Reconocimiento de patrones

Proyecto Final

Estudio de técnicas alternativas para la identificación biométrica mediante las huellas digitales

Vanina Camacho

Guillermo Garella

5 de diciembre de 2016

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República

Índice

1. Introducción

El reconocimiento biométrico automatizado en adultos ha sido desarrollado ampliamente en los últimos años y ha sido sumamente exitoso en el ámbito forense, de la ley y en dispositivos personales como ser los celulares. Las aplicaciones del reconocimiento biométrico están enfocadas principalmente a los adultos y adolescentes. Por lo que los desarrolladores, investigadores y vendedores se han centrado en el desarrollo de la captura de datos y sistemas de reconocimiento para adultos. El problema de la identicación civil de niños es un tema en el que existe un interés creciente y que presenta dificultades a los sistemas biométricos desarrollados para el reconocimiento de adultos. Debido al tamaño de las huellas de los niños los extractores de puntos característicos (minucias) de adultos no funcionan correctamente, por lo que en este proyecto se busca una alternativa para establecer una correspondencia entre huellas. Para luego comprobar su funcionamiento con huellas de niños.

Más precisamente con técnicas no vistas en el curso que pueden ser utilizadas para resolver el problema de matcheo de huellas.

Se estudiará la arquitectura basada en redes neuronales convolucionales siamesas, y su utilización en problemas de identicación.

Se estudiará el mecanismo de transfer learning o fine tuning que permite entrenar parte de la red utilizando características aprendidas con bases de datos más grandes.

Se evaluará si es posible combinar ambos métodos y aplicarlos al problema de identicación de huellas con las bases de datos disponible.

2. Bases de datos

Las bases de datos que se utilizaron para entrenar los modelos utilizados y para entrenar parte de la red con huellas fueron:

- MNIST

  La base de datos MNIST es una base de 60000 imágenes compuesta por dígitos escritos a mano. En la figura 1 se muestran algunas imágenes pertenecientes a la base.

Figura 1: Dígitos MNIST

- ImageNet

  ImageNet es una base de datos de imágenes variadas. El modelo utilizado fue entrenado con 1000000 de imágenes pertenecientes a ImageNet.

- NIST

  Se cuenta con 10000 imágenes de huellas dactilares de adultos pertenecientes a una base de NIST. Las mismas son adquiridas mediante rodamiento con tinta sobre papel y luego escaneadas.

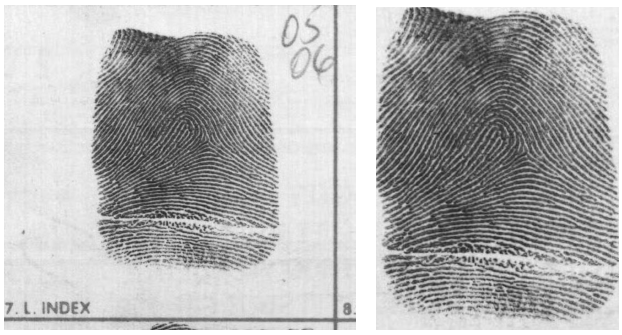
- DNIC

  Se cuenta también con una base privada de datos brindada por la DNIC (Dirección Nacional de Identificación Civil). La misma está compuesta por 310 identidades con varias tomas con la restricción de que la primera toma fuera registrada en el primer mes de vida. En este caso se cuenta tanto con huellas rodadas con tinta como con huellas adquiridas por sensores.

  Tanto las huellas proporcionadas por la DNIC, como las pertenecientes al NIST están divididas en Gallery y Query. El primero corresponde a una toma de la persona y el segundo a otra toma espaciada en el tiempo de la primera.

Pre-Procesamiento de los datos

En cuanto al pre-procesamiento de los datos, se segmentaron las imágenes de forma tal de que la imagen utilizada contuviera solo la huella. Dicha segmentación se realizó mediante operaciones morfológicas, etiquetado de regiones y recortando la imagen de forma de contener la región de mayor área. En la figura 2 se puede ver el resultado de este proceso para una huella de la base del NIST.



(a) Huella original (b) Huella segmentada

Figura 2: Segmentación

Herramientas utilizadas

- Matlab
- Python
- Caffe

Caffe es una potenete librería de deep learning desarrollada por el Centro de Visión y Aprendizaje de Berkeley (BVLC) y por contribuidores de la comunidad.

Dicha librería cuenta con implementaciones que se utilizarán en el correr del informe mostrando su utilización y resultados.

3. Técnicas utilizadas

3.1. Redes neuronales convolucionales siamesas

Las redes neuronales siamesas son un tipo de redes neuronales compuestas por dos subredes idénticas, es decir, contienen la misma configuración con los mismos parámetros y pesos.

En la figura 3 se muestra la estructura típica de una red siamesa.

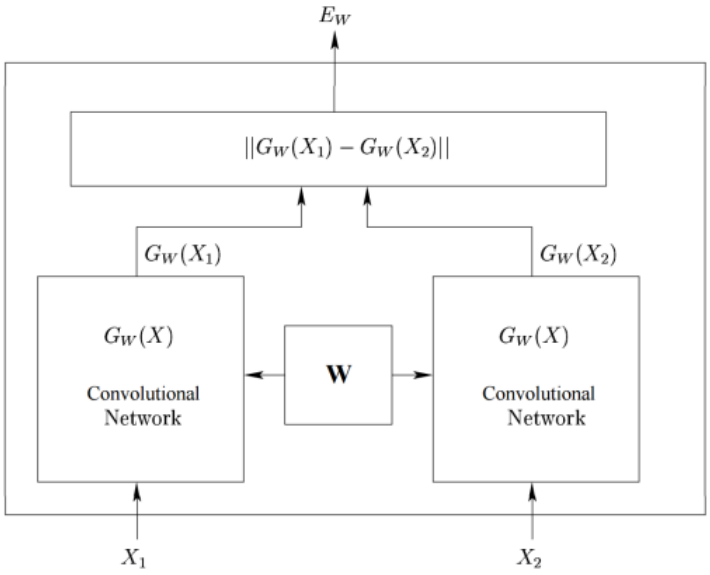


Figura 3: Estructura red siamesa

La entrada al sistema son un par de imágenes con una etiqueta, la cual indica si ambas imágenes son similares o no. Dichas imágenes pasan a través de las subredes, obteniendo dos salidas que se introducen por la función de loss. La función de loss utilizada para este tipo de redes es la función de *Constrative Loss*.

Contrastive Loss

La función de Loss se calcula como:

$$E = \frac{1}{N}\sum_{n=1}^{N} y||a_n - b_n||_2^2 + (1-y)max(margin - ||a_n - b_n||_2, 0)^2 \quad (1)$$

siendo, a_n, b_n los vectores de características al final de la red, y la etiqueta según si el par de imágenes a la entrada es igual o distinto (1 o 0), N la cantidad de imágenes en el batch de la red. Siendo el batch, la cantidad de imágenes que se cargan por iteración.

Esta función consta de dos penalizaciones, penaliza un par similar que está lejos (término de la izquierda) y penaliza un par distinto que esté más cerca que un cierto margen (término de la derecha). Aquellos pares distintos que estén más lejos que el margen no aportan a la función de loss.

Capas

Las redes convolucionales que se utilizaron en este trabajo, se detallan más adelante, están compuestas por distintas capas:

- Convolucionales
- Submuestreo (pooling)
- Totalmente conectadas (InnerProduct)
- De activación
- De normalización

Se detallan a continuación las funciones principales de cada capa.

Convolucionales

Los parámetros de esta capa consisten en componentes de bias y filtros cuyos pesos son aprendidos en el entrenamiento de la red. Cada filtro se convoluciona con el volumen de entrada, por ejemplo con una imagen de entrada de 3x256x256 (canales, ancho, alto). Los tamaños de los filtros son por lo general chicos. La salida de esta capa consta de un volumen dependiendo de la cantidad de filtros y el paso con el que se realiza la convolución. Se puede ver como:

$$Y_j = b_j + \sum_i K_{ij} * Y_i$$

donde Y_i se corresponde a las neuronas de la capa anterior, K_{ij} el núcleo de convolución y b_j la componente de bias.

Pooling

La función principal de esta capa es progresivamente reducir el tamaño del espacio para reducir así la cantidad de parámetros y el costo computacional en la red. La forma más común es utilizar la operación MAX, con filtros de 2x2 aplicados con un paso de submuestreo de 2. Esto es, tomando 4 píxeles formando un cuadrado de cada canal de la entrada y conservar solamente el valor máximo de estos. En la figura 4 se muestra ilustrativamente este proceso.

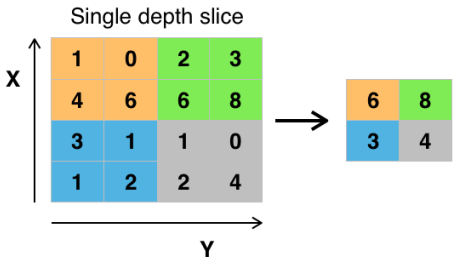


Figura 4: Pooling

Totalmente conectadas

Las neuronas en estas capas tienen conexiones con todas las activaciones de las capas anteriores, cada conexión tiene un peso y una componente de bias.

$$Y_j = b_j + \sum_i w_{ij} Y_i$$

donde Y_i se corresponde a las neuronas de la capa anterior, w_{ij} el peso de la conexión y b_j la componente de bias.

Activación

La capa de activación tiene la función de aplicar una función de activación a las neuronas de salida de las distintas capas de convolución y totalmente conectadas.

Hay distintas funciones posibles de activación pero en las redes de este proyecto se utiliza la función ReLU (Rectified Linear Unit) que se define como:

$$f(x) = max(0, x)$$

Normalización

En esta capa se normalizan regiones de la entrada. Cuando se utilizan neuronas ReLU, al tener activaciones ilimitadas es útil normalizarlas con una capa de Local Response Normalization (LRN).

3.1.1. Red Lenet

La red LeNet es conocida por funcionar bien en aplicaciones de clasificación de dígitos. En caffe se propone la modificación de utilizar funciones de activación ReLU en lugar de Sigmoid.

En la figura 5 se muestra la estructura de esta red.

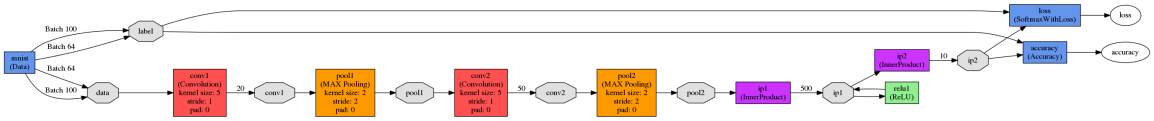


Figura 5: Estructura Lenet

Consiste en una red: *C1-P1-C2-P2-IP1-IP2*
donde C indica una capa convolucional, P pooling, IP capa totalmente conectada.

Esta red se utiliza para luego crear una red siamesa, en el tutorial de caffe [7] se utiliza para clasificar los dígitos de la base MNIST.

En la figura 6 se muestra la estructura resultante al utilizar dos redes Lenet para construir una siamesa.

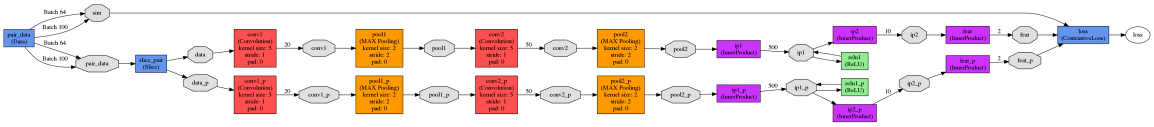


Figura 6: Estructura siamesa Lenet

En este caso se utiliza como función de Loss la función Contrastive Loss a diferencia de la red Lenet.

Es necesario entrenar esta red.

3.1.2. Red Caffenet

Esta red es una variación de la red AlexNet entrenada con ImageNet. El modelo entrenado se encuentra en el model zoo de caffe [8].

En la figura 7 se muestra la estructura de la red.

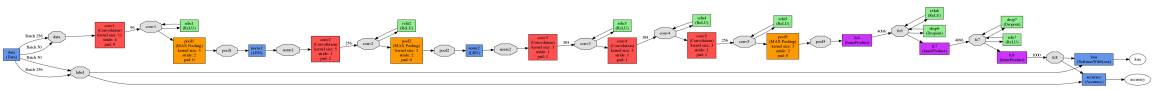


Figura 7: Estructura red Caffenet

Se utilizó la estructura de esta red para construir una red siamesa y utilizar la técnica de Transfer Learning o Fine-Tuning que se detalla en la sección siguiente.

La estructura resultante al crear la red siamesa se muestra en la figura 8

En este caso se agrega una capa creada en python para obtener la etiqueta 1 o 0 necesaria para la función de loss, según si las etiquetas de los datos son iguales o no.

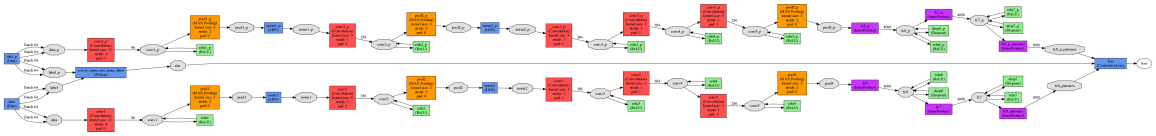


Figura 8: Estructura red siamesa Caffenet

3.2. Transfer Learning

Para entrenar un red convolucional entera desde 0, con inicialización random, es necesario contar con muchos datos y lleva mucho tiempo además de que implica un costo computacional muy alto. En lugar de entrenar toda una red, es posible utilizar la técnica de Transfer Learning que consiste en utilizar una red ya entrenada con muchas imágenes, por ejemplo ImageNet y usarla como inicialización para entrenar o usar la red como extractor de características.

Para usar una red convolucional como un extractor de características fijo se toma una red ya entrenada y en el modelo se remueven capas, y se utiliza la salida como características.

Para usarla como inicialización se cambia el conjunto de datos de entrenamiento y se continúa con backpropagation. Se pueden seguir el entrenamiento tanto de todas las capas como de las últimas, con el fin de evitar el overfitting. Se entrenan las últimas debido a que las características obtenidas por las últimas capas son mas específicas para las clases del conjunto de datos original.

La técnica de transfer learning es útil para distintos escenarios, como ser, contar con conjuntos de datos pequeños, pero similares al conjunto con el cual se entrenó la red, distintos al conjunto de entrenamiento inicial, o contar con conjuntos grandes pero distintos del conjunto original.

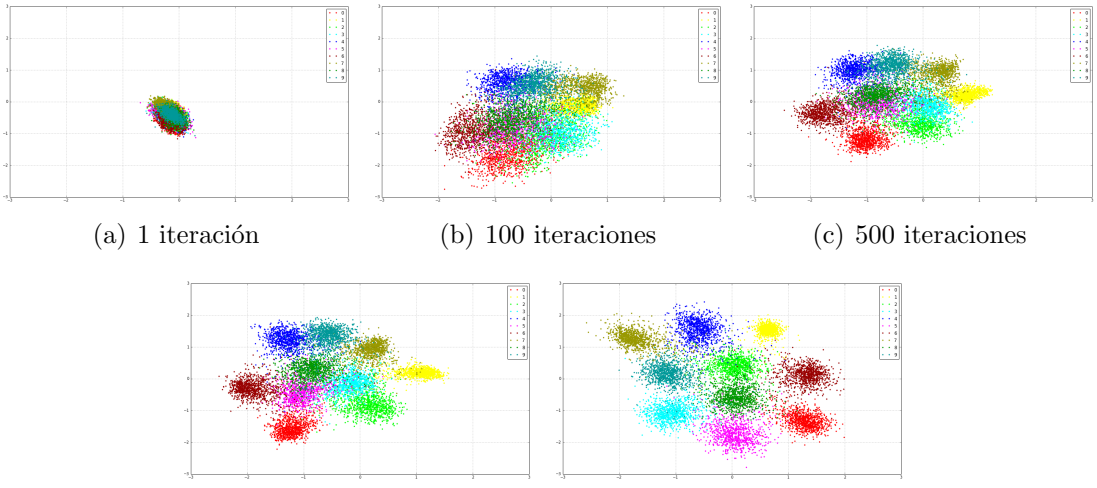
Para el caso de contar con pocos datos pero similares al conjunto original, la mejor opción resulta ser la de entrenar un clasificador como ser SVM. En el caso de datos distintos y pequeños, se puede entrenar un clasificador pero en una salida más temprana que en el caso anterior. En el caso de que el conjunto sea grande y distinto se puede seguir con backpropagation ya que si el conjunto es muy grande no habrá overfitting.

4. Experimentos

4.1. Entrenamiento red siamesa Lenet con MNIST

Se siguieron los pasos del tutorial de caffe para entrenar la red y así obtener un modelo de la misma. Mediante la especificación del solver de caffe es posible guardar modelos de la red según la cantidad de iteraciones. Con el fin de visualizar como se separan los datos a lo largo de las iteraciones se guardaron modelos para 1, 100, 500 ,1000 y 50000 iteraciones.

En la figura 9 se muestran los puntos correspondientes a los dígitos con las características obtenidas a la salida de la última capa totalmente conectada.



(a) 1 iteración (b) 100 iteraciones (c) 500 iteraciones

(d) 1000 iteraciones (e) 50000 iteraciones

Figura 9: Resultado del test para distintas iteraciones

La salida de ésta red, como se observa, es tal que los dígitos se logran mapear como puntos en un plano 2D, esto es debido que cada imagen, luego de pasar por todos los filtros de la red, llegan a representarse mediante 2 características.

Cada color corresponde a un dígito entre el 0 y el 9, se aprecia como a medida que se agregan iteraciones, el modelo separa de manera más efectiva las imágenes correspondientes a dígitos distintos y acerca entre sí aquellas imágenes similares. Se podría decir que a partir de las 5000 iteraciones el modelo tiene un buen desempeño, mostrando una buena separación entre clusters.

También se graficó la salida de la función de loss tanto para el train como para el test, como se muestra en la figura 10.

Como se puede ver tanto para test como para train la función decae. Esto era de esperarse, ya que a medida que transcurren las iteraciones los pesos de las capas se van ajustando. Y lo hacen de manera tal que para las iteraciones siguientes a la número 20000 aproximadamente se produce un sobre entrenamiento para la base de train. Este sobre entrenamiento se puede deber a que se cuenta con una base no muy grande de imágenes comparadas con la cantidad de iteraciones realizadas. Con respecto a la base test, se ob-

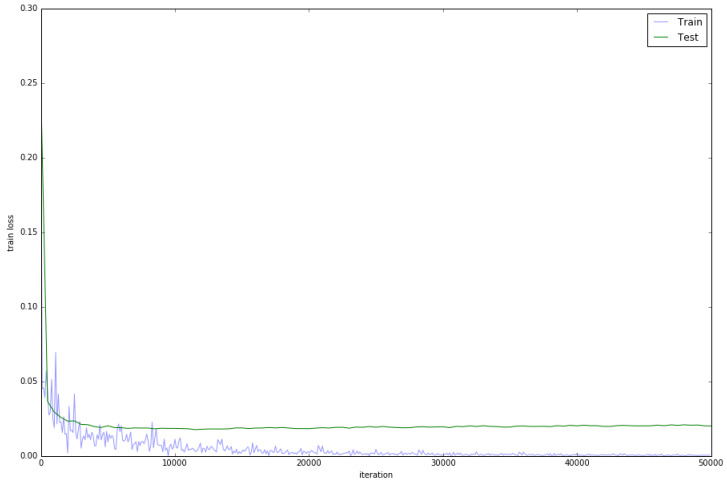


Figura 10: Loss para siamesa lenet con MNIST

serva una uniformidad en la clasificación a partir de la iteración 5000 aproximadamente, como se mencionó de la figura anterior, por lo que ya para ésta iteración se produce una clasificación razonable. Ésta es otra manera de mostrar los resultados vistos en la figura anterior, siendo los resultados compatibles entre las imágenes. Las oscilaciones que se aprecian para el conjunto de train, se pueden deber a que si los batch se toman aleatoriamente imágenes, por lo tanto puede pasar que en un batch no hayan tantos pares de dígitos, y que en el siguiente batch, hayan muchos pares.

Luego, utilizando esta misma red y con el tutorial de fine-tuning [9] de caffe de referencia, se probó de llevar las imágenes de huellas del NIST al tamaño de $28x28$ requerido por la red y entrenar las últimas capas con 8000 datos, dado que no fue posible se probó entrenar toda la red, ya que no es muy grande, pero la función de loss tuvo un valor infinito de forma muy rápida. Por lo que se bajo el learning rate y se prosiguió a entrenar nuevamente la red. El resultado obtenido para el conjunto de test de 2000 datos para un modelo de 1000 iteraciones se muestra en la figura 11.

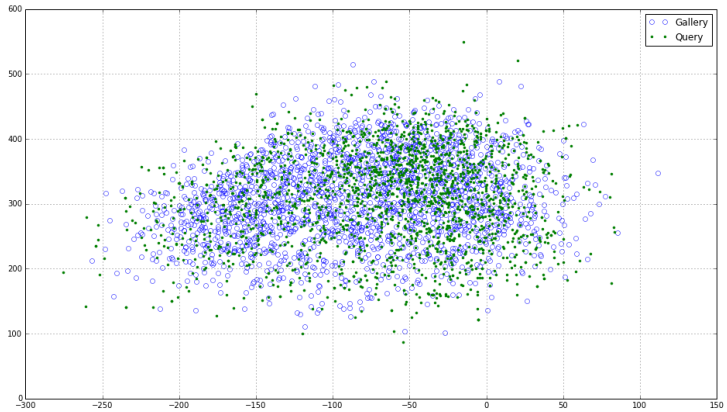


Figura 11: Test de la red lenet entrenada de 0 con imágenes 28x28

Los puntos representados con círculos azules corresponden a la primer toma (Gallery)

de huellas y los puntos verdes a la segunda toma de huellas (Query). Como no es posible apreciar nada en la nube se graficaron sólo algunos puntos (50) correspondientes a genuinos (ver figura 12).

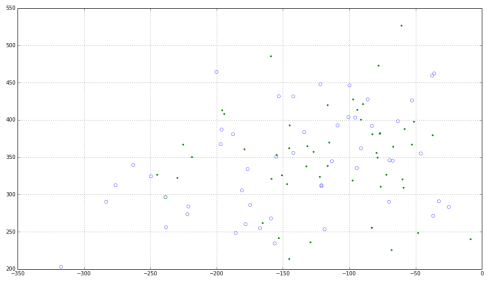
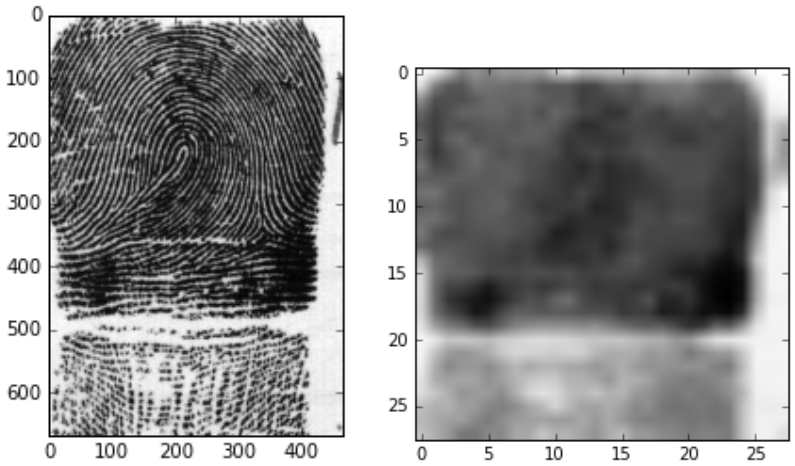


Figura 12: 50 puntos genuinos

Como se puede ver algunos puntos del Query están cercanos a los de Gallery como se espera pero otros están alejados. Se observó el comportamiento de los puntos a lo largo del entrenamiento y se detectó que la aproximación de los puntos alejados era mínima o nula en algunos casos, además de que la distancia entre genuinos en muchos casos era mayor que la distancia entre impostores.

Uno de los posibles problemas es que el problema con las imágenes tan chicas es muy difícil de aprender para la estructura de la red de lenet y no es posible utilizar esta red, ya que las imágenes pierden toda la información relevante y parecen manchas. En la figura 13 se muestra la pérdida de información de la imagen.



(a) Huella original (b) Huella 28x28

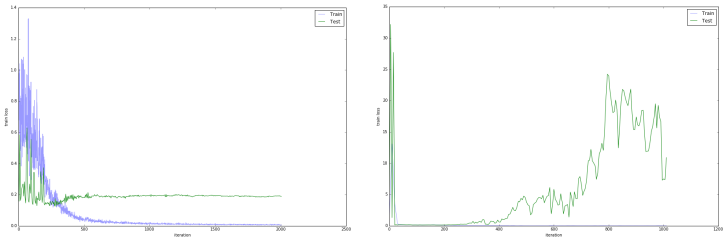
Figura 13: Pérdida de calidad huella

4.2. Entrenamiento red siamesa Caffenet con MNIST

Dado que con la red lenet no se tuvieron resultados prometedores con la red de lenet se prosiguió a utilizar la red de Caffenet mediante la aplicación de la técnica de Fine-Tuning.

Se entrenó la última capa de la red primero con una fracción de la base compuesta por 50 imágenes de train y 50 imágenes de test, con el fin de comprobar el correcto funcionamiento del experimento. Luego se entrenó con 8000 datos para luego realizar el test con 2000. Todas las imágenes fueron llevadas al tamaño requerido por la red de 227x227.

En las figuras 14 se muestran las funciones de loss obtenidas para el entrenamiento y test de la red con dichos datos. Las gráficas fueron obtenidas utilizando distintos tipos de solver admitidos por caffe. El primero es el clásico descenso por gradiente y el segundo denominado Adam es una optimización del método anterior. Con este método, la convergencia es más rápida.



(a) Método descenso por gradiente

(b) Método Adam

Figura 14: Loss con entrenamiento de 50 imágenes de huellas

En el método Adam se nota claramente un sobreentrenamiento a partir de la iteración 250.

Luego, al entrenar la red con 8000 huellas, con el método Adam se obtuvo la gráfica de la figura 15. La misma pertenece constante a partir de la iteración 2500.

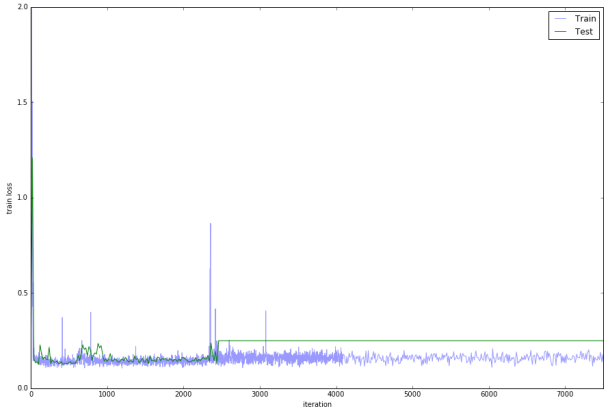
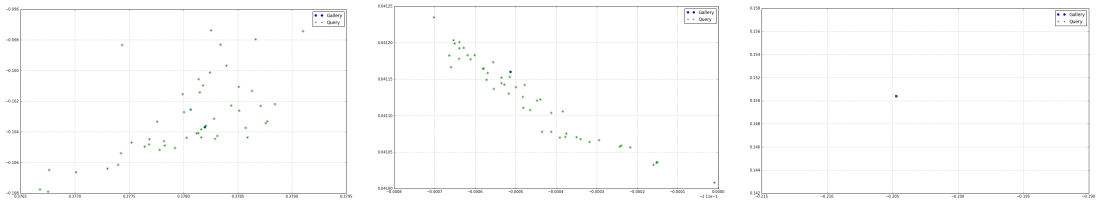


Figura 15: Loss con entrenamiento de 8000 imágenes de huellas

Se graficaron algunos de los puntos de test obtenidos luego de pasar por la red, los mismos se muestran en la figura 16.



(a) 100 iteraciones (b) 500 iteraciones (c) 1000 iteraciones

Figura 16: Resultado del test para distintas iteraciones

Como antes, los puntos representados por el Query son los verdes y los de Gallery círculos azules. El comportamiento no es el esperado, ya que claramente los puntos cada vez se acercan más para finalizar en un único punto, además se comprobó que aquellos pares de huellas que ingresaron a la red juntas quedan en el mismo punto, tanto los pares genuinos como impostores.

No está claro si el método no es adecuado para esta aplicación ya que hay varios factores a tener en cuenta, como ser cuantas capas se entrenan, variación de parámetros para cada capa, como por ejemplo la cantidad de características extraídas en la última capa, y demás. Se puede haber cometido algún error a la hora de armar las bases como también en la definición de la red, que al momento no se detectó.

En el caso de encontrar el problema y de poder solucionarlo, se realizarán las curvas FRR (False Rejection Rate) en función de FAR (False Acceptance Rate) tomando como métrica la distancia entre los puntos.

5. Conclusiones

Se realizó la instalación de la librería Caffe de forma satisfactoria y se llevaron a cabo de manera exitosa las pruebas previstas en el tutorial de la librería Caffe, comprendiendo los resultados y la metodología de los mismos, así como las estructuras de archivos y su contenido.

Con respecto a las pruebas que se realizaron con nuestras implementaciones, los resultados no parecen ser tan vistosos, aunque era de esperarse en ciertos casos. Por ejemplo en el entrenamiento de la red con huellas de tamaño 28x28, se perdía la información importante y no se lograba identificar de buena manera.

Se probaron también implementaciones para la red caffenet con fine tuning, con imágenes de tamaño 227x227. Debido al tamaño de las imágenes, el costo computacional fue mayor, y por lo tanto el tiempo de procesamiento también. Éste fue el mayor problema que se nos presentó, dado que detectábamos errores en las implementaciones luego de haber dejado corriendo los programas y terminaba siendo tiempo perdido.

Trabajo a futuro

Se seguirá con las pruebas para imágenes de tamaño 227x227, tanto para un entrenamiento completo de la red, como utilizando la técnica de fine tuning. Dado que el costo computacional es alto y la demora es prolongada, se intentará conseguir algún equipo que cuente con una GPU adecuada.

Con el motivo de identificar a los bebes mediante las huellas, se tratará de implementar las mismas pruebas realizadas en el proyecto para comprobar la eficacia de los métodos estudiados para éste problema y se intentará conseguir una base más grande, ya que se contaba con poca cantidad para realizar un buen entrenamiento.

Referencias

[1] Quora

[2] Learning a Similarity Metric Discriminatively, with Application to Face Verification

[3] Learning visual similarity for product design with convolutional neural networks

[4] Convolutional Neural Networks for Visual Recognition

[5] Caffe

[6] LeNet

[7] Caffe Siamese

[8] Model Zoo

[9] FineTuning