Búsqueda de palabras manuscritas por similitud visual: Word Spotting

Roger Górriz Gordo

Resumen– En la actualidad existe una gran cantidad de documentos manuscritos que contienen información relevante. Sería de gran utilidad indexar estos documentos para poder realizar búsquedas rápidas por palabras clave. Habitualmente, la técnica para indexar documentos es escanearlos y convertir su contenido en carácteres ASCII mediante un proceso llamado OCR. Sin embargo, este método no se comporta bien con textos manuscritos debido a la varianza de estos. En este documento, se propone un método alternativo para la indexación de documentos manuscritos mediante la similitud visual de sus palabras con otras palabras que ya conocemos. Lo que se llama Word Spotting.

Paraules clau– Fisher Vectors, Aprendizaje Computacional, Descriptores Locales, Word Spotting, Documentos Manuscritos.

Abstract– Nowadays, exsits a huge quantity of handwritten documents which have relevant information. It would be very useful to index those documents in order to be able to do rapid searches by keywords. Usually, the common technique to index documents is to scan them in and convert their content into ASCII characters through OCR. However, this method doesn't work very well with handwritten texts due to its high variance. In this document, we propose an alternative method for the handwritten documents indexing through the visual similarity of its words with other words already known. This technique is also called Word Spotting.

Keywords– Fisher Vectors, Machine Learning, Local Descriptors, Word Spotting, Handwritten documents

1 Introducción

ESTE proyecto pretende ofrecer una alternativa a la indexación de documentos manuscritos mediante Word Spotting. Para hacer esto, este proyecto focalizará el esfuerzo en identificar las palabras de estos documentos. De manera que, dada la imagen de una palabra desconocida de un documento que se quiera indexar, se pueda obtener su 'transcripción' mediante la semejanza con un conjunto de palabras similar.

Para hacer esto, se requerirá una gran base de datos con palabras cuya transcripción conozcamos para entrenar el sistema y así poder buscar similitudes de palabras nuevas con palabras conocidas.

En esencia, se trata de un problema de clasificación de imágenes: esto es, dada una imagen, asociarle una etiqueta o varias según su semejanza con otras imágenes de su misma índole.

Tiene especial interés para este proyecto la clasificación de imágenes a gran escala, es decir, la que tiene que ver con numerosas imágenes y diversas etiquetas. Esto es así porque, para este enfoque, tendremos grandes cantidades de imágenes que contienen cada palabra del vocabulario del idioma en cuestión. Por tanto, se desea obtener una solución que escale sin grandes afectaciones en el rendimiento ni, sobretodo, en la precisión.

En este documento se encontrará toda la información relativa a cómo se encuentra actualmente este campo de investigación, cómo se ha desarrollado el proyecto, los resultados y su análisis, las conclusiones y finalmente la literatura consultada.

2 Estado del arte

Para este proyecto se utilizarán unos descriptores de imagenes llamados Fisher Vectors[1].

Dada la versatilidad de estos Fisher Vectors, se pueden utilizar con un propósito general en lo que a clasificacion de

• E-mail de contacto: roger.gorriz.94@gmail.com
• Mención realizada: Computació
• Trabajo tutorizado por: Marçal Russinyol (CVC)
• Curso 2015/16

imágenes se refiere, ya que se comportan bien con clasificadores lineales, aunque habrá cambios en la implementación según la naturaleza del proyecto.

En el campo de la clasificación de imágenes encontramos múltiples disciplinas que utilizan estos Fisher Vectors, por ejemplo: métodos para la clasificación de caras [2], reconocimiento de escenas[3], o reconocimiento de palabras.

Compitiendo con los Fisher Vectors se encuentran las Redes Neuronales, también con el propósito de la clasificación de imágenes a gran escala. Incluso se proponen modelos híbridos entre estos dos enfoques[4].

Este proyecto es equiparable en cierto modo al realizado por Jon Almazán et al.[5] en el que se consigue un porcentaje de precisión del 92 % en palabras manuscritas utilizando Fisher Vectors combinado con un aprendizaje de atributos.

Debido a la madurez alcanzable en un TFG, este proyecto abarcará los Fisher Vectors, pero no el aprendizaje de atributos.

3 Arquitectura del sistema y flujo de datos

En la aplicación final, el usuario introducirá como input una imagen que contendrá una palabra manuscrita y el sistema devolverá una propuesta de etiqueta basándose en su semejanza con palabras ya conocidas. (Fig 1)

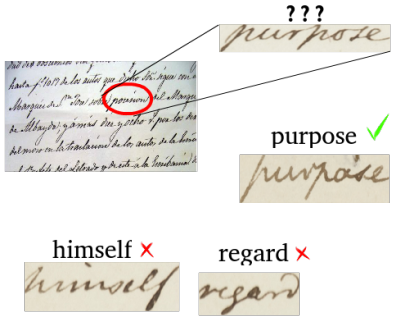

Fig. 1: Representación visual simplificada

Para conseguirlo, previamente el sistema tendrá que ser entrenado con unos Fisher Vectors que serán extraídos de un conjunto de imágenes de las cuales sabemos cual es su transcripción. Una vez el sistema esté entrenado, ya se pueden buscar similitudes de palabras nuevas con palabras ya conocidas.

La primera etapa , que consiste en el entrenamiento del sistema mediante palabras conocidas, se llamará Training. Y la segunda etapa, cuando un usuario introduce una palabra a etiquetar, se llamará Testing.

Para la fase de training se usarán todas las imágenes etiquetadas que tengamos disponibles, éstas pasarán por el modulo de pre-procesamiento. Después se extraerán sus descriptores densos, siguiendo por la generación del Gaussian Mixture Model (GMM), necesario para generar los Fisher Vectors de todas las imágenes. Finalmente, se entrenará el clasificador y ya tendremos el sistema listo para clasificar nuevas imágenes.

Esta fase de training es la más costosa y se realizará una única vez. Fig 2

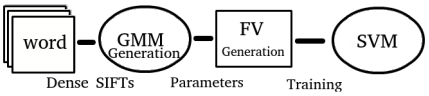

Fig. 2: Proceso de Training

Para la fase de test (Fig 3), el usuario introducirá la imagen de la que quiere saber su transcripción y el sistema seguirá otro flujo. En este caso, se pre-procesará la imagen, se extraerán los descriptores y se generará su Fisher Vector utilizando el GMM generado en la fase de training. Una vez obtenido, este Fisher Vector pasará por el modulo de clasificación donde se comprarará con los Fisher Vectors de las imagenes de training. Así se obtiene la etiqueta del conjunto de palabras al que, por semejanza visual, pertenece la nueva palabra.

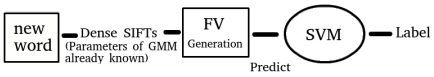

Fig. 3: Proceso de Testing

Lo que se hace en todo este proceso, en el fondo, es obtener unos descriptores generales de las imagenes, los Fisher Vectors. Estos Fisher Vectors son una serie de características de cada imagen codificadas en un vector. Son estos vectores los que representan la imagen y con los que podemos entrenar y clasificar. Cuando se obtienen todos los Fisher Vectors, se tienen todas las imágenes codificadas en un espacio común, y es ahí donde se puede operar matemáticamente y buscar similitudes entre sus características. Así, se puede decidir qué palabra de las que ya conocemos tiene unas características semejantes a las características de la palabra nueva que se quiere transcribir.

4 Desarrollo

En esta sección se descompondrá el sistema en sus diferentes módulos y se explicarán los detalles de la implementación de cada uno de ellos.

4.1. Pre-procesamiento

Dado que las imágenes de palabras manuscritas tienen mucha varianza, es necesario tener una forma estándar de representarlas antes de buscar descriptores. En este módulo se normalizarán las imágenes todo lo posible para que, idealmente, dos palabras iguales tengan la misma representación gráfica. Esto se conseguira mediante la binarización para poder trabajar bien sobre ellas, además de la corrección de los dos factores predominantes en la variación de tipos de escritura, que son las inclinaciones llamadas ‘skew’ y ‘slant’.

4.1.1. Binarizacion

Para trabajar bien con las imágenes, se binariza la imagen utilizando una diferencia de gaussianas (Fig 4). A continua-

ción, se establece un límite, por encima del cual los píxeles seran blancos y por debajo serán negros. Por convención, se usan los píxeles blancos como el objeto a analizar y los píxeles negros como fondo.

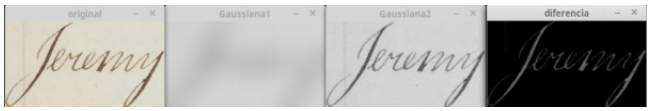

Fig. 4: Secuencia de binarización

4.1.2. Correcion de Skew

Habitualmente las palabras manuscritas tienen una inclinación llamada Skew. (Fig 5)

Fig. 5: Ángulo de skew

El proceso de corrección de Skew alinea el texto con el eje X de la imagen con tal de neutralizar la inclinación.

Para obtener el ángulo que hay que corregir, se divide la imagen en columnas y se selecciona el píxel más bajo perteneciente a la palabra de cada una de ellas.

El conjunto de los píxeles más bajos de las columnas se puede aproximar por una recta de regresión.

Esta recta de regresión, discriminando los puntos más alejados (outliers), es una representación fiel de la inclincación de la palabra.

Mediante una operación de rotación se puede corregir el ángulo de la recta de tal manera que sea paralelo con el eje horizontal.

Fig. 6: Correción del ángulo de Skew

4.1.3. Correcion de Slant

De igual manera que hemos visto en el apartado anterior, las palabras manuscritas también sufren de otra desviación llamada Slant (Fig 7).

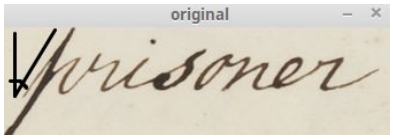

Fig. 7: Ángulo de slant

Una vez se tiene el ángulo de Skew corregido, se puede normalizar el ángulo de slant.

Para corregir el ángulo de slant, se aproxima el contorno vertical de la imagen por un conjunto de líneas rectas. Para cada línea se calcula su inclinación respecto el eje Y y su longitud. Para obtener el ángulo de slant se crea un hash-map, dónde las keys son cada uno de los ángulos de las líneas y el valor es su longitud al cuadrado.

La key con valor más alto corresponderá al ángulo vertical predominante. Se calcula la diferencia respecto la vertical y se corrige con una operación de 'Shear'.

Fig. 8: Corrección del ángulo de Slant

4.2. Generacion de Fisher Vectors

Un Fisher Vector es una representación global de una imagen obtenida mediante las características locales de la misma. Es decir, se ponen en conjunto todos los descriptores locales, SIFT en este caso, y se computa un vector de longitud N que representa las características de la imagen global.

4.2.1. Dense SIFT

Para cada imagen se extraen descriptores SIFT densos, la diferencia al extraer descriptores SIFT no-densos es que los keypoints no son auto-detectados, sino que se obtienen de una forma uniforme por toda la imagen. Para esto, se divide la imagen en cuadrículas con un tamaño (path size) determinado, y se extraen los descriptores para cada una de las regiones locales (Fig 9).

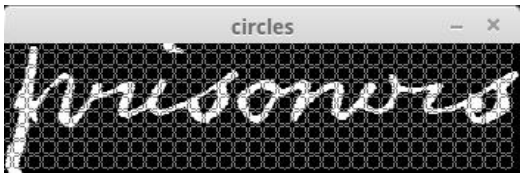

Fig. 9: Representación del dense SIFT.

Ya que las imágenes son de un tamaño variable, en algunas imágenes habrá más descriptores que en otras. Estos descriptores SIFT son vectores de 128 dimensiones y, con grandes cantidades de imagenes, pueden haber problemas de rendimiento. Para subsanar esto, se realiza un Análisis de Componentes Principales (PCA) con tal de reducir la complejidad espacial y temporal, dejando sólo los descriptores más significativos.

4.2.2. XY Enrichment

Estos SIFT que se acaban de extraer tienen un problema importante para el enfoque de este proyecto, ya que no co-

difican explicitamente la posición de las características. Esta información es importante cuando hablamos de palabras manuscritas, ya que sin información espacial dos palabras con las mismas letras y la misma tipografía pero en orden diferente, tendrían los mismos descriptores. Es decir, 'dog' tendría la misma codificación que 'god'.

Para solucionarlo, utilizaremos una técnica llamada 'xy enrichment'. Para esto, se normalizan las posiciones de cada uno de los keypoints, haciendo que el margen superior derecho tenga las coordenadas (0.5,0,5) y el inferior izquierdo (-0.5,-0.5). Estas dos posiciones normalizadas se añaden al SIFT, haciendo que tenga dos dimensiones más. En caso de no haber PCA, obtendríamos un descriptor final de 130 dimensiones.

4.2.3. Gaussian Mixture Model

La codificación de los Fisher Vectors agrupa un gran conjunto de descriptores SIFT como los vistos en el apar-tado anterior. En general, esto se hace entrenando un Gaussian Mixture Model (GMM) con todos los SIFT. En la literatura de la visión por computador, una GMM que modeliza descriptores locales de cualquier imagen se hace llamar 'universal probabilistic visual vocabulary'. Cuando este GMM converja, se habrán agrupado los descriptores SIFT densos según sus características y obtendremos los parámetros necesarios para generar nuestros Fisher Vectors.

En esta etapa, se deberá elegir en cuantos clusters queremos que se agrupen los descriptores. Este parámetro K está directamente relacionado con la precisión final del sistema. En el apartado de Resultados, se verá cómo evoluciona la precisión para diferentes valores de K.

4.2.4. Fisher Vector Generation

Una vez el GMM converja, se procede a la generación de los Fisher Vectors.

Se utiliza la fórmula de la Fig. 10:

$$\Phi_k^{(1)} = \frac{1}{N\sqrt{w_k}} \sum_{p=1}^{N} \alpha_p(k) \left(\frac{x_p - \mu_k}{\sigma_k} \right), \quad \Phi_k^{(2)} = \frac{1}{N\sqrt{2w_k}} \sum_{p=1}^{N} \alpha_p(k) \left(\frac{(x_p - \mu_k)^2}{\sigma_k^2} - 1 \right)$$

Fig. 10: Fórmula para la generación de FV's

Donde:

[w,μ,σ]: Pesos, medias y covarianzas respectivamente del GMM que hemos entrenado con todas las imágenes del conjunto de training.

Xp : Valor de la feature p del descriptor X.

N : Numero total de dimensiones del descriptor.

αp(k): Peso del soft-assignment de la feature p a la gaus-siana k.

Una vez obtenidos los valores de Φ k para todas las K, el Fisher Vector Ø que describe globalmente a esa imagen se obtiene concatenándolos.

$$\phi = \left[\Phi_1^{(1)}, \Phi_1^{(2)}, \ldots, \Phi_K^{(1)}, \Phi_K^{(2)} \right]$$

Fig. 11: Concatenación para la obtención del FV

Así obtenemos un vector de K(1+2D) dimensiones. Donde K es el número de subpoblaciones de la GMM y D la dimensionalidad de los descriptores locales SIFT.

Esta codificación describe cómo la distribución de los descriptores de una imagen en particular difiere de la distribución de los descriptores del resto de imágenes del conjunto de training.

Posteriormente se normaliza el FV, mediante 'l2-normalization' y 'power normalization'.

4.3. Clasificacion de Fisher Vectors

Los Fisher Vectors son un método 'state-of-art' de clasificación, entre otras cosas, porque se comportan muy bien con clasificadores lineales.

En este proyecto se han utilizado SVM (Support Vector Machines) para la clasificación de los FV por su simplicidad y eficiencia. Dado que se quiere clasificar sobre varias clases, se ha seguido la estrategia One-vs-All implementada en el clasificador LinearSVC de la librería sklearn.

En esta etapa, se entrena la SVM con todos los FV del conjunto de training, donde cada uno de estos FV tendrá una etiqueta asociada, que será la 'transcripción' de la palabra.

Cuando la SVM converja, el sistema estará listo para clasificar nuevas imágenes desconocidas.

5 Experimentacion y resultados

En esta sección se detallará el proceso seguido para la validación de los resultados y una discusión sobre estos.

5.1. Dataset

En este proyecto ha servido de apoyo un banco de imagenes que consiste en una serie de documentos de la 'Bentham Collection'[6]. En este dataset tenemos un total de 1016 imágenes de palabras manuscritas, entre las cuales encontramos 173 palabras diferentes que nos servirán para entrenar y testear el sistema.

5.2. Validacion

Para obtener unos resultados fiables para cada una de las pruebas, se han ido escogiendo diferentes conjuntos de training y de test, haciendo 10 iteraciones y promediando la precisión de todas ellas.

Para hacer esto, se ha utilizado un método parecido al K-Fold, pero adaptado a la naturaleza de este proyecto.

Dado que K-Fold escoge un conjunto de datos y los divide en K subconjuntos, se corre el riesgo de que en estos subconjuntos haya un gran sesgo. Puesto que tenemos aproximadamente 5 imágenes de cada palabra, es posible que se asignen al subconjunto de test 4 de ellas y sólo quede una para el conjunto de training. Esto no sería una representación fiel a la realidad, así que los conjuntos se asignan de la siguiente forma:

Para cada uno de los labels, se selecciona una imagen aleatoria que lo represente y se añade al conjunto de test. Así el tamaño del conjunto de test siempre será el numero de palabras diferentes que existen en el dataset.

El resto de palabras, que no son más que repeticiones de las anteriores, se asignan al conjunto de training.

Con esta metodología nos aseguramos que, dada una imagen de entrada, se ha entrenado el sistema para el resto de imagenes con su misma etiqueta.

5.3. Calibración de parámetros

En esta etapa se han hecho pruebas en cuanto a los parámetros más significativos del sistema. Estos son, por una parte, el número de gaussianas K con las que se ha entrenado el Gaussian Mixture Model. Y, por otra parte, el patchsize que se ha utilizado en el DenseSIFT.

Es importante puntualizar que hay un factor de aleatoriedad en los resultados obtenidos. El entrenamiento de la GMM y del SVM, dada su naturaleza, puede resultar en diferentes modelos en diferentes ejecuciones consecutivas.

En lo que concierne al parámetro K (número de gaussianas en la GMM) se han realizado tests con un patchsize fijo de 100px. Obteniendo unos resultados como los de la Figura 15.

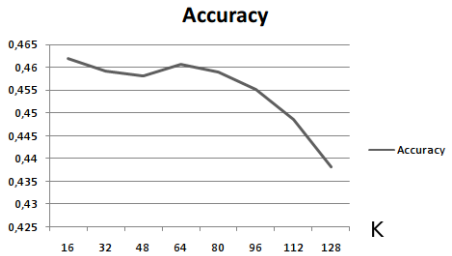

Fig. 12: Evolución de la precisión en funcion de K

Se puede observar una caída de la precisión conforme el parámetro K aumenta. La caída no es muy significativa, ya que se trata aproximadamente de un 2 %. Elegiremos una K=16 para el resto de nuestros experimentos ya que, tendremos mejor rendimiento para una precisión similar. Esta mejora en el rendimiento se debe a la menor dimensionalidad de los Fisher Vectors, que dependen directamente de K.

Otro parámetro importante es el 'patchsize', que determina el tamaño de la cuadrícula en la extracción de descriptores locales. Cuanto más grande sea este parámetro, menos descriptores locales densos obtendremos, y éstos abarcarán más espacio.

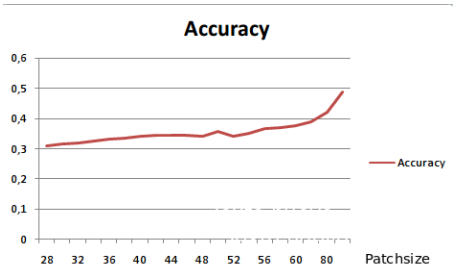

Fig. 13: Evolución de la precisión en funcion del Patchsize

Vemos que la precisión aumenta paulatinamente con la sucesión de patch-sizes que se observan en el eje X. Podemos deducir que es más acertado tener una cuadrícula más grande que abarque más espacio, teniendo descriptores más generales de la imagen.

Con un patch size de 100px se extraen pocos descriptores SIFT densos, pero parecen lo suficientemente ricos para describir la imagen.

También observamos que los patch sizes afectan al rendimiento. A mayor patch size, menos descriptores densos, y más rápida convergencia del GMM.

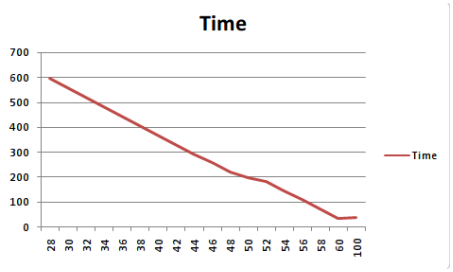

Fig. 14: Evolución de la precisión en funcion del Patchsize

Finalmente, analizamos la composición del tiempo de entrenamiento.

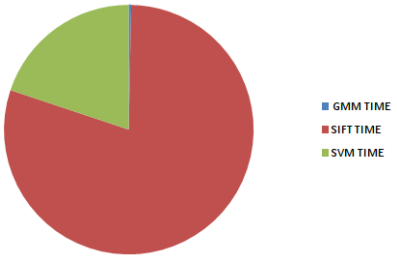

Fig. 15: Proporción del tiempo invertido en diferentes tareas

Como vemos en la figura, la mayor inversión de tiempo es en el momento de extraer descriptores locales SIFT. Lo cual es habitual, ya que se deben extraer varios descriptores locales para cada una de las imágenes del conjunto de test, lo cual es el proceso más costoso. En segundo lugar se encuentra el entrenamiento del SVM y en tercer lugar el entrenamiento del GMM.

Dado que el patchsize es grande, la proporción de tiempo dedicada al entrenamiento de la GMM es muy pequeña. Esto se debe a que con patchsizes grandes se obtienen menos descriptores locales y con poca cantidad de datos, es más fácil que converja el GMM. Con un dataset más grande este tiempo crecería proporcionalmente a la cantidad de descriptores SIFT.

Con estos resultados, se concluye que una buena configuración de parámetros sería K=16, Patchsize=100px

5.4. Otros experimentos

En este apartado se analizan los resultados bajo la métrica de la precisión. Esto es, cuántas imagenes se han etiquetado bien del total de imágenes etiquetadas. También se tiene en cuenta el rendimiento.

5.4.1. Pre-procesamiento

Se han realizado pruebas tanto con el módulo de preprocesamiento como sin él. Sorprendentemente, se han encontrado precisiones no muy dispares, esto podría ser debido a que en este dataset, la imagenes requieren de poco

pre-procesamiento. Es decir, por lo general, las imágenes de este dataset están bien alineadas y con buen contraste respecto al fondo.

	Accuracy
Preprocessing	53 %
Non-Preprocessing	48 %

Tabla 1: COMPARATIVA PRE-PROCESAMIENTO

De todas maneras, sigue siendo preferible utilizar preprocessing para tener una mayor robustez.

5.4.2. Fallos en corrección de skew

A veces, las palabras estan mal recortadas y tenemos intrusiones de fragmentos de otras palabras o de la propia palabra. Si estas intrusiones son por la parte baja de la imagen, el proceso de corrección de skew se puede ver afectado. Dado que opera con los puntos más bajos de la imagen para obtener el baseline.

Fig. 16: Ejemplo de fallo en skew

Dado que los puntos más bajos de la imagen en la Fig 16 no están siendo realmente la 'baseline' de la palabra, el proceso se ve afectado y falla.

6 CONCLUSIONES

En este proyecto se ha visto un método de clasificación para grandes cantidades de imágenes que podría ser útil en la indexación de documentos manuscritos.

Se ha conseguido hacer una primera aproximación mediante Fisher Vectors, un método que requiere gran tiempo de cómputo en la etapa de training, pero que en el momento de la validación es muy rápido. Esto significa que, una vez entrenado el sistema, cualquier transcripción de imagen a palabra será barata en tiempo y recursos.

Se han conseguido unos resultados de 50 %, lo cual es un buen comienzo dado que no se ha entrado en el tema de aprendizaje de atributos. Esto significa que, aprovechando las etiquetas de las palabras en la etapa de training de el GMM, se pueden lograr unas precisiones mucho mayores.

La progresión lógica y trabajo futuro de este proyecto sería mejorarlo mediante el aprendizaje de atributos con un enfo-que parecido al de Jon Almazan et al.[5]

AGRADECIMIENTOS

Gracias a Marçal Russinyol, mi tutor, que me ha guiado para poder sacar el proyecto adelante en un campo muy complejo y abstracto como es el de la Visión por Computador.

REFERENCIAS

[1] Jorge Sanchez, Florent Perronnin, Thomas Mensink, Jakob Verbeek. Image Classification with the Fisher Vector: Theory and Practice. [Research Report] RR-8209, INRIA. 2013.

[2] Sheneman, B. 'Scene Recognition with Bag of Words'.http://www.cc.gatech.edu/~hays/compvision/results/proj4/bsheneman3/index.html

[3] Simonyan et al. 'Fisher Vector Faces in the Wild'. 2013.https://www.robots.ox.ac.uk/~vgg/publications/2013/Simonyan13/simonyan13.pdf

[4] Florent Perronnin et al. 'Fisher Vectors Meet Neural Networks: a Hybrid Classification Architecture'. 2015.

[5] Almazán, J. Gordo, Fornés, A. Valveny E. 'Word Spotting and Recognition with Embedded Attributes' 2014. http://www.cvc.uab.es/~almazan/wpcontent/uploads/2014/01/almazan_pami14.pdf

[6] Transcriptorium Dataset http://transcriptorium.eu/~icdar15kws/data/Validation_QueryByExample.zip