han ayudado y animado durante la realización de este proyecto.

Quisiera expresar mi agradecimiento más sincero a todos los profesores que se preocupan por sus alumnos, que inspiran y resuelven todos los problemas que se les plantea. En especial quisiera agradecer todo el esfuerzo realizado a Jon Ander Gómez, director del proyecto, por toda la ayuda que me ha proporcionado y su gran vocación por la enseñanza.

También me gustaría dar las gracias a todos los compañeros y amigos que he tenido el placer de conocer y me han acompañado durante estos últimos años.

Por último pero no menos importante, quisiera expresar mi más sincero agradecimiento a mi familia por el apoyo incondicional y la ayuda recibida en todo momento, porque siempre están ahí cuando se les necesita.

A todos, de todo corazón, muchas gracias.

Resumen

Este proyecto consiste en adaptar la librería de redes neuronales desarrollada por el director en un sistema de reconocimiento de escritura manuscrita basada en modelos ocultos de Markov.

El objetivo principal es integrar una red neuronal con el objetivo de mejorar los resultados anteriores. La función de la red neuronal es estimar las probabilidades de emisión de los modelos de Markov, anteriormente modeladas con mixturas de gaussianas.

El proyecto incluye un análisis del problema y el diseño del sistema de reconocimiento. También se expone el desarrollo de los programas que implementan dicho sistema de reconocimiento.

Por último, contiene los detalles de los múltiples experimentos realizados para obtener resultados que permitan comparar la técnica anterior con la nueva.

Palabras clave: Aprendizaje Automático, Reconocimiento de Patrones, Reconocimiento de Escritura Manuscrita

Abstract

This project consists in adapt a neuronal network library developed by the director in a hand-written text recognition text based on Markov's hidden models

The main goal is adapt a neural network in order to improve the previoys results. The function of the neural network is providing the emission probabilities of the states in the Markov's model, previously obtained by a Gaussian mixture.

This project includes an analysis of the problem and the handwritten recognition system design. Also, the developments of the programs that implement this recognition system are exposed.

At the end, multiple experiments are performed in order to obtain results for comparing this technique with the previous one.

Keywords: Machine Learning, Pattern Recognition, Hand-written Recognition

Tabla de contenidos

1. Introducción

En esta sección se introduce el concepto de **reconocimiento de escritura manuscrita** y su influencia. También se exponen la motivación del alumno y los objetivos que se intentan alcanzar.

1.1 Reconocimiento de escritura manuscrita

El reconocimiento de escritura manuscrita es un campo dentro del reconocimiento de patrones, rama del aprendizaje automático. Se puede definir el reconocimiento de escritura manuscrita como la habilidad de un ordenador de recibir, aprender e interpretar escritura manuscrita de imágenes. Esta habilidad implica principalmente el reconocimiento de caracteres, aunque un sistema de reconocimiento completo contempla muchos más aspectos, como realizar la segmentación correcta de la imagen en caracteres y encontrar la palabras más adecuadas según su contexto.

Este tipo de sistemas tiene muchas aplicaciones, como pueden ser la lectura de formularios, cheques, direcciones o la digitalización de obras literarias.

Este proyecto se centra en aprender a reconocer carácter a carácter para a continuación realizar segmentaciones de la imagen lo más acertadas posibles.

1.2 Motivación

La elección del tema de este proyecto fue debido a que este sistema tiene muchas aplicaciones prácticas y además, estaba el reto que supone abordar un problema nada sencillo.

La motivación principal del proyecto, de carácter personal, es la introducción al mundo de la inteligencia artificial y el aprendizaje automático, así como conocer y hacer uso de las técnicas utilizadas para la resolución de este tipo de problemas.

Por otra parte, comprobar el potencial que ofrece la integración de la red neuronal en el sistema de reconocimiento y mejorar resultados ya existentes.

Por último, aprender y ampliar los conocimientos de programación en C++, un lenguaje de programación que ha pasado desapercibido durante el grado.

1.3 Objetivos

Los objetivos principales del proyecto son:

- **La integración de la red neuronal en el sistema,** que permita calcular las probabilidades de emisión de los modelos de Markov utilizados en el sistema.
- **Experimentar y realizar una comparación de los resultados** que se obtienen con la técnica anterior, que calcula las probabilidades de emisión utilizando mixturas de gaussianas y comprobar si la nueva técnica genera mejores resultados.
- **Proporcionar una herramienta configurable,** que permita una rápida y sencilla configuración del proceso de aprendizaje mediante ficheros de configuración.
- **Proporcionar ejecutables** que, mediante los archivos de configuración, sean capaces de entrenar modelos de Markov y redes neuronales de una manera general. En concreto, en este proyecto se utilizaran ambos modelos para construir un sistema de reconocimiento de texto manuscrito.

2. Estado del arte

En los últimos años, el número de bibliotecas digitales ha aumentado con lo que también ha aumentado el número de documentos digitalizados publicados en la red, pero aún existe una inmensa cantidad de libros que aún no han sido digitalizados. Disponer de documentos digitalizados aporta una gran serie de ventajas, que proporcionan nuevas maneras de indexar, consultar y filtrar estos documentos.

El proceso de digitalización de los documentos es muy laborioso y es llevado a cabo por expertos en la materia, cuya especialidad es la lectura de textos antiguos de, estilos diferentes. diversos lugares y periodos de tiempo distintos.

El reconocimiento de texto manuscrito (HTR) es aplicable en muchos otros contextos, como la lectura de direcciones, códigos postales [1] o la lectura de formularios y cheques [2]. Estos tipos de problemas son más limitados, por lo que los sistemas actuales de reconocimiento logran obtener unos resultados relativamente altos. En el contexto de la digitalización de documentos, el HTR se ha convertido en un tema de investigación relevante. [3][4]

El objetivo de este proceso es obtener la secuencia de caracteres presentes en una imagen, correspondiente a una línea de texto manuscrito. Para obtener unas imágenes correctas para el entrenamiento y predicción se necesita un preproceso que implica detectar las líneas y regularizar las imágenes [5]. Si los resultados de las transcripciones obtenidas son lo suficientemente bajos pueden ayudar a acelerar el proceso de digitalización de documentos. Sin embargo, en la práctica los resultados de este proceso no son lo suficientemente altos y necesitan de intervención humana para conseguir unas buenas transcripciones. [6]

El paradigma del reconocimiento de formas intenta recrear la manera en que los seres humanos perciben las regularidades y patrones en los objetos [7]. Concretamente en el problema del HTR, el reconocer los trazos característicos de cada símbolo.

En el estado del arte actual los modelos de Markov (HMM) son utilizados para modelar los trazos de los caracteres y se utilizan n-gramas para modelar las secuencias de caracteres [5]. Los resultados obtenidos por estos modelos se intentarán mejorar introduciendo redes neuronales artificiales (ANN), un modelo inspirado en la red neuronal biológica.

Una red neuronal es un modelo que proporciona una alta capacidad de aprendizaje y se integrará en el sistema de reconocimiento de texto manuscrito basado en HMM.

Las redes neuronales pueden ser usadas para abordar el problema del HTR. Tras observar los suficientes caracteres, las redes son capaces de predecir a que carácter pertenece un fragmento de la imagen.

3. Análisis

En los siguientes apartados se expone el análisis realizado para los diferentes aspectos de este proyecto, así como la evolución del diseño planteado para abordar la resolución del problema de una manera eficiente.

3.1 Análisis del conjunto de datos

El conjunto de datos del cual se dispone pertenece al libro "Historia de España del arçobispo Don Rodrigo". Este manuscrito de 853 páginas que data del año 1545, ha sido dividido en bloques y líneas, obteniendo así 20.000 fragmentos aproximadamente con sus respectivas transcripciones. En estos fragmentos, se pueden distinguir hasta 106 caracteres y símbolos distintos.

A partir de estos datos, se define el conjunto de entrenamiento con 10.000 líneas y el conjunto de validación con 5.010 imágenes.

Cada línea corresponde a un archivo de imagen en color (formato PNG), con una altura en píxeles de 50 y una anchura variable para cada imagen. Para simplificar el proceso de carga de los datos, las imágenes PNG fueron transformadas en imágenes en escala de grises (formato PGM), que además, se pueden leer fácilmente como un archivo de texto plano.

3.2 Análisis del problema

El problema que se aborda consiste en reconocer con la máxima precisión posible la secuencia de caracteres que hay escritos en una línea de texto.

El sistema de reconocimiento de escritura manuscrita está basado en modelos ocultos de Markov (hidden Markov models o HMM), que permiten modelar los símbolos presentes en la imagen y las relaciones entre ellos.

Un modelo oculto de Markov es un modelo estadístico cuyo objetivo es determinar los parámetros desconocidos (estados ocultos *x* y probabilidades de transición *a*) a partir de datos observables (*y*).

Nuestro objetivo es entrenar una red neuronal artificial (artificial neural network o ANN) que sea capaz de estimar las probabilidades de emisión *b* de los estados ocultos, anteriormente obtenidas a partir de mixturas de gaussianas.

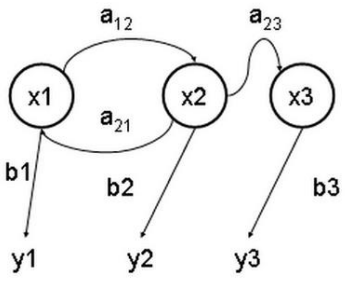

Ilustración 1: HMM

Una red neuronal artificial o ANN es un modelo de aprendizaje inspirado en el sistema nervioso biológico. Una red neuronal está compuesta por neuronas, que reciben una entrada y emiten una salida. Dichas neuronas están organizadas en capas, donde las neuronas de una capa en concreto, reciben la salida de la capa anterior y emiten una salida que será recibida por la capa posterior.

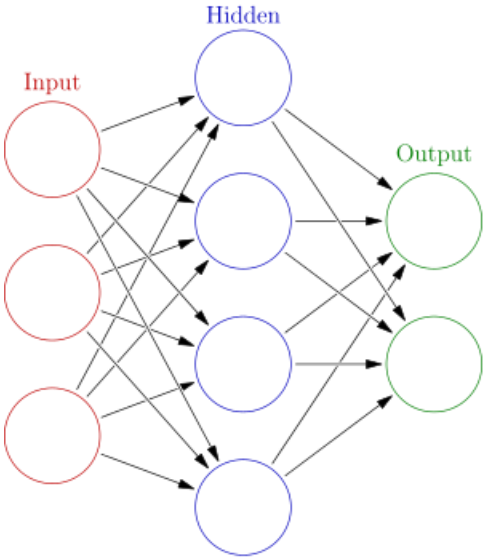

Ilustración 2: ANN

La estructura o topología debe ser cuidadosamente estudiada para abordar de una manera eficaz el problema. Este aspecto especifica el número de capas que componen la ANN, así como las neuronas presentes en cada capa, sus interconexiones, su función de activación y el proceso de aprendizaje.

Un parámetro importante del proceso de aprendizaje es el *learning rate*, que determina el impacto del error de cada muestra procesada en los parámetros. El valor *learning rate* que se utilizará es de 0.01.

Para evitar el sobre-entrenamiento de la red se utilizará la técnica denominada *Dropout* [8], que desactiva de manera aleatoria neuronas durante el entrenamiento. Esta técnica evita que las unidades de la red se adapten entre ellas. El número de neuronas desactivadas se determina por un parámetro entre 0 y 1, que indica la proporción de neuronas que quedarán activas.

Con el objetivo de evitar el sobreentrenamiento de la ANN se añadirá se modificaran ligeramente las muestras añadiendo ruido.

El cometido de la función de activación es convertir la entrada ponderada a la salida emitida. Entre las funciones de activación más comunes podemos encontrar las siguientes:

- **Lineal:** $f(x) = x$
- ***Linear-rectified*:** $f(x) = max(0, x)$
- **Sigmoide:** $f(x) = \frac{1}{1+e^{-x}}$
- **Tangente hiperbólica:** $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- **Softmax:** $f(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}$

Para poder disponer de unos HMM válidos para el reconocimiento de escritura manuscrita, primero se debe estudiar la manera de modelar las secuencias de caracteres y, posteriormente, la manera de poder estimar los parámetros del HMM que permiten calcular la probabilidad de cada secuencia en la imagen.

3.2.1 Análisis del modelo

El primer objetivo es modelar la secuencia de caracteres con un HMM. Para este fin, se debe determinar cómo modelar un solo carácter y concatenar este modelo para obtener la secuencia.

Se decidió modelar un carácter como un HMM de 5 estados de la siguiente manera:

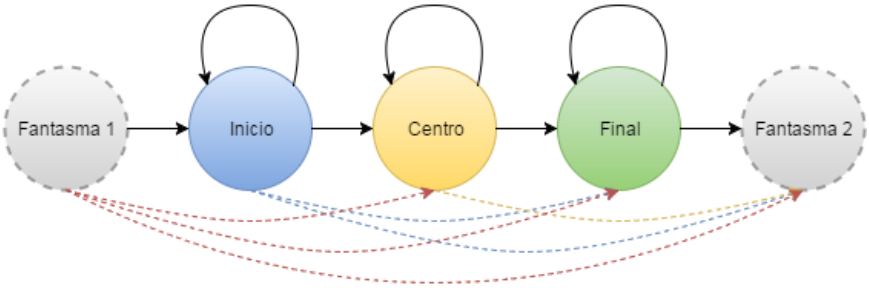

Ilustración 3: HMM carácter

Se distinguen dos tipos de estados diferentes:

- **Estados propios del modelo:** Corresponden al segundo, tercer y cuarto estado del modelo. Tienen la función de especializarse en las características del carácter al que representan, centrándose en la parte inicial, central y final respectivamente.
- **Estados fantasma:** Corresponden al primer y último estado del modelo y no contienen ninguna información sobre el carácter. El primer estado fantasma tiene la función de referenciar al estado final del modelo anterior, mientras que el segundo estado fantasma referencia al estado inicial del siguiente modelo.

En cuanto a las transiciones, el modelo es unidireccional ya que nunca se va a poder visitar un estado que haya sido visto anteriormente. Un estado puede volver a sí mismo, o avanzar al siguiente estado de la secuencia.

Solo se dispondrá de un solo HMM por cada carácter, con identificadores únicos para cada uno de los estados, cuyas probabilidades de emisión serán estimadas por una ANN.

Una vez representados los caracteres, se pueden formar las secuencias de la siguiente manera:

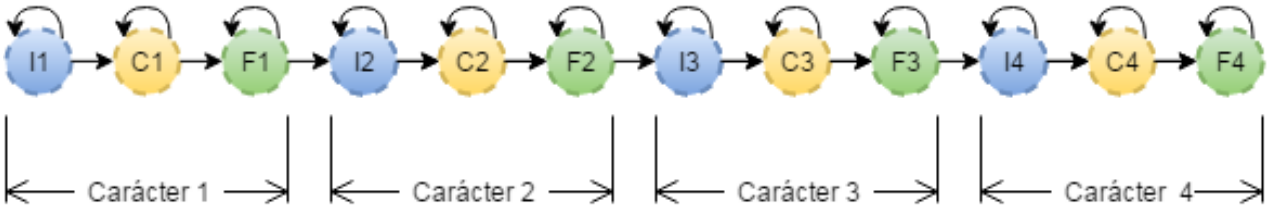

Ilustración 4: HMM secuencia caracteres

Gracias a los estados fantasmas, se puede modelar la secuencia con facilidad y estimar las probabilidades de cambiar de un modelo a otro.

En relación a la topología de la ANN, se decidió utilizar la siguiente estructura:

- **Capa de entrada:** Una capa cuya dimensionalidad debe coincidir con la dimensión de las muestras que se van a tratar. La función de activación de esta capa es lineal.
- **Capas ocultas:** Tres capas ocultas con *D* nodos con la función de activación *linear-rectified*. La cantidad de neuronas dependerá de la dimensionalidad de la capa de entrada.
- **Capa de salida:** Una capa cuya dimensionalidad debe coincidir con el número total de estados presentes en los HMM (por norma general, 3 veces el número de caracteres

distintos). Esta capa tiene la función de activación *soft-max*, que emite como salida valores entre 0 y 1 para cada nodo donde la suma de todos los valores emitidos es 1. Estos valores se usarán para estimar las probabilidades de emisión en los HMM.

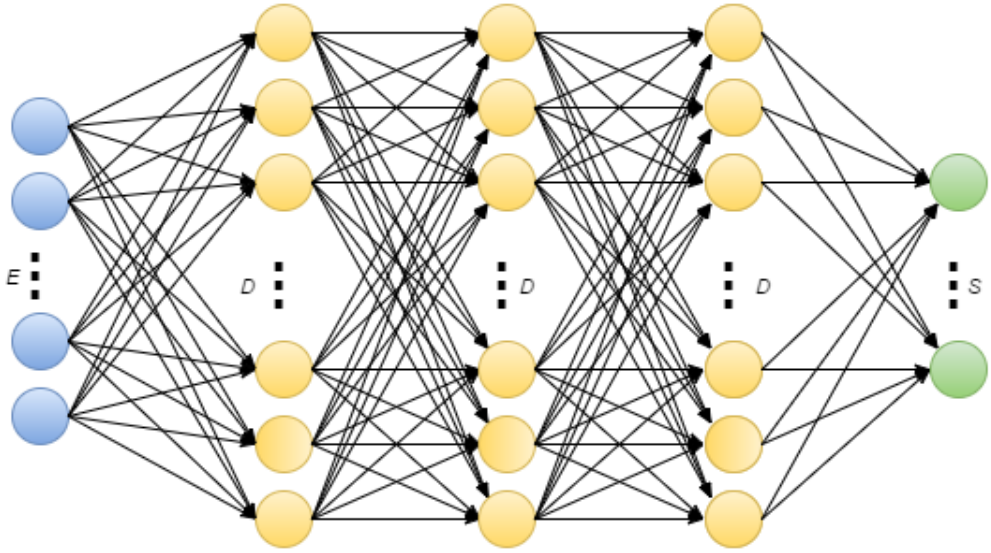

Ilustración 5: Estructura ANN

3.2.2 Análisis de la extracción de características

El entrenamiento de los HMM asociados a cada carácter y la ANN se realizará a partir de los datos de entrenamiento que se disponen.

Para extraer las características de cada uno de los caracteres en las imágenes se utilizará una técnica que consiste en dividir la imagen en fragmentos más pequeños. Todas estas sub-imágenes denominadas marcos o ventanas tienen la misma anchura y se van obteniendo desplazando el marco una distancia fija.

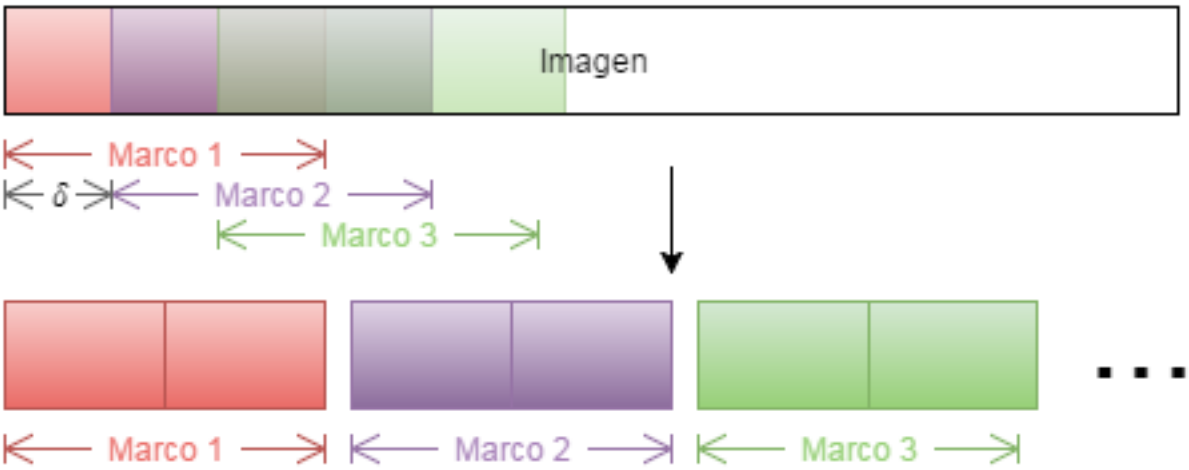

Ilustración 6: Extracción marcos

Para obtener unos buenos datos de entrenamiento, es recomendable que exista un solapamiento entre los marcos para intentar captar las relaciones que existen entre las diferentes unidades. La anchura mínima recomendada es de 5 píxeles, donde el marco se centra en un pixel y recoge el contexto de 2 pixeles por cada lado.

La dimensión de estos marcos depende de la altura de la imagen (50 pixeles) y la anchura de marco fijada. Para la anchura mínima fijada, el tamaño de cada marco sería de 250 valores, que podría aumentar si se decide aumentar el tamaño del marco.

Trabajar con muchas dimensiones es contraproducente en dos sentidos:

- **Ralentiza los cálculos**, ya que se realizan muchísimos más cálculos porque aumenta el número de parámetros que se deben aprender.
- **Maldición de la dimensionalidad**. Los datos posiblemente contienen información irrelevante que perjudica el entrenamiento, ya que se necesitan muchísimos datos para asegurarse que existen múltiples muestras con cada una de las diferentes combinaciones de valores.

Para paliar estos dos efectos, se decidió reducir la dimensión de las muestras utilizando una red neuronal. Este preproceso de los datos agilizará la fase de entrenamiento y concentrará la información relevante de los marcos en pocas dimensiones.

La red usada para reducir las dimensiones se denomina **autoencoder.** La topología de esta ANN corresponde a:

- **Capa de entrada:** Una capa cuyo tamaño es la dimensión inicial de las muestras que se quieren procesar.
- **Capas ocultas:** Capas ocultas, que van dirigiendo la reducción de dimensionalidad hasta la dimensión deseada.
- **Capa de salida:** Esta capa tiene la dimensión a la cual se quiere reducir las muestras procesadas. De esta capa se obtiene una codificación equivalente a la muestra, pero con la información más comprimida.

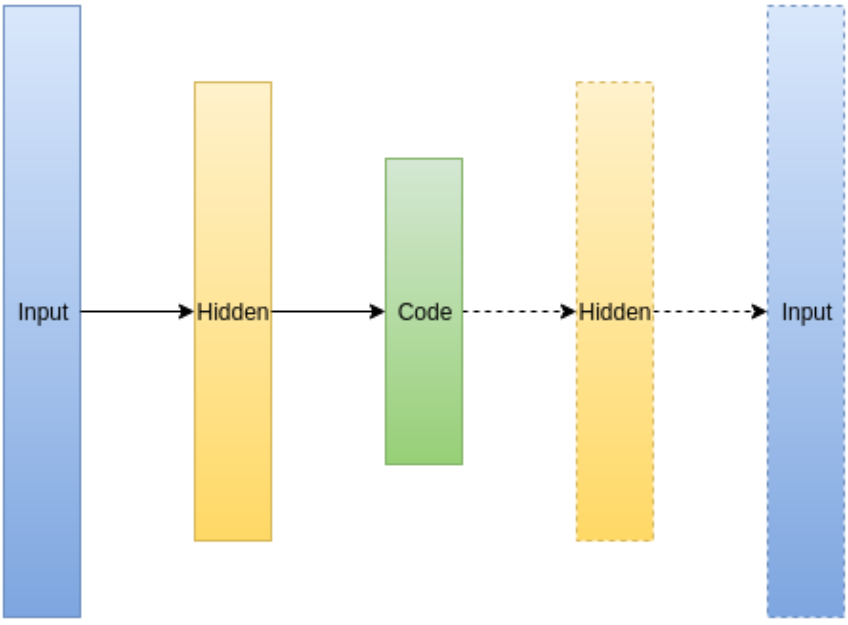

Ilustración 7: Estructura *autoencoder*

El entrenamiento de esta ANN solo requiere las muestras que se quieren tratar. El proceso de aprendizaje consiste en codificar y decodificar cada muestra. Cada una de las decodificaciones

es comparada con la muestra inicial y se actualizan los parámetros para que las decodificaciones se vuelvan más precisas.

Una vez entrenado el *autoencoder*, se eliminan las capas duplicadas en la fase de entrenamiento y se puede usar para codificar las muestras en datos más compactos.

3.2.3 Análisis de la fase de entrenamiento

Definimos X como el conjunto de marcos y S como el conjunto de estados.

El objetivo principal es entrenar los HMM y la ANN.

Para entrenar la ANN, se necesitan pares de muestra-etiqueta (marco – estado al que pertenece), que a priori no se conocen, por lo tanto, la primera fase del entrenamiento será obtener dichos pares.

Las redes neuronales son muy sensibles en los primeros instantes del aprendizaje, pero una vez entrenadas, es muy difícil reajustar los parámetros aprendidos. Es aconsejable entrenar la red neuronal con los datos más precisos posibles para obtener unos buenos resultados.

Para conseguir estos buenos datos se decidió utilizar la técnica que se intenta mejorar en este proyecto. Se utilizará la segmentación que se obtiene al usar mixturas de gaussianas en los HMM y entrenar la ANN que permitirá mejorar dicha segmentación.

Podemos dividir el entrenamiento en 3 fases:

1. *Entrenamiento HMM con mixtura de gaussianas:*

La primera fase del entrenamiento se centrara en estimar los parámetros de los modelos de Markov que modelan los caracteres usando mixturas de gaussianas en los estados. Una vez entrenados los HMM, se podrá determinar los fragmentos de la imagen que corresponden a cada carácter. Con estos datos, se pueden obtener los pares muestra-etiqueta necesarios para el entrenamiento de la ANN.

Los HMM se entrenan con todos los marcos resultantes de cada imagen y su correspondiente transcripción. El algoritmo utilizado es el de Baum-Welch que, observando la secuencia de HMM que corresponde a cada una de las imágenes del conjunto de entrenamiento, permite estimar los parámetros ocultos de cada uno de los modelos asociados a cada carácter.

El entrenamiento de los HMM empieza asumiendo que los caracteres de la imagen están distribuidos uniformemente a lo largo de la imagen. A medida que progrese el aprendizaje con los datos de entrenamiento, se ajustarán las fronteras y se podrá generar la segmentación de la imagen.

2. *Entrenamiento de la red neuronal:*

Dada la segmentación generada por los HMM entrenados con gaussianas podemos asociar cada uno de los marcos a uno de los estados de los HMM y generar los pares muestra-etiqueta.

El algoritmo de entrenamiento de las ANN será el de Back Propagation. Este algoritmo consiste en procesar cada muestra a través de la ANN para obtener una salida. Esta salida se compara con la salida esperada (etiqueta) calculando el error. Dicho error se propaga desde la salida hasta

la entrada, actualizando los parámetros de la red neuronal para obtener resultados más semejantes a los esperados.

Una vez entrenada la ANN con todos los datos de entrenamiento, la red neuronal será capaz de proporcionar la probabilidad de que un marco X_t pertenezca a un estado, expresada como $P(S|X_t)$.

3. Refinamiento e incorporación de la ANN en los HMM:

La última fase del entrenamiento consiste en refinar los parámetros aprendidos en los HMM y la ANN incorporando la ANN entrenada en los HMM.

La función de la ANN dentro del modelo de Markov consiste en proporcionar las probabilidades de emisión de los estados. La probabilidad de emisión se puede expresar como $P(X_t|S_i)$, que no es la misma que proporciona la red neuronal $P(S|X_t)$.

Esta probabilidad puede ser obtenida mediante el teorema de Bayes, que relaciona estas dos probabilidades tal que:

$$P(X_t|S_i) = \frac{P(S_i|X_t) * P(X_t)}{P(S_i)}$$

Asumiendo que todos los marcos de las imágenes tienen la misma probabilidad de aparecer, la expresión anterior puede aproximarse a:

$$P(X_t|S_i) \approx \frac{P(S_i|X_t)}{P(S_i)}$$

La probabilidad $P(S_i)$ puede ser estimada por conteo dada la segmentación obtenida en la primera fase del entrenamiento como:

$$P(S_i) = \frac{Número\ de\ marcos\ asociados\ a\ S_i}{Número\ total\ de\ marcos}$$

Una vez calculadas estas probabilidades es posible utilizar la ANN para estimar las probabilidades de emisión de los HMM, reentrenar los HMM obtenidos en la primera fase y generar una nueva segmentación de la imagen.

Con esta nueva segmentación se puede reentrenar la ANN y refinar la estimación de las probabilidades de emisión. Este proceso se puede repetir tantas veces como se desee.

3.2.4 Diseño

El diseño del proceso de entrenamiento es el siguiente:

1.1 Conjunto de datos de entrenamiento original.
1.2 Preproceso de los datos (cambio de formato, reducción de dimensionalidad, …)
2.1 Creación de los HMM.
2.2 Modificación manual de los HMM (reducir estados de un carácter, modificar probabilidades de transición, …)
2.3 Distribución uniforme o *flat start.*

2.4 Entrenamiento de los HMM con gaussianas. En cada iteración se permite aumentar el número de gaussianas por mixtura.

3. Generar la segmentación de la imagen que permite generar los datos para entrenar la ANN y las probabilidades a priori de cada estado.

4. Entrenar la ANN.

5. Reentrenar los HMM con la ANN entrenada y las probabilidades a priori de los estados. Volver a paso 3.

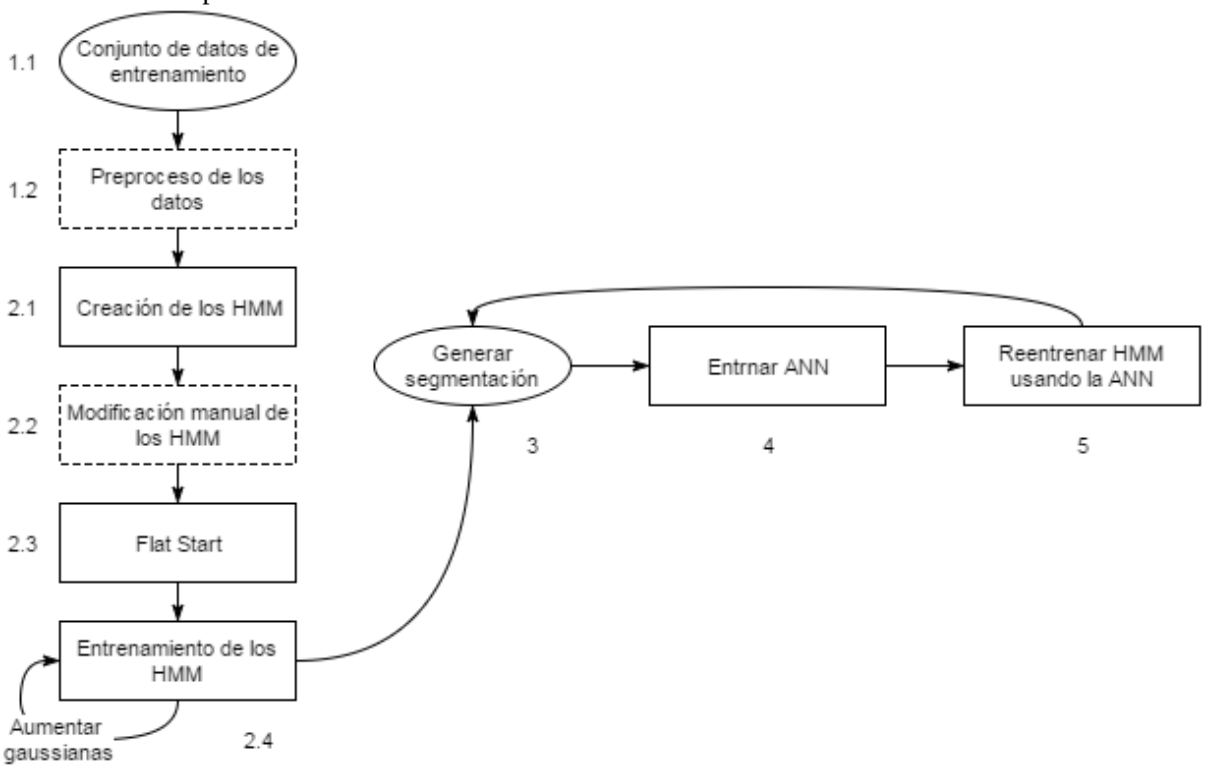

Ilustración 8: Diseño proceso aprendizaje

Para la implementación de este proceso de aprendizaje es necesario el desarrollo de distintas herramientas:

- **Librería de HMM y ANN**, proporcionada por el director de este proyecto, que permite crear, entrenar y salvar estos modelos.
- **Librería para la lectura y tratamiento de los datos**. Esta librería permite cargar los datos y organizarlos de una manera eficiente en estructuras de datos.
- **Librería de configuración**, cuya función es la de proporcionar la posibilidad de configurar el proceso y sus parámetros mediante ficheros de configuración.

Haciendo uso de estas herramientas es posible la implementación de dos programas con las siguientes funciones:

- **Entrenamiento HMM.** La función del primer programa es utilizar las librerías anteriores para inicializar, entrenar y finalmente, disponer de unos HMM válidos. Este programa soporta HMMs basados en mixturas de gaussianas y HMMs basados en ANNs.
- **Entrenamiento ANN.** Este programa permite el entrenamiento de una red neuronal a partir de los datos de entrenamiento y sus correspondientes etiquetas. Una vez entrenada, el programa permite guardar en el sistema los parámetros aprendidos.

4. Desarrollo

En esta sección se explica el desarrollo de los programas implementados para llevar a cabo el proceso de entrenamiento diseñado en el anterior apartado. Se muestran los detalles correspondientes a las librerías y programas implementados por el alumno.

4.1 Lectura y tratamiento de los datos.

Esta librería contiene todas las herramientas necesarias para leer los datos y almacenarlos de una manera estructurada y fácilmente accesibles.

4.1.1 Lectura de datos:

Todas las muestras se encuentran en un directorio específico, donde cada imagen del corpus se representa como un archivo distinto.

La información asociada a las muestras se ubica en dos ficheros de texto. El primero contiene una lista de todos los símbolos presentes en el corpus utilizado y el segundo contiene la lista de las transcripciones de cada uno de los archivos.

Los símbolos se representan como una secuencia de caracteres (*String)* y son almacenados en una lista. Las transcripciones son representadas por un vector de símbolos y se guardan en un *Hashmap*, que permite acceder al vector de símbolos correspondiente proporcionando el nombre del fichero.

Las muestras se representan como vectores de números reales y se almacenan en un *Hashmap*. Este *Hashmap* no almacena directamente el vector, sino que tiene la referencia a un objeto que contiene información adicional a cada imagen.

4.1.2 Tratamiento de los datos

Las imágenes cargadas están representadas por un vector de reales, correspondientes a los valores de cada píxel. Esta información se almacena en un objeto denominado *Sample* o *Muestra* y es el encargado de realizar la extracción de los marcos de cada imagen.

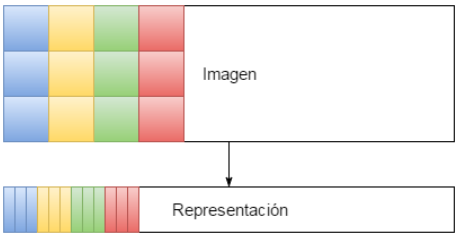

Ilustración 9: Representación imagen

El número de marcos que se pueden extraer de la imagen depende de la anchura de la imagen (W*)*, la anchura del marco (M*)* y la distancia entre marcos (*δ)*. Se puede obtener mediante la siguiente fórmula:

$$N\,marcos = \frac{W - M}{\delta} + 1$$

Los marcos no se representarán como vectores de reales, ya que esta información es redundante y se encuentra en el vector de la imagen. En cambio, se representarán como un único valor, que hace referencia a la dirección de memoria correspondiente al primer píxel de cada marco.

De esta manera, podemos representar todos los marcos obtenidos de la imagen como un vector de punteros a la dirección de memoria del píxel donde empieza cada marco.

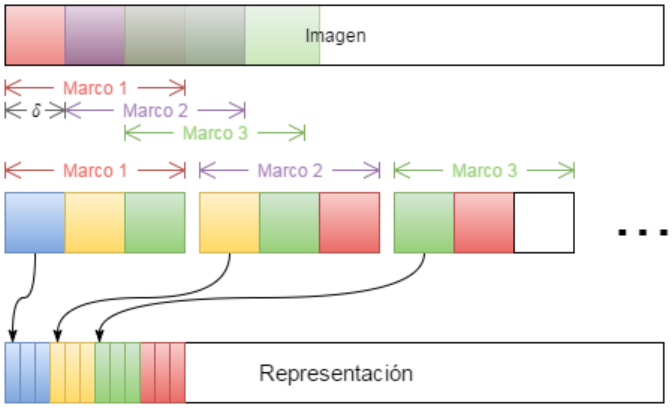

Ilustración 10: Representación marcos

4.2 Configuración

Esta librería proporciona la posibilidad de configurar todos los parámetros que se utilizarán en los programas implementados. Estas variables permiten adaptar los programas implementados al tipo de experimento que se desea realizar.

Los parámetros son los siguientes:

- **Archivo de configuración (parameters_cfg):** Se permite pasar un archivo de configuración, que se procesará antes. Esto permite aislar los parámetros comunes en un mismo fichero y no duplicar información en los distintos ficheros.
- **Ruta al fichero de símbolos (path_symbols):** Fichero que contiene los símbolos presentes en el corpus.
- **Ruta al fichero de las transcripciones (path_transcriptions):** Fichero donde se ubican las transcripciones de las imágenes.
- **Ruta al directorio de las muestras (path_samples):** Directorio donde están contenidas todas las muestras del corpus.
- **Extensión (extension):** Extensión de los archivos de las muestras. La utilidad de este parámetro es que permite usar el programa para trabajar con distintos tipos de archivos, por ejemplo, se podría trabajar con archivos PNG o PGM.
- **Número de muestras (n_samples):** Se permite elegir el número de muestras que se van a usar para el entrenamiento. Si se quiere usar todo el corpus, este valor debe ser -1.
- **Ruta *hhmdefs* (path_hmmdefs):** Esta es la ruta al directorio donde se guardarán todos los modelos de Markov entrenados.
- **MLF (mlf_filename):** Este parámetro hace referencia a un archivo donde se guardará o ya contiene la segmentación manual de las imágenes del corpus. Este fichero se generará en el programa HMM y se leerá en el ANN.
- **ANN (ann_filename):** Este parámetro contiene la ruta al fichero de configuración de la ANN, donde se especifica la topología y todos los parámetros adicionales.

- **Probabilidades a priori de los estados (states_a_priori_probs_filename):** Archivo donde se guardará o se contiene las probabilidades a priori de cada estado. Este fichero se generará en el programa ANN y se leerá en el HMM.
- **Set o corpus (set):** Este parámetro permite elegir el corpus con el que se desea trabajar. Este parámetro es muy importante, porque dependiendo del corpus, la lectura de los datos es distinta.
- **Tamaño de marco (window_length):** El tamaño de los marcos que se obtienen de las imágenes.
- **Distancia entre marcos (delta_length):** Distancia entre los marcos que se generan.
- **Índice HMM (hmm_index):** Cada vez que se realiza una acción en el entrenamiento de los HMM se guarda la definición de los mismos. Este parámetro permite elegir en qué etapa empezará el entrenamiento de los HMM, que cargará los modelos correspondientes a este valor y empezará a entrenar.
- **Acciones (ACTIONS):** Los archivos de configuración permiten definir la secuencia de acciones que se realizarán en el entrenamiento de los HMM. Estas acciones serán explicadas con detalle en el apartado del entrenamiento de los HMM.

```
parameters_cfg      cfg/parameters.cfg
hmm_index           1

ACTIONS
FLAT_START
# Components per Gaussian 1
BAUM_WELCH          2
SPLIT_GAUSSIANS
# Components per Gaussian 2
BAUM_WELCH          2
SPLIT_GAUSSIANS
# Components per Gaussian 4
BAUM_WELCH          2
SPLIT_GAUSSIANS
# Components per Gaussian 8
BAUM_WELCH          2
SPLIT_GAUSSIANS
# Components per Gaussian 16
BAUM_WELCH          2
SPLIT_GAUSSIANS
# Components per Gaussian 32
BAUM_WELCH          18
FORCED_ALIGNMENT
```

```
path_symbols                        /home/common/datasets/HTR/Rodrigo/tfg_joapuiib/data/text/symbol_list.txt
path_transcriptions                 /home/common/datasets/HTR/Rodrigo/tfg_joapuiib/data/text/transcriptions.mlf
path_samples                        /home/common/datasets/HTR/Rodrigo/tfg_joapuiib/data/embedded5
extension                           embedded
n_samples                           -1
path_hmmdefs                        hmmdefs
mlf_filename                        mlf/ann_inicial.mlf
states_a_priori_probs_filename      lib/propri_probs.txt
set                                 Rodrigo
window_length                       5
delta_length                        1
~
~
```

Ilustración 11: Ejemplo archivos de configuración

4.3 Entrenamiento HMM

Haciendo uso de las librerías utilizadas, este programa llamado *hmm_train* permite crear, inicializar y entrenar los modelos de Markov.

En primer lugar, se procesa el fichero de configuración donde se especifican todos los parámetros y acciones que debe realizar el programa. A continuación, se cargan todos los datos necesarios para realizar el entrenamiento, organizándolos de la manera previamente explicada. Una vez cargados los datos, es posible empezar el aprendizaje.

El aprendizaje está definido mediante las acciones cargadas del fichero de configuración, que se ejecutarán secuencialmente. El programa permite realizar las siguientes acciones:

- **Inicializar:** Esta acción crea una definición de los HMM inicializada a los valores por defecto. Esta acción se ejecutará si se encuentra el comando INITIALIZE o el parámetro *hmm_index* es igual a 0.
- ***Flat start*:** Asigna de una manera uniforme los marcos de cada imagen con los estados de los HMM correspondientes a la transcripción imagen. Este proceso no modifica las probabilidades de transición, pero sí que modifica la mixtura de gaussianas para adaptarse a los datos. Esta acción se ejecuta con el comando FLAT_START.
- **Baum-Welch:** Mediante el comando BAUM_WELCH "i" se informa al programa de cuantas iteraciones del algoritmo Baum-Welch debe realizar. El número de iteraciones se define en el parámetro *i* del comando. En cada una de las iteraciones se procesan todos los marcos de todas las imágenes, con sus correspondientes transcripciones, para actualizar las probabilidades de transición y las gaussianas.
- **Aumentar gaussianas:** Esta acción divide las mixturas de gaussianas, duplicando el número de gaussianas por cada mixtura. Aumentar el número de gaussianas por mixtura permite estimar de una manera más precisa las probabilidades de emisión de los estados, pero ralentiza el aprendizaje. Es comando proporcionado para ejecutar esta acción es SPLIT_GAUSSIANS. El número de gaussianas inicial es 1.
- **Segmentación:** El comando FORCED_ALIGNMENT realiza la segmentación de cada imagen con respecto a su transcripción. Este proceso guarda cada segmentación en el fichero MLF especificado en la configuración (mlf_filename). La segmentación también se realiza para el conjunto de validación, que se utilizará en el siguiente programa.

Si en los archivos de configuración están definidos los ficheros correspondientes a la definición de una ANN y las probabilidades a priori de los estados, el entrenamiento de los HMM se realizará con una red neuronal en vez de las mixturas de gaussianas.

4.4 Entrenamiento ANN

Haciendo uso de la segmentación generada por el programa anterior, el programa llamado *ann_train* permite generar las etiquetas de las muestras y entrenar una ANN.

Inicialmente, este programa carga los archivos de configuración y los datos al igual que *hmm_train*. A continuación, se carga el archivo MLF definido en el parámetro *mlf_filename* y se generan las etiquetas. La etiqueta de los marcos corresponde al estado que pertenece el píxel central del cada marco. En esta lectura, también se puede realizar el conteo necesario para estimar la probabilidad a priori de cada estado $P(S_i)$. Este proceso también tiene en cuenta los datos de validación, que se utilizarán para comprobar de una manera más objetiva el error de la red, ya que estos datos no influyen en el aprendizaje.

Una vez cargados los datos, se procesa el archivo de configuración de la red neuronal y empieza el entrenamiento. En dicho archivo se define la topología, funciones de activación, *drop-out*, de cada capa, *learning-rate* y el número mínimo de iteraciones o *epochs* que se realizarán antes de terminar el aprendizaje.

Una vez terminado el aprendizaje, se salvan en el sistema todos los parámetros aprendidos de la red en un fichero especificado en el archivo de configuración.

5. Experimentos y resultados

En el presente apartado se expone el proceso de los distintos experimentos realizados, las decisiones tomadas durante la experimentación y sus respectivos resultados.

Los resultados serán obtenidos mediante la comparación de las segmentaciones obtenidas por el programa. Estas segmentaciones serán comparadas con una única segmentación manual compuesta por 49 imágenes elegidas al azar.

Esta comparación es realizada calculando el error entre las fronteras definidas en la segmentación obtenida y la manual. Determinamos que una frontera está bien determinada si:

$$|frontera\ obtenida - frontera\ manual| < tolerancia$$

Paralelamente a estos experimentos también se han realizado pruebas para abordar un problema de reconocimiento de habla. El corpus utilizado llamado Albayzin, contiene 4.800 muestras compuestas por vectores de 23 valores reales.

Todos los experimentos han sido ejecutados en máquinas que soportan hasta 32 hilos de ejecución.

5.1 Experimento sobre los PGM

El primer experimento realizado corresponde al entrenamiento de los HMM utilizando las 10.000 imágenes PGM en escala de grises directamente. La anchura del marco de este experimento es de 5 píxeles, y la distancia entre marcos de 1 píxel. Esta configuración genera 6.835.219 de marcos de 250 dimensiones.

El experimento consiste en entrenar y aumentar progresivamente el número de gaussianas por mixtura, hasta llegar a 32. A continuación, se refinan las mixturas y se obtiene la segmentación de los datos.

El proceso de entrenamiento usado es el siguiente:

- Inicializar los HMM.
- *Flat start.*
- 2 iteraciones de *Baum-Welch.*
- Aumentar gaussianas (2 gaussianas).
- 2 iteraciones de *Baum-Welch.*
- Aumentar gaussianas (4 gaussianas).
- 2 iteraciones de *Baum-Welch.*
- Aumentar gaussianas (8 gaussianas).
- 2 iteraciones de *Baum-Welch.*
- Aumentar gaussianas (16 gaussianas).
- 2 iteraciones de *Baum-Welch.*
- Aumentar gaussianas (32 gaussianas).
- 18 iteraciones de *Baum-Welch.*
- Segmentación.

Este proceso duró aproximadamente 2 semanas y se obtuvieron los siguientes resultados:

Tolerancia en píxeles	Porcentaje de aciertos (%)
1	9.16
2	24.16
3	37.48
4	50.55
5	59.93
10	86.67
15	96.10

Tabla 1: Resultados segmentación PGM

Paralelamente a este experimento se lanzó un experimento con el set de reconocimiento del habla con la misma configuración, obteniendo 1.453.781 marcos de 115 dimensiones. Este proceso tuvo una duración de 36 horas aproximadamente y el acierto para una tolerancia de 30 milisegundos (comparable a 5 píxeles en escritura manuscrita) es de 93.10%.

Tras analizar los malos resultados obtenidos en escritura manuscrita con respecto a los obtenidos en reconocimiento del habla, tanto del porcentaje de acierto como de duración del entrenamiento, se decidió aplicar una técnica de reducción de dimensionalidad a los marcos.

5.2 Experimento reducción de dimensionalidad

Tras obtener los anteriores resultados se decidió reducir la dimensionalidad de las muestras mediante una ANN en modo *autoencoder*.

Para realizar una buena compactación de los datos se han realizado experimentos con dos topologías diferentes para la ANN y dos anchuras de marco distintas.

- **Capa de entrada:** El *drop-out* es de 1 para ambos casos.
  - Para una anchura de marco de 5 píxeles, esta capa contiene 250 neuronas y una función de activación sigmoide, cuyos valores emitidos están entre 0 y 1.
  - Para una anchura de marco de 7 píxeles, esta capa contiene 350 neuronas y una función de activación lineal.
- **Capas ocultas:** Para ambos experimentos se utilizan dos capas de 1024 unidades. La función de la primera es *linear-rectified* y el *drop-out* 0.5 para ambos casos. La función de activación de la segunda capa oculta es:
  - Para 5 píxeles, tangente hiperbólica.
  - Para 7 pixeles, *linear-rectified*.

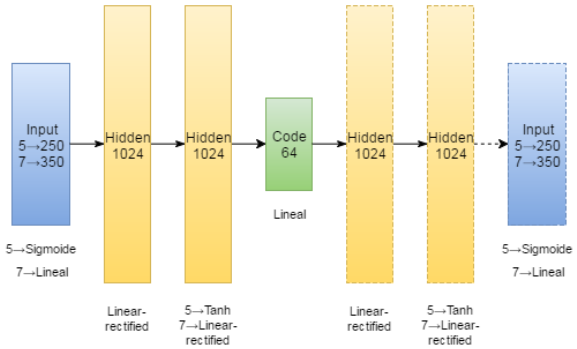

Ilustración 12: Especificación autoencoder

- **Capa de salida:** La capa de salida es común para los dos experimentos. Esta capa está compuesta por 64 neuronas con una función de activación linear y un *drop-out* de 1.

Se decidió reducir la dimensión de las muestras a 64 dimensiones en todos los casos para poder utilizar la misma topología de la red neuronal en los siguientes pasos.

Una vez realizado el entrenamiento de las redes neuronales se generaron y se guardaron en disco los datos correspondientes a la reducción de dimensiones de los marcos obtenidos mediante una anchura de 5 píxeles y 7 píxeles, nombrados *Embedded-5* y *Embedded-7* respectivamente.

1. Entrenamiento HMM

Con los datos obtenidos se lanzó el mismo proceso de entrenamiento de los HMM usado en el experimento anterior, que concluyó en 2 días aproximadamente y se obtuvieron los siguientes resultados:

Tolerancia en píxeles	**Porcentaje de aciertos (%)**		
	PGM	**Embedded-5**	**Embedded-7**
1	9.16	19.82	12.51
2	24.16	51.28	34.77
3	37.48	74.40	55.35
4	50.55	86.43	70.69
5	59.93	91.62	79.80
10	86.67	98.53	96.22
15	96.10	99.62	99.14

Tabla 2: Comparación resultados PGM, Emedded-5 y Embedded-7

Los resultados de los experimentos con reducción de la dimensionalidad son muy buenos en comparación a los resultados obtenidos con las imágenes en escala de grises. La duración se reduce a una séptima parte con respecto al experimento anterior, y el error, para 5 píxeles de tolerancia y de anchura de marco, supera la barrera del 90%.

Para obtener los resultados anteriores se tuvieron que realizar unos pequeños ajustes.

El primer ajuste se decidió al realizar la segmentación manual. La modificación que se realizó fue añadir un espacio en blanco, identificado como <SPACE> al inicio y al final de cada transcripción, ya que las imágenes empiezan y terminan con un espacio que no estaba reflejado en las transcripciones proporcionadas. Este cambió mejoró la estimación de los HMM correspondientes al espacio en blanco y a todas aquellas unidades que aparecen al principio o final de alguna imagen.

Otro ajuste realizado fue la reducción de estados del modelo de Markov asociado al espacio en blanco, eliminando 2 de los estados propios del modelo. El objetivo de este cambio es mejorar la estimación del modelo correspondiente al espacio, ya que este "carácter" no dispone de unas características diferentes a lo largo de su aparición. Esta modificación también mejora la segmentación manual en aquellas imágenes donde aparecen espacios muy pequeños, ya que como mínimo se asigna 1 píxel a esta unidad (uno por estado).

Cada una de estas modificaciones mejoró un 2% aproximadamente el acierto obtenido para 5 píxeles de tolerancia.

Analizando los resultados obtenidos se dedujo que la mejor topología para reducir la dimensionalidad de los marcos es la estructura utilizada en *Embedded-5*, y se decidió continuar la experimentación con dichos resultados.

2. *Entrenamiento ANN*

Tras obtener la segmentación, se entrenó una red neuronal con la siguiente topología:

- **Capa de entrada:** 64 nodos que coincide con la dimensión de las muestras, dunción de activación linear y *drop-out* de 1.
- **Capas ocultas:** Tres capas ocultas de 512 nodos, una función de activación *linear-rectified* y un *drop-out* de 0.5.
- **Capa de salida:** Esta capa contiene 316 neuronas (106 símbolos * 3 estados – 2 estados del espacio). La función de activación es *soft-max* y el *drop-out* de 1.

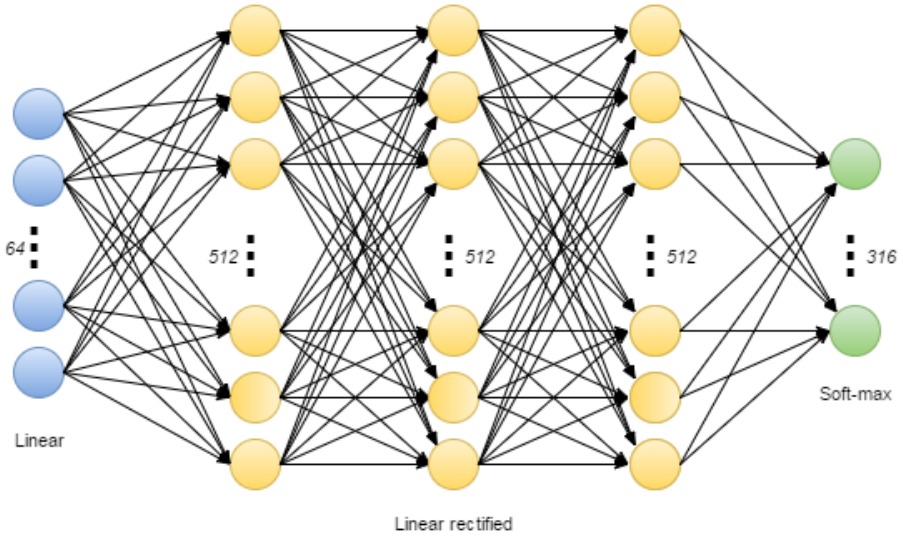

Ilustración 13: Especificación ANN

El entrenamiento de la red neuronal se realizó en 100 iteraciónes y se completó en 2,5 días aproximadamente. La evolución del error se puede visualizar en la siguiente gráfica:

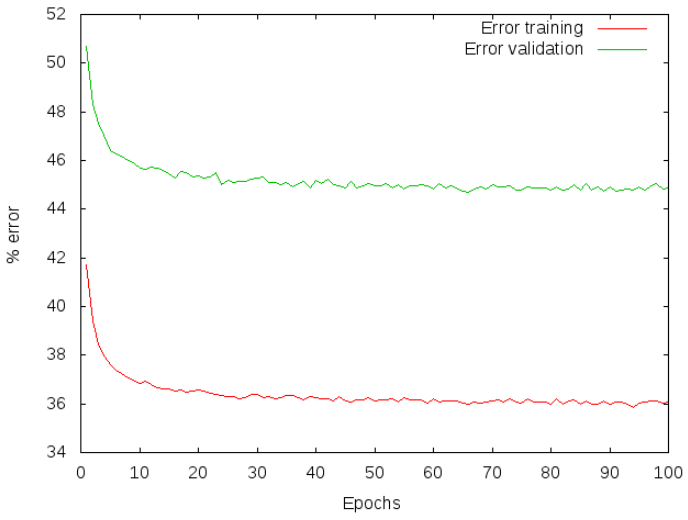

Ilustración 14: Error ANN

El error es relativamente alto (Inicio: 50.692%, Final: 44.854%) pero se usará la ANN para intentar mejorar los resultados obtenidos de la segmentación anterior.

3. *Incorporación ANN en los HMM*

Una vez entrenada la red neuronal es posible integrarla en los modelos de Markov y refinar ambos modelos.

El proceso que seguiremos para reestimar los parámetros será el siguiente:

1. **Reestimación HMM,** que incorporara la red neuronal entrenada. Este proceso realizará una sola iteración del algoritmo Baum-Welch y generará una nueva segmentación.
2. **Reestimación ANN,** que procesando la anterior segmentación, generará nuevas etiquetas y realizará una sola iteración del algoritmo Back Propagation. La red podrá volver a usarla en el punto anterior.

Repitiendo este proceso 20 veces se obtuvieron los siguientes resultados:

Iteración	**Porcentaje de aciertos (%) / Tolerancia en píxeles**						
	1	**2**	**3**	**4**	**5**	**10**	**15**
*	**19.82**	51.28	74.40	86.43	91.62	98.53	99.62
1	18.95	53.35	75.80	87.84	93.50	98.71	99.50
2	19.00	54.58	77.33	89.00	93.92	98.71	99.54
3	19.06	54.50	77.15	89.03	93.95	98.83	99.58
4	19.20	54.65	78.01	89.25	94.13	98.96	99.71
5	19.10	54.34	77.94	89.24	94.04	98.87	99.62
6	19.10	**54.67**	77.77	89.45	93.99	98.87	99.71
7	18.89	54.59	78.27	89.45	93.98	98.87	99.62
8	18.71	54.25	78.38	89.54	94.21	99.04	99.71
9	18.78	54.38	**78.63**	89.73	94.32	99.04	99.71
10	18.62	54.46	78.46	89.79	94.75	99.00	99.71
11	18.45	53.44	78.43	90.00	94.63	99.00	99.79
12	17.68	53.38	78.11	89.87	94.62	98.87	99.71
13	18.20	53.31	78.47	90.17	95.04	99.04	99.79
14	17.59	52.86	78.41	90.12	95.08	99.00	99.79
15	17.80	53.23	78.49	90.00	95.16	**99.17**	**99.83**
16	17.86	53.75	78.18	90.09	**95.17**	99.08	99.83
17	17.68	53.21	78.11	**90.45**	95.12	99.17	99.79
18	17.82	52.80	78.30	90.15	95.03	99.17	99.79
19	17.60	52.52	77.78	90.05	95.01	99.00	99.75
20	17.83	53.21	78.08	90.08	95.08	99.08	99.79

Tabla 3: Evolución error reestimación Embedded5

*Segmentación inicial generada mediante gaussianas.

Cada iteración tarda aproximadamente 3 horas en completarse. Estos resultados son notablemente más buenos que los resultados obtenidos al usar solamente gaussianas. Se puede observar que el acierto para 4 píxeles de tolerancia supera la barrera del 90% y que tan solo con una tolerancia de 5 píxeles, se consigue sobrepasar el 95%. También se puede observar el gran impacto que genera la incorporación de la red neuronal en los HMMs, ya que el error disminuye considerablemente en la primera iteración del proceso. Para 5 píxeles de tolerancia el error pasa de 8.38% a 6.5% (-1.88%).

5.3 Experimento reentrenamiento ANN

Las redes neuronales, explicadas en anteriores apartados, son muy sensibles en las primeras iteraciones del entrenamiento, pero una vez ajustados los parámetros es muy complicado reestimarlos.

El presente experimento intenta demostrar esta afirmación entrenando de cero una red neuronal, aprovechándose de las segmentaciones generadas en el experimento anterior. Para ello, utilizaremos la segmentación generada en la primera iteración del refinamiento, cuyo acierto para una tolerancia de 5 píxeles mejora un 1,88%.

La topología de la nueva ANN es la misma que en el experimento anterior y la evolución del error con la nueva segmentación se puede visualizar en la siguiente gráfica:

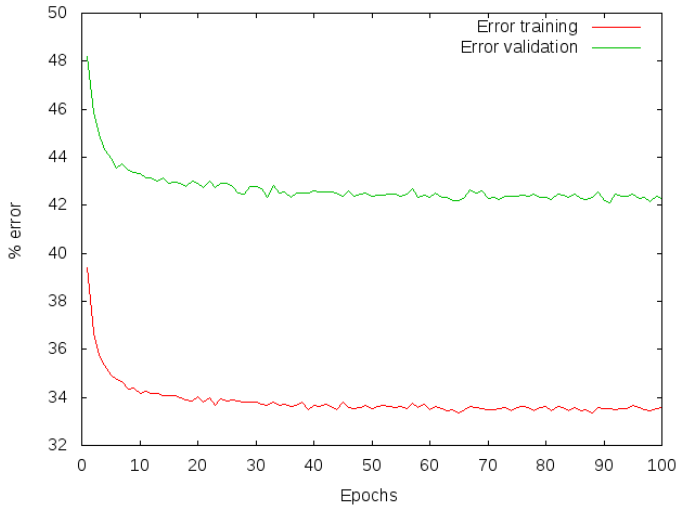

Ilustración 15: Error ANN reentrenamiento

Como puede observarse, el error es menor durante todo el entrenamiento en comparación a la primera red neuronal entrenada (Inicio: 48.213%, Final: 42.299%).

Repitiendo x*/ veces el anterior proceso de reestimación de los parámetros se obtienen los siguientes resultados:

Iteración	**Porcentaje de aciertos (%) / Tolerancia en píxeles**						
	1	**2**	**3**	**4**	**5**	**10**	**15**
Inicial	18.95	53.35	75.80	87.84	93.50	98.71	99.50
1	18.76	53.53	76.68	88.97	93.52	98.79	99.62
2	18.74	53.21	77.00	88.98	93.53	98.91	99.67
3	19.16	53.57	76.45	89.19	93.74	99.12	99.71
4	19.14	53.79	77.15	89.12	93.99	99.08	99.67
5	19.03	54.09	77.05	89.48	94.20	99.08	99.75
6	**19.24**	**55.01**	77.55	89.27	94.07	98.91	99.75
7	18.82	54.80	77.09	89.36	94.53	99.08	**99.79**
8	18.44	54.40	77.14	89.49	94.58	**99.12**	99.79
9	18.42	54.40	77.53	90.00	94.75	99.08	99.79
10	18.60	53.75	**77.65**	**90.12**	**95.00**	99.08	99.79
11	18.47	53.73	77.72	90.09	94.67	99.08	99.75
12	18.50	54.47	77.84	89.98	94.72	99.09	99.79

13	18.23	53.79	77.87	90.10	94.74	99.17	99.79
14	18.31	53.65	77.53	90.16	94.73	99.13	99.83

Tabla 4: Evolución resultados reentrenamiento ANN con Embedded-5

Los resultados obtenidos son ligeramente mejores que los obtenidos anteriormente. Se puede observar que la barrera de los 90% para 4 píxeles de tolerancia se sobrepasa en la iteración 9, dos iteraciones antes que en el anterior modelo. Analizando todos los valores, se puede observar que el acierto es ligeramente superior en la gran mayoría de los casos.

Un detalle que diferencia este experimento del anterior es que no hay una gran mejora en la primera iteración con respecto a la segmentación inicial. Esto demuestra que sí que hay una mejora "instantánea" al usar una ANN en vez de las mixturas de gaussianas.

5.4 Trabajos futuros

En este apartado se exponen los experimentos que no se han realizado y podrían ser usados para obtener mejores resultados.

1. *Probar nuevas topologías para reducir la dimensionalidad con una ANN*

Como se ha observado, elegir una buena topología en la compactación de los datos influye de manera muy notable en el aprendizaje y los resultados posteriormente obtenidos. Uno de los experimentos pendientes seria buscar nuevas topologías que mejorasen los resultados obtenidos.

2. *Probar con diferentes tamaños de marco.*

Este es uno de los experimentos futuros más importantes que no se ha llevado a cabo de momento, dada la gran duración de los procesos y la reducción de dimensionalidad (hasta 1 semana). Este experimento habría consistido en probar la reducción de dimensionalidad elegida y realizar el entrenamiento llevado a cabo en los anteriores experimentos con diferentes tamaños de marco (7, 9 y 11).

3. *Realizar más iteraciones en la reestimación*

Repetir el proceso realizado en la reestimación de los modelos hasta que no se mejoren los resultados. Con esta prueba se podría buscar la mejor segmentación que el proceso es capaz de proporcionar.

4. *Reentrenar diferentes ANN en el proceso de reestimación*

Este experimento sería el siguiente paso del último experimento realizado. Consistiría en realizar el proceso de reestimación durante *x* iteraciones. Una vez realizadas, elegir la mejor segmentación obtenida y reentrenar la ANN desde cero, para volver a repetir el proceso. El mayor inconveniente de esta prueba sería la alta duración del experimento.

6. Conclusiones

6.1 Conclusión del proyecto

Analizando los resultados obtenidos en los experimentos realizados, y a falta de completar algunos experimentos importantes, se puede constatar que el hecho de utilizar redes neuronales para estimar las probabilidades de emisión de los HMMs aporta una mejora significativa con respecto al uso de mixturas de gaussianas.

Otra conclusión relevante de este proyecto es el hecho de haber podido comprobar la importancia de trabajar con una buena representación de los datos. Analizando los resultados de los experimentos, el haber hecho uso de *autoencoders* para compactar los datos ha permitido obtener resultados significativamente mejores que si se utilizan los datos sin procesar. Esta mejora en los resultados se obtiene en el entrenamiento inicial de los HMM con gaussianas, por lo que dicha mejora es independiente del uso de redes neuronales para calcular las probabilidades de emisión de los HMM.

En cuanto a la incorporación de la ANN en los modelos de Markov, el error disminuye en gran medida en la primera iteración tras su incorporación. Esto permite corroborar que se obtienen unos muy buenos resultados a la hora de segmentar las imágenes con esta técnica. En el mejor de los casos, solo 5 de cada 100 fronteras están mal determinadas por el alineamiento forzado con una tolerancia de 5 píxeles.

6.2 Aplicaciones en otras áreas

Este mismo proceso de aprendizaje puede ser aplicado en otras áreas, como puede ser en el reconocimiento del habla. En este mismo proyecto se han realizado experimentos adicionales sobre un corpus de voz, con el objetivo de constatar que la utilización en los ANN para calcular las probabilidades de emisión en los HMM mejoran los resultados obtenidos con gaussianas.

7. Bibliografía

[1] S.N. Srihari, E.J. Keubert, Integration of handwritten address interpretation technology into the United States postal service remote computer reader system, in: Fourth International Conference on Document Analysis and Recognition, vol. 2, Ulm, Germany, 1997, pp. 892–896.

[2] G. Dimauro, S. Impedovo, R. Modugno, G. Pirlo, A new database for research on bank-check processing, in: 8th International Workshop on Frontiers in Handwriting Recognition, 2002, pp. 524–528.

[3] Alejandro H. Toselli , Verónica Romero, Moisés Pastor, Enrique Vidal, Multimodal interactive transcription of text images, in: Pattern Recognition 43 (2010) 1814–1825, 2009, pp. 1814.

[4] Marti, U.-V., Bunke, H., 2001. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. Internat. J. Pattern Recognit. Artif. Intell 15 (1), 65–90.

[5] Vicent Alabau, Carlos-D. Martínez-Hinarejos, Verónica Romero, Antonio-L. Lagarda, An iterative multimodal framework for the transcription of handwritten historical documents, in: Pattern Recognition Letters 35 (2014) 195–203, 2012, pp. 195.

[6] Daniel Martín-Albo, Verónica Romero, Alejandro H. Toselli, Enrique Vidal, Multimodal Computer-Assisted Transcription Of Text Images At Character-Level Interaction, in: International Journal of Pattern Recognition and Artifcial Intelligence Vol. 26, No. 5, 2012, pp. 1.

[7] Marti, U.-V., Bunke, H., 2001. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. Internat. J. Pattern Recognit. Artif. Intell 15 (1), pp. 1

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov; Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 15(Jun):1929−1958, 2014.

8. Ilustraciones y Tablas